

# Workflow Patterns



## Workflow Data Patterns

### Download of the data patterns paper:

N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst.

[Workflow Data Patterns](#). (PDF, 423 Kb).

QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004.

### Introduction

Workflow systems seek to provide an implementation vehicle for complex, recurring business processes. Notwithstanding this common objective, there are a variety of distinct features offered by commercial workflow management systems. These differences result in significant variations in the ability of distinct tools to represent and implement the plethora of requirements that may arise in contemporary business processes. Many of these requirements recur quite frequently during the requirements analysis activity for workflow systems and abstractions of these requirements serve as a useful means of identifying the key components of workflow languages. Previous work has identified a number of Workflow Control-flow Patterns which characterise the range of control flow constructs that might be encountered when modelling and analysing workflow. In this web site you will find a series of Workflow Data Patterns that aim to capture the various ways in which data is represented and utilised in workflows. By delineating these Patterns in a form that is independent of specific workflow technologies and modelling languages, we are able to provide a comprehensive treatment of the workflow data perspective and we subsequently use these Patterns as the basis for a detailed comparison of a number of commercially available workflow management systems and business process modelling languages.

Before you view the different data patterns, you may wish to examine one of the following options to gain a better understanding of the work presented to you:

- To get a better understanding of the data characteristics that occur repeatedly in workflow systems, go to the [data characterisation](#) web page.
- To get a better understanding of the basic terms and concepts used in this body of work, go to the [workflow structure](#) web page.

### Data Visibility

Within the context of a workflow engine, there are a variety of distinct ways in which data elements can be defined and utilised. Typically these variations relate to the manner in which they are declared and the main workflow construct to which they are anchored. More importantly, they directly influence the way in which the data element may be used e.g. to capture production information, to manage control data or for communication with the external environment. Here we consider each of the potential contexts in which a data construct can be defined and utilised.

#### 1. [Task Data](#)

2. [Block Data](#)
3. [Scope Data](#)
4. [Multiple Instance Data](#)
5. [Case Data](#)
6. [Folder Data](#)
7. [Workflow Data](#)
8. [Environment Data](#)

## Data Interaction

Here we examine the various ways in which data elements can be passed between components in a workflow process and how the characteristics of the individual components can influence the manner in which the trafficking of data elements occurs. Of particular interest is the distinction between the communication of data between components within a workflow engine as against the data-oriented interaction of a workflow element with the external environment.

### Internal Data Interaction

9. [Data Interaction - Task to Task](#)
10. [Data Interaction - Block Task to Sub-Workflow Decomposition](#)
11. [Data Interaction - Sub-Workflow Decomposition to Block Task](#)
12. [Data Interaction - to Multiple Instance Task](#)
13. [Data Interaction - from Multiple Instance Task](#)
14. [Data Interaction - Case to Case](#)

### External Data Interaction

15. [Data Interaction - Task to Environment - Push-Oriented](#)
16. [Data Interaction - Environment to Task - Pull-Oriented](#)
17. [Data Interaction - Environment to Task - Push-Oriented](#)
18. [Data Interaction - Task to Environment - Pull-Oriented](#)
19. [Data Interaction - Case to Environment - Push-Oriented](#)
20. [Data Interaction - Environment to Case - Pull-Oriented](#)
21. [Data Interaction - Environment to Case - Push-Oriented](#)
22. [Data Interaction - Case to Environment - Pull-Oriented](#)
23. [Data Interaction - Workflow to Environment - Push-Oriented](#)
24. [Data Interaction - Environment to Workflow - Pull-Oriented](#)
25. [Data Interaction - Environment to Workflow - Push-Oriented](#)
26. [Data Interaction - Workflow to Environment - Pull-Oriented](#)

## Data Transfer Mechanisms

Here we consider the manner in which the actual transfer of data elements occurs between one workflow component and another. These Patterns aim to capture the various mechanisms by which data elements can be passed across the interface of a workflow component. The specific style of data passing that is used in a given scenario depends on a number of factors including whether the two components share a common address space for data elements, whether it is intended that a distinct copy of an element is passed as against a reference to it and whether the component receiving the data element can expect to have exclusive access to it. These variations give rise to a number of distinct Patterns as described below.

27. [Data Transfer by Value - Incoming](#)
28. [Data Transfer by Value - Outgoing](#)
29. [Data Transfer - Copy In/Copy Out](#)
30. [Data Transfer by Reference - Unlocked](#)

- 31. [Data Transfer by Reference - With Lock](#)
- 32. [Data Transformation - Input](#)
- 33. [Data Transformation - Output](#)

## Data-based Routing

Whereas the above sections have examined characteristics of data elements in isolation from other workflow perspectives (i.e. control, resource, organisational etc.), the following Patterns capture the various ways in which data elements can interact with other perspectives and influence the overall operation of the workflow.

- 34. [Task Precondition - Data Existence](#)
- 35. [Task Precondition - Data Value](#)
- 36. [Task Postcondition - Data Existence](#)
- 37. [Task Postcondition - Data Value](#)
- 38. [Event-based Task Trigger](#)
- 39. [Data-based Task Trigger](#)
- 40. [Data-based Routing](#)

## Disclaimer

We, the authors and the associated institutions, assume no legal liability or responsibility for the accuracy and completeness of any product-specific information contained in this body of work. All possible efforts have been made to ensure that the results presented are, to the best of our knowledge, up to date and correct.

---

© 2007 *Workflow Patterns Initiative*

0025696

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 1 (Task Data)

[FLASH animation of Task Data pattern](#)

### Description

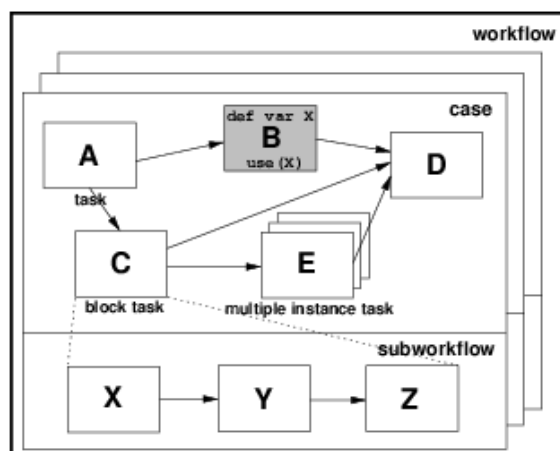
Data elements can be defined by tasks which are accessible only within the context of individual execution instances of that task.

### Example

The *working trajectory* variable is only used within the *Calculate Flight Path* task.

### Motivation

To provide data support for local operations at task level. Typically these data elements will be used to provide working storage during task execution for control data or intermediate results in the manipulation of production data. Figure 2 illustrates the declaration of a task data element (variable X in task B) and the scope in which it can be utilised (shown by the shaded region and the use() function). Note that it has a distinct existence (and potential value) for each instance of task B (i.e. in this example it is instantiated once for each workflow case since task B only runs once within each workflow).



**Figure 2:** Task level data visibility

### Implementation

The implementation of task data in a workflow system takes one of two forms - either data elements are defined as parameters to the task making them available for use within the task or they are declared within the definition of the task itself. In either case, they are bound in scope to the task block and have a lifetime that corresponds to that of the execution of an instance of that task.

## Issues

One difficulty that can arise is the potential for a task to declare a data element with the same name as another data element declared elsewhere (either within another task or at a different level within the workflow hierarchy (e.g. block, case, process level) that can be accessed within the task. A second issue that may require consideration can arise where a task is able to execute more than once (e.g. in the case of a multi-merge [\[AHKB03\]](#)). When the second (or later) instance commences, should the data elements contain the values from the first instance or should they be re-initialised.

## Solutions

The first issue can be addressed through the use of a tight binding approach at task level restricting the use of data elements within the task to those explicitly declared by the task itself and those which are passed to the task as formal parameters. All of these data element names must be unique within the context of the task. An alternative approach is employed in BPEL4WS, which only allows access to the data element with the innermost scope in the event of name clashes. Indeed, this approach is proposed as a mechanism of "hiding" data elements from outer scopes by declaring another with the same name at task level. The second issue is not a major consideration for most workflow tools which initialise data elements at the commencement of each task instance. One exception to this is FLOWer which provides the option for a task instance which comprises part of a loop in a workflow to either refresh data elements on each iteration or to retain their values from the previous instance (in the preceding or indeed any previous loop iteration).

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	+/-	Indirectly supported by 3GL program implementations
FLOWer	3.0	+/-	Use of restricted option allows update of data element to be limited to single step but other steps can view data value.
COSA	4.2	+	Directly supported by (non-persistent) STD attributes at the activity level
XPDL	1.0	-	Only workflow processes can have data elements
BPEL4WS	1.1	+/-	Scopes provide a means of limiting the visibility of variables to smaller blocks approaching task level binding
BPMN	1.0	+	The smallest operational unit in a BPMN diagram is Task. Task data is defined through the attribute Properties of a Task.

			The Properties defined for a Task are local and can only be used within the Task.
UML	2.0	+/-	Indirectly supported where a local action language is utilised which provides action-specific variables
Oracle BPEL	10.1.2	+/-	A task must be wrapped into a scope

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Data can be explicitly declared at task level with task level scoping</li><li>2. Data support exists within the workflow tool</li></ol>	<ol style="list-style-type: none"><li>1. Task data can be indirectly specified via a programmatic implementation of a task</li><li>2. Task data can be specified at task level but is visible (although not mutable) outside</li><li>3. Facilities do not exist to ensure data elements are initialised for each task instance</li></ol>

© 2007 *Workflow Patterns Initiative*

0025696

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))



# Workflow Patterns



## Pattern 2 (Block Data)

[FLASH animation of Block Data - Global pattern](#)

[FLASH animation of Block Data - Reference pattern](#)

[FLASH animation of Block Data - Value pattern](#)

### Description

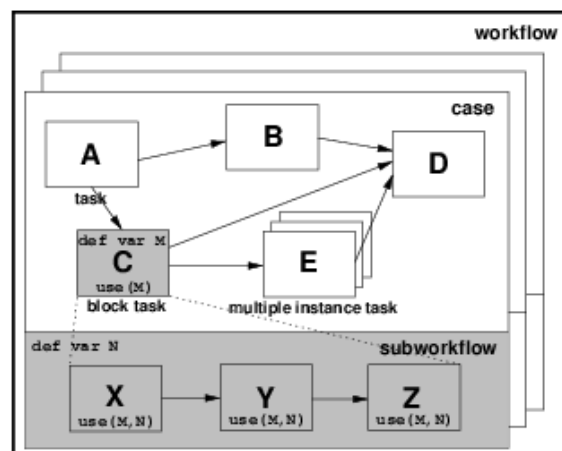
Block tasks (i.e. tasks which can be described in terms of a corresponding sub-workflow) are able to define data elements which are accessible by each of the components of the corresponding sub-workflow.

### Example

All components of the sub-workflow which define the *Assess Investment Risk* block task can utilise the *security details* data element.

### Motivation

The manner in which a block task is implemented is usually defined via its decomposition into a sub-workflow. It is desirable that data elements available in the context of the undecomposed block task are available to all of the components that make up the corresponding sub-workflow. Similarly, it is useful if there is the ability to define new data elements within the context of the sub-workflow that can be utilised by each of the components during execution. Figure 3 illustrates both of these scenarios, data element M is declared at the level of the block task C and is accessible both within the block task instance and throughout each of the task instances (X, Y and Z) in the corresponding sub-workflow. Similarly data element N is declared within the context of the sub-workflow itself and is available to all task instances in the sub-workflow. Depending on the underlying workflow system, it may also be accessible at the level of the corresponding block task.



**Figure 3:** Block level data visibility

## Implementation

The concept of block data is widely supported by workflow systems and all but one of the offerings examined in this survey which supported the notion of sub-workflows (footnote [1](#)) implemented it in some form. Staffware allows sub-workflows to specify their own data elements and also provides facilities for parent processes to pass data elements to sub-workflows as formal parameters. In Websphere MQ Workflow, sub-workflows can specify additional data elements in the data container that is used for passing data between task instances within the sub-workflow and restrict their scope to the sub-workflow. FLOWer and COSA also provide facilities for specifying data elements within the context of a sub-workflow.

## Issues

A major consideration in regard to block-structured tasks within a workflow is the handling of block data visibility where cascading block decompositions are supported and data elements are implicitly inherited by sub-workflows. As an example, in the preceding diagram block data sharing would enable a data element declared within the context of task C to be utilised by task X, but if X were also a block task would this data element also be accessible to task instances in the sub-workflow corresponding to X?

## Solutions

One approach to dealing with this issue adopted by workflow tools such as Staffware is to only allow one level of block data inheritance by default i.e. data elements declared in task instance C are implicitly available to X, Y and Z but not to further sub-workflow decompositions. Where further cascading of data elements is required, then this must be specifically catered for. COSA allows a sub-workflow to access all data elements in a parent process and provides for arbitrary levels of cascading (footnote [2](#)), however updates to data elements in sub-workflows are not automatically propagated back to the parent task.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Directly supported - each (sub)procedure can maintain a distinct set of data fields
Websphere MQ Workflow	3.4	+	Supported via global data containers for a process
FLOWer	3.0	+	Each plan can have its own data elements
COSA	4.2	+	Workflows can be nested and attributes of the form INSTANCE.name provide a means of supporting block data



XPDL	1.0	+	Supported through the block activity construct
BPEL4WS	1.1	-	Sub-processes are not supported
BPMN	1.0	+	Through the attribute Properties of a Sub-Process. The Properties defined for a Sub-Process are local and accessible to all Sub-Process components.
UML	2.0	+	Directly supported through parameters to activities which are accessible to all activity components
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Data can be explicitly declared at block task level with block task level scoping.</li><li>2. Data support exists within the workflow tool</li><li>3. Facilities exist for formal parameter passing to and from a block</li></ol>	<ol style="list-style-type: none"><li>1. Block data can be indirectly specified via a programmatic implementation of a block task or sub-workflow</li></ol>

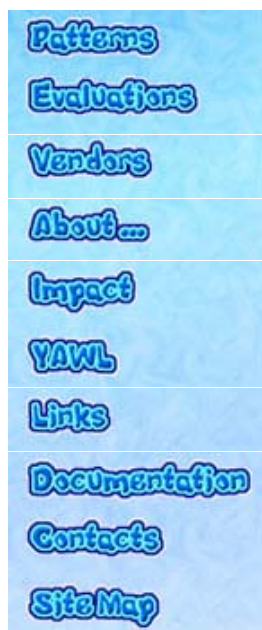
© 2007 *Workflow Patterns Initiative*

0025696

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 3 (Scope Data)

[FLASH animation of Scope Data pattern](#)

### Description

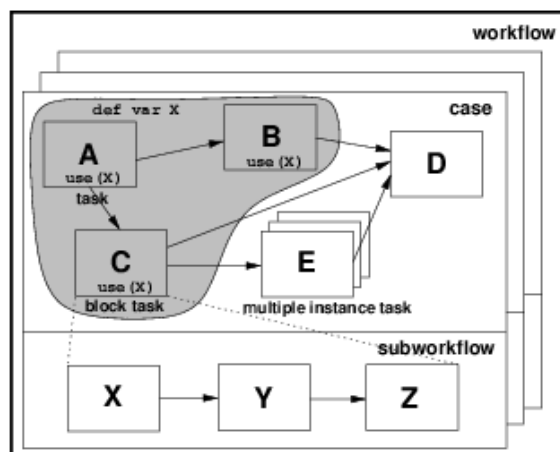
Data elements can be defined which are accessible by a subset of the tasks in a case.

### Example

The *initial tax estimate* variable is only used within the *Gather Return Data*, *Examine Similar Claims* and *Prepare Initial Return* tasks in the *Prepare Tax Return* process.

### Motivation

Where several tasks within a workflow coordinate their actions around a common data elements or set of data elements, it is useful to be able to define data elements that are bound to that subset of tasks in the overall workflow process. Figure 4 illustrates the declaration of data element X which is scoped to tasks A, B and C. It can be freely accessed by these tasks but is not available to tasks D and E. One of the major justifications for scopes in workflows is that they provide a means of binding data elements, error and compensation handlers to sets of related tasks within a case. This allows for more localised forms of recovery action to be undertaken in the event that errors or concurrency issues are detected.



**Figure 4:** Scope level data visibility

### Implementation

The definition of scope data elements requires the ability to define the portion of the workflow process to which the data elements are bound. This is potentially difficult in workflows that are based on a graphical process notation but less difficult for those

that utilise a textual definition format such as XML. A significant distinction between scopes and blocks in a workflow context is that scopes provide a grouping mechanism within the same address space (or context) as the surrounding case elements. They do not define a new context and data passing to tasks within the scope does not rely on any specific data passing mechanisms other than normal task-to-task data transfer facilities. BPEL4WS is the only offering examined that fully supports the notion of scope data elements. It provides support for a scope construct which allows related activities, variables and exception handlers to be logically grouped together. FLOWer supports "restricted data elements" which can have their values set by nominated tasks although they are more widely readable.

## Issues

Potential exists for variables named within a scope to have the same name as a variable in the surrounding block in which the scope is defined.

## Solutions

The default handling for this BPEL4WS is that the innermost context in which a variable is defined indicates which variable should be used in any given situation. Variables within a given scope must be unique.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+/-	Restricted data elements allow data update to be limited to defined activities
COSA	4.2	-	Not supported
XPDL	1.0	-	Not supported
BPEL4WS	1.1	+	Directly supported by scope construct
BPMN	1.0	-	Not supported. The Group is a construct introduced purely for visualisation purposes and it does not provide any data handling for the objects it groups together.
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Directly supported by <scope>

## Summary of Evaluation

+ Rating	+/- Rating

1. Direct workflow support for binding data elements to a subset of the task instances in the workflow process	1. Ability to limit data update or read actions to defined tasks 2. Achievable via programmatic extensions
--	---

---

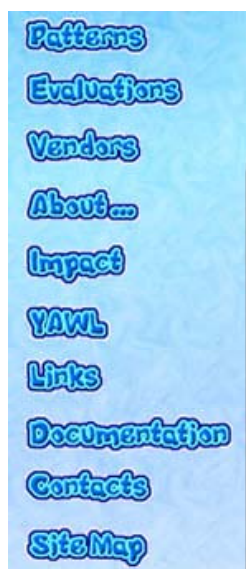
© 2007 *Workflow Patterns Initiative*

0025696

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 4 (Multiple Instance Data)

[FLASH animation of Multiple Instance Data pattern](#)

### Description

Tasks which are able to execute multiple times within a single workflow case can define data elements which are specific to an individual execution instance.

### Example

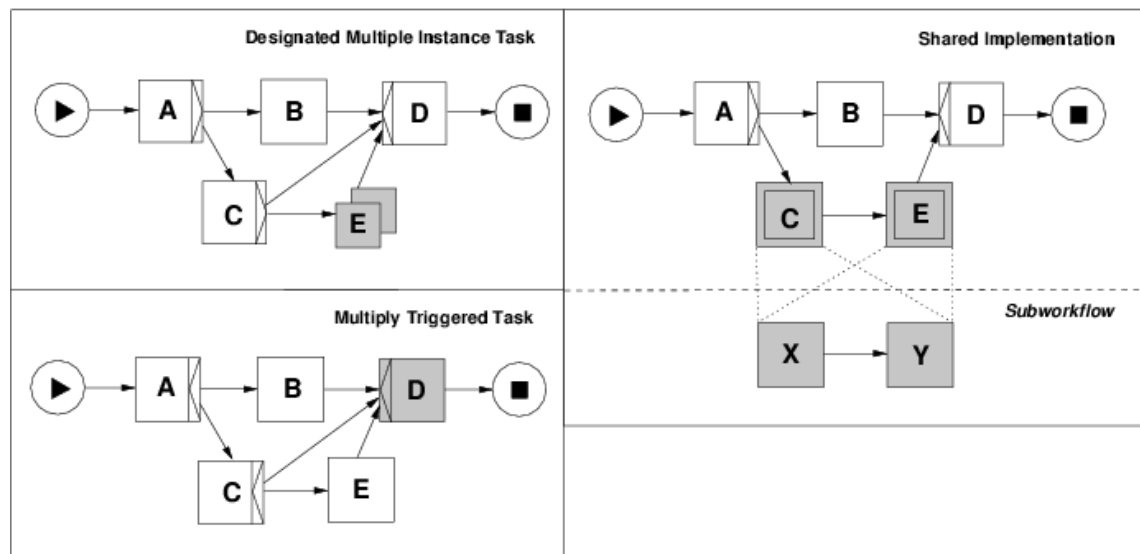
Each instance of the *Expert Assessment* task is provided with the *case history* and *test results* at commencement and manages its own *working notes* until it returns a *verdict* at completion.

### Motivation

Where a task executes multiple times, it is useful to be able to define a set of data elements which are specific to each individual execution instance. The values of these elements may be passed to the task instance at commencement and at the conclusion of its execution they can be made available (either on an individual basis or as part of an aggregated data element) to subsequent tasks. There are three distinct scenarios in which a task could be executed more than once:

1. Where a particular task is designated as a multiple instance task and once it is enabled, multiple instances of it are initiated simultaneously.
2. Where distinct tasks in a workflow process share the same implementation.
3. Where a task can receive multiple initiation triggers (i.e. multiple tokens in a Petri-net sense) during the operation of a workflow case.

Each of these scenarios is illustrated in Figure 5 using the YAWL notation. In the top lefthand diagram, task E illustrates a multiple instance task. In the bottom lefthand diagram, task D corresponds to both an OR-join followed by a task invocation. When the OR-join construct receives a control flow triggering from any of the incoming arcs, it immediately invokes the associated task. If it receives multiple triggers at distinct points in time, this results in the task being invoked multiple times. In the righthand diagram, tasks C and E both share the same implementation that is defined by the subworkflow containing tasks X and Y. Further details on YAWL can be found in [\[AH05\]](#).



**Figure 5:** Alternative implementations of multiple instance tasks

## Implementation

The ability to support distinct data elements in multiple task instances presumes the workflow engine is also able to support data elements that can be bound specifically to individual tasks (i.e. Pattern 1) in some form. Workflow engines lacking this capability are unable to facilitate the isolation of data elements between task instances for any of the scenarios identified above. In addition to this, there are also other prerequisites for individual scenarios as described below. In order to support multiple instance data in the first of the scenarios identified above, the workflow engine must also support designated multiple instance tasks and it must provide facilities for composite data elements (e.g. arrays) to be split up and allocated to individual task instances for use during their execution and for these data elements to be subsequently recombined for use by later tasks. For the second scenario, it must be possible for two or more distinct block tasks to share the same implementation (i.e. the same underlying sub-workflow) and the workflow engine must support block-level data. Additionally these data elements must be able to be allocated values at commencement of the sub-workflow and for their values to be passed back to the calling workflow task once the sub-workflow has completed execution. The third scenario requires the workflow engine to provide task-level data binding and support the capability for a given task to be able to receive multiple triggers. Each of the instances should have a mutually distinct set of data elements which can receive data passed from preceding tasks and pass them to subsequent tasks. Of the various multiple instance scenarios identified, FLOWer provides support for the first of them (footnote [3](#)). Websphere MQ Workflow, COSA and XPD L can support the second and Websphere MQ Workflow and COSA directly support the third scenario. Staffware can potentially support the third scenario also, however programmatic extensions would be necessary to map individual instances to distinct case level variables.

## Issues

A significant issue that arises for workflow systems that support designated multiple instance tasks such as FLOWer involves the partitioning of composite data elements (such as an array) in a way that ensures each task instance receives a distinct portion of the data element and also that the entire data element is passed to one of the multiple task instances.

## Solutions

FLOWer has a unique means of addressing this problem through mapping objects which allow sections of composite data element in the form of an array (e.g. X[1], X[2] and so on) to be allocated to individual instances of a multiple instance task (known as a dynamic plan). Each multiple task instance only sees the element it has been allocated and each task instance has the same naming for each of these elements (i.e. X). At the conclusion of all of the multiple instances,



the data elements are coalesced back into the composite form together with any changes that have been made.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Only scenerio supported is multiple task triggering but there is not direct means of ensuring data independence of each invocation
Websphere MQ Workflow	3.4	+	Support for distinct data elements in multiply triggered tasks and tasks with shared sub-workflow decompositions
FLOWer	3.0	+	Fully supported through dynamic plans
COSA	4.2	+	Support for distinct data elements in multiply triggered tasks and tasks with shared sub-workflow decompositions
XPDL	1.0	+	Support for distinct data elements in tasks with shared sub-workflow decompositions
BPEL4WS	1.1	-	Not supported. Data elements are scoped at case level
BPMN	1.0	+/-	Two out of three possible scenarios are supported, namely: i) Where a task can be triggered multiple times, e.g., as part of a loop or as a task following a Multiple Merge construct. ii) Where two tasks share the same decomposition. This is supported through the notion of Independent Sub-Processes. iii) Where a task is specifically designated as having multiple instances in the process model is not supported. The lack of any Properties attribute for the MI (in Table 18, MI Loop Activity Attributes), makes it impossible to handle any instance-specific data for the different instances of a task.
UML	2.0	+	Directly supported through the Expansion Kind and data objects.
Oracle BPEL	10.1.2	+/-	Partial support dependents on the type of the MI task

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"> <li>1. The data element is capable of being replicated or partitioned across multiple tasks</li> <li>2. Each of these data instances exist in their own address space</li> <li>3. The instances are able to be accessed from a higher level in the process hierarchy</li> </ol>	<ol style="list-style-type: none"> <li>1. Later instances of the same task retain data values from the execution of an earlier instance</li> <li>2. Data isolation between multiple task instances can be achieved through programmatic extensions</li> </ol>

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) | [Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 5 (Case Data)

[FLASH animation of Case Data pattern](#)

### Description

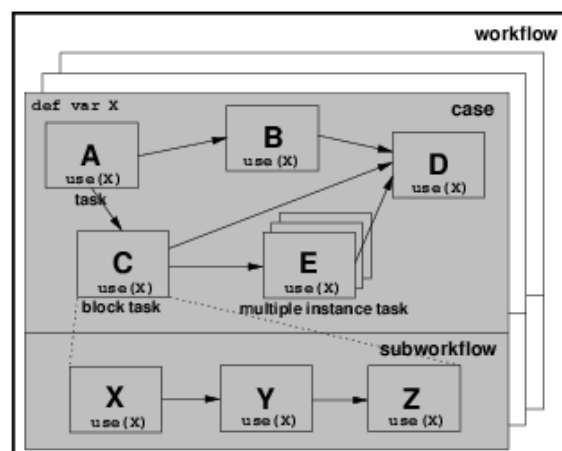
Data elements are supported which are specific to a process instance or case of a workflow. They can be accessed by all components of the workflow during the execution of the case.

### Example

The *employee assessment results* can be accessed by all of the tasks during this execution instance of the *Performance Assessment* workflow.

### Motivation

Data elements defined at case level effectively provide global data storage during the execution of a specific case. Through their use, data can be made accessible to all workflow components without the need to explicitly denote the means by which it is passed between them. Figure 6 illustrates the use of the case level data element X which is utilised by all of the tasks throughout a workflow case (including those in sub-workflows).



**Figure 6:** Case level data visibility

### Implementation

Most workflow engines support the notion of case data in some form, however the approaches to its implementation vary widely. Staffware implements a data management strategy that is based on the notion of a common data store for each workflow case although individual data fields must be explicitly passed to sub-workflows in order to make them accessible to the tasks within them. Websphere

MQ Workflow takes a different approach with a global data store being defined for each workflow case for case level data elements but distinct data passing conventions needing to be specified to make this data accessible to individual tasks. COSA provides a series of tiers of data constructs with the INSTANCE tool agent corresponding to case level data elements which are globally accessible throughout a workflow case (and associated sub-workflows) by default. In FLOWer, XPDL and BPDL4WS, the default binding for data elements is at case level and they are visible to all of the components in a process.

## Issues

1. One consideration that arises with the use of case level data is in ensuring that these elements are accessible, where required, to the components of a sub-workflow associated with a specific workflow case (e.g. as the definition of a block task).
2. Perhaps the most significant issue associated with case level data is in managing concurrent access to it by multiple processes.

## Solutions

1. In some workflow tools, sub-workflows that are linked to a workflow process do not seem to be considered to be part of the same execution context as the main workflow process. To remedy this issue, tools such as Staffware and Websphere MQ Workflow require that case level data elements be explicitly passed to and from sub-workflows as parameters in order to make them visible to the various components at this level. In COSA, FLOWer and XPDL, they are visible to all sub-workflows by default.
2. Concurrency management is not an area that is currently well-addressed by commercial workflow engines. Where a workflow engine provides for case level data, the general solutions to the concurrency problem tend to be either to allow for the use of a third party transaction manager (e.g. Staffware which allows Tuxedo to be used for transaction management) or to leave the problem to the auspices of the workflow developer (e.g. COSA). There has been significant research interest over recent years in the various forms of advanced transaction support that are required for business processes [ [Gre02](#) , [RS95](#) , [AAA+96](#) , [WS97](#) , [DHL01](#) ] and several prototypes have been constructed which provide varying degrees of concurrency management and transactional support for workflow systems e.g. METEOR [\[KS95\]](#), EXOTICA [\[AAA+96\]](#), WIDE [\[GVA01\]](#), CrossFlow [\[VDGK00\]](#) and ConTracts [\[WR92\]](#), although most of these advances have yet to be incorporated in mainstream commercial products.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Main workflow procedure can maintain data fields for each case but they must be explicitly passed to sub-workflows
Websphere MQ Workflow	3.4	+	Supported through the default data structure

FLOWer	3.0	+	Default binding for a data element
COSA	4.2	+	Supported via tool agent INSTANCE (INSTANCE.name).
XPDL	1.0	+	A workflow process can have "data elements. In case of nesting the scope is not 100% clear
BPEL4WS	1.1	+	The default scoping for variables is process level
BPMN	1.0	+	Supported through the attribute Properties of a Process
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Bound to outermost scope in the process definition

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"> <li>1. Direct tool support for data elements at case level with case level scoping</li> <li>2. Case data visible to all components of the case</li> </ol>	<ol style="list-style-type: none"> <li>1. Effect of case level data sharing can be achieved through programmatic extensions</li> <li>2. Data elements must be passed to sub-workflows</li> </ol>

© 2007 *Workflow Patterns Initiative*

0025696

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
 ([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 6 (Folder Data)

[FLASH animation of Folder Data pattern](#)

### Description

Data elements can be defined which are accessible by multiple cases on a selective basis.

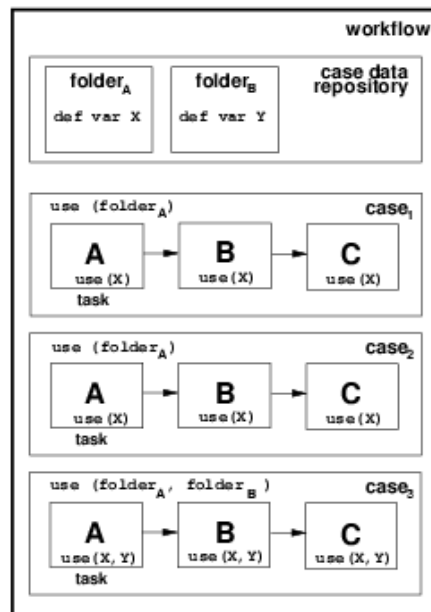
### Example

All instances of the *Approve Travel Request* task can access the *current cash reserves* data element regardless of the case in which they execute.

### Motivation

Folder data provides a mechanism for sharing a data element between related task instances in different workflow cases. This is particularly useful where tasks in multiple workflow cases are working on a related problem and require access to common working data elements. Figure 7 illustrates the notion of folder data. In essence, "folders" of related data elements are declared in the context of a workflow process prior to the execution of individual cases. Individual cases are able to nominate one or more of these "folders" that their task instances should have access to during execution. Access may be read-only or read-write. In Figure 7, two folders are declared (A and B) containing data elements X and Y respectively. During the course of execution, case and case have access to folder A whereas case has access to both folders A and B. As there is only one copy of each folder maintained, should any of case, case or case execute concurrently, then they will in effect share access to data element X. As a general rule, for folder data to be useful in a workflow engine, the cardinality of the accessibility relationship between folders and cases needs to be m-n i.e. data elements in a given folder need to be accessible to more than one case and a given case needs to be able to access more than one data folder during execution.





**Figure 7:** Folder data visibility

## Implementation

Of the workflow engines examined, only COSA offers the facility to share data elements between cases on a selective basis. It achieves this by allowing each case to be associated with a folder at commencement. At any time during execution of the case, it is possible for the default folder to be changed. The type of access (read-only or read-write) and a range of access controls can be specified for each folder at design time. It is also possible to use folders as repositories for more complex data elements such as documents.

## Issues

1. As each folder defines its own context, one consideration that arises where a case (or a task instance within a case) has access to multiple folders is how naming clashes are resolved where two data elements in distinct folders share the same name.
2. Concurrency control for folder level data is also a potential source of difficulty.

## Solutions

1. In the case of COSA, this problem is addressed by only allowing a case to access attributes from one folder at a time. This is achieved using a specific Tool Agent for folders. It is possible to change the folder to which a case refers at any time (again using a specific Tool Agent call).
2. Similar considerations apply to those discussed for case data. In the case of COSA, there is no direct system support to address this problem.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	-	Not supported
COSA	4.2	+	Fully supported via folders accessible to nominated activities across multiple cases
XPDL	1.0	-	Not supported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Direct tool support for data elements at case level with case level scoping</li><li>2. Ability to support the binding of folder data elements to cases on an m-n basis</li></ol>	<ol style="list-style-type: none"><li>1. Achievable via programmatic extensions</li></ol>

© 2007 *Workflow Patterns Initiative*

0025696

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 7 (Workflow Data)

[FLASH animation of Workflow Data pattern](#)

### Description

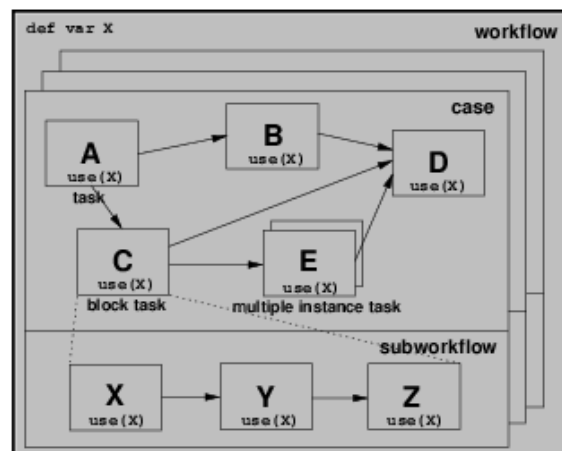
Data elements are supported which are accessible to all components in each and every case of the workflow and are within the control of the workflow system.

### Example

- The *risk/premium matrix* can be utilised by all of the cases of the *Write Insurance Policy* workflow and all tasks within each case.
- During the course of workflow execution, a number of cases will be selected for auditing each hour. The number selected will be based on the *average case execution time* for the previous six hours.

### Motivation

Some data elements have sufficiently broad applicability that it is desirable to make them accessible to every component in all cases of workflow execution. Data that falls into this category includes startup parameters to the workflow engine, global application data that is frequently used and production information that governs the potential course of execution that each workflow case may take. Figure 8 illustrates the extent of workflow data visibility. Note that in contrast to case level data elements which are typically only visible to tasks in the main workflow case in which they are declared, workflow-level data elements are visible globally throughout all workflow cases - both to the main body of the case and also to sub-workflows linked to it.



**Figure 8:** Workflow data visibility

### Implementation

In order to make data elements broadly accessible to all workflow cases, most workflow engines address this requirement by utilising persistent storage, typically in the form of a database. This may be provided directly as an internal facility by the workflow engine (tables, persistent lists, DEFINITION tool agents and packages in the case of Staffware, Websphere MQ Workflow, COSA and XPDL respectively) or by linking in or providing access facilities to a suitable third-party product.

## Issues

The main issue associated with workflow-level data is in managing concurrent access to it by multiple processes.

## Solutions

As discussed for Patterns 5 and 6. There has been significant research interest over recent years in the various forms of advanced transaction support that are required for business processes [ [Gre02](#) , [RS95](#) , [AAA+96](#) , [WS97](#) , [DHL01](#) ] and several prototypes have been constructed which provide varying degrees of concurrency management and transactional support for workflow systems e.g. METEOR [\[KS95\]](#), EXOTICA [\[AAA+96\]](#), WIDE [\[GVA01\]](#), CrossFlow [\[VDGK00\]](#) and ConTracts [\[WR92\]](#), although most of these advances have yet to be incorporated in mainstream commercial products.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Supported through tables and lists
Websphere MQ Workflow	3.4	+	Supported through persistent lists
FLOWer	3.0	-	Not supported
COSA	4.2	+/-	COSA allows for attributes of the form DEFINITION.name however these are fixed at design time
XPDL	1.0	+/-	It appears that the "data fields" of a package can be used for this. Its not clear how the values can be modified
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported
UML	2.0	+	Directly supported through Object-Nodes which are potentially accessible to all of the components in a UML 2.0 AD
Oracle BPEL	10.1.2	+	Supported via deployment descriptor properties

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Workflow data visible to all components of a workflow</li><li>2. Direct tool support for workflow level data</li></ol>	<ol style="list-style-type: none"><li>1. Effect of workflow level data sharing can be achieved through programmatic extensions</li><li>2. Unable to update workflow data elements during execution</li></ol>

---

© 2007 *Workflow Patterns Initiative*

0025697

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 8 (Environment Data)

[FLASH animation of Environment Data pattern](#)

### Description

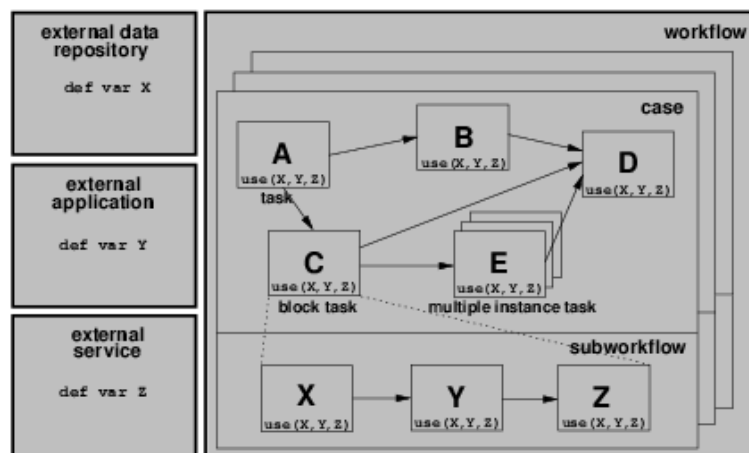
Data elements which exist in the external operating environment are able to be accessed by components of the workflow during execution.

### Example

Where required, tasks in the *System Monitoring* workflow can access the *temperature sensor data* from the operating environment.

### Motivation

Direct access to environmentally managed data by workflow tasks or cases during execution can significantly simplify workflow processes and improve their ability to respond to changes in the broader operational context. External data may be sourced from a variety of distinct locations including external databases, applications that are currently executing or can be initiated in the operating environment and services that mediate access to various data repositories and distribution mechanisms e.g stock price feeds. These scenarios are illustrated in Figure 9.



**Figure 9:** Environment data visibility

### Implementation

The ability to access external data elements generally requires the ability to connect to an interface or interprocess communication (IPC) facility in the operating environment or to invoke an external service which will supply data elements. Facilities for achieving this may be either *explicitly* or *implicitly* supported by the workflow engine. *Explicit support* involves the direct provision by the workflow



engine of constructs for accessing external data sources. Typically these take the form of specific elements that can be included in the design-time workflow model. Staffware provides the Automatic Step construct as well as script commands which enable specific items of data to be requested from an external applications. COSA provides the Tool Agent interfaces which provide a number of facilities for accessing external data elements. FLOWer implements Mapping Objects which allow data elements to be copied from external databases into the workflow engine, updated as required with case data and copied back to the underlying database. It also allows text files to be utilised in workflow actions and has a series of interfaces for external database integration. BPEL4WS provide facilities that enable external web services to be invoked. *Implicit support* occurs in workflow engines such as Websphere MQ Workflow where the actual implementation of individual workflow tasks is achieved by the development of associated programs in a procedural language such as C++ or Java. In this situation, access to external data occurs within individual task implementations by extending the program code to incorporate the required integration capabilities.

## Issues

There are a multitude of ways in which external data elements can be utilised within a workflow system. It is infeasible for any workflow tool to support more than a handful of them. This raises the issue of the minimal set of external integration facilities required for effective external data integration.

## Solutions

There is no definitive answer to this problem as the set of facilities required depends on the context in which the tool will be utilised. For the purposes of this research however, we consider it sufficient if a tool can demonstrate the ability to access data files (in text or binary format) in the operating environment and is able to access an external API (e.g. an XML, DDE or ODBC interface) through which data requests can be dynamically framed.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Supported through script functions and Automatic Steps
Websphere MQ Workflow	3.4	+/-	Indirectly accessible via program implementations
FLOWer	3.0	+	Supported via mappings, document types and actions
COSA	4.2	+	External data elements can be accessed via DDE, ODBC, XML, OS tool agents
XPDL	1.0	-	No explicit support
BPEL4WS	1.1	+	Accessing to environment data via web service calls

BPMN	1.0	-	Not supported
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Synchronous message interaction <invoke>, <receive>

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Workflow tool provides direct support for accessing external data elements both from files and via external applications or APIs</li><li>2. No limitations on source or format of data elements that can be accessed</li></ol>	<ol style="list-style-type: none"><li>1. Access to external data elements can be achieved through programmatic extensions</li></ol>

© 2007 *Workflow Patterns Initiative*

0025697

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 9 (Task to Task)

[FLASH animation of Data Interaction - Task to Task - Distinct Control and Data Channels pattern](#)

[FLASH animation of Data Interaction - Task to Task - Integrated Control and Data Channels pattern](#)

[FLASH animation of Data Interaction - Task to Task - Global Data Store pattern](#)

### Description

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

### Example

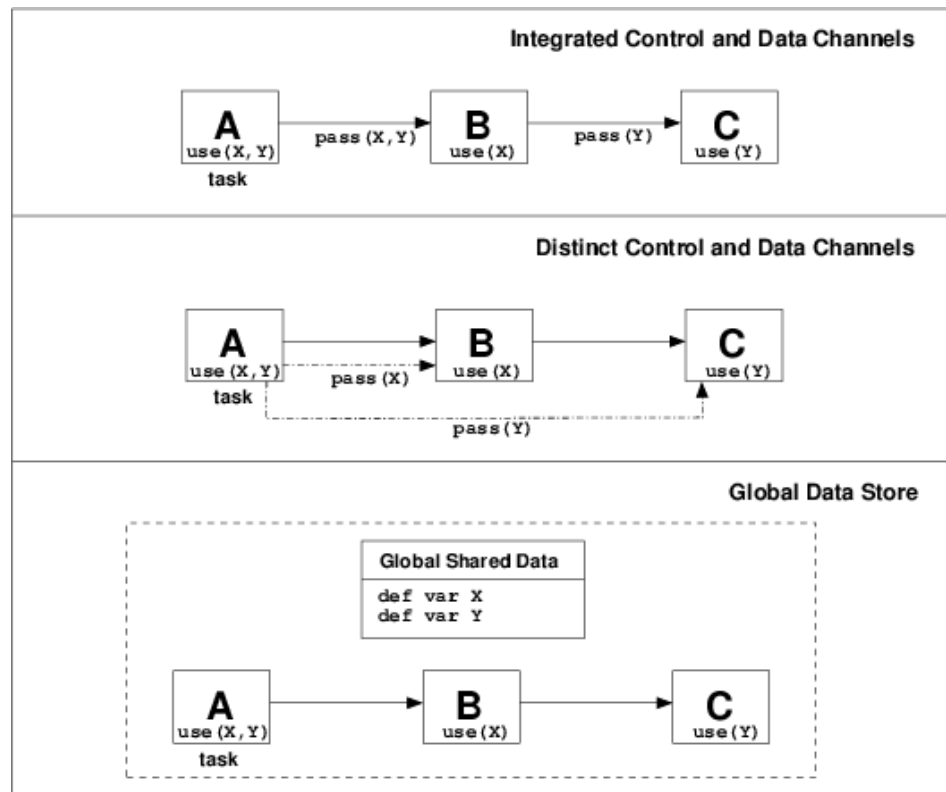
The *Determine Fuel Consumption* task requires the coordinates determined by the *Identify Shortest Route* task before it can proceed.

### Motivation

The passing of data elements between tasks is a fundamental aspect of workflow systems. In many situations, individual tasks execute in their own distinct address space and do not share significant amounts of data on a global basis. This necessitates the ability to move commonly used data elements between distinct tasks as required.

### Implementation

All workflow engines support the notion of passing parameters from one task to another however, this may occur in a number of distinct ways depending on the relationship between the data perspective and control flow perspective within the workflow. There are three main approaches as illustrated in Figure [10](#).



**Figure 10:** Approaches to data interaction between tasks

The distinctions between each of these are as follows:

- *Integrated control and data channels* - where both control flow and data are passed simultaneously between tasks utilising the same channel. In the example, task B receives the data elements X and Y at exactly the same time that control is passed to it. Whilst conceptually simple, one of the disadvantages of this approach to data passing is that it requires all data elements that may be used some time later in the workflow process to be passed with the thread of control regardless of whether the next task will use them or not. E.g. task B does not use data element Y but it is passed to it because task C will subsequently require access to it.
- *Distinct data channels* - in which data is passed between workflow tasks via explicit data channels which are distinct from the process control links within the workflow design. Under this approach, the coordination of data and control passing is usually not specifically identified. It is generally assumed that when control is passed to a task that has incoming data channels, the data elements specified on these channels will be available at the time of task commencement.
- *Global data store* - where tasks share the same data elements (typically via access to globally shared data) and no explicit data passing is required (cf. Patterns 6 and 7). This approach to data sharing is based on tasks having shared *a priori* knowledge of the naming and location of common data elements. It also assumes that the implementation is able to deal with potential concurrency issues that may arise where several task instances seek to access the same data element.

The majority of offerings examined adopt the third strategy. Staffware, FLOWer, COSA and XPDL all facilitate the passing of data through case-level data repositories accessible by all tasks. BPEL4WS utilises a combination of the first and third approaches. Variables can be bound to scopes within a process definition which may encompass a number of tasks, but there is also the ability for messages to be passed

between tasks when control passes from one task to another. Websphere MQ Workflow adopts the second mechanism with data elements being passed between tasks in the form of data containers via distinct data channels.

## Issues

1. Where there is no data passing between tasks and a common data store is utilised by several tasks for communicating data elements, there is the potential for concurrency problems to arise, particularly if the case involves parallel execution paths. This may lead to inconsistent results depending on the task execution sequence that is taken.
2. In the situation where data and control flow are passed along the same channel, there is the potential for two (potentially differing) copies of the same data element to flow to the same task (footnote [4](#)), necessitating that the task decide which is the correct version of the data element to use in its processing activities.

## Solutions

1. Concurrency control is handled in a variety of different ways by the tools examined. FLOWer avoids the problem by only allowing one active user or process that can update data elements in a case at any time (although other processes and users can access data elements for reading). BPEL4WS supports serialisable scopes which allow compensation handlers to be defined for groups of tasks that access the same data elements. A compensation handler is a procedure that aims to undo or compensate for the effects of the failure of a task on other tasks that may rely on it or on data that it has affected. Staffware provides the option to utilise an external transaction manager (Tuxedo) within the context of the workflow cases that it facilitates.
2. There is no direct solution to this problem identified in BPEL4WS.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Data interaction between tasks is based on use of common block-level data fields
Websphere MQ Workflow	3.4	+	Facilitated via data containers passed between tasks on data channels
FLOWer	3.0	+	Supported by shared data elements
COSA	4.2	+	Supported by data sharing (implicit data passing)
XPDL	1.0	+	All tasks refer to the same data
BPEL4WS	1.1	+	Data elements can be passed between activities in messages or via shared variables
BPMN	1.0	+	i) Through integrated control and data channels. Supported through the notion of Data Objects associated to Sequence Flows.

			ii) Through distinct control and data channels. Supported through the notion of Data Objects with its Properties attribute. iii) Through global shared data. Global shared data is supported through the attribute Properties of a Process.
UML	2.0	+	Directly supported by the ObjectNode construct.
Oracle BPEL	10.1.2	+	No data passing; data elements shared between tasks via access to globally shared data

## Summary of Evaluation

+ Rating	+/- Rating
1. Data elements can be directly passed from one task to another	1. Data Interaction involves some form of intermediation or can only occur via shared data repositories

© 2007 *Workflow Patterns Initiative*

0025698

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))



# Workflow Patterns



## Pattern 10 (Block Task to Sub-Workflow Decomposition)

[FLASH animation of Data Interaction - Block Task to Sub-Workflow Decomposition - Explicit Data Passing via Data Channels pattern](#)

[FLASH animation of Data Interaction - Block Task to Sub-Workflow Decomposition - Explicit Data Passing via Parameters pattern](#)

[FLASH animation of Data Interaction - Block Task to Sub-Workflow Decomposition - Implicit Data Passing pattern](#)

### Description

The ability to pass data elements from a block task instance to the corresponding sub-workflow that defines its implementation.

### Example

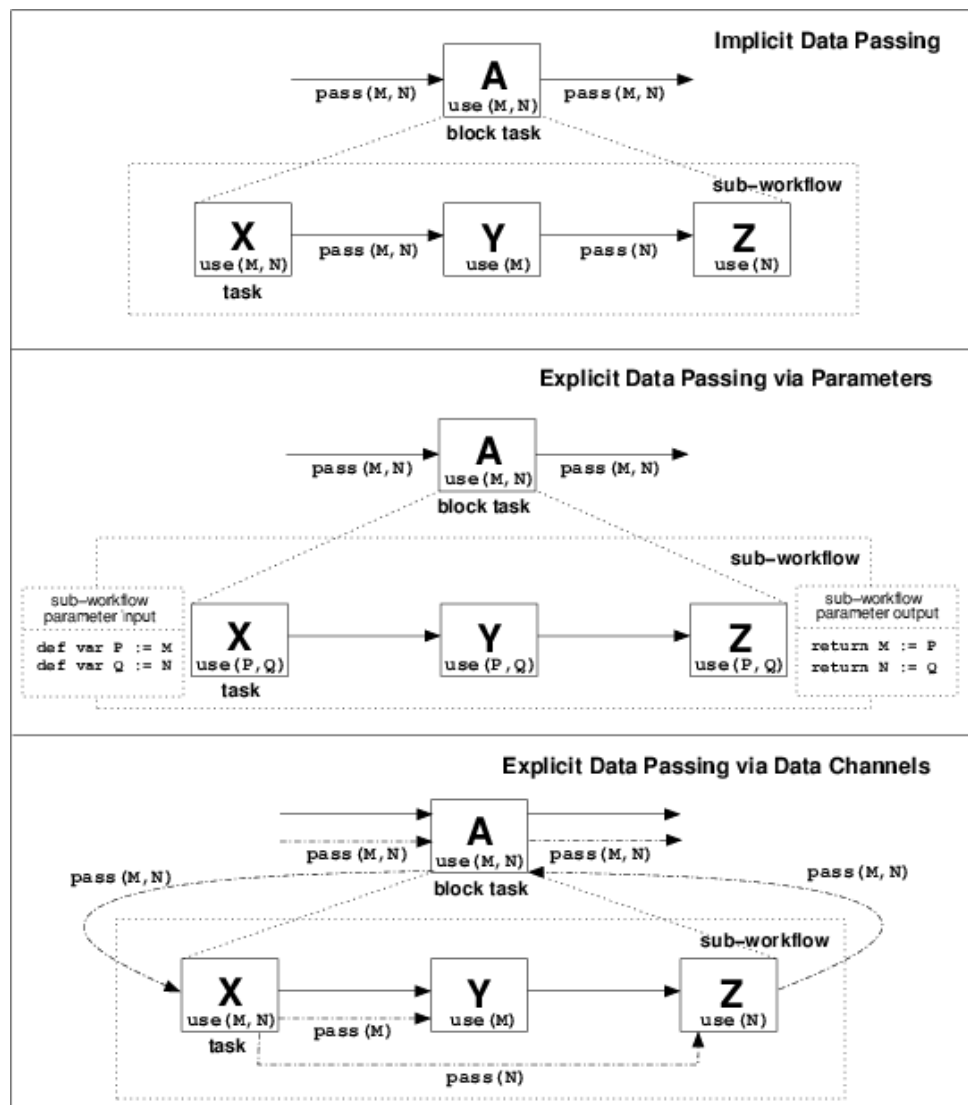
*Customer claims* data is passed to the *Calculate Likely Tax Return* block task whose implementation is defined by a series of tasks in the form of a sub-workflow. The *customer claims* data is passed to the sub-workflow and is visible to each of these sub-tasks.

### Motivation

Most workflow systems support the notion of composite or block tasks in some form. These are analogous to the programmatic notion of procedure calls and indicate a task whose implementation is described in further detail at another level of abstraction (typically elsewhere in the workflow design) utilising using the same range of workflow constructs. The question that arises when data is passed to a block element is whether it is immediately accessible by all of the tasks that define its actual implementation or if some form of explicit data passing must occur between the block task and the sub-workflow.

### Implementation

Typically one of three approaches is taken to handling the communication of parameters from a block task to the underlying implementation. Each of these is illustrated in Figure [11](#). The characteristics of each approach are as follows:



**Figure 11:** Approaches to data interaction from block tasks to corresponding sub-workflows

- *Implicit data passing* - data passed to the block task is immediately accessible to all sub-tasks which make up the underlying implementation. In effect the main block task and the corresponding sub-workflow share the same address space and no explicit data passing is necessary.
- *Explicit data passing via parameters* - data elements supplied to the block task must be specifically passed as parameters to the underlying sub-workflow implementation. The second example in Figure 11 illustrates how the specification of parameters can handle the mapping of data elements at block task level to data elements at sub-workflow level with distinct names. This capability provides a degree of independence in the naming of data elements between the block task and sub-workflow implementation and is particularly important in the situation where several block tasks share the same sub-workflow implementation.
- *Explicit data passing via data channels* - data elements supplied to the block task are specifically passed via data channels to all tasks in the sub-workflow that require access to them.

It is important to note that the first approach does not involve any actual data passing between block activity and implementation, rather the block level data elements are made accessible to the sub-workflow. This strategy is adopted by FLOWer and COSA

for data passing to sub-workflows. In both cases, the sub-workflow is presented with the entire range of data elements utilised by the block task and there is no opportunity to limit the scope of items passed. The third approach relies on the passing of data elements between tasks to communicate between block task and sub-workflow. Both Staffware and XPDL utilise this mechanism for passing data elements to sub-workflows. In contrast, the second approach necessitates the creation of new data elements at sub-workflow level to accept the incoming data values from the block activity. Websphere MQ Workflow follows this strategy and sink and source nodes are used to pass data containers between the parent task and corresponding sub-workflow.

## Issues

One consideration that may arise where the explicit parameter passing approach is adopted is whether the data elements in the block task and the sub-workflow are independent of each other (i.e. whether they exist in independent address spaces). If they do not, then the potential exists for concurrency issues to arise as tasks executing in parallel update the same data element.

## Solutions

This issue can arise with Staffware where sub-workflow data elements are passed as fields rather than parameters. The resolution to this issue is to map the input fields to fields with distinct names not used elsewhere during execution of the case.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Indirectly supported via field mappings
Websphere MQ Workflow	3.4	+	Facilitated via data containers and source/sink nodes
FLOWer	3.0	+/-	Supported via implicit data passing between shared data elements but no ability to restrict the range of data elements passed
COSA	4.2	+/-	Supported by data sharing (implicit data passing) but no ability to limit the range of items passed
XPDL	1.0	+	The element "subflow" provides actual parameters which should be matched by the formal parameters of the corresponding workflow process. This is not defined out in detail, but the intention is clear.
BPEL4WS	1.1	-	Not supported
BPMN	1.0	+	i) Implicit data passing through global shared data is applied when a decomposition is realised via an Embedded Sub-Process. ii) Explicit data passing via

			parameters is applied when a decomposition is realised through an Independent Sub-Process. In such a case the data-transfer is defined through the Input - and OutputPropertyMaps Expression for the Sub-Process
UML	2.0	+	Supported by parameter passing between the block task and the decomposition
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Data elements available to a block task are able to be passed to or are accessible in the associated sub-workflow</li><li>2. There is some degree of control over which elements at block task level are made accessible in the sub-workflow</li></ol>	<ol style="list-style-type: none"><li>1. The range of elements accessible within the sub-workflow is more limited than those available in the block task</li></ol>

© 2007 *Workflow Patterns Initiative*

0025698

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 11 (Sub-Workflow Decomposition to Block Task)

[FLASH animation of Data Interaction - Sub-Workflow Decomposition to Block Task pattern](#)

### Description

The ability to pass data elements from the underlying sub-workflow back to the corresponding block task instance.

### Example

The *Determine Optimal Trajectory* sub-workflow passes the coordinates of the *launch* and *target locations* and *flight plan* back to the block task.

### Motivation

At the conclusion of the underlying sub-workflow, the data elements which have resulted from its execution need to be made available to the block activity which called it.

### Implementation

The approaches taken to handling this Pattern are essentially the same as those identified for Pattern 10. Where data elements are passed on an explicit basis, a mapping needs to be specified for each output parameter indicating which data element at block level will receive the relevant output value.

### Issues

One difficulty that can arise with the implementation of this Pattern occurs when there is not a strict correspondence between the data elements returned from the sub-workflow and the receiving data elements at block task level. E.g. the sub-workflow returns more data elements than the block task is expecting, possibly as a result of additional data elements being created during its execution.

### Solutions

This problem can be solved in one of two ways. Some workflow engines such as Staffware support the ability for block tasks to create data elements at block task level for those data items at sub-workflow level which it has not previously encountered. Other products such as Websphere MQ Workflow require a strict mapping to be defined between sub-workflow and block task data elements to prevent this situation from arising.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Indirectly supported via field mappings
Websphere MQ Workflow	3.4	+	Facilitated via data containers and source/sink nodes
FLOWer	3.0	+/-	As for Pattern 10
COSA	4.2	+/-	As for Pattern 10
XPDL	1.0	+	As for Pattern 10
BPEL4WS	1.1	-	Not supported
BPMN	1.0	+	i) Implicit data passing through global shared data is applied when a decomposition is realised via an Embedded Sub-Process. ii) Explicit data passing via parameters is applied when a decomposition is realised through an Independent Sub-Process. In such a case the data-transfer is defined through the Input - and OutputPropertyMaps Expression for the Sub-Process
UML	2.0	+	Directly supported by the ExpansionRegion construct (where the ExpansionKind mode is set to parallel).
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Data elements at sub-workflow level can be passed to or made accessible in the corresponding block task	1. There are limited facilities for controlling which sub-workflow data elements can be made available to the block task

© 2007 *Workflow Patterns Initiative*

0025698

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 12 (To Multiple Instance Task)

[FLASH animation of Data Interaction - to Multiple Instance Task - Shared Data Passed by Reference pattern](#)

[FLASH animation of Data Interaction - to Multiple Instance Task - Instance Specific Data Passed by Value pattern](#)

[FLASH animation of Data Interaction - to Multiple Instance Task - Instance Specific Data Passed by Reference pattern](#)

### Description

The ability to pass data elements from a preceding task instance to a subsequent task which is able to support multiple execution instances. This may involve passing the data elements to all instances of the multiple instance task or distributing them on a selective basis.

### Examples

- The *Identify Witnesses* task passes the *witness list* to the *Interview Witnesses* task. This data is available to each instance of the *Interview Witnesses* task at commencement.
- The *New Albums List* is passed to the *Review Album* task and one task instance is started for each entry on the list. At commencement, each of the *Review Album* task instances is allocated a distinct entry from the *New Albums List* to review.

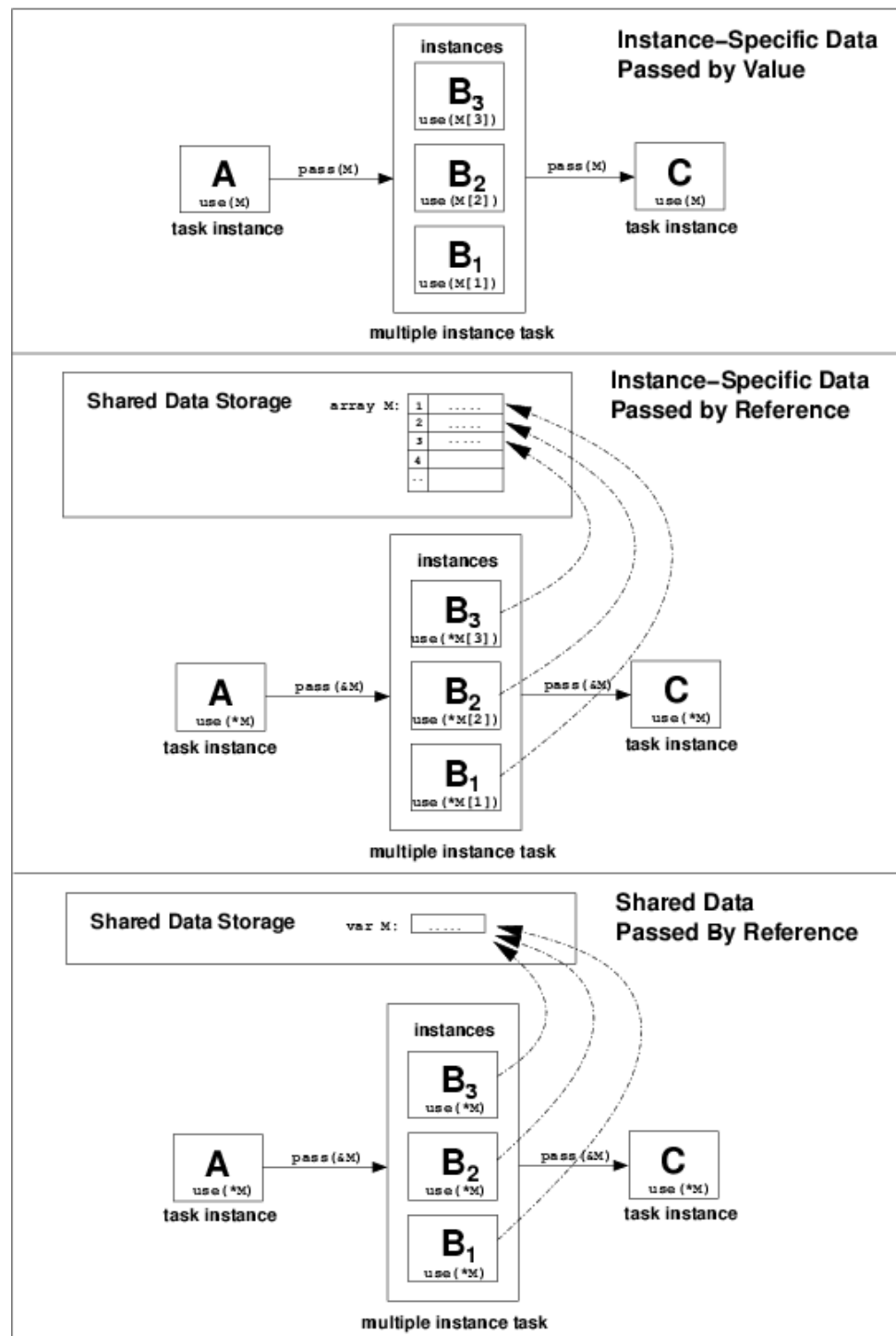
### Motivation

Where a task is capable of being invoked multiple times, a means is required of controlling which data elements are passed to each of the execution instances. This may involve ensuring that each task instance receives all of the data elements passed to it (possibly on a shared basis) or distributing the data elements across each of the execution instances on some predefined basis.

### Implementation

There are three potential approaches to passing data elements to multiple instance tasks as illustrated in Figure [12](#).





**Figure 12:** Data interaction approaches for multiple instance tasks

As a general rule, it is possible either to pass a data element to all task instances or to distribute one item from it (assuming it is a composite data element such as an array or a set) to each task instance. Indeed the number of task instances that are initiated may be based on the number of individual items in the composite data element. The specifics of each of these approaches are discussed below.

- *Instance-specific data passed by value* - this involves the distribution of a data element passed by value to task instances on the basis of one item of the data element per task instance (in the example shown, task instance receives M[1], receives M[2] and so on). As the data element is passed by value, each task

instance receives a copy of the item passed to it in its own address space. At the conclusion of each of the task instances, the data element is reassembled from the distributed items and passed to the subsequent task instance.

- *Instance-specific data passed by reference* - this scenario is similar to that described above except that the task instances are passed a reference to a specific item in the data element rather than the value of the item. This approach obviates the need to reassemble the data element at the conclusion of the task instances.
- *Shared data passed by reference* - in this situation all task instances are passed a reference to the same data element. Whilst this allows all task instances to access the same data element, it does not address the issue of concurrency control should one of the task instances amend the value of the data element (or indeed if it is altered by some other component of the workflow).

FLOWer provides facilities for instance-specific data to be passed by reference whereby an array can be passed to a designated multiple instance task and specific sub-components of it can be mapped to individual task instances. It also allows for shared data elements to be passed by reference to all task instances.

## Issues

Where a task is able to execute multiple times but not all instances are created at the same point, an issue that arises is whether the values of data elements are set for all execution instances at the time at which the multiple instance task is initiated or whether they can be fixed after this occurs but prior to the actual invocation of the task instance to which they relate.

## Solutions

In FLOWer, the Dynamic Plan construct allows the data for individual task instances to be specified at any time prior to the actual invocation of the task. The passing of data elements to specific task instances is handled via Mapping Array data structures. These can be extended at any time during the execution of a Dynamic Plan, allowing for new task instances to be created "on the fly" and the data corresponding to them to be specified at the latest possible time.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Multiple instance tasks not supported
Websphere MQ Workflow	3.4	-	No support for multiple instances
FLOWer	3.0	+	Each instance of a dynamic plan can have discrete data elements which can be passed to/from it
COSA	4.2	-	Not supported
XPDL	1.0	-	No support for multiple instances
BPEL4WS	1.1	-	Not supported

BPMN	1.0	-	Not supported
UML	2.0	+	Directly supported by the ExpansionRegion construct (where the ExpansionKind mode is set to parallel).
Oracle BPEL	10.1.2	+/-	Not supported by all variants of MI tasks

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Multiple instance tasks directly supported</li><li>2. Data elements can be passed from an atomic task to all instances of a multiple instance task</li><li>3. Workflow handles synchronization of data passing and any necessary data replication</li><li>4. Facilities are available to allocate sections of an aggregate data element to specific task instances</li><li>5. Data elements in each task instance are independent of those in other task instances</li></ol>	<ol style="list-style-type: none"><li>1. Multiple instances not directly supported</li><li>2. Programmatic intervention required to facilitate data passing</li><li>3. No facilities for refreshing/resetting data elements for re-entrant task instances</li></ol>

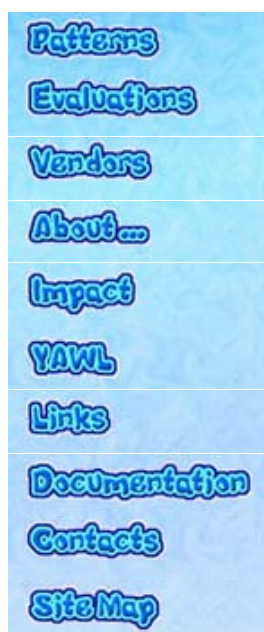
© 2007 *Workflow Patterns Initiative*

0025698

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 13 (From Multiple Instance Task)

[FLASH animation of Data Interaction - from Multiple Instance Task pattern](#)

### Description

The ability to pass data elements from a task which supports multiple execution instances to a subsequent task.

### Example

At the conclusion of the various instances of the *Record Lap Time* task, the *list of lap times* is passed to the *Prepare Grid* task.

### Motivation

Each execution instance of a multiple instance task effectively operates independently from other instances and as such, has a requirement to pass on data elements at its conclusion to subsequent tasks.

### Implementation

In general, data is passed from a multiple instance task to subsequent tasks when a certain number of the task instances have concluded. The various scenarios in which this may occur are illustrated in Figure [12](#). These usually coincide with the passing of control from the multiple instance task although data and control flow do not necessarily need to be fully integrated. In the case where data elements are passed by reference, the location rather than the values are passed on at task completion (obviously this implies that the data values may be accessed by other components of the workflow prior to completion of the multiple instance task as they reside in shared storage).

### Issues

One issue that arises with multiple instance tasks is the point at which the output data elements from them are available (in some aggregate form) to subsequent tasks.

### Solutions

In the case of FLOWer, this issue is dependent on where the data elements are accessed from. The FLOWer engine allows parallel task instances to access output data elements from other instances even if these instances have not yet completed (i.e. the values obtained, if any, may not be final). However subsequent task instances (i.e. those after the multiple instance task), can rely on the complete composite data

element being available.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Multiple instance tasks not supported
Websphere MQ Workflow	3.4	-	No support for multiple instances
FLOWer	3.0	+	As for Pattern 12
COSA	4.2	-	Not supported
XPDL	1.0	-	No support for multiple instances
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported
UML	2.0	+	Directly supported by the ExpansionRegion construct (where the ExpansionKind mode is set to parallel).
Oracle BPEL	10.1.2	+/-	For non-directly supported MI task, a certain number of instances should be complete before the data is passed to the next task

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Multiple instance tasks directly supported</li><li>2. Data elements can be aggregated from multiple task instances and forwarded to subsequent task instance(s)</li><li>3. Workflow handles synchronization of data passing and any necessary data replication</li></ol>	<ol style="list-style-type: none"><li>1. Multiple instances not directly supported</li><li>2. Programmatic intervention required to facilitate data passing</li></ol>

© 2007 *Workflow Patterns Initiative*

0025698

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 14 (Case to Case)

[FLASH animation of Data Interaction - Case to Case pattern](#)

### Description

The passing of data elements from one case of a workflow during its execution to another case that is executing concurrently.

### Example

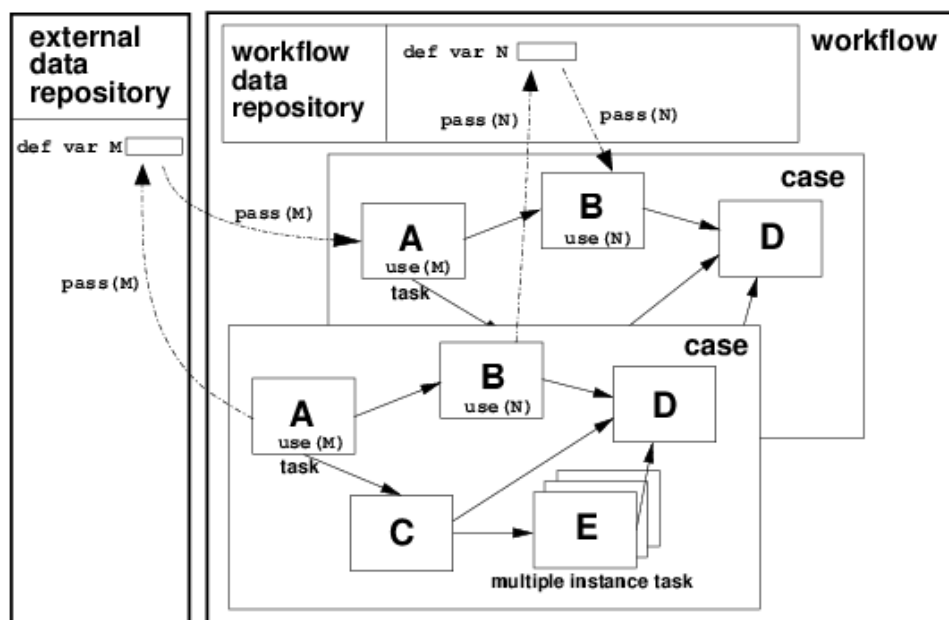
During execution of a case of the *Re-balance Portfolio* workflow the *best price* identified for each security is passed to subsequent cases.

### Motivation

Where the work completed and the results obtained during the course of one workflow case are likely to be of use to subsequent cases, a means of communicating them to both currently executing and subsequent cases is required.

### Implementation

Direct support for this function requires the availability of a function within the workflow tool that supports a given case initiating the communication (and possibly updating) of data elements with a nominated workflow case. Alternatively, it is possible to achieve the same outcome indirectly by writing them back to a shared store of persistent data known to both the initiating and receiving cases. This may be either an internally maintained facility at workflow level or an external data repository (footnote [5](#)). Both of these scenarios are illustrated in Figure [13](#).



**Figure 13:** Data interaction between workflow cases

Each of these approaches requires the communicating cases to have *a priori* knowledge of the location of the data element that is to be used for data passing. They also require the availability of a solution to address the potential concurrency issues that may arise where multiple workflow cases wish to communicate with each other. COSA was the only one of the offerings examined that supported a direct method for communicating data elements between workflow cases. It achieves this using shared "folder" data elements (c.f. Pattern 4). In the case of the other offerings, it is possible to achieve the same result indirectly for each of them using the methods described above.

## Issues

The main consideration associated with this Pattern is in establishing an effective means of linking related workflow cases and their associated data together in a meaningful way.

## Solutions

None of the workflow engines examined address this need, however there are offerings that provide solutions to the issue. In Vectus (footnote [6](#)) it is possible to relate cases. There are four types of relationships "parent", "child", "sibling", and "master". These relationships can be used to navigate through cases at runtime. For example, one case can schedule tasks in another flow, terminate a task in another flow, create new cases, etc. It is also possible to share data using a dot notation similar to Object Constraint Language [\[OMG03\]](#). The "master" relation can be used to create a proxy shared among related cases to show all tasks related to these cases. A similar construct is also present in the BizAgi (footnote [7](#)) product.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.

- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Achievable via script functions
Websphere MQ Workflow	3.4	+/-	Indirectly achievable by including case communication facilities in program implementations
FLOWer	3.0	+/-	Indirectly supported via external database
COSA	4.2	+	Supported via folder data
XPDL	1.0	+/-	Indirectly achievable via coordinated web services operations in sending and receiving cases
BPEL4WS	1.1	+/-	Indirectly achievable using the invoke construct
BPMN	1.0	-	Not supported
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Data elements can be directly passed between workflow cases	1. Interaction occurs via an external data repository 2. Not directly achievable or programmatic interaction required

© 2007 *Workflow Patterns Initiative*

0025698

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)



# Workflow Patterns



## Pattern 15 (Task to Environment - Push)

[FLASH animation of Data Interaction - Task to Environment - Push-Oriented pattern](#)

### Description

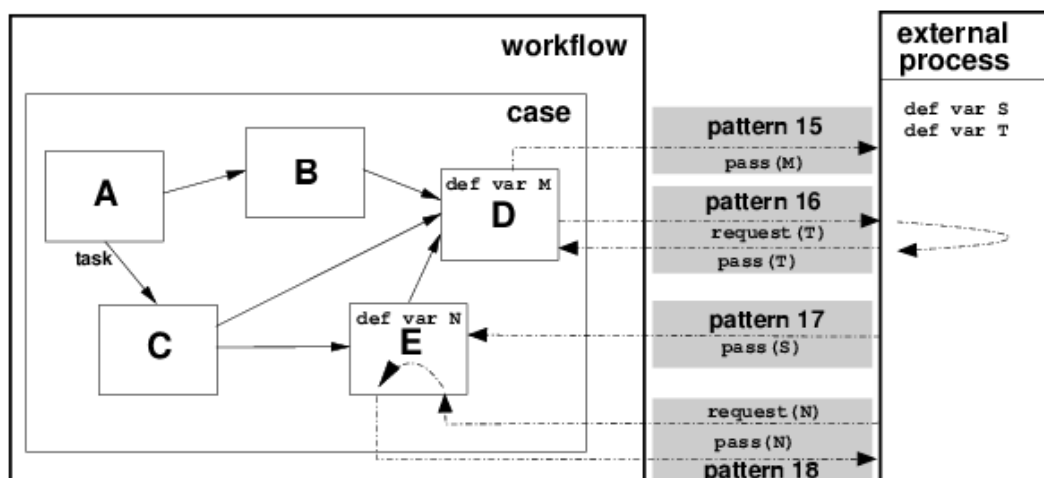
The ability of a task to initiate the passing of data elements to a resource or service in the operating environment.

### Example

The *Calculate Mileage* task passes the *mileage data* to the *Corporate Logbook* database for permanent recording.

### Motivation

The passing of data from a workflow task to an external resource is most likely to be initiated by the task itself during its execution. Depending on the specific requirements of the data passing interaction, it may connect to an existing API or service interface in the operating environment or it may actually invoke the application or service to which it is forwarding the data. Figure 14 illustrates the various data passing scenarios between workflow tasks and the operating environment.



**Figure 14:** Data interaction between workflow tasks and the operating environment

### Implementation

There are a wide range of ways in which this Pattern is achieved in workflow engines, however these tend to divide into two categories:

- *Explicit integration mechanisms* - where the workflow system provides specific constructs for passing data to the external environment.
- *Implicit integration mechanisms* - where the data passing occurs implicitly within the programmatic implementations that make up tasks in the workflow process and is not directly supported by the workflow environment.

In both cases, the data passing activity is initiated by the task and occurs synchronously with task execution. Although, it is typically task-level data elements that are trafficked, any data items that are accessible to the task (i.e. parameters, case and workflow data) may be transferred. Staffware provides the Automatic Step construct for passing data elements from task instances to external programs. FLOWer enables data to be passed to external applications using the Operation Action construct and to external databases using Mapping Objects. COSA provides a broad number of Tool Agents which facilitate the passing of data to external applications and databases. Websphere MQ Workflow is more limited in its support and delegates the passing of data elements to user-defined Program implementations. XPDL and BPEL4WS are limited to passing data elements to web services.

## Issues

One difficulty that can arise when sending data to an external application is knowing whether it was successfully delivered. This is particularly a problem when the external application does not provide immediate feedback on delivery status.

## Solutions

The most general solution to this problem is for a subsequent task in the case to request an update on the status of the delivery using a construct which conforms to Pattern 16 and requires an answer to be provided by the external application. Alternatively, the external application can lodge a notification of the delivery using some form of event (i.e. Pattern 38) or by passing data back to the workflow (i.e. Pattern 17). This is analogous to the messaging notion of asynchronous callback in a workflow context.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Directly supported via Automatic Steps and script functions
Websphere MQ Workflow	3.4	+/-	Indirectly achievable by including communication facilities in program implementations
FLOWer	3.0	+	Supported via the Action Operation and Mapping Object (using Out/Out&Insert type) constructs
COSA	4.2	+	Supported through tool agents e.g. ODBC

XPDL	1.0	+	Data elements can be passed to other web services
BPEL4WS	1.1	+	The invoke construct supports the passing of data elements to other web services on an asynchronous basis
BPMN	1.0	+	Captured through a message flow flowing from a Task to the boundry of a Pool representing the Environment
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Directly supported by means <i>inputVariable</i> of <invoke>

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support for task-initiated data passing to external applications	1. Limitations on the type of data elements that can be passed 2. Achievable via programmatic extensions

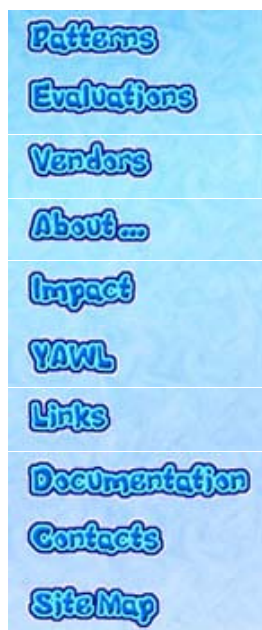
© 2007 *Workflow Patterns Initiative*

0025699

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 16 (Environment to Task - Pull)

[FLASH animation of Data Interaction - Environment to Task - Pull-Oriented pattern](#)

### Description

The ability of a workflow task to request data elements from resources or services in the operational environment.

### Example

The *Determine Cost* task must request *cattle price* data from the *Cattle Market System* before it can proceed.

### Motivation

Workflow tasks require the means to proactively seek the latest information from known data sources in the operating environment during their execution. This may involve accessing the data from a known repository or invoking an external service in order to gain access to the required data elements.

### Implementation

Similar to Pattern 15, distinct workflow engines support this Pattern in a variety of ways however these approaches divide into two categories:

- *Explicit integration mechanisms* - where the workflow system provides specific constructs for accessing data in the external environment.
- *Implicit integration mechanisms* - where access to external data occurs at the level of the programmatic implementations that make up tasks in the workflow process and is not directly supported by the workflow engine. Interaction with external data sources typically utilises interprocess communication (IPC) facilities provided by the operating system facilities such as message queues or remote procedure calls, or enterprise application integration (EAI) mechanisms such as DCOM (footnote [8](#)), CORBA (footnote [9](#)) or JMS (footnote [10](#)).

Staffware provides two distinct constructs that support this objective. Automatic Steps allow external systems to be called (e.g. databases or enterprise applications) and specific data items to be requested. Scripts allow external programs to be called either directly at system level or via system interfaces such as DDE (footnote [11](#)) to access required data elements. FLOWer utilises Mapping Objects to extract data elements from external databases. COSA has a number of Tool Agent facilities for requesting data from external applications. XPDL and BPEL4WS provide facilities for the synchronous request of data from other web services. In contrast, Websphere MQ Workflow does not provide any facilities for external integration and requires the underlying programs that implement workflow tasks to provide these capabilities

where they are required.

## Issues

One difficulty with this style of interaction is that it can block progress of the requesting case if the external application has a long delivery time for the required information or is temporarily unavailable.

## Solutions

The only potential solution to this problem is for the requesting case not to wait for the requested data (or continue execution after a nominated timeout) and to implement some form of asynchronous notification of the required information (possibly along the lines of Pattern 17). The disadvantage of this approach is that it complicates the overall interaction by requiring the external application to return the required information via an alternate path and necessitating the workflow to provide notification facilities.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Directly supported via Automatic Steps and script functions
Websphere MQ Workflow	3.4	+/-	Indirectly achievable by including communication facilities in program implementations
FLOWer	3.0	+	Supported via the mapping construct using In type
COSA	4.2	+	Supported through tool agents e.g. ODBC
XPDL	1.0	+	Data can be synchronously requested from other web services
BPEL4WS	1.1	+	The invoke construct also supports the request of data elements from other web services on a synchronous basis
BPMN	1.0	+	Captured through a message flow flowing from a Task to the boundary of a Pool representing the Environment, followed by a message flow from the Pool (representing the Environment) back to the task
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Direct support via <invoke> and <receive>

## Summary of Evaluation

+ <b>Rating</b>	+/- <b>Rating</b>
1. Direct workflow support for task instances to initiate requests for data and receive replies from external applications	1. Achievable via programmatic extensions

---

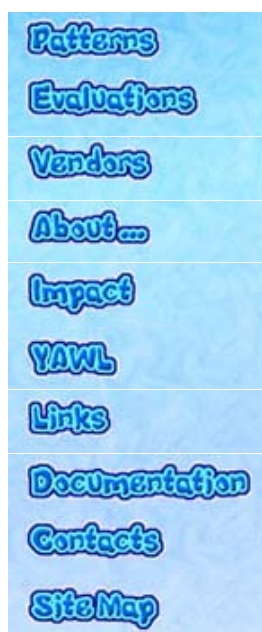
© 2007 *Workflow Patterns Initiative*

0025699

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 17 (Environment to Task - Push)

[FLASH animation of Data Interaction - Environment to Task - Push-Oriented pattern](#)

### Description

The ability for a workflow task to receive and utilise data elements passed to it from services and resources in the operating environment on an unscheduled basis.

### Example

During execution, the *Review Performance* task may receive new *engine telemetry* data from the *Wireless Car Sensors*.

### Motivation

An ongoing difficulty for workflow tasks is establishing mechanisms that enable them to be provided with new items of data as they become available from sources outside of the workflow system. This is particularly important in areas of volatile information where updates to existing data elements may occur frequently but not on a scheduled basis e.g. price updates for equities on the stock market. This Pattern relates to the ability of tasks to receive new items of data as they become available without needing to proactively request them from external sources or suspend execution until updates arrive.

### Implementation

As for Patterns 15 and 16, approaches to this Pattern can be divided into *explicit* and *implicit* mechanisms. The main difficulty that arises in its implementation is in providing external processes with the addressing information that enables the routing of data elements to a specific task instance. Potential solutions to this include the provision of externally accessible execution monitors by workflow engines which indicate the identity of currently executing tasks or task instances recording their identity with a shared registry service in the operating environment. An additional consideration is the ability of workflow tasks to be able to asynchronously wait for and respond to data passing activities without impacting the actual progress of the task. This necessitates the availability of asynchronous communication facilities at task level - either provided as some form of ( *explicit* ) task construct by the workflow engine or able to be ( *implicitly* ) included in the programmatic implementations of tasks. A number of the offerings examined support this Pattern in some form. COSA provides the ability to pass data elements to tasks via the trigger construct as well as via various APIs. BPEL4WS provides a similar capability with the receive construct and event handlers. FLOWer allows the value of data elements associated with task forms to be updated via the `chp_frm_setfield` API call. In Staffware, this Pattern can be indirectly achieved by using the Event Step construct which allows external processes to update field data during the execution of a workflow case. However, a

"dummy" task is required to service the query request.

## Issues

The major difficulty associated with this Pattern is in providing a means for external applications to identify the specific task instance in which they wish to update a data element.

## Solutions

In general the solution to this problem requires the external application to have knowledge of both the case instance and the specific task in which the data element resides. Details of currently executing cases can only be determined with reference to the workflow engine and most offerings provide a facility or API call to support this requirement. The external application will most likely require a priori knowledge of the identity of the task in which the data element resides.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Indirectly supported by Event Steps
Websphere MQ Workflow	3.4	+/-	Indirectly achievable by including communication facilities in program implementations
FLOWer	3.0	+/-	Values of data elements in forms associated with tasks can be updated via CHIP library API
COSA	4.2	+	Supported via command line and C/COM/Java APIs
XPDL	1.0	-	Not reported
BPEL4WS	1.1	+/-	Indirectly achievable via the receive construct or event handlers although the data update operation requires specific coding
BPMN	1.0	+	Captured through a message flow flowing from the boundary of a Pool representing the Environment to a Task
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Directly support via <receive> and event handlers

## Summary of Evaluation

+ Rating	+/- Rating



1. Direct workflow support for task instances to accept externally generated data during execution	1. Achievable via programmatic extensions
--	---

---

© 2007 *Workflow Patterns Initiative*

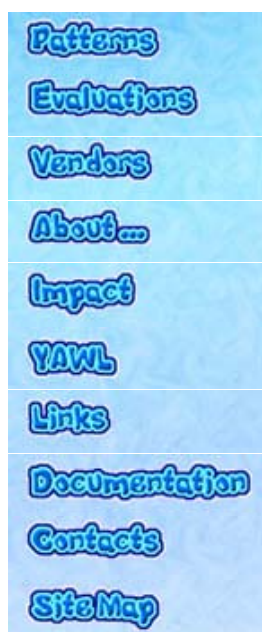
0025699

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) | [Impact](#)

[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 18 (Task to Environment - Pull)

[FLASH animation of Data Interaction - Task to Environment - Pull-Oriented pattern](#)

### Description

The ability of a workflow task to receive and respond to requests for data elements from services and resources in the operational environment.

### Example

During execution, the *Assess Quality* task may receive requests for current data from the *Quality Audit* web service handler. It must provide this service with details of all current data elements.

### Motivation

In some cases, the requests for access to task instance data are initiated by processes outside the workflow environment. These requests need to be handled as soon as they are received but should be processed in a manner that minimises any potential impact on the task instance from which they are sourcing data.

### Implementation

The ability to access data from a task instance can be handled in one of three ways:

1. The workflow engine can provide a means of accessing task instance data from the external environment.
2. During their execution, tasks instances publish data values to a well-known location e.g. a database that can be accessed externally.
3. Task instances incorporate facilities to service requests for their data from external processes.

In practice, this facility is not widely supported as an explicit construct by the offerings examined. BPEL4WS provides direct support for the Pattern (via the receive construct and event handlers). Staffware provides the EIS Report construct and EIS Case Data Extraction facilities to enable the values of data elements to be requested from workflow cases. COSA provides APIs for getting and setting the values of various types of data elements from the external environment (both at command line level and also programmatically). FLOWer provides an indirect solutions via the `chp_frm_getfield` API call although this method has limited generality.

### Issues

Similar difficulties exist with the utilisation of this Pattern to those identified for

Pattern 17.

## Solutions

As for Pattern 17.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Indirectly supported by EIS Reports and EIS Case Data extraction facility
Websphere MQ Workflow	3.4	+/-	Indirectly achievable by including communication facilities in program implementations
FLOWer	3.0	+/-	Values of data elements in forms associated with tasks can be queried via CHIP library API
COSA	4.2	+	Supported via command line and C/COM/Java APIs
XPDL	1.0	-	Not reported
BPEL4WS	1.1	+/-	Indirectly achievable via the receive/reply constructs or event handlers although the operation requires specific coding
BPMN	1.0	+	Captured through a message flow flowing from the boundary of a Pool representing the Environment to a Task followed by a message flow flowing from the same Task back to the Pool representing the Environment
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Directly supported through <receive> and <reply>

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support for tasks to respond to external requests for data	1. Achievable via programmatic extensions

© 2007 Workflow Patterns Initiative

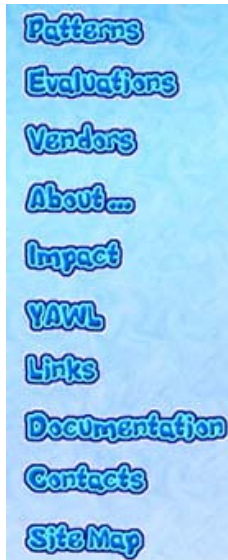
0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)

[Map](#)

# Workflow Patterns



## Pattern 19 (Case to Environment - Push)

[FLASH animation of Data Interaction - Case to Environment - Push-Oriented pattern](#)

### Description

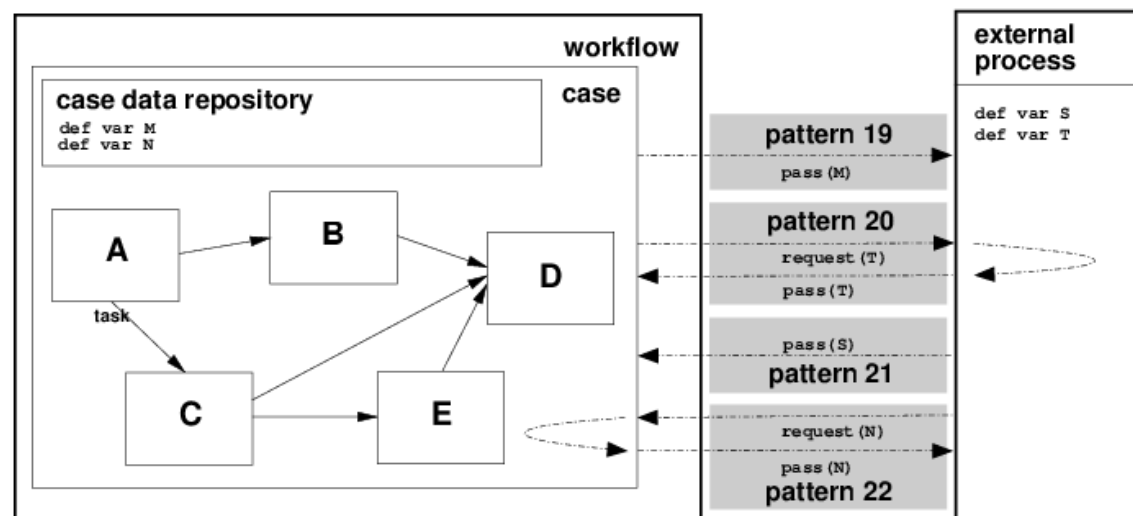
The ability of a workflow case to initiate the passing of data elements to a resource or service in the operational environment.

### Example

At its conclusion, each case of the *Perform Reconciliation* workflow passes its results to the *Corporate Audit* database for permanent storage.

### Motivation

An alternative (or possible extension) to task-level data passing is to provide facilities for passing data at the level of the workflow case. The various options for this approach to data passing are illustrated in Figure 15. This Pattern is analogous to Pattern 15 except that it operates in the context of a process instance.



**Figure 15:** Data interaction between cases and the operating environment

### Implementation

This Pattern has not been widely observed in the tools evaluated. Its implementation requires explicit support by the workflow engine for case level data passing which is independent of the task instances that comprise the case. Maximal flexibility for this Pattern is gained through the use of a rule-based approach to the triggering of the data passing action. Potential invocation criteria include state-based conditions (e.g. case initiation, case completion), data-based conditions (e.g. pass the data element when a nominated data condition is satisfied) and temporal conditions (e.g. emit the value of a data element periodically during case execution). FLOWer provides a mechanism for synchronizing case data elements with an external database through the use of mapping constructs which are triggered for each activity in a plan. This provides a mechanism for providing external visibility of data elements during the execution of a case.

## Issues

The main issue associated with this Pattern is determining an appropriate time to undertake the "push" action of the data element from the case to the environment.

## Solutions

There are many points during case execution at which the external communication of a data element does not seem to be sensible. In the case of FLOWer, this action occurs at the conclusion of a specific task (or tasks) within the case. It would also seem to be appropriate at the conclusion of a case.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+	Supported via the mapping construct using Out/Out&Insert type
COSA	4.2	-	Not supported
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported. Message flows can indeed be drawn between the boundaries of two pools, one representing the Process the Case instantiates, and other one representing the Environment. However, the semantics of this construct does not capture data exchange at any moment during the execution of the case
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support for case-initiated data passing to external applications at any time	1. Achievable via programmatic extensions

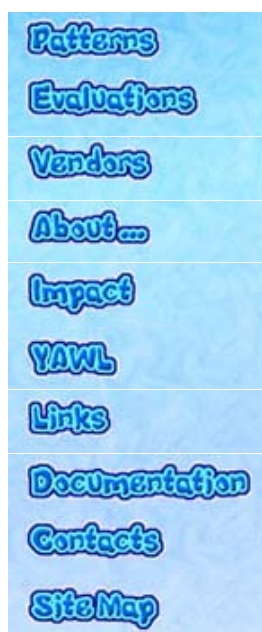
© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) | [Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)

# Workflow Patterns



## Pattern 20 (Environment to Case - Pull)

[FLASH animation of Data Interaction - Environment to Case - Pull-Oriented pattern](#)

### Description

The ability of a workflow case to request data from services or resources in the operational environment.

### Example

At any time cases of the *Process Suspect* workflow may request additional data from the *Police Records System*.

### Motivation

In the event that a workflow case requires access to the most current values of external data elements during execution, it may not be sufficient for these values to be specified at the initiation of the case. Instead, facilities may be required for them to be accessed during the course of execution at the point in the case where they are most likely to be needed.

### Implementation

Similar to Pattern 19, this Pattern has not been widely observed in the tools evaluated. Once again, its implementation requires explicit support by the workflow engine for the extraction of data elements from external applications. FLOWer provides this capability via mapping functions linked to each activity in a plan which extract required data elements from an external database.

### Issues

The main issue associated with this Pattern is determining an appropriate time to undertake the "pull" action of the data element from the case to the environment.

### Solutions

As for the previous Pattern, there are many points during case execution at which the update of a data element from a source in the operating environment does not seem to be sensible. In the case of FLOWer, this action occurs at the beginning of a specific task (or tasks) within the case. It would also seem to be appropriate to pass required data elements from the environment at the beginning of a case.

### Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+	Supported via the mapping construct using In type
COSA	4.2	-	Not supported
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported. Message flows can indeed be drawn between the boundaries of two pools, one representing the Process the Case instantiates, and other one representing the Environment. However, the semantics of this construct does not capture data exchange at any moment during the execution of the case
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support for cases to initiate requests for data and receive replies from external applications at any time	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

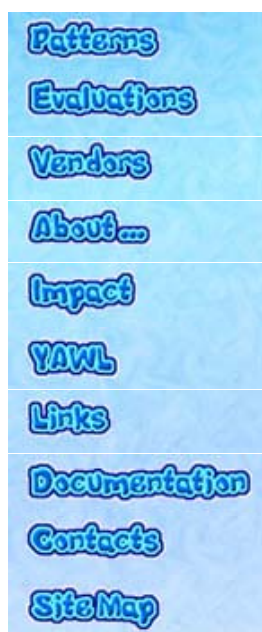
0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)



# Workflow Patterns



## Pattern 21 (Environment to Case - Push)

[FLASH animation of Data Interaction - Environment to Case - Push-Oriented pattern](#)

### Description

The ability of a workflow case to accept data elements passed to it from services or resources in the operating environment.

### Example

During execution, each case of the *Evaluate Fitness* workflow may receive additional *fitness measurements* data from the *Biometry system*.

### Motivation

During the course of the execution of a workflow case, new case-level data elements may become available or the values of existing items may change. A means of communicating these data updates to the workflow case is required.

### Implementation

There are two distinct mechanisms by which this Pattern can be achieved:

- Values for data elements can be specified at the initiation of a specific case.
- The workflow can provide facilities for enabling update of data elements during the execution of a case.

Staffware supports the former approach allowing startup values to be specified for cases. Websphere MQ Workflow also allows values to be specified for data elements at case commencement. FLOWer provides direct support for update of data elements during case execution through the `chp_dat_setval` API call.

### Issues

None observed.

### Solutions

N/A.

### Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it

complies with each of the criteria specified.

- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Limited supported for passing parameters at case commencement
Websphere MQ Workflow	3.4	+/-	Parameter values can be specified at the initiation of each case
FLOWer	3.0	+	Supported via CHIP library API
COSA	4.2	+	Trigger functions allow process instance and folder data to be set
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported. Message flows can indeed be drawn between the boundaries of two pools, one representing the Process the Case instantiates, and other one representing the Environment. However, the semantics of this construct does not capture data exchange at any moment during the execution of the case
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support for cases to accept externally generated data at any time during execution	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)

# Workflow Patterns



## Pattern 22 (Case to Environment - Pull)

[FLASH animation of Data Interaction - Case to Environment - Pull-Oriented pattern](#)

### Description

The ability of a workflow case to respond to requests for data elements from a service or resource in the operating environment.

### Example

Each case of the *Handle Visa* workflow must respond to data requests from the *Immigration Department*.

### Motivation

The rationale for this Pattern is similar to that for Pattern 20 in terms of supporting data passing from a case to a resource or service external to the workflow, however in this case the data passing is initiated by a request from the external process.

### Implementation

For workflow engines supporting case level data, the most widely common approach to supporting this Pattern is to provide the ability for external processes to interactively query case-level data elements. This Pattern is not widely implemented amongst the workflow systems examined. Only FLOWer provides direct support via the `chp_dat_getval` API call.

### Issues

None observed.

### Solutions

N/A

### Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+	Supported via CHIP library API
COSA	4.2	+	Trigger functions allow process instance and folder data to be read
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported. Message flows can indeed be drawn between the boundaries of two pools, one representing the Process the Case instantiates, and other one representing the Environment. However, the semantics of this construct does not capture data exchange at any moment during the execution of the case
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support for cases to respond to external requests for data at any time	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns

[Patterns](#)[Evaluations](#)[Vendors](#)[About...](#)[Impact](#)[YAWL](#)[Links](#)[Documentation](#)[Contacts](#)[Site Map](#)

## Pattern 23 (Workflow to Environment - Push)

[FLASH animation of Data Interaction - Workflow to Environment - Push-Oriented pattern](#)

### Description

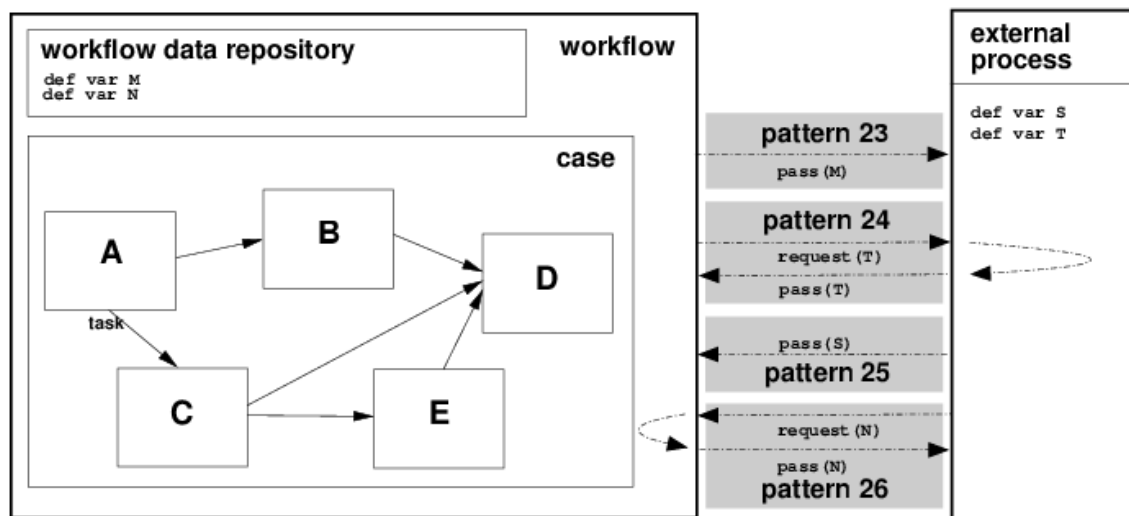
The ability of a workflow engine to pass data elements to resources or services in the operational environment.

### Example

At any time the *Process Tax Return* workflow may save its working data to an external data warehouse facility.

### Motivation

Where a workflow engine stores data elements at workflow level, it is desirable if facilities are available for communicating this information to external applications. This is particularly useful as a means of enabling external applications to be kept informed of aggregate workflow relevant data (eg. % successful cases, resource usage etc.) without needing to continually poll the workflow engine. The various approaches to passing workflow level data to and from the external environment are illustrated in Figure 16.



**Figure 16:** Data interaction between a workflow system and the operating system

### Implementation

Whilst this Pattern serves as a useful mechanism for proactively communicating data elements and runtime information to an external application on a whole of workflow basis, it is not widely implemented. Websphere MQ Workflow provides a limited ability to communicate the creation of and changes to run-time data elements (e.g. work items) to an external application via its push data access model.

## Issues

None observed.

## Solutions

N/A

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	+/-	Push data access model provides limited facilities for communicating runtime workflow data to external applications
FLOWer	3.0	-	Not supported
COSA	4.2	-	Not supported
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	As workflow data is not supported, any data interaction to and from a workflow is not applicable
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct support for the workflow engine to initiate data passing to external applications	1. Achievable via programmatic extensions

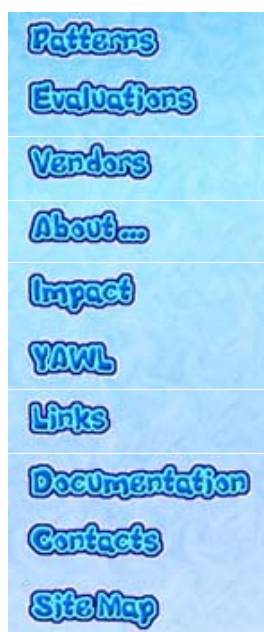
© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) | [Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 24 (Environment to Workflow - Pull)

[FLASH animation of Data Interaction - Environment to Workflow - Pull-Oriented pattern](#)

### Description

The ability of a workflow to request workflow-level data elements from external applications.

### Example

The *Monitor Portfolios* workflow is able to request *new market position* download from the *Stock Exchange* at any time.

### Motivation

This Pattern is motivated by the need for a workflow (or elements within it) to request data from external applications for subsequent use by some or all components of the workflow engine.

### Implementation

None of the workflow systems examined provide direct support for this Pattern.

### Issues

None observed.

### Solutions

N/A.

### Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation



Staffware	9	+/-	Indirect support for importing tables and lists via the swutil function
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	-	Not supported
COSA	4.2	-	Not supported
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	As workflow data is not supported, any data interaction to and from a workflow is not applicable
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct support for the workflow engine to initiate requests for data and receive replies from external applications	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

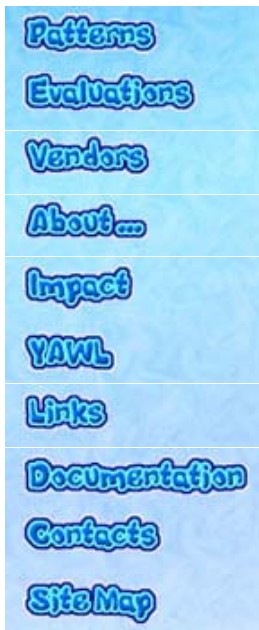
0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))



# Workflow Patterns



## Pattern 25 (Environment to Workflow - Push)

[FLASH animation of Data Interaction - Environment to Workflow - Push-Oriented pattern](#)

### Description

The ability of services or resources in the operating environment to pass workflow-level data to a workflow process.

### Example

All *mining permits* data currently available is passed to the *Develop Mining Leases* workflow.

### Motivation

The rationale for this Pattern is to provide applications independent of the workflow engine with the ability to create or update workflow-level data elements in a workflow engine.

### Implementation

There are three ways in which this objective might be achieved, all of which require support from the workflow engine:

1. Data elements are passed into the workflow engine (typically as part of the command line) at the time the workflow engine is started. (This assumes that the external application starts the workflow engine.)
2. The external application initiates an import of workflow-level data elements by the workflow engine.
3. The external application utilises an API call to set data elements in the workflow engine.

Staffware provides limited support for the first of these options to set workflow configuration data. It also supports the second of these options in a limited sense through the *swutil* option which allows workflow tables and lists to be populated from external data files in predefined formats. Websphere MQ Workflow provides support for the third alternative and allows persistent lists and other runtime workflow data elements to be updated via API calls.

### Issues

None observed.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	+/-	Indirect support for manipulation persistent lists via API calls
FLOWer	3.0	-	Not supported
COSA	4.2	-	Not supported
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	As workflow data is not supported, any data interaction to and from a workflow is not applicable
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct support for the workflow engine to accept externally generated data during execution	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 26 (Workflow to Environment - Pull)

[FLASH animation of Data Interaction - Workflow to Environment - Pull-Oriented pattern](#)

### Description

The ability of a workflow engine to handle requests for workflow-level data from external applications.

### Example

At any time, the *Monitor Portfolios* workflow may be required to respond to requests to provide data on any portfolios being reviewed to the *Securities Commissioner*.

### Motivation

Similar to the previous Pattern, however in this instance, the request for workflow level data is initiated by the external application.

### Implementation

Generally there are two distinct approaches to achieving this requirement:

1. External applications can call utility programs provided by the workflow engine to export workflow-level data to a file which can be subsequently read by the application.
2. External applications can utilise API facilities provided by the workflow engine to access the required data programmatically.

Staffware adopts the first of these approaches to provide external applications with access to table and list data that is stored persistently by the workflow engine. Websphere MQ Workflow utilises the latter strategy allowing external applications to bind in API calls providing access to workflow-level data items such as persistent lists and lists of work items. COSA provides support for both approaches.

### Issues

None observed.

### Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Access to table and field data is supported via the swutil function
Websphere MQ Workflow	3.4	+	Directly supported via the Runtime API
FLOWer	3.0	-	Not supported
COSA	4.2	+	Supported via command line and C/COM/Java APIs
XPDL	1.0	-	Not reported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	As workflow data is not supported, any data interaction to and from a workflow is not applicable
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support for workflows to respond to external requests for data	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 27 (Data Transfer by Value - Incoming)

[FLASH animation of Data Transfer by Value - Incoming and Outgoing patterns](#)

### Description

The ability of a workflow component to receive incoming data elements by value relieving it from the need to have shared names or common address space with the component(s) from which it receives them.

### Example

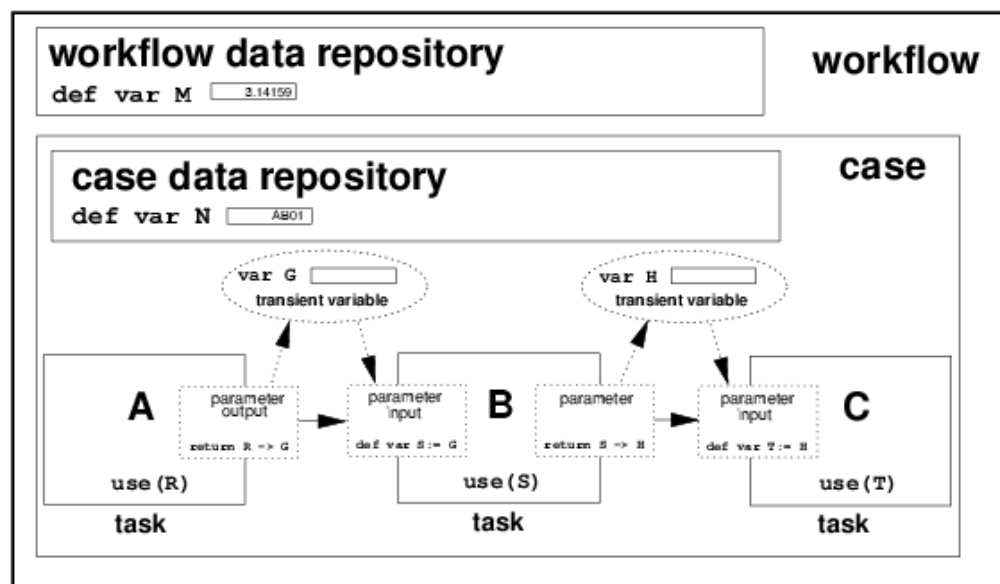
At commencement, the *Identify Successful Applicant* task receives values for the *required role* and *salary* data elements.

### Motivation

Under this scenario, data elements are passed as values between communicating workflow components. There is no necessity for each workflow component to utilise a common naming strategy for the data elements or for components to share access to a common data store in which the data elements reside. This enables individual components to be written in isolation without specific knowledge of the manner in which data elements will be passed to them or the context in which they will be utilised.

### Implementation

This approach to data passing is commonly used for communicating data elements between tasks that do not share a common data store or wish to share task-level (or block-level) data items. The transfer of data between workflow components is typically based on the specification of mappings between them identifying source and target data locations. In this situation, there is no necessity for common naming or structure of data elements as it is only the data values that are actually transported between interacting components. Figure 17 illustrates the passing of the value of data element R in task instance A to task instance B where it is assigned to data element S. In this example, a transient variable G (depending on the specific workflow engine, this could be a data container or a case or workflow level variable) is used to mediate the transfer of the data value from task instance A to task instance B which do not share a common address space.



**Figure 17:** Data transfer by value

Websphere MQ Workflow utilises this approach to data passing in conjunction with distinct data channels. Data elements from the originating workflow task instance are coalesced into a data container. A mapping is defined from this data container to a distinct data container which is transported via the connecting data channel between the communicating tasks. A second mapping is then defined from the data container on the data channel to a data container in the receiving task. BPEL4WS provides the option to pass data elements between activities using messages - an approach which relies on the transfer of data between workflow components by value. Similarly COSA provides analogous support using triggers although this does not allow for data types to be maintained during transfer. XPDL provides more limited support for data transfer by value between a block task and subflow. As all data elements are case level, there is no explicit data passing between tasks.

## Issues

None observed.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	+	All data passing via containers is by value

FLOWer	3.0	-	Not supported
COSA	4.2	+/-	Achievable via triggers although limitation on range and typing of parameters passed
XPDL	1.0	+/-	Although call by value is the standard mechanism (see Section 7.1.2.0), there is not explicit data passing between activities. However, the element "subflow" provides actual parameters which should be matched by the formal parameters of the corresponding workflow process
BPEL4WS	1.1	+	option to pass data elements between activities by value (using messages)
BPMN	1.0	+	Supported though the notion of InputSets of Activities
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Directly supported by the attributes of <assign> wizard

## Summary of Evaluation

+ Rating	+/- Rating
1. Workflow components are able to accept data elements passed to them as values	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))



# Workflow Patterns



## Pattern 28 (Data Transfer by Value - Outgoing)

[FLASH animation of Data Transfer by Value - Incoming and Outgoing patterns](#)

### Description

The ability of a workflow component to pass data elements to subsequent components as values relieving it from the need to have shared names or common address space with the component(s) to which it is passing them.

### Example

Upon completion, the *Identify Successful Applicant* task passes the *successful applicant name* to the next task.

### Motivation

Similar to Pattern 27 although in this scenario, the emphasis is on minimising the coupling between a component and the subsequent components that may receive its output data.

### Implementation

As for Pattern 27.

### Issues

None observed.

### Solutions

N/A.

### Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported



Websphere MQ Workflow	3.4	+	All data passing via containers is by value
FLOWer	3.0	-	Not supported
COSA	4.2	+/-	As for Pattern 27
XPDL	1.0	+/-	As for Pattern 27
BPEL4WS	1.1	+	Option to pass data elements between activities by value (using messages)
BPMN	1.0	+	Supported through the notion of OutputSets of Activities
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Directly supported by the attributes of <assign> wizard

## Summary of Evaluation

+ Rating	+/- Rating
1. Workflow components are able to pass data elements to subsequent components by value	1. Achievable via programmatic extensions

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 29 (Data Transfer - Copy In/Copy Out)

[FLASH animation of Data Transfer - Copy In/Copy Out pattern](#)

### Description

The ability of a workflow component to copy the values of a set of data elements into its address space at the commencement of execution and to copy their final values back at completion.

### Example

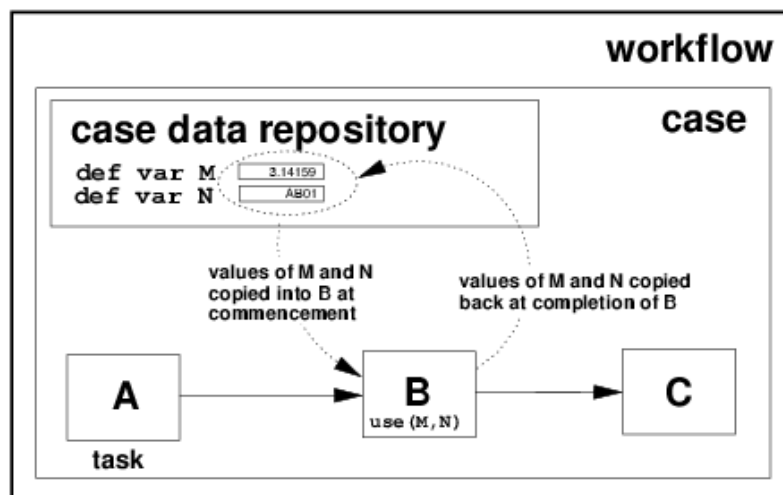
When the *Review Witness Statements* task commences, copy in the *witness statement* records and copy back any changes to this data at task completion.

### Motivation

This facility provides components with the ability to make a local copy of data elements that can be referenced elsewhere in the workflow. This copy can then be utilised during execution and any changes that are made can be copied back at completion. It enables components to function independently of data changes and concurrency constraints that may occur in the broader workflow context.

### Implementation

Whilst not a widely supported data passing strategy, some workflow engines do offer limited support for it. In some cases, its use necessitates the adoption of the same naming strategy and structure for data elements within the component as used in the environment from which their values are copied. The manner in which this style of data passing operates is shown in Figure 18.



**Figure 18:** Data transfer - copy in/copy out

FLOWer utilises this strategy for accessing data from external databases via the InOut construct. XPDL adopts a similar strategy for data passing to and from subflows.

## Issues

Difficulties can arise with this data transfer strategy where data elements are passed to a sub-workflow that executes independently (asynchronously) of the calling workflow. In this situation, the calling workflow continues to execute once the sub-workflow call has been made and this can lead to problems when the sub-workflow completes and the data elements are copied back to the calling workflow as the point at which this copy back will occur is indeterminate.

## Solutions

There are two potential solutions to this problem:

- Do not allow asynchronous sub-workflow calls.
- In the event of asynchronous sub-workflow calls occurring, do not copy back data elements at task completion - this is the approach utilised by XPDL.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+/-	Supported via InOut mapping construct
COSA	4.2	-	Not supported
XPDL	1.0	+/-	As for 27
BPEL4WS	1.1	-	Not supported
BPMN	1.0	+/-	Partially supported. It occurs when decomposition is realised through Independent Sub-Process. The data attributes to be copied into/out of the Independent Sub-Process are specified through the Input- and OutputPropertyMaps attributes
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	+	Directly supported by means of two <assign> constructs

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct workflow support exists for a workflow component to copy in data elements from an external source (either within or outside the workflow) when commencing execution and to copy the (possibly updated) values back to the external sources at the conclusion of execution	1. Indirectly achievable via programmatic extensions

---

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 30 (Data Transfer by Reference - Unlocked)

[FLASH animation of Data Transfer by Reference - Unlocked pattern](#)

### Description

The ability to communicate data elements between workflow components by utilising a reference to the location of the data element in some mutually accessible location. No concurrency restrictions apply to the shared data element.

### Example

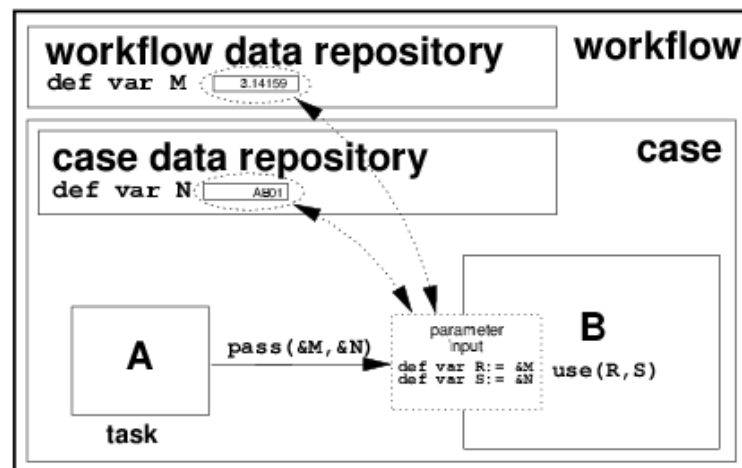
The *Finalise Interviewees* task passes the location of the *interviewee shortlist* to all subsequent tasks in the *Hire* process.

### Motivation

This Pattern is commonly utilised as a means of communicating data elements between workflow components which share a common data store. It involves the use of a named data location (which is generally agreed at design time) that is accessible to both the origin and target components and, in effect, is an implicit means of passing data as no actual transport of information occurs between the two components.

### Implementation

Reference-based data passing requires the ability for communicating workflow components to have access to a common data store and to utilise the same reference notation for elements that they intend to use co-jointly. There may or may not be communication of the location of shared data elements via the control channel or data channel (where supported) at the time that control flow passes from one component to the next. This mechanism for data passing is employed within the Staffware, FLOWer and COSA workflow engines and also within XPDL and BPEL4WS. Figure [19](#) illustrates the passing of two data elements M and N (workflow and case level respectively) by reference from task instance A to B. Note that task instance B accepts these elements as parameters to internal data elements R and S respectively.



**Figure 19:** Data transfer by reference - unlocked

## Issues

None observed.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Achievable via fields
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+	Standard means of data passing
COSA	4.2	+	Standard means of data passing
XPDL	1.0	+	Directly supported where data is passed implicitly by shared variables
BP4WS	1.1	+	No concurrency control is applied by default
BPMN	1.0	-	Not supported
UML	2.0	-	Not supported
Oracle BP4L	10.1.2	+	No concurrency restrictions for accessing global data

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Block, case or workflow level data can be accessed by all associated workflow components</li><li>2. Programmatic extensions required</li></ol>	<ol style="list-style-type: none"><li>1. Restricted data visibility limits the use of this strategy</li></ol>

---

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 31 (Data Transfer by Reference - With Lock)

[FLASH animation of Data Transfer by Reference - With Lock pattern](#)

### Description

The ability to communicate data elements between workflow components by passing a reference to the location of the data element in some mutually accessible location. Concurrency restrictions are implied with the receiving component receiving the privilege of read-only or dedicated access to the data element.

### Example

At conclusion, the *Allocate Client Number* task passes the locations of the *new client number* and *new client details* data elements to the *Prepare Insurance Document* task, which receives dedicated access to these data items.

### Motivation

This approach is an extension of Pattern 30, in which there is also the expectation that the originating workflow component had acquired either read-only or exclusive access to the specific data elements being passed (i.e. a read or write lock). The workflow component that receives these data elements can choose to relinquish this access level (thus making them available to other workflow components) or it may choose to retain it and pass it on to later components. There is also the potential for the access level to be promoted (i.e. from a read to a write lock) or demoted (i.e. from a write to a read lock).

### Implementation

This Pattern is not widely implemented in the workflow engines examined. BPEL4WS provides an approach to concurrency control through the use of serializable scopes which allow fault and compensation handlers to be defined for groups of process activities. FLOWer provides limited support for a style of write lock which allows the primary case to modify data whilst still enabling it to be accessible for reading by other processes.

### Issues

None observed.

### Solutions

N/A.



## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+/-	Limited concurrency support. Case can only be updated by one user at a time but others can still view case data
COSA	4.2	-	Not supported
XPDL	1.0	-	No support, see Section 7.1.2.1 of WPMC-TC-1025 (Oct. 2002).
BPEL4WS	1.1	+/-	Some measure of concurrency control can be achieved through the use of serializable scopes
BPMN	1.0	+	As BPMN adopts a token-oriented approach to data passing, the parameters - which typically relate to objects - are consumed (i.e. locked) at activity commencement.
UML	2.0	+	Directly supported. The standard means of passing data elements to an activity is via parameters based on data tokens.
Oracle BPEL	10.1.2	-	The serializable scope feature does not work according to its semantics

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"> <li>1. Block, case or workflow level data can be access by all workflow components</li> <li>2. Locking/concurrency control facilities are provided</li> </ol>	<ol style="list-style-type: none"> <li>1. Restricted data visibility limits the use of this strategy</li> <li>2. Programmatic extensions required</li> </ol>

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)

# Workflow Patterns



## Pattern 32 (Data Transformation - Input)

[FLASH animation of Data Transformation - Input and Output patterns](#)

### Description

The ability to apply a transformation function to a data element prior to it being passed to a workflow component.

### Example

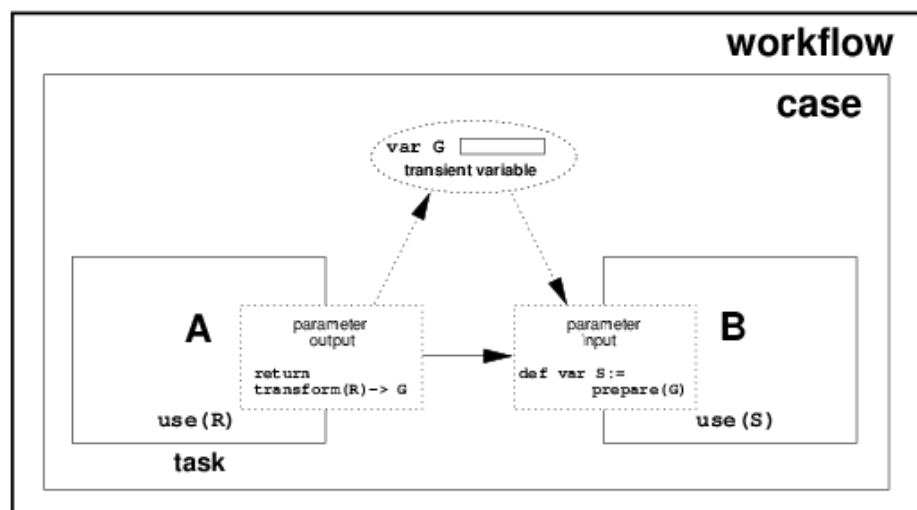
Prior to passing the *transform voltage* data element to the *Project\_demand()* function, convert it to standard ISO measures.

### Motivation

The ability to specify transformation functions provides a means of handling potential mismatches between data elements and formal parameters to workflow components in a declarative manner. These functions could be internal to the workflow or could be externally facilitated.

### Implementation

In the example shown in Figure 20, the *prepare()* function intermediates the passing of the data element *G* between task instances *A* and *B*. At the point at which control is passed to *B*, the results of performing the *prepare()* transformation function on data element *G* are made available to the input parameter *S*.



**Figure 20:** Data transformation - input

Staffware provides the ability for a task to call a function capable of manipulating the data elements passed to the task prior to its commencement through the use of the form initial facility. The function called may be based on the Staffware script language or external 3GL capabilities and hence can support relatively complex transformation functionality. FLOWer provides limited facilities for the transformation of input data elements through the use of mappings and derived elements.

## Issues

None observed.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Supported via form initial function for tasks that have forms
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+/-	Indirectly supported via mapping and derived elements
COSA	4.2	-	Not supported
XPDL	1.0	-	Not mentioned
BPEL4WS	1.1	-	Not supported
BPMN	1.0	+/-	Partially supported. It occurs when decomposition is realised through Independent Sub-Process. As the Input- and OutputPropertyMaps are in form of Expressions, we assume that transformation function can be specified through them
UML	2.0	+	Supported by the ObjectFlow transformation behaviour which allows transformation functions to be applied to data tokens as they are passed along connecting edges between activities.
Oracle BPEL	10.1.2	-	Directly supported by the attributes of <assign> wizard

## Summary of Evaluation

<b>+ Rating</b>	<b>+/- Rating</b>
1. Direct workflow support 2. Transformations can be specified on all data element	1. Only achievable via programmatic extensions

---

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 33 (Data Transformation - Output)

[FLASH animation of Data Transformation - Input and Output patterns](#)

### Description

The ability to apply a transformation function to a data element immediately prior to it being passed out of a workflow component.

### Example

Summarise the *spatial telemetry data* returned by the *Satellite Download* task before passing to subsequent activities.

### Motivation

As for Pattern 32.

### Implementation

As described for Pattern 32 except that the facilities identified are used for transforming the data elements passed from a task instance rather than for those passed to it.

### Issues

None observed.

### Solutions

N/A.

### Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Supported via form release function for tasks that have forms

Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+/-	Indirectly supported via mappings and derived elements
COSA	4.2	-	Not supported
XPDL	1.0	-	Not mentioned
BPEL4WS	1.1	-	Not supported
BPMN	1.0	+/-	Partially supported. It occurs when decomposition is realised through Independent Sub-Process. As the Input- and OutputPropertyMaps are in form of Expressions, we assume that transformation function can be specified through them
UML	2.0	+	Supported by the ObjectFlow transformation behaviour which allows transformation functions to be applied to data tokens as they are passed along connecting edges between activities.
Oracle BPEL	10.1.2	-	Directly supported by the attributes of <assign> wizard

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Direct workflow support</li><li>2. Transformations can be specified on all data elements</li></ol>	<ol style="list-style-type: none"><li>1. Only achievable via programmatic extensions</li><li>2. Limited transformation capability or range</li></ol>

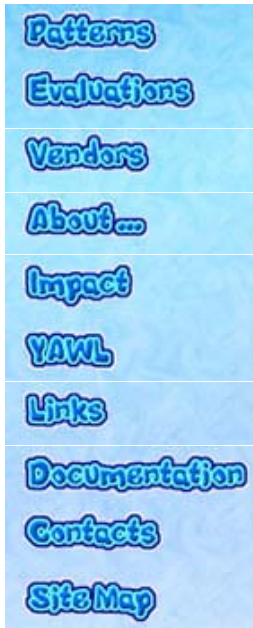
© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 34 (Task Precondition - Data Existence)

[FLASH animation of Task Precondition - Data Existence pattern](#)

### Description

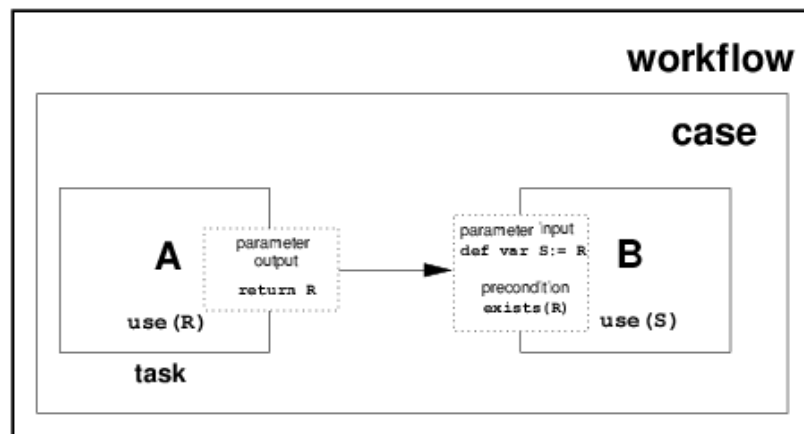
Data-based preconditions can be specified for tasks based on the presence of data elements at the time of execution.

### Example

Only execute the *Run Backup* task when *tape\_loaded\_flag* exists.

### Motivation

The ability to deal with missing data elements at the time of task invocation is desirable. This allows corrective action to be taken during workflow execution rather than necessitating the raising of an error condition and halting action.



**Figure 21:** Task precondition - data existence

### Implementation

Typically data existence preconditions are specified on task input parameters (footnote<sup>12</sup>) in the workflow design model as illustrated in Figure 21. In this context, data existence refers to the ability to determine whether a required parameter has been defined and provided to the task at the time of task invocation and whether it has been assigned a value. One of five actions are possible where missing parameters are identified:

- Defer task commencement until they are available.
- Specify default values for parameters to take when they are not available.

- Request values for them interactively from workflow users.
- Skip this task and trigger the following task(s).
- Kill this thread of execution in the workflow case.

This Pattern is implemented in a variety of different ways amongst several of the tools examined. Staffware provides the ability to set default values for fields which do not have a value recorded for them via the initial form command facility but only for those tasks that have forms associated with them. Conditional actions can also be used to route control flow around (i.e. skip) a task where required data elements are not available. FLOWer provides the milestone construct which, amongst other capabilities, provides data synchronization support allowing the commencement of a subsequent task to be deferred until nominated data elements have a value specified. COSA also provides transition conditions on incoming arcs and in conjunction with the CONDITION.TOKEN and STD tool agents, it enables the passing of control to a task to be delayed until the transition condition (which could be data element existence) is achieved. BPEL4WS provides exception handling facilities where an attempt to utilise an uninitialised variable is detected. These can be indirectly used to provide data existence precondition support at task level but require each task to be defined as having its own scope with a dedicated fault handler to manage the uninitialised parameters accordingly.

## Issues

A significant consideration in managing preconditions that relate to data existence is being able to differentiate between data elements that have an undefined value and those that are merely empty or null.

## Solutions

Staffware addresses this issue by using distinct tokens (SW\_NA) for data elements that have undefined values. Uninitialised fields have this value by default and it can be tested for in workflow expressions and conditions. FLOWer also provides facilities to test whether data elements have been assigned a value. BPEL4WS provide internal capabilities to recognise uninitialised data elements although these facilities are not directly accessible to workflow developers. Other workflow engines examined do not differentiate between undefined and empty (or null) values.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Uninitialised fields can be detected via the SW_NA value. Initial scripts can be used to set default values for missing parameters where tasks have forms. Conditional actions can be used to skip tasks where required parameters are missing
Websphere MQ Workflow	3.4	-	Not supported



FLOWer	3.0	+	Supported in various ways e.g. milestone defined for a mandatory data element can defer subsequent task initiation until required data element is available
COSA	4.2	+	By specifying transition conditions for tasks based on the CONDITION.TOKEN and STD.exists tool agents, subsequent task invocation can be delayed until required data element is available
XPDL	1.0	-	Not supported
BPEL4WS	1.1	+/-	Can be indirectly facilitated via exception handlers dealing with attempts to use uninitialised parameters but requires dedicated scope for each activity
BPMN	1.0	+	In the cases data transfer is captured through Data Objects, the Boolean attribute RequiredForStart can capture a precondition for data existence
UML	2.0	+	Directly supported. Both action and activity constructs include local preconditions and postconditions based on logical expressions framed in OCL.
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
1. Direct precondition support for evaluation data element existence at task instance level	1. Programmatic extensions required

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)

# Workflow Patterns



## Pattern 35 (Task Precondition - Data Value)

[FLASH animation of Task Precondition - Data Value pattern](#)

### Description

Data-based preconditions can be specified for tasks based on the value of specific parameters at the time of execution.

### Example

Execute the *Rocket Initiation* task when *countdown* is 2.

### Motivation

The ability to specify value-based preconditions on parameters to tasks provides the ability to delay execution of the task (possibly indefinitely) where a precondition is not satisfied.

### Implementation

There are three possible alternatives where a value-based precondition is not met:

- The task can be skipped and the subsequent task(s) initiated.
- Commencement of the task can be delayed until the required precondition is achieved.
- This thread of execution in the workflow case can be terminated.

This Pattern is directly implemented by FLOWer through the milestone construct which enables the triggering of a task to be delayed until a parameter has a specified value. Similarly COSA provides the ability to delay execution of a task where a precondition is not met through the specification of transition conditions based on the required data values on the incoming arc to the task. By specifying alternate data conditions (which correspond to the negation of the required data values) on the outgoing arc from the state preceding the contingent task to the subsequent state, it is also possible to support the skipping of tasks where data preconditions are not met. Both approaches assume the transition conditions are specified in conjunction with the CONDITION.TOKEN tool agent. In a more limited way, Staffware provides the ability to delay task execution through the specification of scripts which test for the required data values at task commencement. However, this approach is only possible for tasks which have forms associated with them. A better solution is to use condition actions to test for required parameters and skip the task invocation where they are not available. XPDL provides support for a task to be skipped when a nominated data value is not achieved through the specification of an additional edge in the workflow schema which bypasses the task in question (i.e. it links the preceding and subsequent tasks) and has a transition condition which is the negation of required data values. A

similar effect can be achieved in BPEL4WS through the use of link conditions. By specifying a link condition to the task (call it A) which corresponds to the required data values and creating a parallel *empty* task in the business process that has a link condition that is the negation of this, task A will be skipped if the required data values are not detected.

## Issues

None identified.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Initial scripts can be used to delay invocation depending on parameter values where tasks have forms. Conditional actions can be used to skip tasks where parameter values are not equal to specified values
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+	Milestones can also have a condition delaying invocation of subsequent task until condition is met
COSA	4.2	+	By specifying transition conditions for tasks based on the required data values and CONDITION.TOKEN tool agent, subsequent task invocation can be delayed until required data values are met
XPDL	1.0	+	Inclusion of additional edge around task with transition condition that is the negation of required data values allows task to be skipped when values not met
BPEL4WS	1.1	+	Inclusion of additional link around task with transition condition that is the negation the required data values allows task to be skipped when data values not met.
BPMN	1.0	-	Not supported
UML	2.0	+	Directly supported. Both action and activity constructs include local preconditions and postconditions based on logical expressions framed in OCL.

Oracle BPEL	10.1.2	+	Directly supported via joinCondition evaluation the status of links
-------------	--------	---	---

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Direct precondition support at task instance level</li><li>2. Support for a broad range of preconditions</li></ol>	<ol style="list-style-type: none"><li>1. Limited range of preconditions supported</li><li>2. Programmatic extensions required</li></ol>

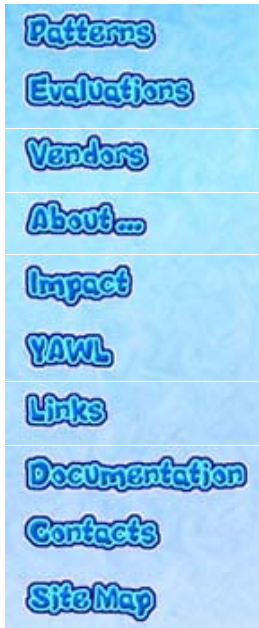
© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 36 (Task Postcondition - Data Existence)

[FLASH animation of Task Postcondition - Data Existence pattern](#)

### Description

Data-based postconditions can be specified for tasks based on the existence of specific parameters at the time of execution.

### Example

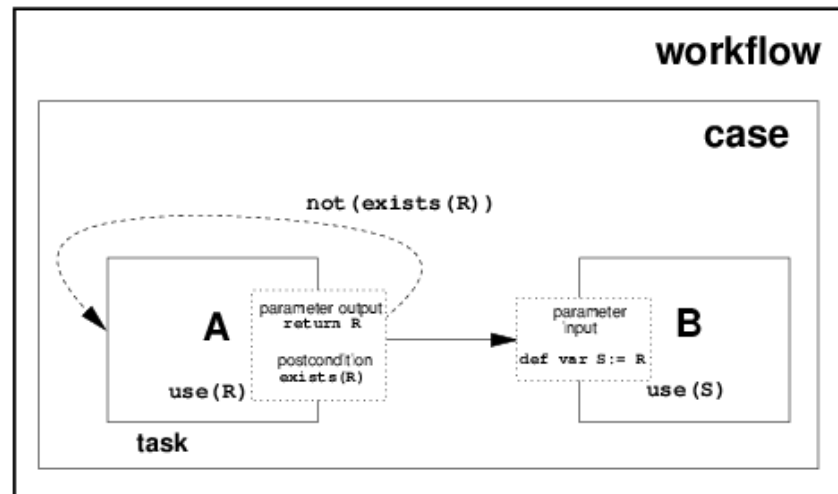
Do not complete the *Rocket Initiation* task until *ignition data* is available.

### Motivation

Implementation of this Pattern ensures that a task cannot complete until specified output parameters exist and have been allocated a value.

### Implementation

Two alternatives exist for the implementation of this Pattern. Where tasks that have effectively finished all of their processing but have a nominated data existence postcondition that has not been satisfied, either the task could be suspended until the required postcondition is met or the task could be implicitly repeated until the specified postcondition is met. Figure 22 illustrates this Pattern. The specification of a data-based postcondition on a task effectively establishes an implicit control loop back to the beginning of the task that corresponds to the negation of the postcondition. Depending on the semantics of control flow within a specific workflow engine, the re-routing of control back to the beginning of the task may or may not result in the task being executed again. The implication of both scenarios however, is that the task does not pass on control flow until the required parameters exist and have defined values. FLOWer provides direct support for this Pattern by allowing data element fields in task constructs (called plan elements) to be specified as mandatory, thus ensuring they have a value specified before the plan element can complete execution. Websphere MQ Workflow implements this Pattern through the specification of exit conditions on tasks. If the required exit condition is not met at the time task processed is completed, then the task is restarted. A specific IS NULL function is available to test if a parameter has been assigned a value. Staffware provides indirect support through the inclusion of release scripts with tasks which evaluate whether the required data elements have been specified. Completion of the task is deferred until the required condition is satisfied although this approach is limited to tasks that have forms associated with them. Similarly COSA indirectly supports this Pattern through the use of condition-based triggers which only pass control to a subsequent task once a required data existence condition is set.



**Figure 22:** Task postcondition - data existence

## Issues

As for Pattern 34.

## Solutions

As for Pattern 34.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Indirectly achievable via validation functions linked to release form commands
Websphere MQ Workflow	3.4	+	Directly supported via exit conditions and the IS NULL function which allow parameter value assignment to be tested. Where exit condition is not met, task automatically repeats
FLOWer	3.0	+	Achieved by making a data element mandatory. Task will not complete until the data element had a defined value
COSA	4.2	-	Not supported
XPDL	1.0	-	Not supported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	+	In the cases data transfer is captured through Data Objects, the Boolean attribute ProducedAtCompletion can capture a postcondition for data existence

UML	2.0	+	Directly supported. Both action and activity constructs include local preconditions and postconditions based on logical expressions framed in OCL
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Direct postcondition support for evaluating data element existence at task level</li><li>2. Must support internal task repetition</li></ol>	<ol style="list-style-type: none"><li>1. Limited range of postconditions</li><li>2. Programmatic extensions required</li></ol>

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 37 (Task Postcondition - Data Value)

[FLASH animation of Task Postcondition - Data Value pattern](#)

### Description

Data-based postconditions can be specified for tasks based on the value of specific parameters at the time of execution.

### Example

Execute the *Fill Rocket* task until *rocket-volume* is 100%.

### Motivation

Implementation of this Pattern would ensure that a task could not complete until specific output parameters had a particular data value or are in a specified range.

### Implementation

Similar to Pattern 36, two options exist for handling the achievement of specified values for data elements at task completion:

- Delay execution until the required values are achieved.
- Implicitly re-run the task.

The implementation methods for this Pattern adopted by the various tools examined are identical to those described in Pattern 36.

### Issues

None identified.

### Solutions

N/A.

### Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.



Product/Language	Version	Score	Motivation
Staffware	9	+/-	Indirectly achievable via validation functions linked to release form commands
Websphere MQ Workflow	3.4	+	Directly supported by specifying exit condition based on the required parameter value. Where exit condition is not met, task automatically repeats
FLOWer	3.0	+	Modelling elements can have a postcondition. Task will not complete until the postcondition (which should be based on the required parameter value) is met
COSA	4.2	-	Not supported
XPDL	1.0	-	Not supported
BPEL4WS	1.1	-	Not supported
BPMN	1.0	-	Not supported
UML	2.0	+	Directly supported. Both action and activity constructs include local preconditions and postconditions based on logical expressions framed in OCL
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"> <li>1. Direct postcondition support at task level</li> <li>2. Must support internal task repetition</li> </ol>	<ol style="list-style-type: none"> <li>1. Limited range of postcondtions supported</li> <li>2. Programmatic extensions required</li> </ol>

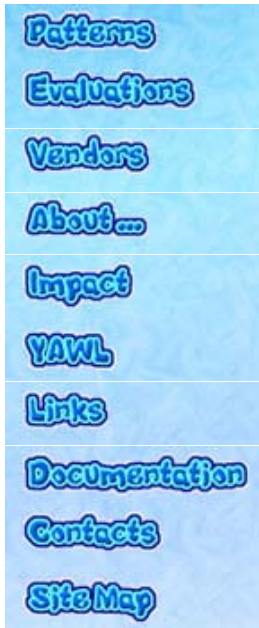
© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
[webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com)

# Workflow Patterns



## Pattern 38 (Event-Based Task Trigger)

[FLASH animation of Event-based Task Trigger \(Persistent\) pattern](#)

[FLASH animation of Event-based Task Trigger \(Transient\) pattern](#)

### Description

The ability for an external event to initiate a task.

### Example

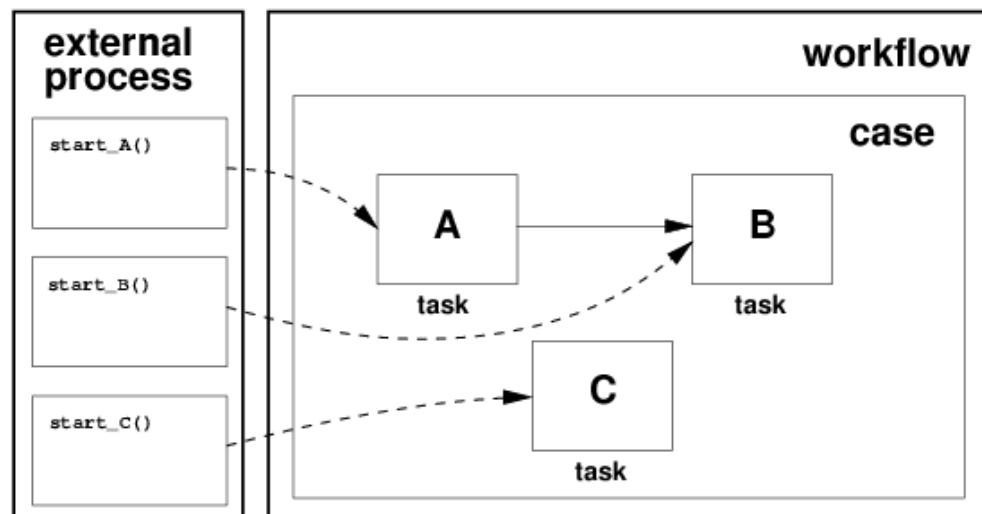
Initiate the *Emergency Shutdown* task immediately after the *Power Alarm* event occurs.

### Motivation

This Pattern centres on the ability of an external event to trigger the initiation or resumption of a specific workflow task instance. This enables the control flow of the workflow to be influenced by external applications.

### Implementation

This facility generally takes the form of an external workflow interface that provides a means for applications to trigger the execution of a specific task instance. There are three distinct scenarios that may arise in the context of this Pattern as illustrated in Figure 23.



**Figure 23:** Event-based task trigger

The first alternative (illustrated by the function in Figure 23) is that the task instance to be initiated is the first task in the workflow. This is equivalent in control terms to starting a new workflow case in which A is the first task instance. The second alternative (illustrated by the function) is that the external event is triggering the resumption of a task instance that is in the middle of a workflow process. The task instance has already had control passed to it but its execution is suspended pending occurrence of the external event trigger. This situation is shown in Figure 23 above with task instance B already triggered as a result of task instance A completing but halted from further progress until the event from occurs. The third alternative is that the task instance is isolated from the main control flow in the workflow and the only way in which it can be initiated is by receiving an external event stimulus. Figure 23 shows task instance C which can only be triggered when the event stimulus from is received. All three variants of this Pattern are directly supported by Staffware, FLOWer, COSA and BPEL4WS and in all cases, the passing of data elements as well as process triggering is supported. Websphere MQ Workflow provides indirect support for all three variants but requires event handling to be explicitly coded into activity implementations.

## Issues

None identified.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+	Events are directly supported. New workflow cases can also be externally invoked
Websphere MQ Workflow	3.4	+/-	Activity triggering is supported through various interfaces. Denotation of events needs to be explicitly included in activity implementations using facilities such as the suspend() function
FLOWer	3.0	+	Milestones can wait for data element to have (specific) value set. External processes can directly update value of data elements via CHIP library and cause case execution to resume. CHIP library also provides facilities to create new cases and plans
COSA	4.2	+	Triggers provide direct support for all three Pattern variants
XPDL	1.0	-	Not mentioned

BPEL4WS	1.1	+	Direct event support via event handlers and the receive construct. New process instances can also be invoked externally
BPMN	1.0	+	Supported through the Message, Timer, Error and Cancel Event constructs.
UML	2.0	+	Supported by the AcceptEventAction construct.
Oracle BPEL	10.1.2	+	Directly supported via <receive> and event handlers

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Direct workflow support for task instance initiation/suspension on an event trigger</li><li>2. Provision of facilities for external processes to identify the correct task instance to signal</li></ol>	<ol style="list-style-type: none"><li>1. Programmatic extensions</li></ol>

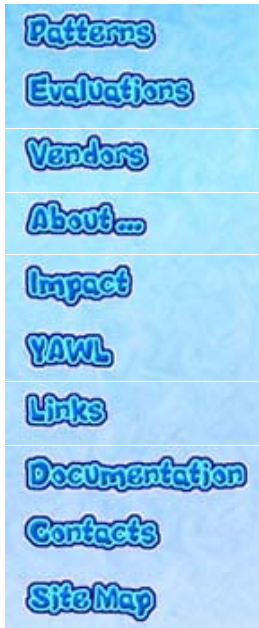
© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 39 (Data-Based Task Trigger)

[FLASH animation of Data-based Task Trigger pattern](#)

### Description

The ability to trigger a specific task when an expression based on workflow data elements evaluates to true.

### Example

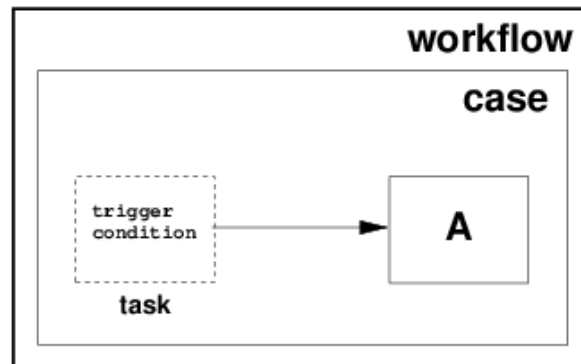
Trigger the *Re-balance Portfolio* task when the *loan margin is less than 85%*.

### Motivation

This Pattern provides a means of triggering the initiation or resumption of a task instance when a condition based on workflow data elements is satisfied.

### Implementation

This Pattern is analogous to the notion of active rules [\[Har93\]](#) or event-condition-action (ECA) rules [\[PD99\]](#) found in active databases [\[DGG95\]](#). This Pattern is directly supported in FLOWer through the specification of a condition corresponding to the required data expression on a milestone construct immediately preceding the to-be-triggered task. When the data condition is met, the task is then triggered. Similarly the Pattern can be directly implemented in COSA through the use of transition conditions which incorporate the data condition being monitored for on the incoming edges to the to-be-triggered task. Depending on the semantics required for the triggering, the transition condition may or may not include the `CONDITION.TOKEN` tool agent (footnote [13](#)). Many workflow systems do not directly support this Pattern, however in some cases it can be constructed for workflows that support event-based triggering (i.e. Pattern 38) by simulating event-based triggering within the context of the workflow. This is achieved by nominating an event that the triggered task should be initiated/resumed on and then establishing a data monitoring task that runs in parallel with all other workflow tasks and monitors data values for the occurrence of the required triggers. When one of them is detected, the task that requires triggering is initiated by raising the event that it is waiting on. The only caveat to this approach is that the workflow must not support the *Implicit Termination* Pattern [\[AHKB03\]](#), e.g. Staffware. If the workflow were to support this Pattern, then problems would arise for each workflow case when the final task was completed as this would not imply that other outstanding task instances should also terminate. Since the monitoring task could potentially run indefinitely, it could not be guaranteed that it would cease execution at case completion. This scenario is illustrated in Figure [24](#). Task instance A is to be triggered when *trigger condition* evaluates to true. A task instance is set up to monitor the status of *trigger condition* and to complete and pass control to A when it occurs.



**Figure 24:** Data-based task trigger

By adopting this strategy, the Pattern can be indirectly implemented in BPEL4WS. Although it could be similarly constructed in Staffware and variants which replaced the event link with a control edge could be implemented in Websphere MQ Workflow and XPDL, all of these workflows support implicit termination and hence would lead to problematic implementations of this Pattern.

## Issues

None identified.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	-	Not supported
Websphere MQ Workflow	3.4	-	Not supported
FLOWer	3.0	+	Directly supported via inclusion of milestones with data-based conditions preceding tasks that are waiting on the data condition
COSA	4.2	+	Directly supported by including a transition condition (which may include CONDITION.TOKEN) corresponding to the required data condition on the incoming arc to the task to be triggered
XPDL	1.0	-	Not supported
BPEL4WS	1.1	+/-	Indirectly achievable by including a data condition monitoring task in the process and triggering the to-be-triggered task via a

			link or alternatively using message events for triggering
BPMN	1.0	+	Supported through the Rule Event construct
UML	2.0	-	Not supported
Oracle BPEL	10.1.2	-	Not supported

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Direct workflow support for monitoring conditions based on data elements</li><li>2. Specific task instances can be initiated when trigger condition is achieved</li></ol>	<ol style="list-style-type: none"><li>1. Limited range of potential data triggers</li><li>2. Programmatic extensions required</li></ol>

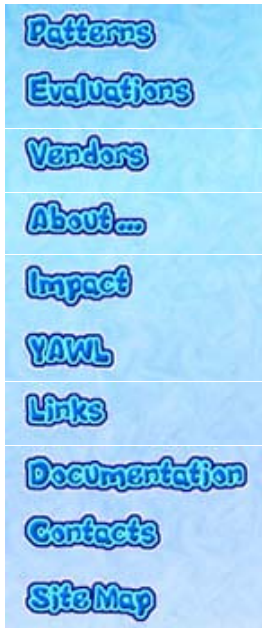
© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |  
[Impact](#)  
[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)  
[Map](#)

For any problems or questions, please [contact us](#)  
**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))

# Workflow Patterns



## Pattern 40 (Data-Based Routing)

[FLASH animation of Data-based Routing pattern](#)

### Description

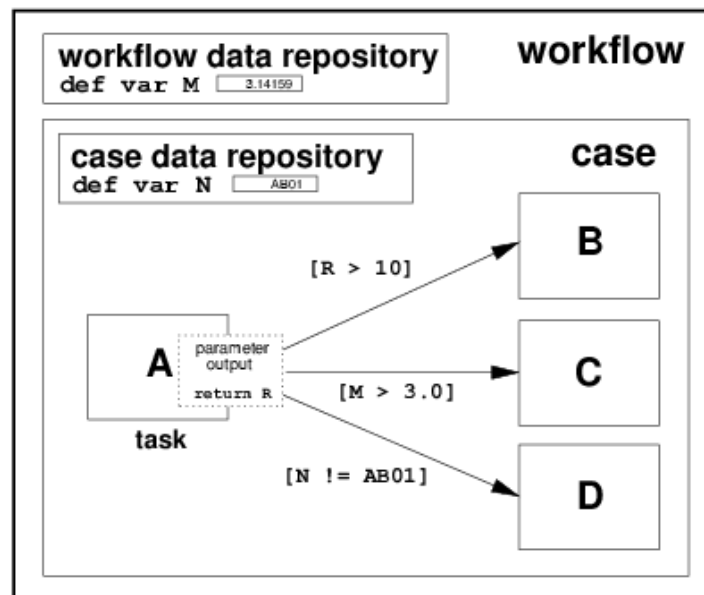
The ability to alter the control flow within a workflow case as a consequence of the value of data-based expressions.

### Example

If *alert* is *red* then execute the *Inform Fire Crew* task after the *Handle Alert* task otherwise run the *Identify False Trigger* task.

### Motivation

Without data-based routing constructs, it would not be possible for a workflow to alter its operation in response to the data it was processing. In effect, the function of the workflow would be rigidly fixed at design time. Data-based routing provides the ability for the completion of one task instance to trigger one or several subsequent task instances depending on the outcome of an expression based on the values of various workflow data elements. Figure 25 illustrates the routing expressions as constructs on the control channel from one task instance to another. Data in these expressions can be task-level data passed from the completing task instance, case or workflow level data elements. In the example shown, task instance C will be triggered once A completes (as the value of the workflow level data element M is greater than 3.0), task instance D will not and task instance B may be triggered depending on the value of data element R that is passed from A.





**Figure 25: Data-based routing**

## Implementation

This Pattern serves as an aggregation of two major Control-Flow Patterns [\[AHKB03\]](#) that depend on workflow data:

- *exclusive choice* - where control flow is passed to one of several subsequent task instances depending on the outcome of a decision or the value of an expression based on workflow data elements.
- *multi-choice* - where, depending on the outcome of a decision or the value of an expression based on workflow data elements, control flow is passed to several subsequent task instances.

Both variants of this construct are supported by Websphere MQ Workflow, FLOWer, COSA, XPDL and BPEL4WS. Staffware only directly supports the Exclusive Choice Pattern.

## Issues

None identified.

## Solutions

N/A.

## Product Evaluation

- To achieve a given + rating, a workflow engine must demonstrate that it complies with each of the criteria specified.
- To achieve a +/- rating it must satisfy at least one of the criteria listed.
- Otherwise a - rating is recorded.

Product/Language	Version	Score	Motivation
Staffware	9	+/-	Only exclusive choice supported
Websphere MQ Workflow	3.4	+	Directly supported via transition conditions and start conditions
FLOWer	3.0	+/-	Choices can depend on data but only exclusive choice supported
COSA	4.2	+	Supported by conditions on input and output arcs. Both exclusive choice and multi-choice supported
XPDL	1.0	+	Through transition conditions and transition restrictions. Both exclusive choice and multi-choice supported
BPEL4WS	1.1	+	Both exclusive choice and multi-choice supported
BPMN	1.0	+	Supported, as Condition Expressions are possible to specify for Sequence Flows
UML	2.0	+	Supported via the DecisionNode construct and guard conditions on ActivityEdges.

Oracle BPEL	10.1.2	+	Directly supported via links and <switch>
-------------	--------	---	---

## Summary of Evaluation

+ Rating	+/- Rating
<ol style="list-style-type: none"><li>1. Any data element accessible at case level can be utilised in a routing construct</li><li>2. Direct workflow support</li><li>3. Support for both exclusive choice and multi-choice constructs</li></ol>	<ol style="list-style-type: none"><li>1. Limited range of data elements can be used in routing constructs</li><li>2. Limited range of routing constructs supported</li><li>3. Programmatic extensions required</li></ol>

© 2007 *Workflow Patterns Initiative*

0025700

[Patterns](#) | [Evaluations](#) | [Vendors](#) | [About](#) |

[Impact](#)

[YAWL](#) | [Links](#) | [Documentation](#) | [Contacts](#) | [Site](#)

[Map](#)

For any problems or questions, please [contact us](#)

**Webmaster:** Jessica Prestedge  
([webmaster@workflowpatterns.com](mailto:webmaster@workflowpatterns.com))