

Basic Control Patterns

Pattern 1: Sequence

Execute activities in sequence.

Sequence is the most basic workflow pattern. It is required when there is a dependency between two or more tasks so that one task cannot be started (scheduled) before another task is finished.



Description:

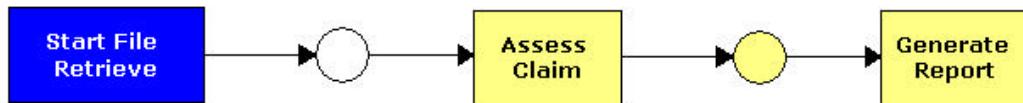
An activity in a workflow process is enabled after the completion of another activity in the same process.

Synonyms:

Sequential routing, serial routing.

Example:

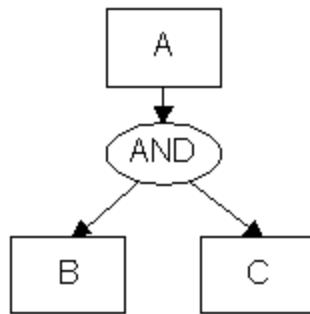
1. Activity send_bill is executed after the execution of activity send_goods.
2. An insurance claim is evaluated after the client's file is retrieved.
3. Activity add_air_miles is executed after the execution on activity book_flight.



Pattern 2: Parallel Split

Execute activities in parallel.

Parallel Split is required when two or more activities need to be executed in parallel. Parallel Split is easily supported by most workflow engines except for the most basic scheduling (timed) systems that do not require any degree of concurrency.



Description:

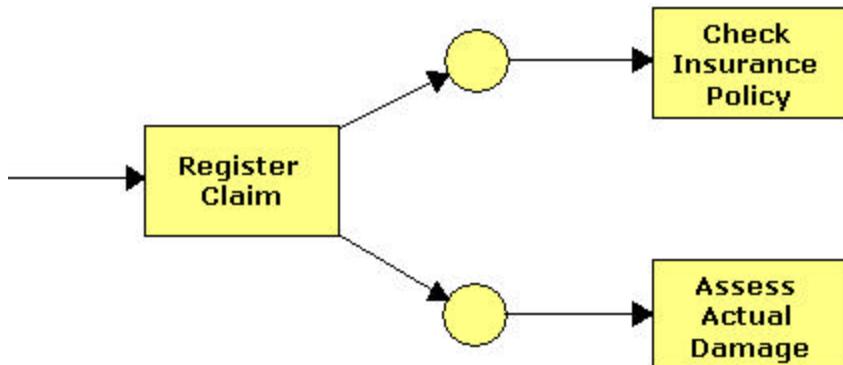
A point in the workflow process where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.

Synonyms:

AND-split, parallel routing, fork.

Example:

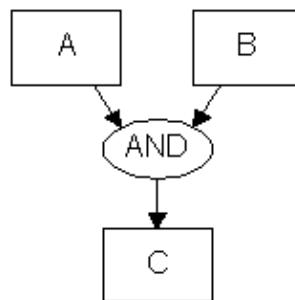
1. The execution of the activity *payment* enables the execution of the activities *ship_goods* and *inform_customer*.
2. After registering an insurance claim two parallel sub-processes are triggered: one for checking the policy of the customer and one for assessing the actual damage.



Pattern 3: Synchronization

Synchronize two parallel threads of execution.

Synchronization is required when an activity can be started only when two parallel threads complete.



Description:

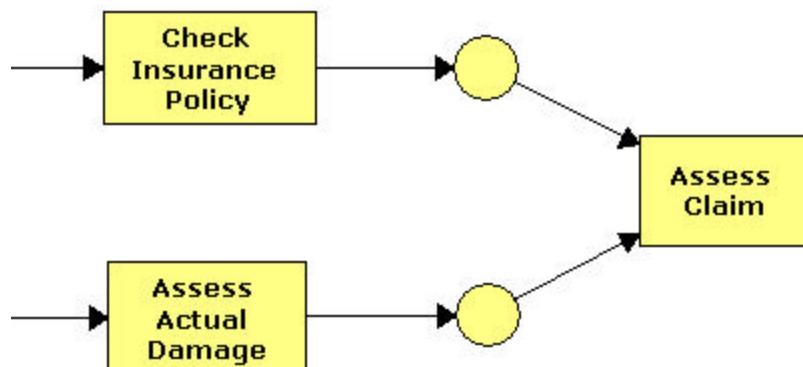
A point in the workflow process where multiple parallel sub-processes/activities converge into one single thread of control.

Synonyms:

AND-join, synchronizer.

Example:

1. Insurance claims are evaluated after the policy has been checked and the actual damage has been valuated.

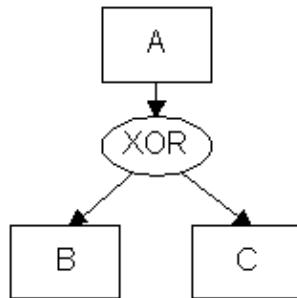


PECTRA Technology Inc.

Synchronization is made by using an identification number. The instances that were created from an AND Split have this same identification number, therefore in the synchronization, from the state before an activity, instances with the same identification number are selected. In a Join with synchronization the activity executes when there are tokens with the same identification number in all the states previous to that activity.

Pattern 4: Exclusive Choice

Choose one execution path from many alternatives.



Description:

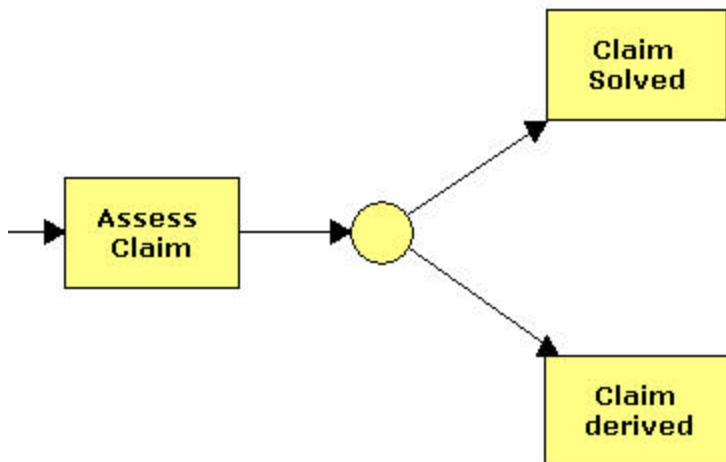
A point in the workflow process where based on a decision or workflow control data, one of several branches is chosen.

Synonyms:

XOR-split, conditional routing, switch, decision.

Example:

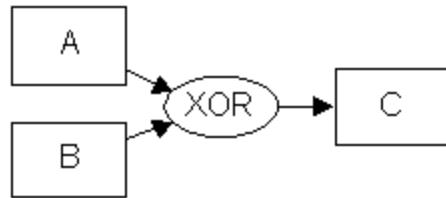
1. In the process *client_claim* two requirement types are evaluated and two options are possible: claim solved, or claim derived. The process will be split depending on the decision taken.



Pattern 5: Simple Merge

Merge two alternative execution paths.

Merge is required if we want to merge two alternative execution paths into one.



Description:

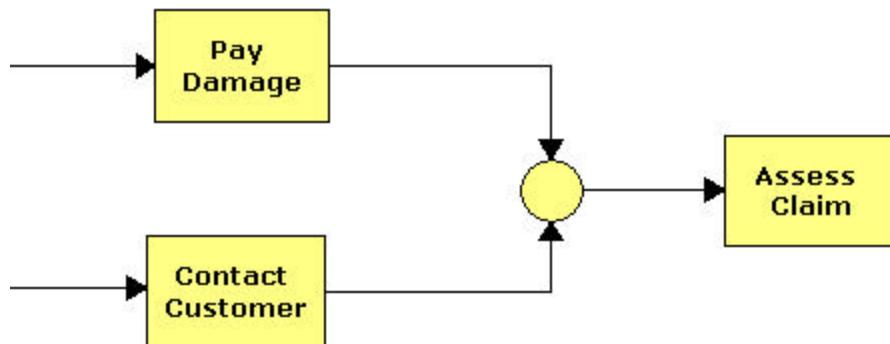
A point in the workflow process where two or more alternative branches come together without synchronization. In other words the merge will be triggered once any of the incoming transitions are triggered.

Synonyms:

XOR-join, asynchronous join, merge.

Example:

1. Activity *assess_claim* is enabled after either *pay_damage* or *contact_customer* is executed.

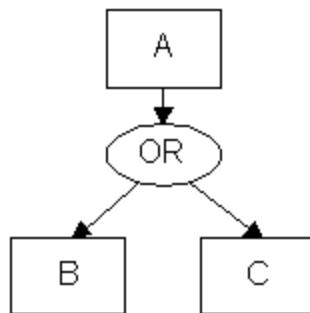


Advanced Branching and Synchronization Patterns

Pattern 6: Multiple Choice

Choose several execution paths from many alternatives.

Pattern Exclusive Choice assumes that exactly one of the alternatives is selected and executed, i.e. it corresponds to an exclusive OR. Sometimes it is useful to deploy a construct which can choose multiple alternatives from a given set of alternatives. Therefore, we introduce the (inclusive) multi-choice.



Description:

A point in the workflow process where based on a decision or workflow control data, one or more branches are chosen.

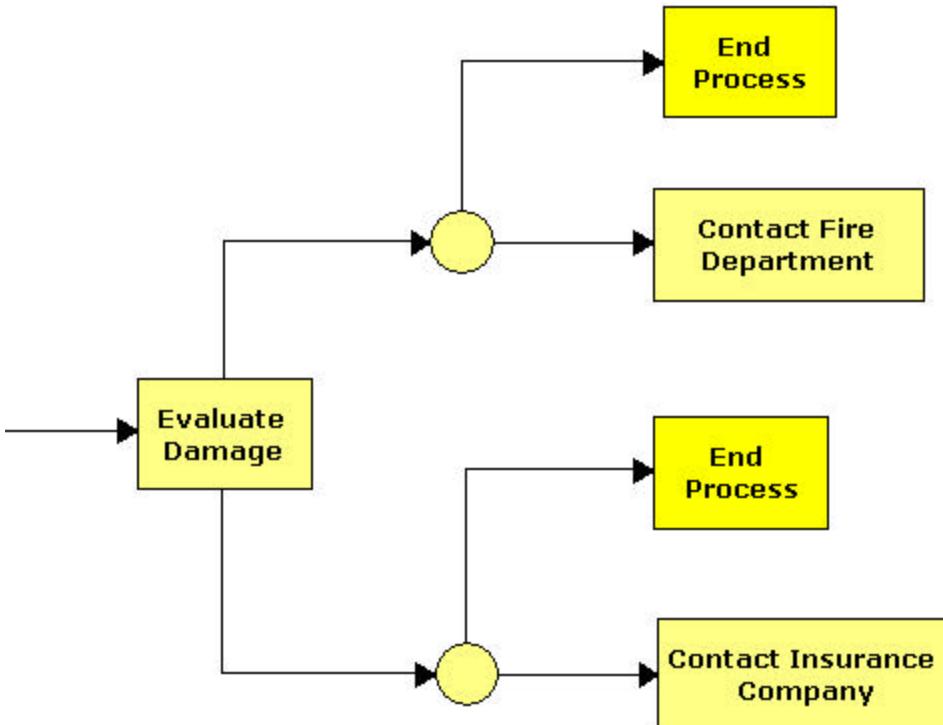
Synonyms:

Conditional routing, selection, OR-split.

Example:

- After executing the activity `evaluate_damage` the activity `contact_fire_department` or the activity `contact_insurance_company` is executed. At least one of these activities is executed. However, it is also possible that both need to be executed.

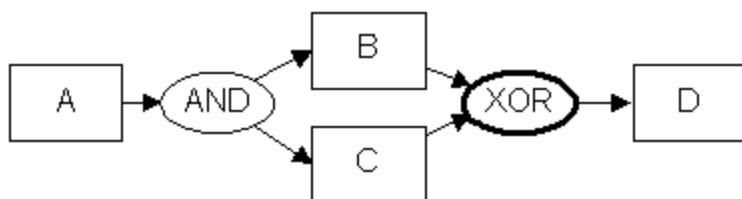




Pattern 7: Multiple Merge

Merge many execution paths without synchronizing.

This pattern aims to address the problem mentioned in Simple Merge. That is the situation when more than one incoming transition of a merge is being activated.

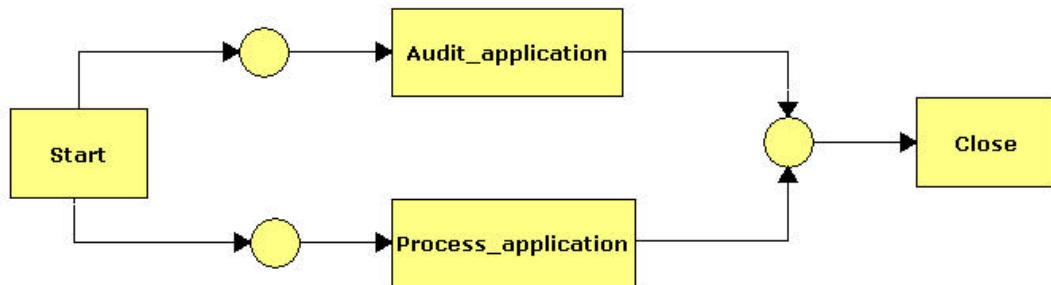


Description:

Multiple Merge is a point in the workflow process where two or more branches converge without synchronization. If more than one branch is executed, possibly at the same time, the activity that follows the merge starts every time for each incoming branch (i.e. in the figure above, D will be instantiated twice).

Example:

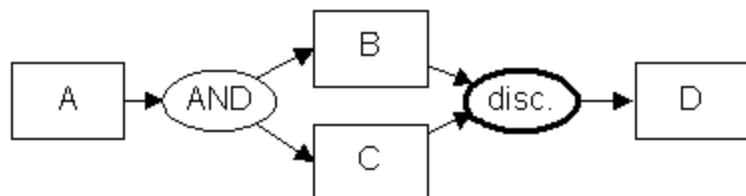
1. Sometimes two or more parallel branches share the same ending. Instead of replicating this (potentially complicated) process for every branch, a multi-merge can be used. A simple example of this would be two activities *audit_application* and *process_application* running in parallel which should both be followed by an activity *close_case*.



Pattern 8: Discriminator

Merge many execution paths without synchronizing. Execute the subsequent activity only once.

This pattern can be seen as the converse of the multi-merge. It should be implied when our semantics is that only one activity should be instantiated after merge.

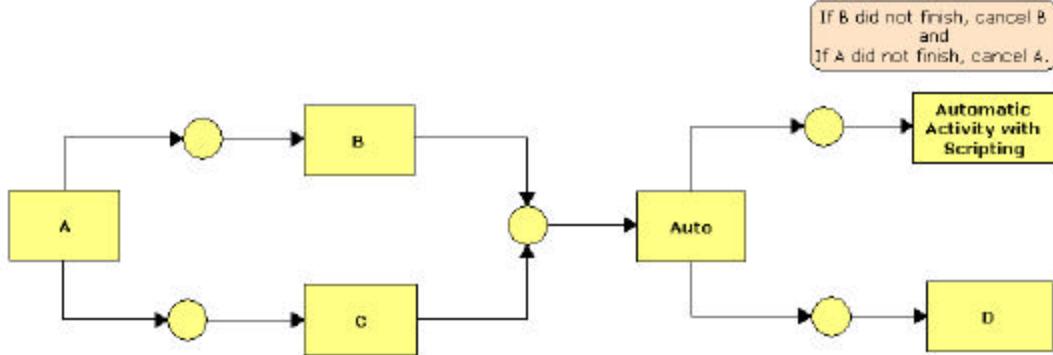


Description:

The discriminator is a point in a workflow process that waits for a number of incoming branches to complete before activating the subsequent activity. From that moment on it waits for all remaining branches to complete or ignores them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again.

Example:

1. To improve query response time, a complex search is sent to two different databases over the Internet. The first one that comes up with the result should proceed the flow. The second result is ignored.



Pattern 9: N-out-of-M Join

Merge many execution paths. Perform partial synchronization and execute subsequent activity only once.

The following pattern can be seen as a generalization of the basic Discriminator. We would like to synchronize N threads from M incoming transitions.

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.

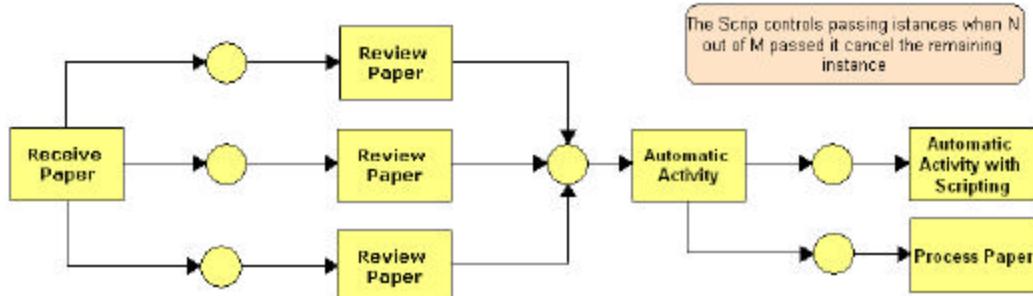
Synonyms:

Partial join, discriminator, custom join.

Example:

1. A paper needs to be sent to three external reviewers. Upon receiving two reviews the paper can be processed. The third review can be ignored.





Pattern 10: Synchronizing Join

Merge many execution paths. Synchronize if many paths are taken. Simple merge if only one execution path is taken.

The Multiple Choice pattern can be handled quite easily by today's workflow products. Unfortunately, the implementation of the corresponding merge construct (OR-join) is much more difficult to realize. The OR-join should have the capability to synchronize parallel flows and to merge alternative flows. The difficulty is to decide when to synchronize and when to merge. Synchronizing alternative flows leads to potential deadlocks and merging parallel flows may lead to the undesirable multiple execution of activities.

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 reconverge without synchronization.

Synonyms:

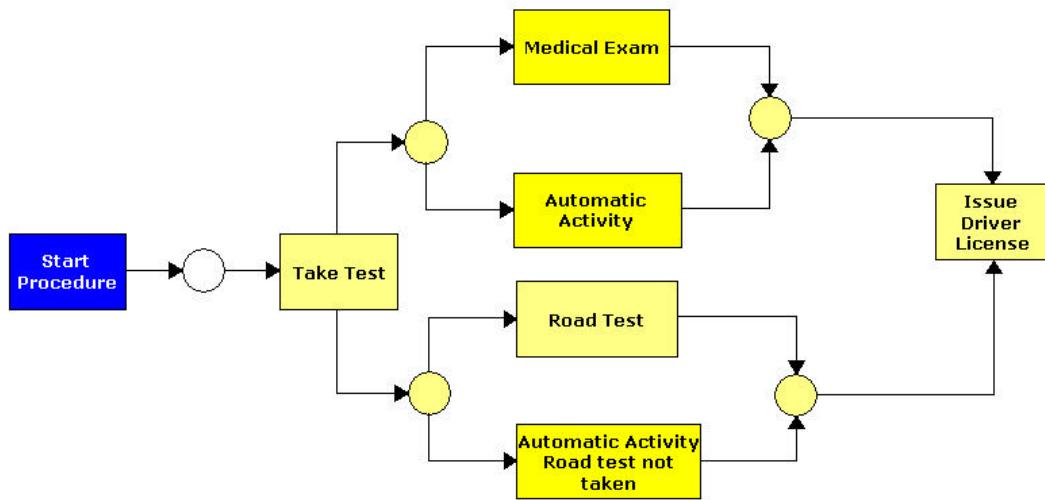
Synchronizing join.

Example:

- After one or both activities *medical exam* and *road test* have been completed (depending on if they have been executed at all), the activity *issue_driven_license* needs to be executed just once.

PECTRA Technology Inc.

Synchronization is made by using the identification number, the instances that were created from an AND Split have this identification number equal, therefore in the synchronization, from the status before to an activity, all instances with the same identification number are selected. In a Join with synchronization the activity splits up when there are tokens with the same identification number in all the status previous to that activity.



Structural Patterns

Pattern 11: Arbitrary Cycles

Execute workflow graph w/out any structural restriction on loops.

During the workflow analysis/design time it is undesirable to be exposed to various syntactical constraints of the specific workflow enactment tool such as for example that there should be only one entry and one exit point to the loop. In fact, to achieve proper abstraction, the workflow engine should allow for execution of unconstrained models as typically they are much more suitable for the end-users to trace the execution of the process.

Description:

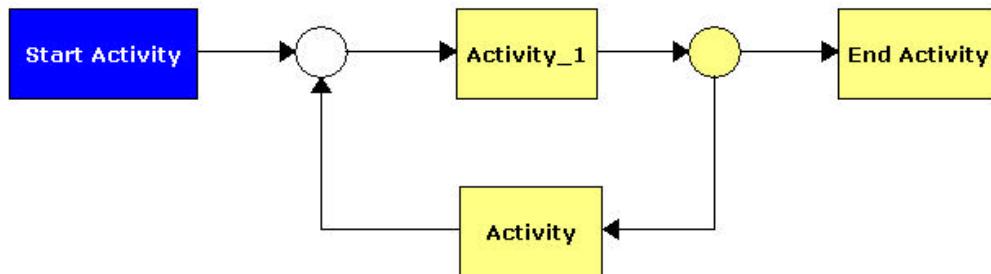
A point in a workflow process where one or more activities can be done repeatedly.

Synonyms:

Loop, iteration, cycle.

Example:

1. Most of the initial workflow models at the analysis stage contain arbitrary cycles (if they contain cycles at all). The end of the path can be reached, when activity 1 is executed and handled correctly. Through the use of more variables and conditions a loop can have more or less complex appearances.



Pattern 12: Implicit Termination

Terminate if there is nothing to be done.

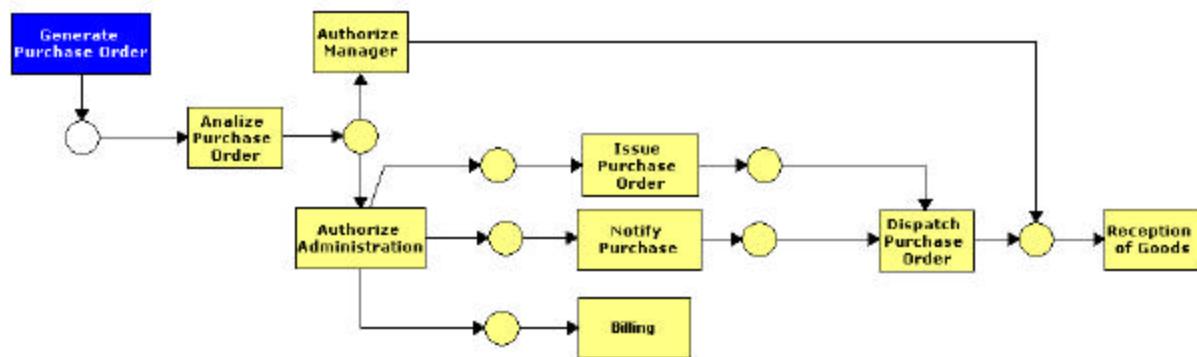
Another example of the requirement imposed by some of the workflow engines on a modeler is that the workflow model is to contain only one ending node, or in case of many ending nodes, the workflow model will terminate when the first one is reached. Again, most business models do not follow this pattern - it is more natural to think of a business process as terminated once there is nothing else to be done.

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).

Example:

1. The process *generate_purchase_order* ends with the activity *goods_reception*. The instance ends when activity *authorize_manager* or activity *authorize_administration* are terminated. (See figure in next page).



Problem Description:

Most workflow engines terminate the process when an explicit *Final* node is reached. Any current activities that happen to be running by that time will be aborted which may be confusing to end-users.

Pectra Technology Inc. has a workflow model that can contain more than one ending node. It is natural to think of a business process as terminated once the first ending node is executed. But with PECTRA Technology, when a parallel instance has concluded, it does not necessarily imply that all other instances finish automatically.

Patterns Involving Multiple Instances

Pattern 13: MI with a priori known design time knowledge

Generate many instances of one activity when a number of instances are known at the design time.

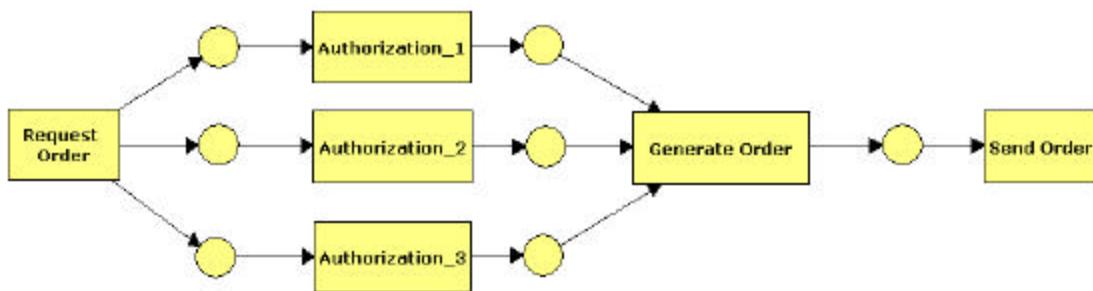
We would like to create many instances for one activity. The number of instances is known at design time.

Description:

For one case an activity is enabled multiple times. The number of instances of a given activity for a given case is known at design time.

Example:

1. The requisition of hazardous material requires three different authorizations.



Pattern 14: MI with a priori known runtime knowledge

Generate many instances of one activity when a number of instances can be determined at some point during the runtime (as in FOR loop).

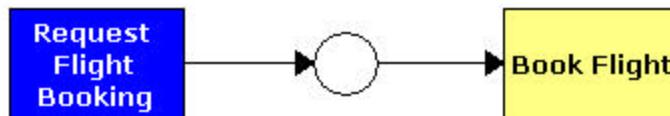
We would like to be able to generate many instances of an activity. The number of instances is dynamic, i.e. not known at the design time. It is known though at some point before all instances need to be executed. You can think about this pattern as a FOR loop that instantiates an activity.

Description:

For one case an activity is enabled multiple times. The number of instances of a given activity for a given case is variable and may depend on characteristics of the case or availability of resources, but is known at some stage during runtime, before the instances of that activity have to be created.

Example:

- When booking a trip, the activity *book_flight* is executed multiple times if the trip involves multiple flights.



PECTRA Technology Inc. handles this problem by using method End Activity. More than once in "Request flight booking", in the same transaction, it generates as many instances as needed; each instance may have different values in the attributes. This technique is used often with filtering of activity in "Book flight" activity.

Problem Description:

Only a few workflow management systems offer a construct for the multiple activation of one activity for a given case. Most systems have to resort to a fixed number of parallel instances of the same activity or an iteration construct where the instances are processed sequentially.



Pattern 15: MI with no a priori runtime knowledge

Generate many instances of one activity when a number of instances cannot be determined (as in WHILE loop).

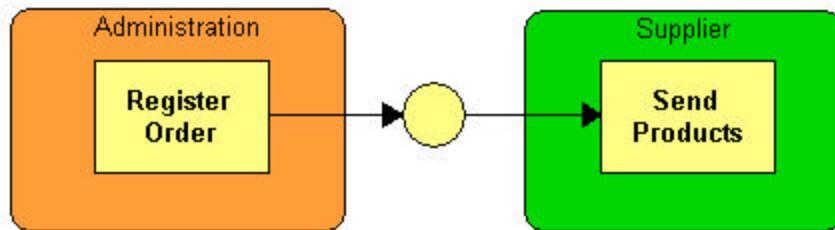
We would like to be able to generate many instances of an activity. The number of instances is dynamic, i.e. not known at the design time nor is it known at any stage during the execution of the process before all these instances needs to be activated. You can think about this pattern as a WHILE loop that instantiates an activity.

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 activities are completed, some other activity needs to be started. The difference with Pattern 14 is that even while some of these instances are being executed or already completed, new ones can be created.

Example:

The requisition of 100 computers involves an unknown number of deliveries. The number of computers per delivery is unknown and therefore the total number of deliveries is not known in advance. Once each delivery is obtained, it can be determined whether a next delivery is to come by comparing the total number of delivered goods so far with the number of the goods requested.



PECTRA Technology Inc. handles this problem by using method End Activity. More than once in "Register Order" in the same transaction, it generates as many instances as needed; each instance may have different values in the attributes. This technique is used often with filtering of activity in "Send Products".

In "Register Order" can generate instances as many as needed and continue running allowing new instance to be created.

Problem Description:

- Most workflow engines do not allow more than one instance of the same activity to be active at the same time.



Pattern 16: MI requiring synchronization

Generate many instances of one activity and synchronize them afterwards.

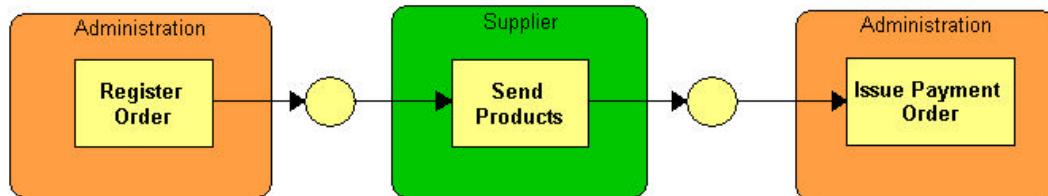
The other multiple-instances related patterns do not consider the synchronization of created instances. For example, spawning off a variable number of sub-processes from the main process does only launch multiple instances without considering synchronization issues. But sometimes it is required to continue the process only after all instances are completed, possibly w/out any a priori knowledge of how many instances were created.

Description:

For one case an activity is enabled multiple times. The number of instances may not be known at design time. After completing all instances of that activity another activity has to be started.

Example:

1. The requisition of 100 computers results in a certain number of deliveries. Once all deliveries are processed, the requisition has to be closed by means of *payment_order*.



PECTRA Technology Inc.

Within in the activity *Send_products* there must be a Script *Pay Order*. It enables not to execute the next activity until previous instance were completed under certain conditions. For this case, the *Order Payment* will not be issue until all 100 computers were delivered.

Problem Description:

Most workflow engines do not allow multiple instances. Languages that do allow multiple instances do not provide any construct that would allow for synchronization of these instances. Languages that support the *Release* construct do not provide any construct that would allow for synchronization of spawned off sub-processes.

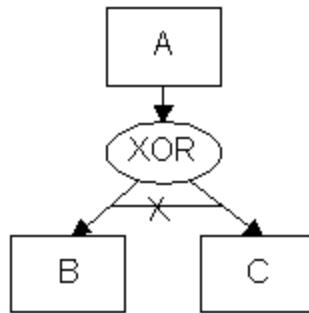
PECTRA Technology Inc. solves this problem using a combination of scripting, "In box" filtering, and mono instance. Mono instance allows the instances to be grouped according to a common attribute. These instances will be subsequently visualized in Digital Gate as one single instance (mono instance) because of the indicated condition, and inside it, all the instances with the same attributes for which they were grouped.

Stated Based Patterns

Pattern 17: Deferred Choice

Execute one of the two alternatives threads. The choice which thread is to be executed should be implicit.

Moments of choice, such as supported by constructs as XOR-splits/OR-splits, in workflow management systems are typically of an explicit nature, i.e. they are based on data or they are captured through decision activities. This means that the choice is made a-priori, i.e. before the actual execution of the selected branch starts an internal choice is made. Sometimes this notion is not appropriate. We may want to have a situation where two threads are "enabled" for an execution (suppose one thread enables an activity A, the other enables activity B). We would like to see both activities on a worklist). Once that one of the thread is started, the other thread should be disabled (i.e. once activity A gets started, B should disappear from the worklist).



Description:

A point in the workflow process where one of several branches are 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.



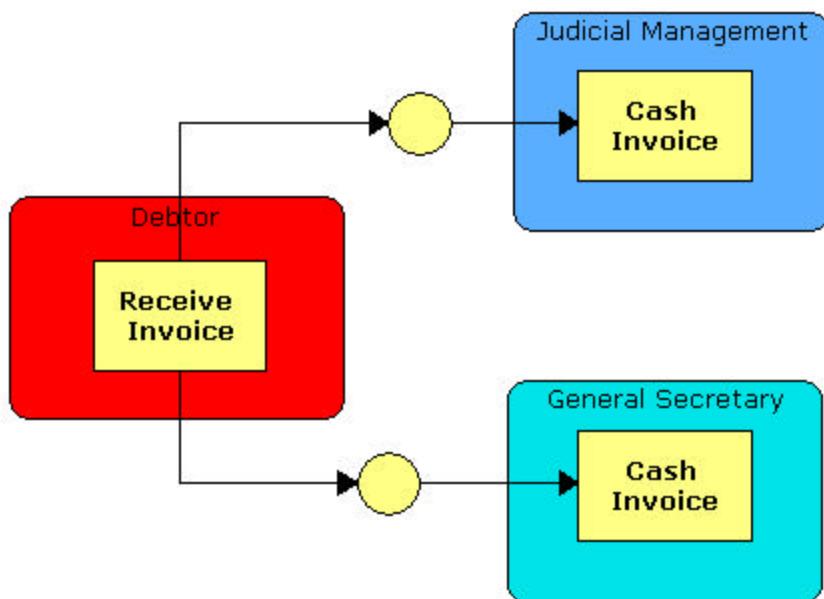
Synonyms:

External choice, implicit choice.

Example:

1. A debtor must pay his invoices as much to "Judicial Management" as to "General Secretariat". The debtor has the possibility to pay all in one place, i.e. he can pay all his invoices at "General Secretariat" - the ones that correspond to that area, and also the ones that correspond to "Judicial Management" area-. When paying all invoices to "General Secretariat", automatically the charge of the invoices corresponded to "Judicial Management" are canceled.

PECTRA Technology Inc. handles this problem when canceling or blocking an instance; all other activities are totally cancelled by scripting. A dispatcher of Notification can be used notifying the process cancellation.



Pattern 18: Interleaved Parallel Routing

Execute two activities in random order, but not in parallel.

Patterns Parallel Split and Synchronizing Join are typically used to specify parallel routing. Most workflow management systems support true concurrency, i.e. it is possible that two activities are executed for the same case at the same time. If these activities share data or other resources, true concurrency may be impossible or lead to anomalies such as lost updates or deadlocks.

Description:

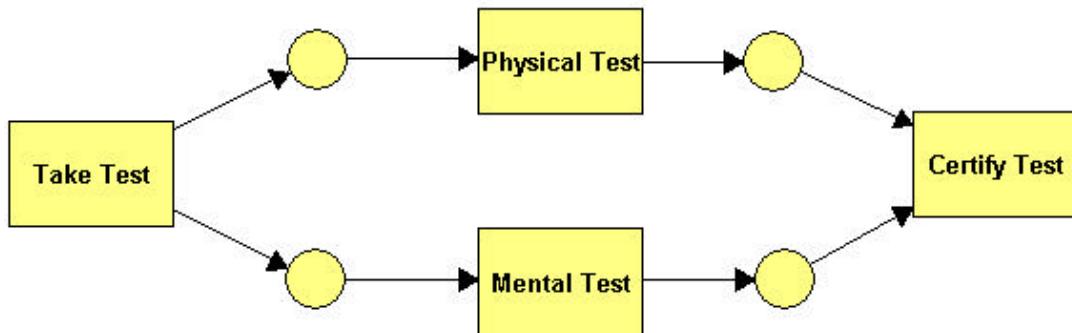
A set of activities is executed in an arbitrary order: Each activity in the set is executed, the order is decided at run-time, and no two activities are executed at the same moment (i.e. no two activities are active for the same workflow instance at the same time).

Synonyms:

Unordered sequence.

Example:

1. The Navy requires every job applicant has to take two tests: *physical_test* and *mental_test*. These tests can be conducted in any order but not at the same time.



PECTRA Technology Inc. handles this by locking one activity when the activity *physical_test* is executing it, blocks the execution of the activity *mental_test*. When activity *physical_test* has been executed, *mental_test* can be executed.

When both activities have been executed, they join synchronically to execute the activity *certify_test*.

Problem Description:

Since most workflow management systems support true concurrency when using constructs such as the Parallel Split and Synchronizing Join, it is not possible to specify interleaved parallel routing.



Pattern 19: Milestone

Enable an activity until a milestone is reached

This pattern allows for testing whether a workflow process has reached a certain phase. Upon reaching some phase we would like to disable the activities that were previously enabled.

Description:

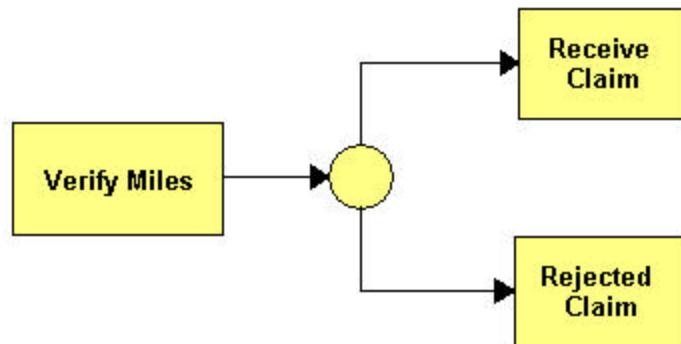
The enabling of an activity depends on the case being in a specified state, i.e. the activity is only enabled if a certain milestone has been reached which did not expire yet. Consider three activities *A*, *B*, and *C*. Activity *A* is only enabled if activity *B* has been executed and *C* has not been executed yet, i.e. *A* is not enabled before the execution *B* and *A* is not enabled after the execution *C*.

Synonyms:

Arc test, deadline, state condition.

Example:

1. A customer can claim air miles until six months after the flight. Expired that period of time, the client cannot claim for the air miles.



PECTRA Technology Inc. solves problem using attribute setting and Inbox filtering. Those Filters are used to modify the view of the Inbox by applying a condition by which some instances are hidden.

Using attribute setting we set if the claim can be proceeded.



Cancellation Patterns

Pattern 20 y 21: Cancel Activity and Cancel Case

Cancel (disable) the process.

PECTRA
Technology

Description:

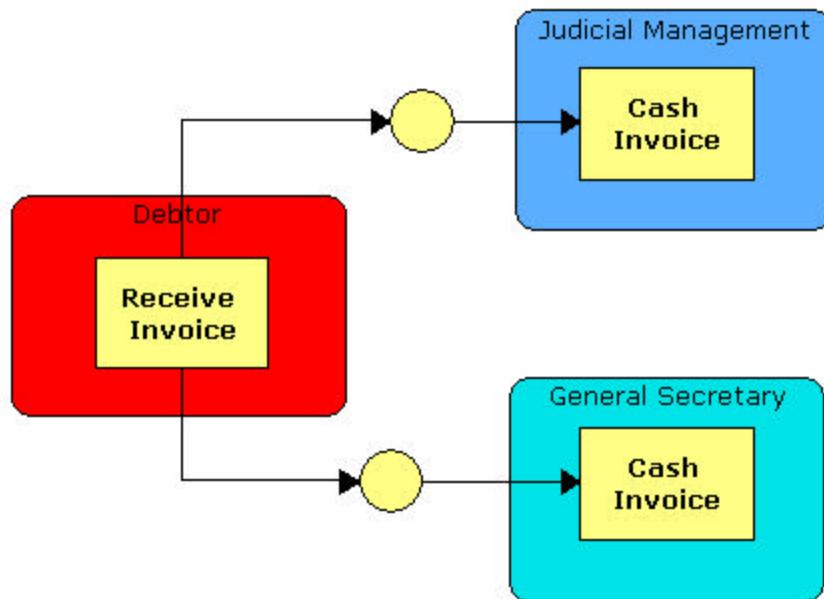
A case, i.e. workflow instance, is removed completely.

Synonyms:

Withdraw case.

Example:

1. A debtor must pay his invoices as much to "Judicial Management" as to "General Secretariat". The debtor has the possibility to pay all in one place, i.e. he can pay all his invoices at "General Secretariat" - the ones that correspond to that area, and also the ones that correspond to "Judicial Management" area-. When paying all invoices to "General Secretariat", automatically the charge of the invoices corresponded to "Judicial Management" are canceled.





PECTRA
Technology

PECTRA Technology Inc. cancels running instances by means of scripting. cancelation is passed to the motor of workflow that cancels the instance.

Problem Description:

Workflow management systems typically do not support the withdrawal of an entire case using the (graphical) workflow language.





Index

Basic Control Patterns.....	1
Pattern 1: Sequence.....	1
Pattern 2: Parallel Split	1
Pattern 3: Synchronization	2
Pattern 4: Exclusive Choice	4
Pattern 5: Simple Merge	5
Advanced Branching and Synchronization Patterns.....	6
Pattern 6: Multiple Choice	6
Pattern 7: Multiple Merge	7
Pattern 8: Discriminator.....	8
Pattern 9: N-out-of-M Join.....	9
Pattern 10: Synchronizing Join	10
Structural Patterns	11
Pattern 11: Arbitrary Cycles.....	11
Pattern 12: Implicit Termination	12
Pattern 13: MI with a priori known design time knowledge	13
Pattern 14: MI with a priori known runtime knowledge	14
Pattern 15: MI with no a priori runtime knowledge.....	15
Pattern 16: MI requiring synchronization	16
Stated Based Patterns.....	17
Pattern 17: Deferred Choice	17
Pattern 18: Interleaved Parallel Routing	18
Pattern 19: Milestone	20
Cancellation Patterns	21
Pattern 20 y 21: Cancel Activity and Cancel Case	21
Index.....	23