CommerceQuest

ENABLING THE DYNAMIC ENTERPRISE

Workflow Pattern Analysis

of

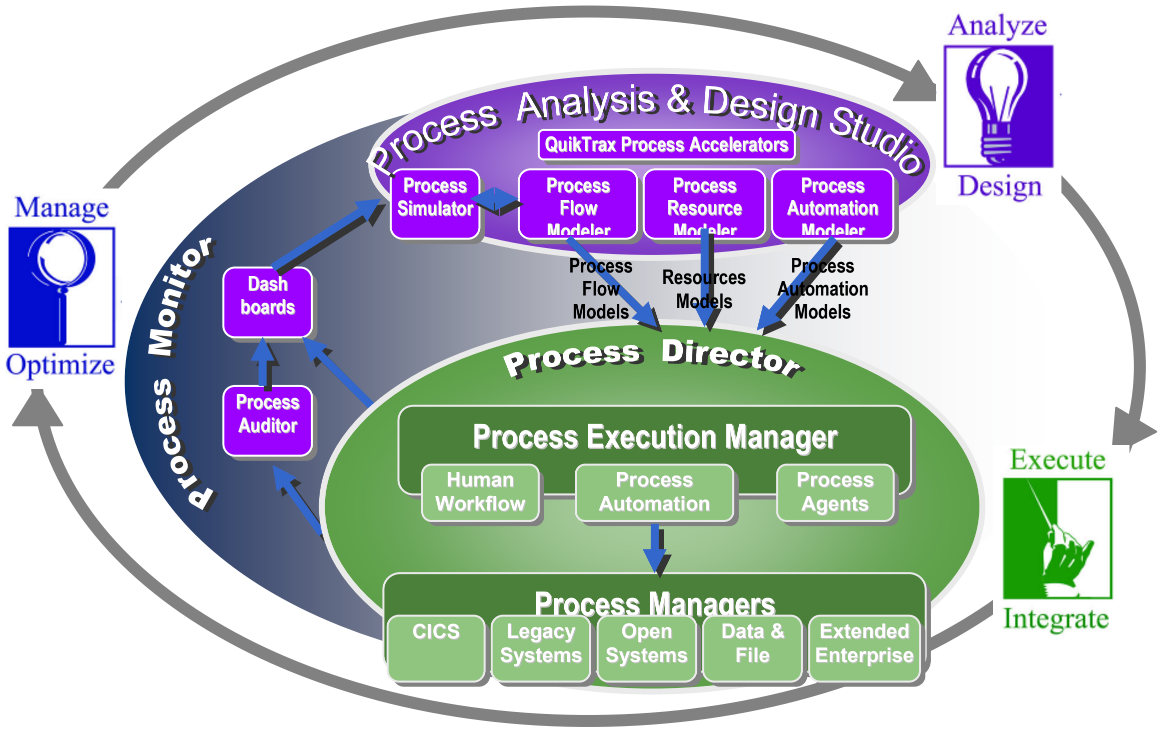
TRAXION BPM Suite

Prepared by: P. Eksteen (pieter.eksteen@commercequest.com)
April 2004
Version 1.1

TABLE OF CONTENTS

Introduction

CommerceQuest's Business Process Management Suite (TRAXION BPMS) allows companies to analyze, design, execute, integrate, manage and optimize processes in a service-based architecture for the real-time enterprise. TRAXION BPMS is a powerful multi-platform solution for business process and resource collaboration. It acts as both a strategic, comprehensive infrastructure for a business' process strategy and as a series of tactical, incremental solutions that enable customers to improve performance today in a risk-free environment. CommerceQuest's TRAXION BPMS is seamlessly integrated to ensure frictionless and rapid deployment of business processes.



Capabilities include:

- Analysis and Design – TRAXION BPMS offers a highly intuitive set of process tools via "The Process Analysis and Design Studio." The tools allow for highly flexible and customized business process and resource modeling analysis.
- Execution and Integration -- TRAXION BPMS offers a dynamic set of tactical capabilities via its "Process Director" toolset. Process Director contains all the tools required to reach out to, extract, and manage the coordination of enterprise assets, people, systems and resources.
- Management and Optimization -- TRAXION BPMS offers several powerful tools including, "Process Monitor" and "Process Simulator" that allows for continuous improvement opportunities via real-time access to information across an organization's activities and operations.

The BPMS is underscored by a range of Process Accelerators that allow an organization to implement a new process quickly and effectively, or serve as a benchmark against which to measure their own processes.

TRAXION Principles

TRAXION BPMS defines workflow as a set of Activities joined by Transitions. Activities consist of a task or tasks carried out by a resource. Resources can be defined as people or system(s). Human resources are defined in terms of their position in a hierarchy e.g. departments, divisions; their participation as a member of a workgroup; any roles that they might fulfil; and in terms of their cost and availability. System resources can consist of on of the following:

- of a coded 'agent'
- an external system resource accessed through the integration infrastructure.
- another process (either contained within the current engine or another engine).

Activities also have data fields associated with them. These fields are updated by their defining activity or subsequent activities, and are used in the transitions to implement conditions under which the transitions happen.

Upon completion of an activity routing algorithms interrogate each of the (one or more) transitions from that activity, determining which path or paths to follow in the process 'map'. It is possible to implement coded overrides before the routing in order to influence the routing algorithm.

Transitions are links between activities that can have associated conditions. These conditions determine whether or not that link should be traced to its destination activity. It is also possible to implement coded post-routing logic.

Activities can have any number of transitions coming to them (merge/join), and any number of transitions to other subsequent activities (splits). Merges (synchronization) is the default behaviour when more than one transition is made to an activity. If the 'Never Merge' flag is set on the activity – it will be spawned (with its associated subsequent activities) for each incoming transition that is completed. If no conditions for routing can be met, or there are no subsequent transitions from an activity – the process will end (termination of a process is always implicit).

Scoring TRAXION against the patterns.

The analysis of the workflow patterns in relation to the TRAXION product results in 3 possible results:

Directly supported

The product supports the definition of the pattern both in terms of modelling and execution.

Supported at runtime only

The product does not have any modelling constructs to describe the pattern, although the runtime can be used to support the functionality. A short description of how this might be achieved is supplied.

Not supported

The product does not support the pattern at all. This result is sometimes accompanied by a short description of our implementation of the feature, and any similarly named features are described to minimise confusion.

Workflow Patterns

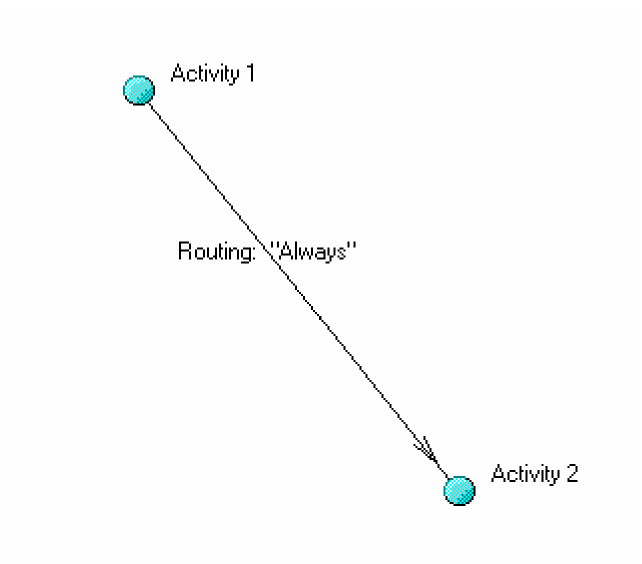
Pattern 1 (Sequence)

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

Synonyms: Sequential routing, serial routing.

TRAXION Support: Directly supported

Support is provided by means of the linking of two activities.



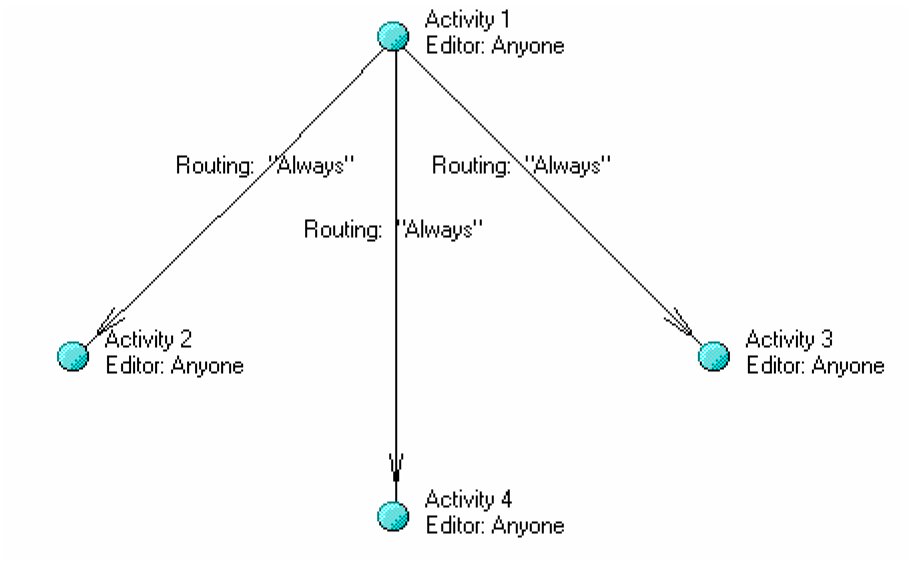
Pattern 2 (Parallel Split)

Description: A point in the workflow process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.

Synonyms: AND-split, parallel routing, fork.

TRAXION Support: Directly supported.

This is supported via the ability to support by linking the source activity to multiple destination activities – each of which is activated in parallel.



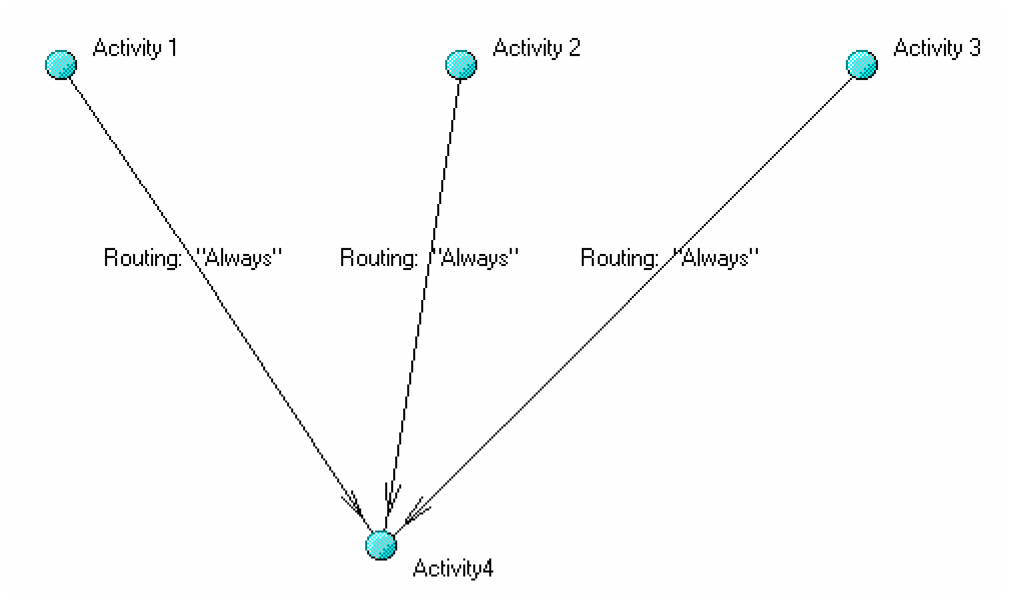
Pattern 3 (Synchronization)

Description: A point in the workflow process where multiple parallel sub-processes/activities converge into one single thread of control, thus synchronizing multiple threads. It is an assumption of this pattern that each incoming branch of a synchronizer is executed only once (if this is not the case, then see Patterns 13-15 (Multiple Instances Requiring Synchronization)).

Synonyms: AND-join, rendezvous, synchronizer.

TRAXION Support: Directly supported.

Synchronization is supported by linking multiple source activities to a single destination. Control over associated data can be determined at design or run-time, by defining precedence for each item of associated data.



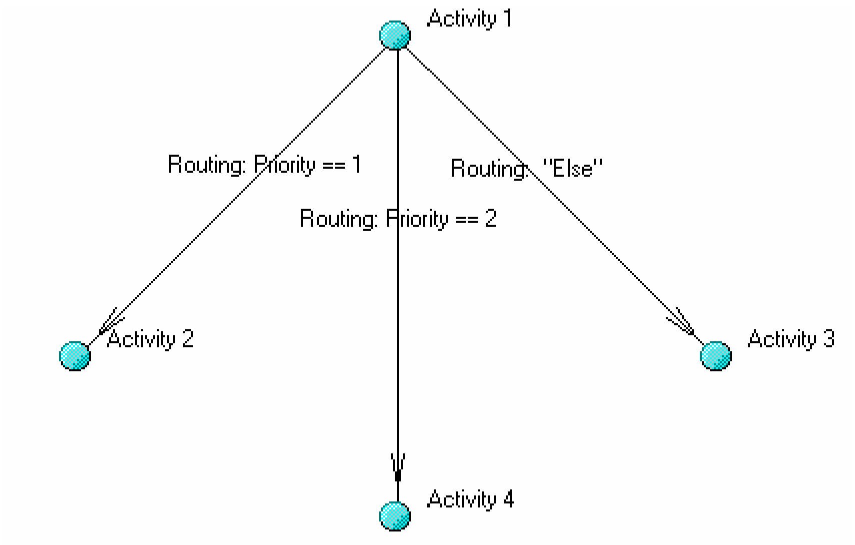
Pattern 4 (Exclusive Choice)

Description: A point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen.

Synonyms: XOR-split, conditional routing, switch, decision.

TRAXION Support: Directly supported

This pattern is supported via conditional routing within the TRAXION product. As many conditional routings can be applied – with one (optional) 'else' route if none of the others can be satisfied.



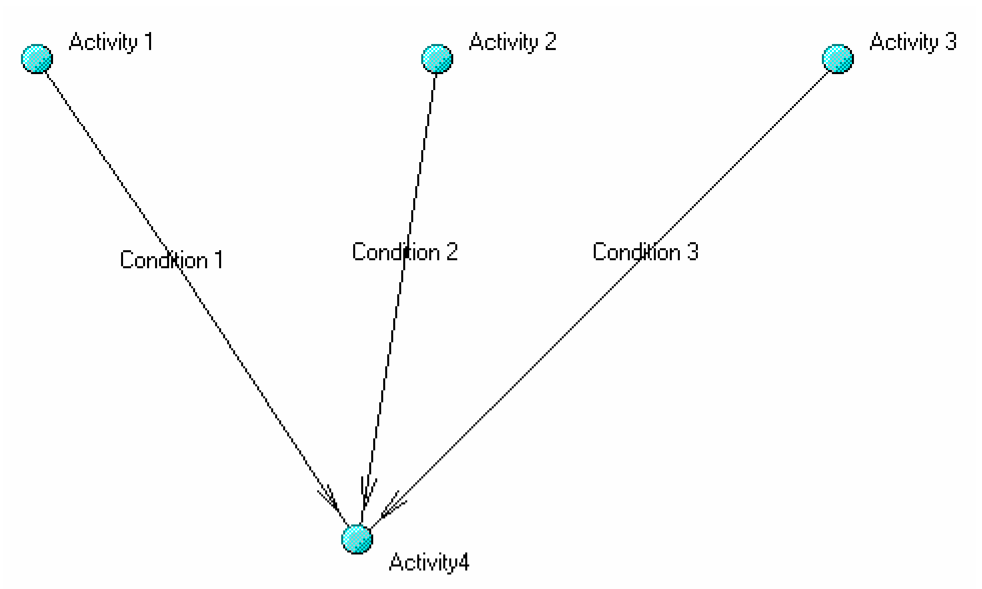
Pattern 5 (Simple Merge)

Description: A point in the workflow process where two or more alternative branches come together without synchronization. It is an assumption of this pattern that none of the alternative branches is ever executed in parallel (if this is not the case, then see Pattern 8 (Multi-merge) or Pattern 9 (Discriminator)).

Synonyms: XOR-join, asynchronous join, merge.

TRAXION Support: Directly supported

This pattern is supported through the ability of the engine to evaluate whether or not any other link will ever be activated. The first time a link is made to Activity 4, it checks to see if any of the transitions on the remaining inbound links were satisfied - if not – activity 4 is executed.



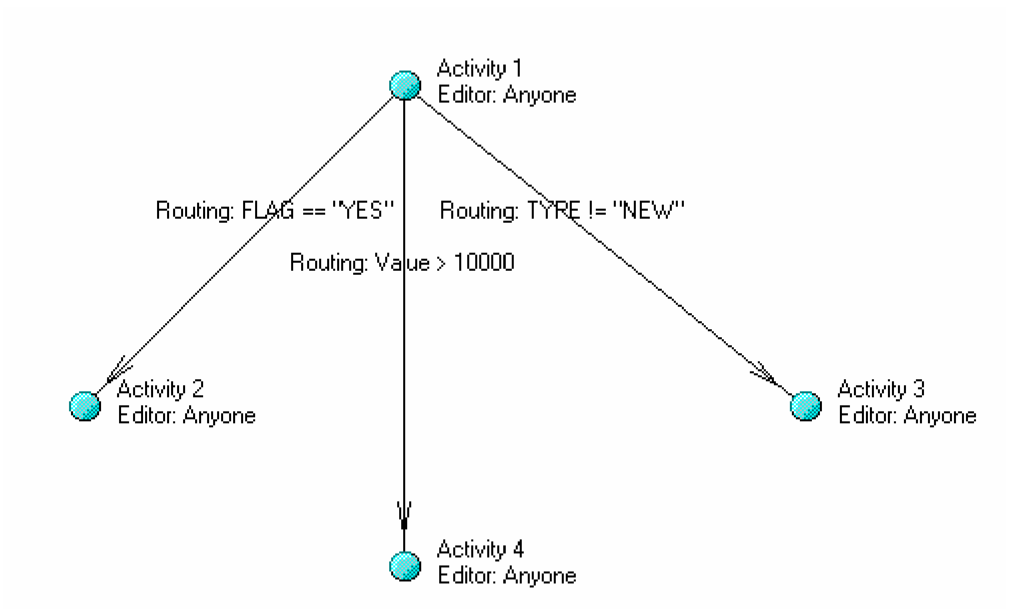
Pattern 6 (Multi-choice)

Description: A point in the workflow process where, based on a decision or workflow control data, a number of branches are chosen.

Synonyms: Conditional routing, selection, OR-split.

TRAXION Support: Directly supported.

This pattern is supported through the ability to specify individual conditions on each link between activities.



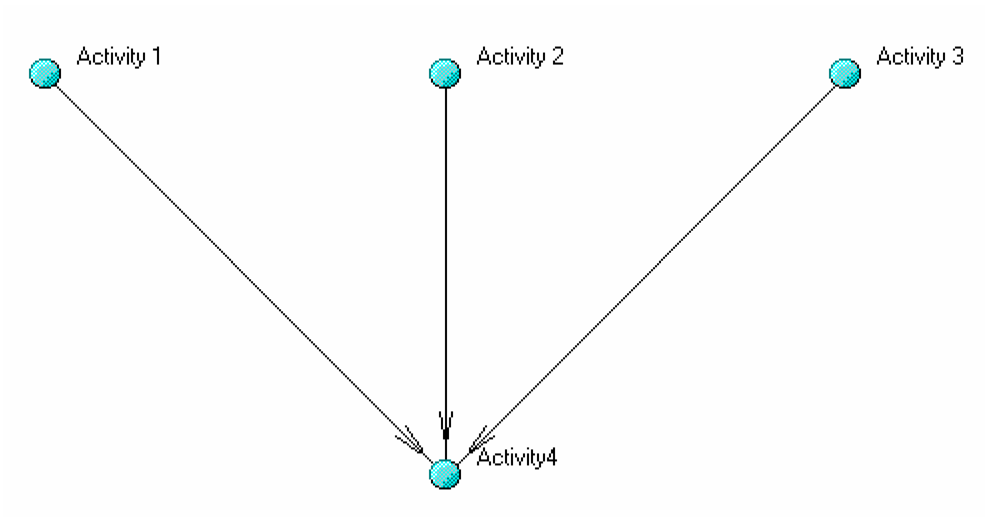
Pattern 7 (Synchronizing Merge)

Description: A point in the workflow process where multiple paths converge into one single thread. If more than one path is taken, synchronization of the active threads needs to take place. If only one path is taken, the alternative branches should re-converge without synchronization. It is an assumption of this pattern that a branch that has already been activated, cannot be activated again while the merge is still waiting for other branches to complete.

Synonyms: Synchronizing join.

TRAXION Support: Directly supported.

This pattern is implicitly supported by default (see Pattern 5 above). Each and every incoming link is synchronized, unless it was / will never be activated. Synchronization (and activation of Activity 4) depends on all (in this case) 3 activities being completed or excluded from ever being activated higher up the process map. This is the default behavior (Never Merge not active).

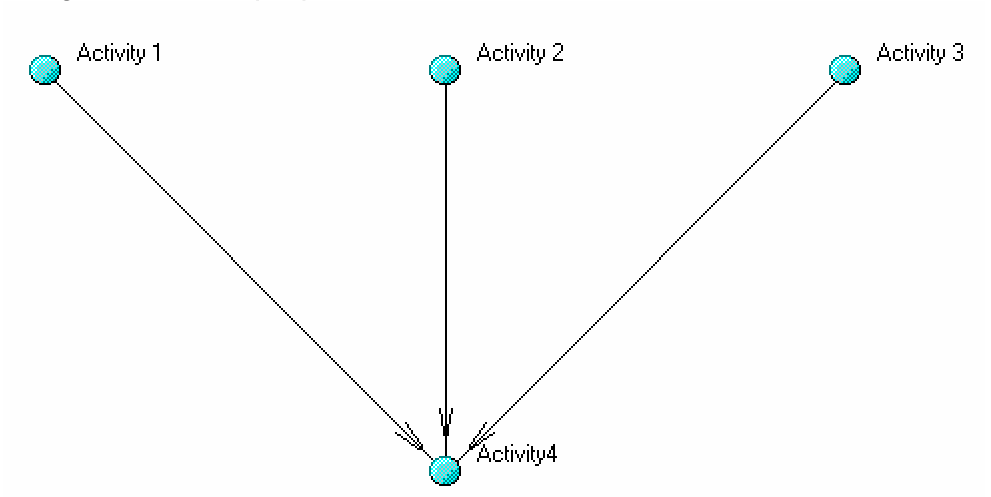


Pattern 8 (Multi-merge)

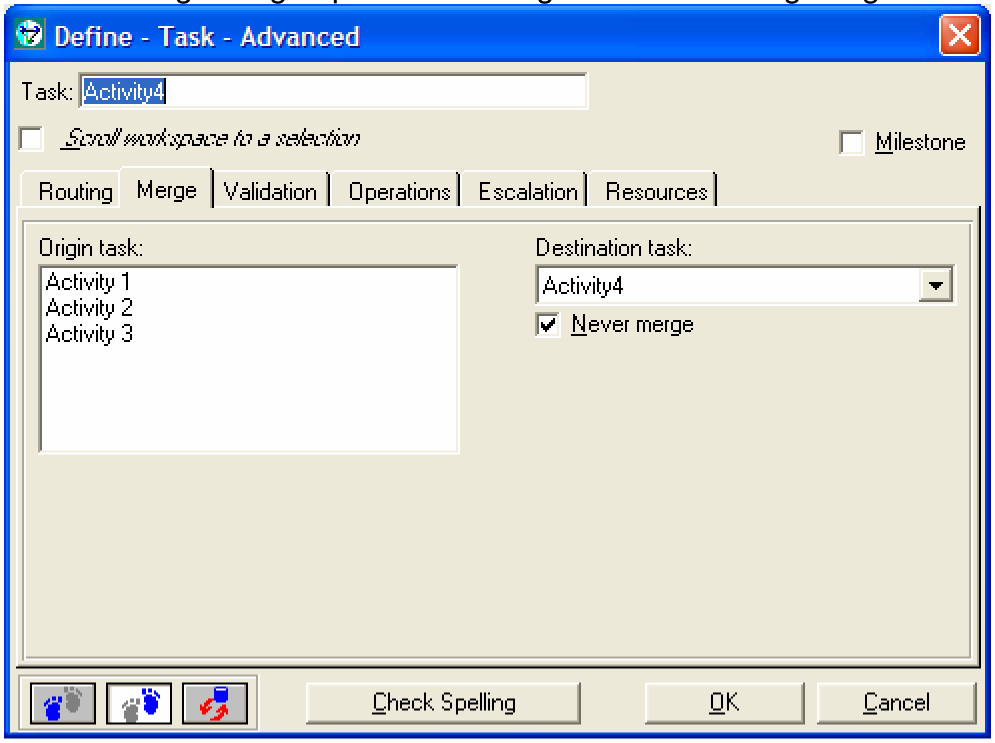
Description: A point in a workflow process where two or more branches re-converge without synchronization. If more than one branch gets activated, possibly concurrently, the activity following the merge is started for every activation of every incoming branch.

TRAXION Support: Directly supported.

This pattern is supported by flagging in the Activity 4 in the example as 'Never merge' within the properties of the destination.



The following dialog depicts the setting of a 'never' merge flag.



Task: Activity4

[ ] Milestone

Origin task:
Activity 1
Activity 2
Activity 3

Destination task: Activity4

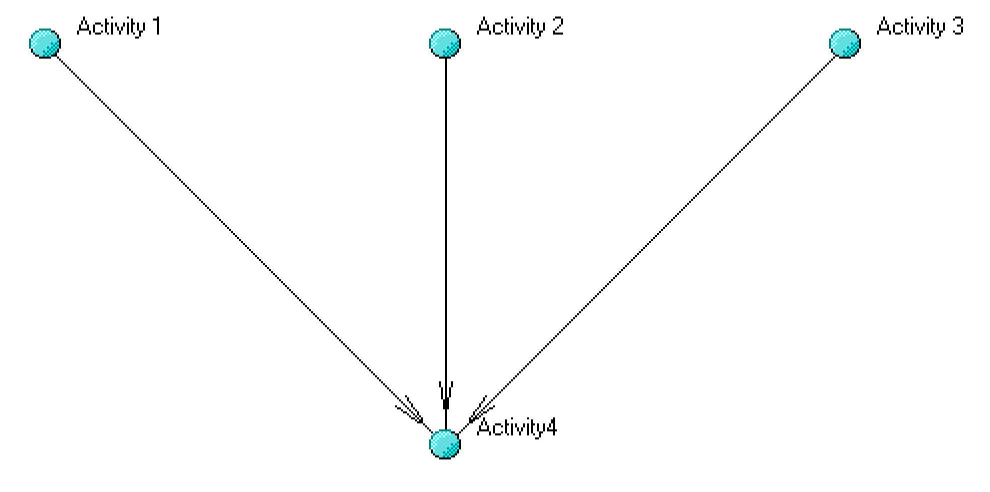
[✓] Never merge

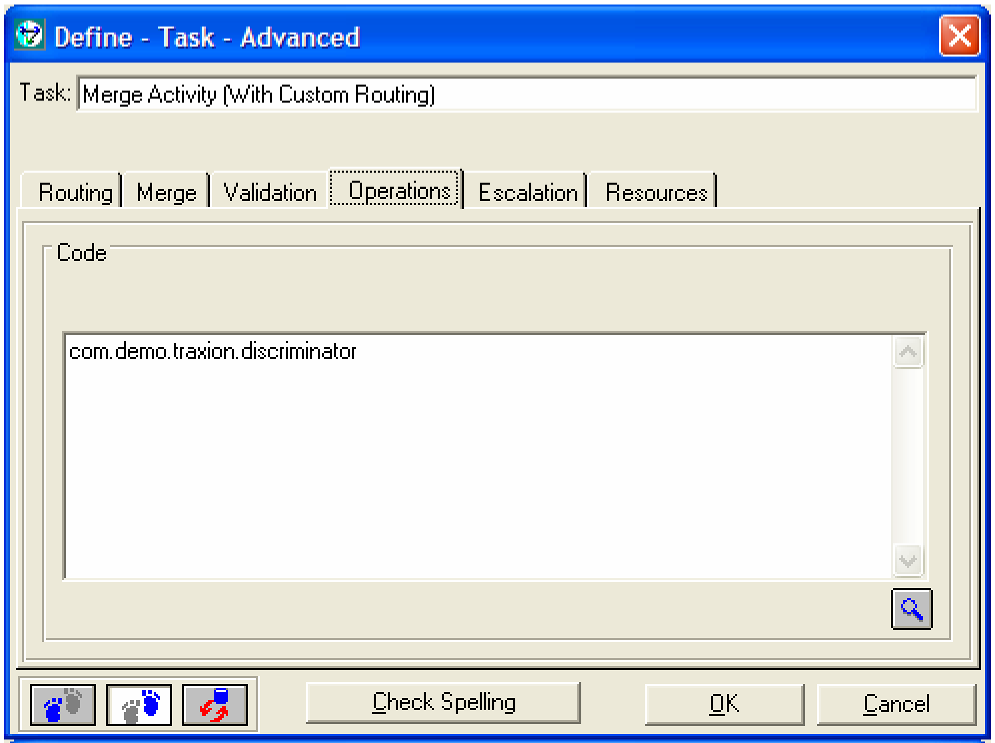
Pattern 9 (Discriminator)

Description: The discriminator is a point in a workflow process that waits for one of the incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to complete and ignores them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again (which is important otherwise it could not really be used in the context of a loop).

TRAXION Support: Not supported.

This pattern can only be implemented by putting an additional activity (with Never Merge enabled) into the flow which has a custom routing operation which precludes a second instance being fired. This will have the effect of taking firing Activity 4 for each incoming link – except that the custom routing would track when Activity 4 has already been fired – and prevent the subsequent (in time) links from firing it again.

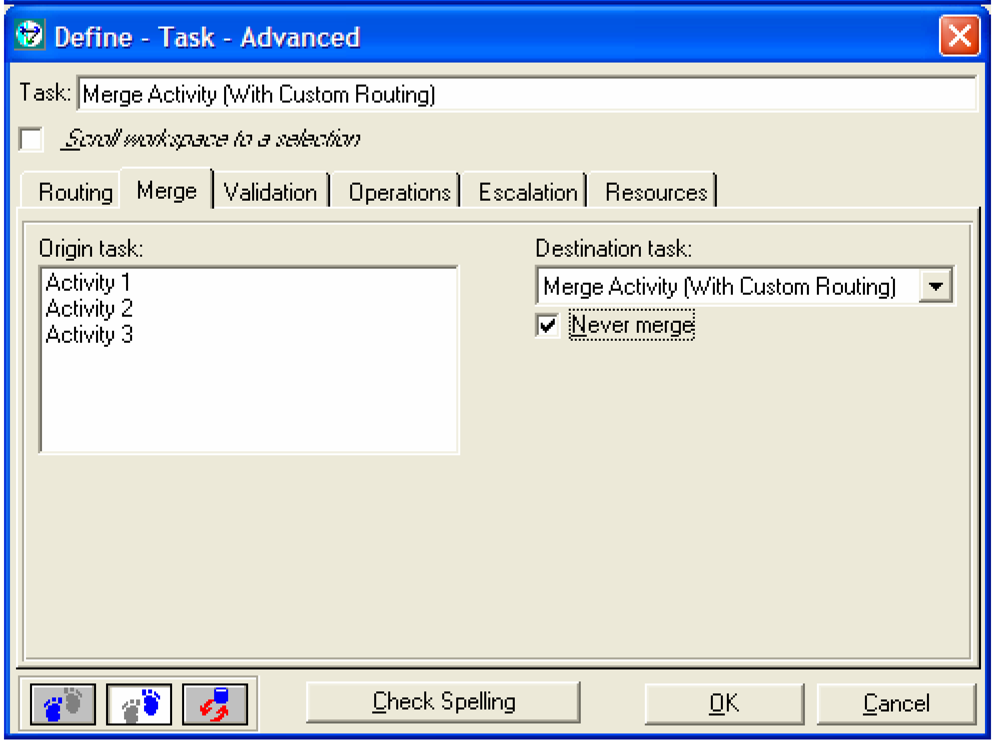



Define - Task - Advanced

Task: Merge Activity (With Custom Routing)

Code

```
com.demo.traxion.discriminator
```


Define - Task - Advanced

Task: Merge Activity (With Custom Routing)

☐ Scroll workspace to a selection

Origin task:
Activity 1
Activity 2
Activity 3

Destination task: Merge Activity (With Custom Routing)

☑ Never merge

Pattern 9a (N out of M join)

Description: N-out-of-M Join is a point in a workflow process where M parallel paths converge into one. The subsequent activity should be activated once N paths have completed. Completion of all remaining paths should be ignored. Similarly to the discriminator, once all incoming branches have "fired", the join resets itself so that it can fire again.

TRAXION Support: Not supported.

This pattern could, like Pattern 9: Discriminator, be supported by the use of an additional ‘auto-complete activity’ with custom routing. The custom routing would count the number of completed tasks, and allow the routing to complete, whilst suppressing subsequent completions from triggering an additional instance.

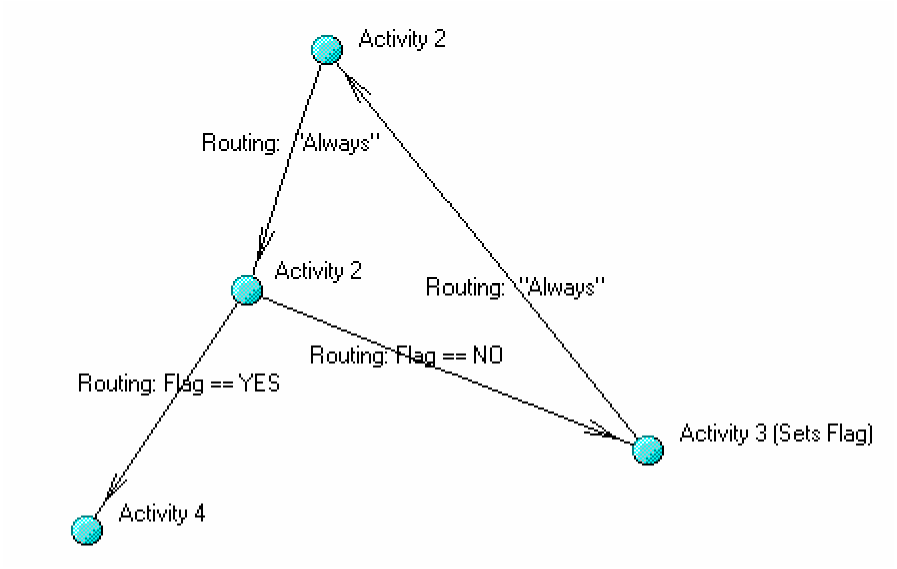
Pattern 10 (Arbitrary Cycles)

Description: A point in a workflow process where one or more activities can be done repeatedly.

Synonyms: Loop, iteration, cycle.

TRAXION Support: Supported through choice routing and repeat of tasks.

This is supported through use of the choice routing of tasks. Evaluation of the choice is made at each activation of the link. Support for looping is dependant on synchronisation issues. When used with Never Merge disabled, the possibility of 'hung' processes will arise. The Structure and Integrity checks carried out by the Process Designer will highlight possible structures where such a condition may arise.

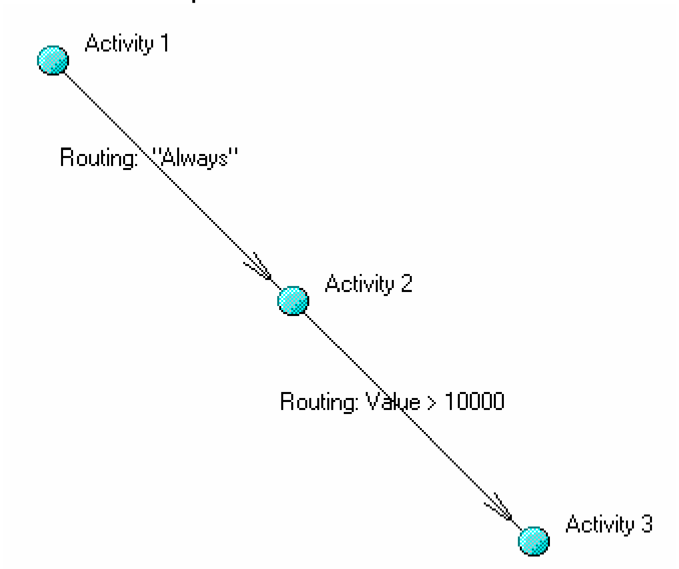


Pattern 11 (Implicit Termination)

Description: A given sub-process should be terminated when there is nothing else to be done. In other words, there are no active activities in the workflow and no other activity can be made active (and at the same time the workflow is not in deadlock).

TRAXION Support: Directly supported.

This pattern is supported by virtue of the fact that transitions are optional, and if there are no further transitions, or none of the existing transitions condition can be met – the process is terminated.



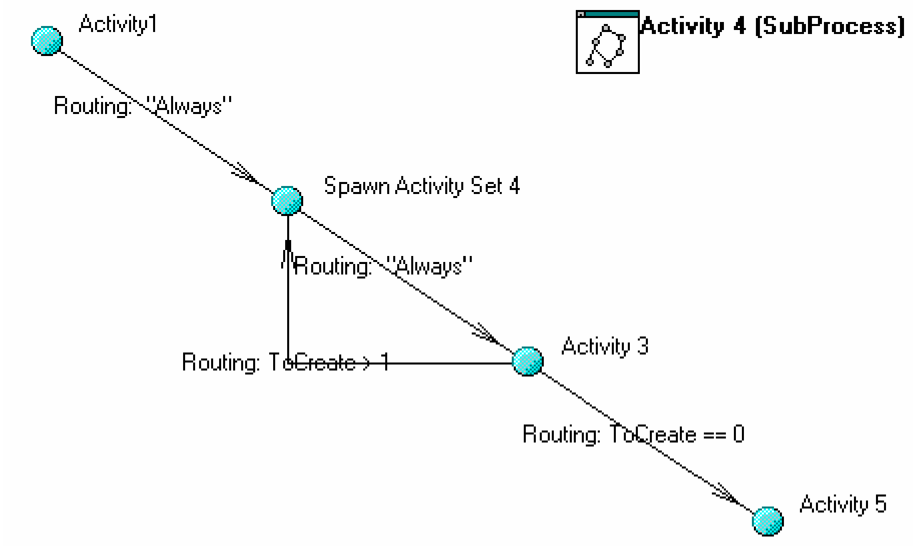
In this example Activity 3 will only be fired if the condition is met – otherwise the process will terminate at Activity 2.

Pattern 12 (Multiple Instances without Synchronization)

Description: Within the context of a single case (i.e., workflow instance) multiple instances of an activity can be created, i.e., there is a facility to spawn off new threads of control. Each of these threads of control is independent of other threads. Moreover, there is no need to synchronize these threads.
Synonyms: Multi threading without synchronization, Spawn

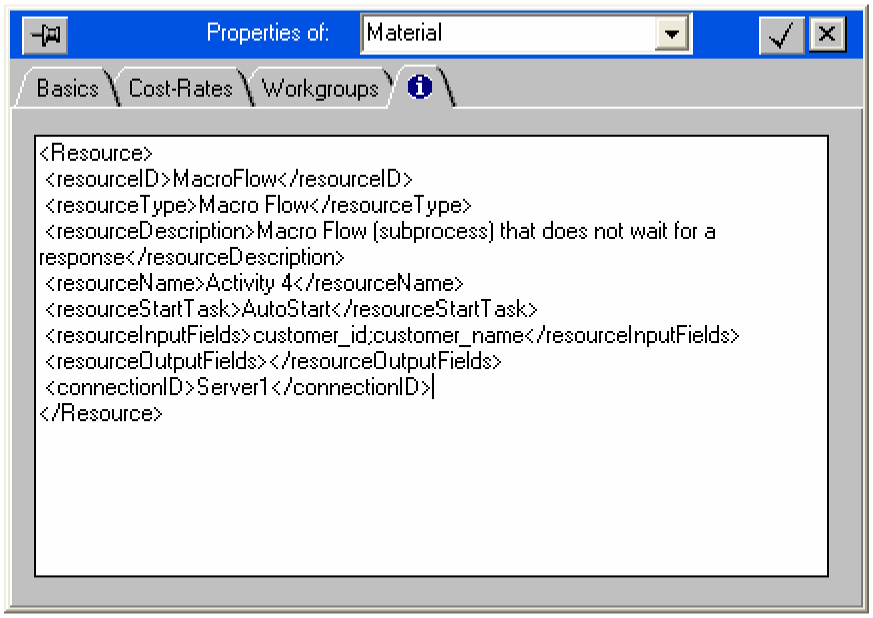
TRAXION Support: Directly supported.

This pattern is supported through a *Macro Flow without Response* Connector – a special resource type within the TRAXION BPMS system. Essentially this allows one to spawn a sub-process and continue evaluation of any following links - without waiting for a response.



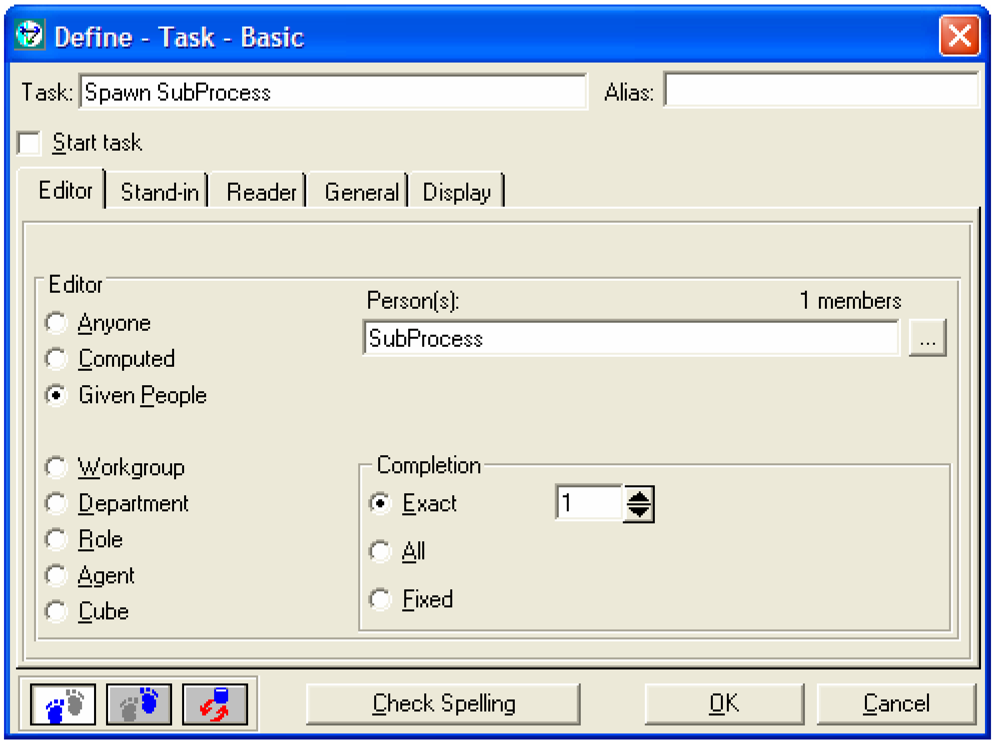
The concept of using a resource to instantiate additional process instances (or sub-processes) is a relatively new addition to the execution engine. At present the modelers (both Flow and Resource) support for defining these explicitly is rudimentary.

The following dialog depicts an example of a Resource definition that is used to spawn a new process. The sub-process can consist of a single activity, or a whole new process.



```
<Resource>
 <resourceID>MacroFlow</resourceID>
 <resourceType>Macro Flow</resourceType>
 <resourceDescription>Macro Flow (subprocess) that does not wait for a
response</resourceDescription>
 <resourceName>Activity 4</resourceName>
 <resourceStartTask>AutoStart</resourceStartTask>
 <resourceInputFields>customer_id;customer_name</resourceInputFields>
 <resourceOutputFields></resourceOutputFields>
 <connectionID>Server1</connectionID>
</Resource>
```

The support for ‘Special Material’ resources is incomplete in the modelers (hence the use of the ‘Material’ type of resource. Within the Process Flow Modeler, the Material is referenced as an individual person:



Task: Spawn SubProcess
Alias:
☐ Start task

Editor: Anyone [✓ Given People] Computed Workgroup Department Role Agent Cube
Person(s): SubProcess — 1 members
Completion: [✓ Exact] 1 All Fixed

Pattern 13 (Multiple Instances with a Priori Design Time Knowledge)

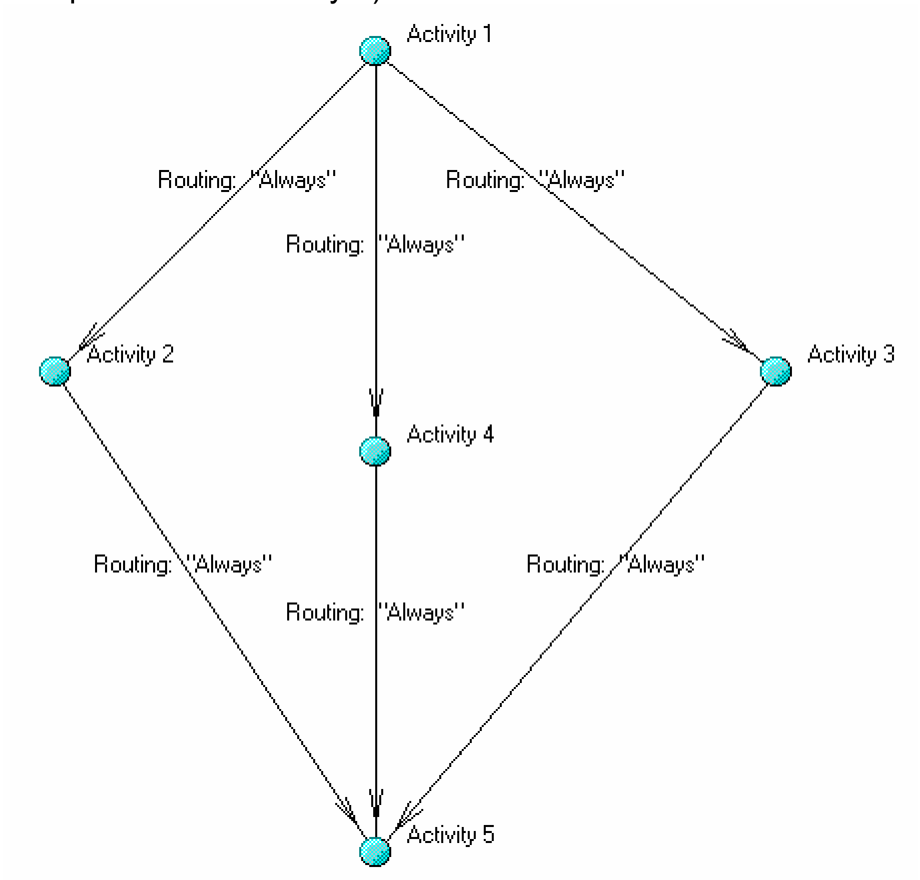
Description: For one process instance an activity is enabled multiple times. The number of instances of a given activity for a given process instance is known at design time. Once all instances are completed some other activity needs to be started.

TRAXION Support: Directly supported

This pattern is supported through 2 mechanisms.

1. By creating multiple links to any number of activities – all of which will be started simultaneously.
2. By utilizing either of the *Macro Flow* Connectors (*with and without Response).* By simply creating as many links to activities utilizing the Macro Flow resource – any number of instances of sub-process (series of activities) can be started. These can either return (for synchronization) or not – depending on the variant utilized.

Upon completion they will (depending on the conditions associated with the transitions branching from them) cause Activity 5 to be triggered once they have all completed (synchronized); or will each trigger Activity 5 (Never Merge option set on Activity 5).

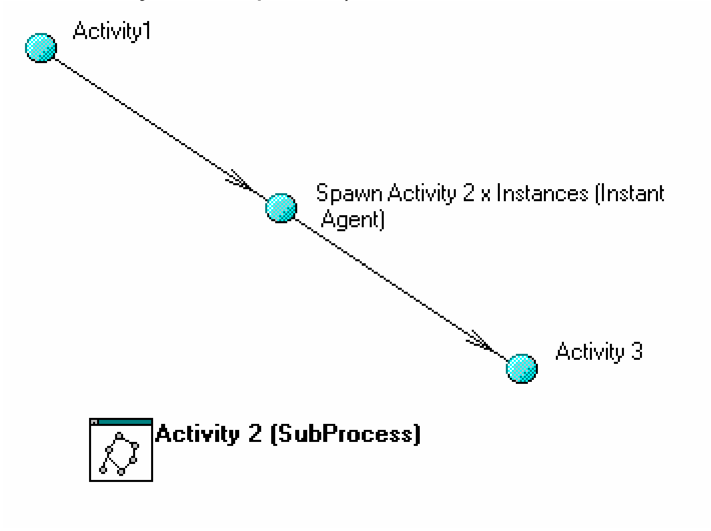


Pattern 14 (Multiple Instances with a Priori Runtime Knowledge)

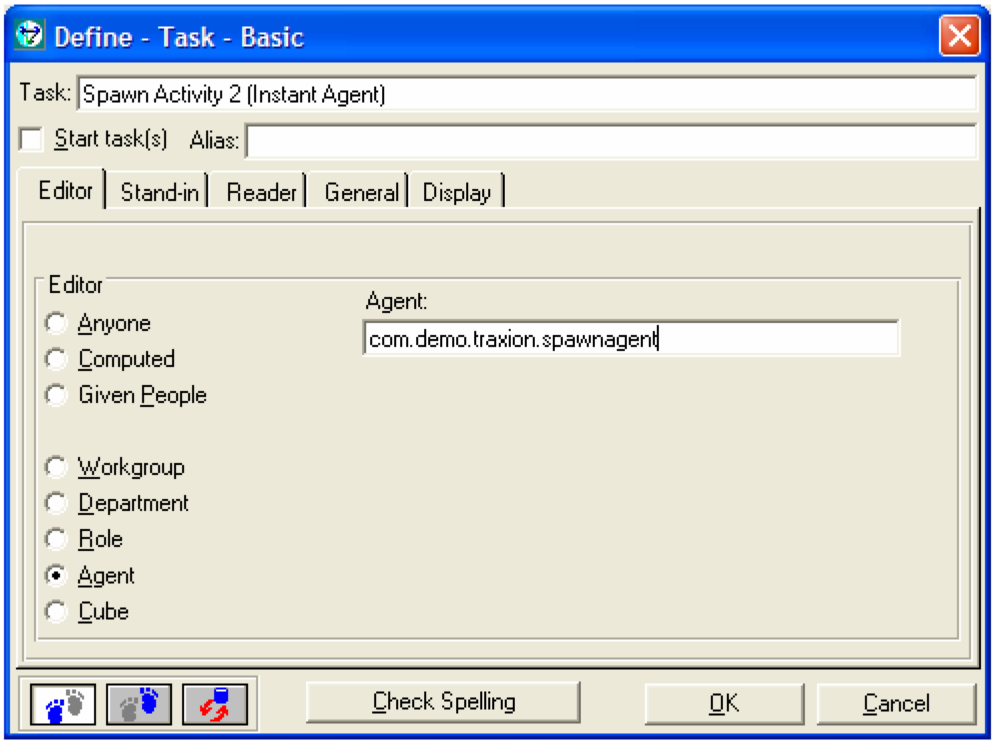
Description: For one case an activity is enabled multiple times. The number of instances of a given activity for a given case varies and may depend on characteristics of the case or availability of resources [CCPP98, JB96], but is known at some stage during runtime, before the instances of that activity have to be created. Once all instances are completed some other activity needs to be started.

TRAXION Support: Supported at runtime only.

This pattern can be implemented in one of 2 ways: by writing a custom Instant Agent with Response that utilizes the API to spawn as many instances of a sub-process as are determined by logic based on the process data; and using the Macro Flow with Response resources to specify a process to run (analogous to that described in Pattern 12 – except that the sub-process must complete before the activity is completed).



Option 1 is described in the following dialog.


Define - Task - Basic

Task: Spawn Activity 2 (Instant Agent)

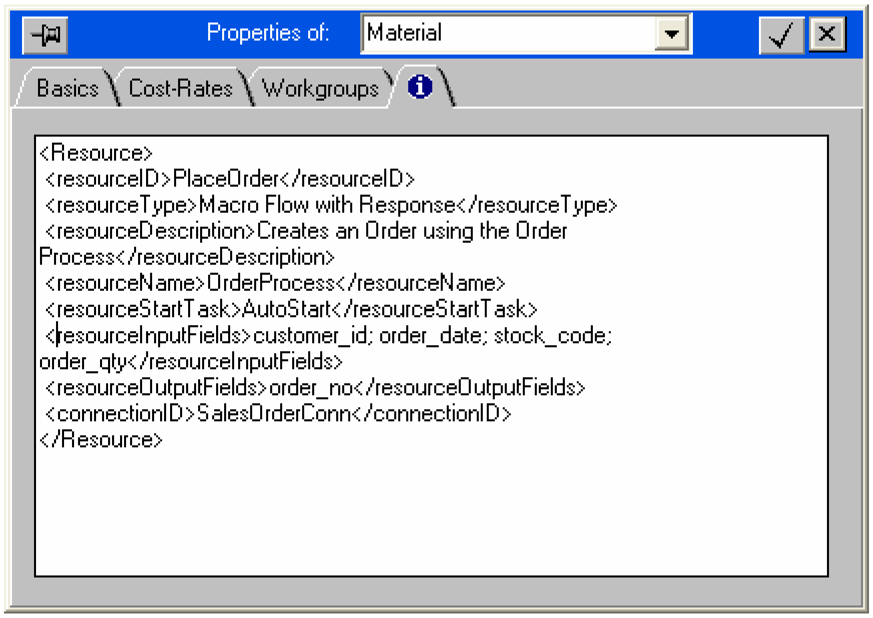
Start task(s) Alias:

Editor

( ) Anyone
( ) Computed
( ) Given People
( ) Workgroup
( ) Department
( ) Role
[✓ Agent]
( ) Cube

Agent: com.demo.traxion.spawnagent

Option 2 is described in the following resource definition:


Properties of: Material

```
<Resource>
 <resourceID>PlaceOrder</resourceID>
 <resourceType>Macro Flow with Response</resourceType>
 <resourceDescription>Creates an Order using the Order
Process</resourceDescription>
 <resourceName>OrderProcess</resourceName>
 <resourceStartTask>AutoStart</resourceStartTask>
 <resourceInputFields>customer_id; order_date; stock_code;
order_qty</resourceInputFields>
 <resourceOutputFields>order_no</resourceOutputFields>
 <connectionID>SalesOrderConn</connectionID>
</Resource>
```

Again the material resource would be specified as a ‘person’ – due to the runtime engine outstripping the modelers in terms of functionality.

Pattern 15 (Multiple Instances without a Priori Runtime Knowledge)

Description: For one case an activity is enabled multiple times. The number of instances of a given activity for a given case is not known during design time, nor is it known at any stage during runtime, before the instances of that activity have to be created. Once all instances are completed some other activity needs to be started. The difference with Pattern 14 is that even while some of the instances are being executed or already completed, new ones can be created.

TRAXION Support: Supported at runtime only.

This pattern is supported by option 1 described in pattern 14. By utilising the ability to synchronise the execution (Response from Process) or not synchronised (no Response from Process) it is possible to support all the possibilities. The Activity in the example shown in Pattern 13 is a sub-process consisting of 1 or more activities. The number of instances of this sub-process spawned can be determined programmatically inside the Agent (using workflow or external data).

Pattern 16 (Deferred Choice)

Description: A point in the workflow process where one of several branches is chosen. In contrast to the XOR-split, the choice is not made explicitly (e.g. based on data or a decision) but several alternatives are offered to the environment. However, in contrast to the AND-split, only one of the alternatives is executed. This means that once the environment activates one of the branches the other alternative branches are withdrawn. It is important to note that the choice is delayed until the processing in one of the alternative branches is actually started, i.e. the moment of choice is as late as possible.

Synonyms: External choice, implicit choice, deferred XOR-split.

TRAXION Support: Not supported.

Due to the lack of intrinsic support for discriminators, deferred choice is almost impossible to implement – even with the ability to execute activities via Web Services or the TRAXION API.

Pattern 17 (Interleaved Parallel Routing)

Description: A set of activities is executed in an arbitrary order: Each activity in the set is executed, the order is decided at run-time, and no two activities are executed at the same moment (i.e. no two activities are active for the same workflow instance at the same time).

Synonyms: Unordered sequence.

TRAXION Support: Not supported

There is no direct coordination between routes. To implement this would require heavy customisation for every activity and link in each of the streams of activity.

Pattern 18 (Milestone)

Description: The enabling of an activity depends on the case being in a specified state, i.e. the activity is only enabled if a certain milestone has been reached which did not expire yet.
Consider three activities named A, B, and C. Activity A is only enabled if activity B has been executed and C has not been executed yet, i.e. A is not enabled before the execution of B and A is not enabled after the execution of C. Figure 16 illustrates the pattern. The state in between B and C is modeled by place m. This place is a milestone for A. Note that A does not remove the token from M: It only tests the presence of a token.

Synonyms: Test arc, deadline (cf. [JB96]), state condition, withdraw message.

TRAXION Support: Not directly supported.

TRAXION supports milestones in terms of scheduling and prioritisation. An activity marked as a Milestone in the TRAXION product will get preference as far as resource allocation and scheduling is concerned. Tracking milestones to support this pattern can be implemented by ‘milestone flags’ as data within the process, and implementing custom routing code to interrogate these and route accordingly.

Pattern 19 (Cancel Activity)

Description: An enabled activity is disabled, i.e. a thread waiting for the execution of an activity is removed.

Synonyms: Withdraw activity.

TRAXION Support: Supported at runtime only.

The runtime engine allows for the cancellation of a single activity. The modeller does not implicitly allow or deny this on an activity by activity basis though – it is possible on any activity. This is implemented in the runtime engine by making use of the Process API.

Pattern 20 (Cancel Case)

Description: A case, i.e. workflow instance, is removed completely (i.e., even if parts of the process are instantiated multiple times, all descendants are removed).

Synonyms: Withdraw case.

TRAXION Support: Supported at runtime only.

This pattern is supported via the execution engine's ability to have a system administrator delete an entire process instance. Alternatively a process Supervisor can put a process instance on hold, and subsequently delete all activities that have occurred.

Both possibilities are implemented via the Process API.

References

Workflow Patterns

W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros
www.workflowpatterns.com