Process Patterns

Software Engineering White Paper

November 2003

Houston, USA
2425 West Loop South
Suite 200
Houston, Texas 77027 - USA
Phone : (713) 335 5562
Fax : (713) 297 8864

Buenos Aires, Argentina
Av. Leandro N. Alem 1050
8vo piso Dto. A
Capital Federal, C1001AAD
Argentina
Phone : +54 (11) 4313 8485
Fax: Ext. 121

Córdoba, Argentina
Bv. Las Heras 402
Cordoba, X5000FMR
Argentina
Phone : +54 (351) 4245756
Fax : +54 (351) 4237168

E-mail : info@pectra.com
http://www.pectra.com

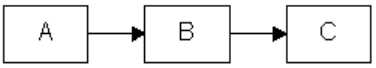
3132744341623231012293014132736121620131327481323321624374

Basic Control Patterns

Pattern 1: Sequence

Execute activities in sequence.

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



Description:

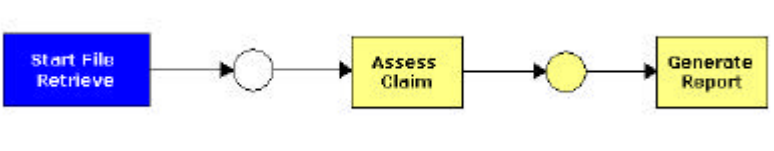
An activity in a workflow process is enabled after the completion of another activity same process.

Synonyms:

Sequential routing, serial routing.

Examples:

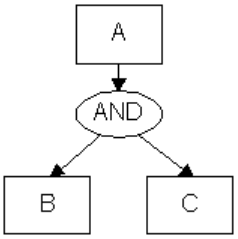
1. Activity send_bill is executed after the execution of activity send_goods.
2. An insurance claim is evaluated after the client's file is retrieved.
3. Activity add_air_miles is executed after the execution on activity book_flight.



Pattern 2: Parallel Split

Execute activities in parallel.

Parallel Split is required when two or more activities need to be executed in parallel. P Split is easily supported by most workflow engines except for the most basic sche (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 tre control which can be executed in parallel, thus allowing activities to be exe simultaneously or in any order.

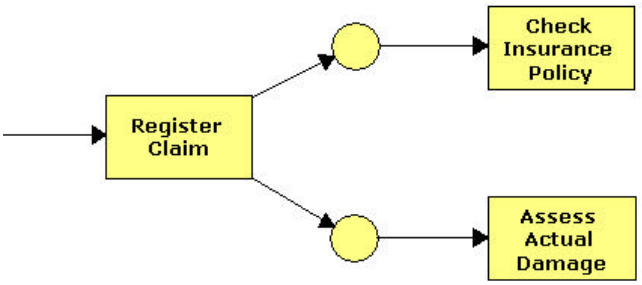
Synonyms:

AND-split, parallel routing, fork.

Examples:

1. The execution of the activity *payment* enables the execution of the activities *ship_* and *inform_customer*.

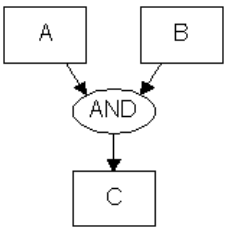
2. After registering an insurance claim two parallel sub-processes are triggered: o checking the policy of the customer and one for assessing the actual damage.



Pattern 3: Synchronization

Synchronize two parallel threads of execution.

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



Description:

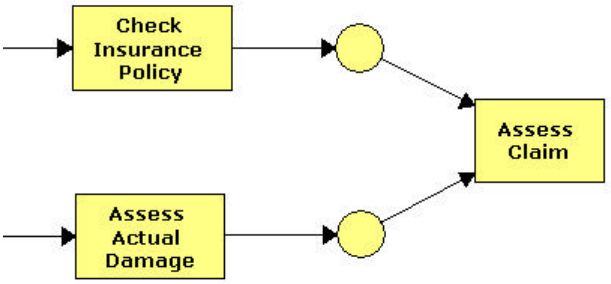
A point in the workflow process where multiple parallel sub-processes/activities cor
into one single thread of control.

Synonyms:

AND-join, synchronizer.

Examples:

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

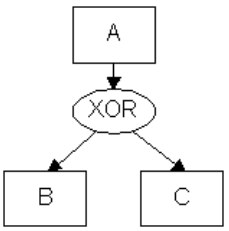


PECTRA Technology Inc.

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

Pattern 4: Exclusive Choice

Choose one execution path from many alternatives.



Description:

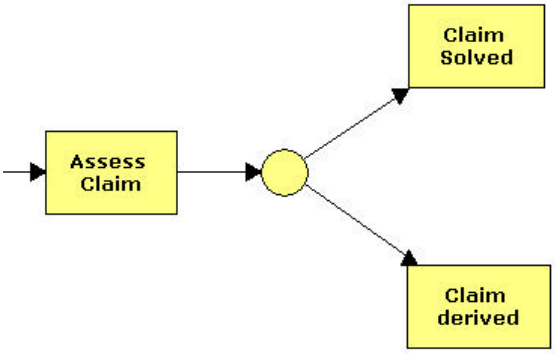
A point in the workflow process where based on a decision or workflow control data, o several branches is chosen.

Synonyms:

XOR-split, conditional routing, switch, decision.

Examples:

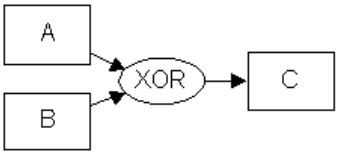
1. In the process *client_claim* two requirement types are evaluated and two option possible: claim solved, or claim derived. The process will be split depending on the de taken.



Pattern 5: Simple Merge

Merge two alternative execution paths.

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



Description:

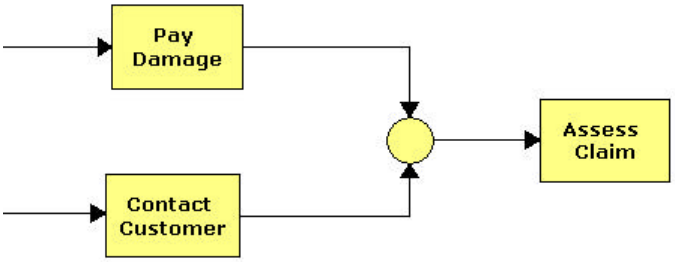
A point in the workflow process where two or more alternative branches come tog without synchronization. In other words the merge will be triggered once any c incoming transitions are triggered.

Synonymes:

XOR-join, asynchronous join, merge.

Examples:

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

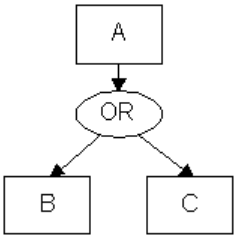


Advanced Branching and Synchronization Patterns

Pattern 6: Multiple Choice

Choose several execution paths from many alternatives.

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



Description:

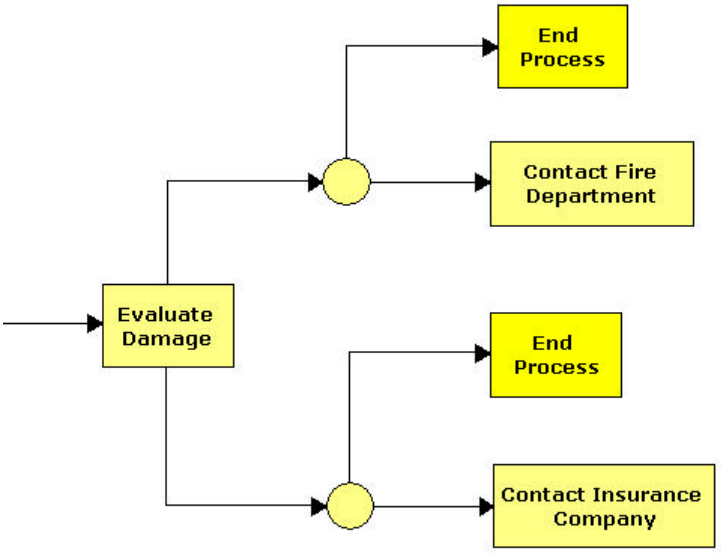
A point in the workflow process where based on a decision or workflow control data, c
more branches are chosen.

Synonyms:

Conditional routing, selection, OR -split.

Examples:

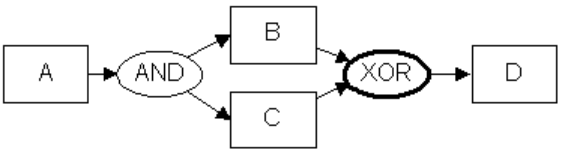
After executing the activity *evaluate_damage* the activity *contact_fire_department*
activity *contact_insurance_company* is executed. At least one of these activities is exe
However, it is also possible that both need to be executed.



Pattern 7: Multiple Merge

Merge many execution paths without synchronizing.

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

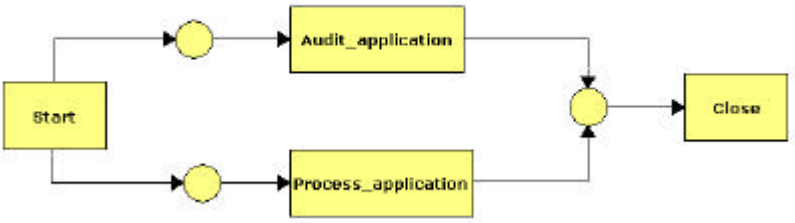


Description:

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

Examples:

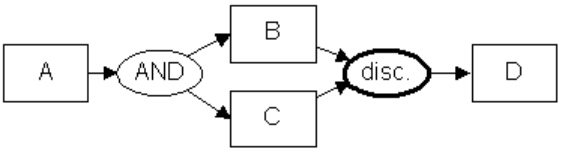
1. Sometimes two or more parallel branches share the same ending. Instead of repli
this (potentially complicated) process for every branch, a multi-merge can be used. A
example of this would be two activities *audit_application* and *process_application* runr
parallel which should both be followed by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



Pattern 8: Discriminator

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

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

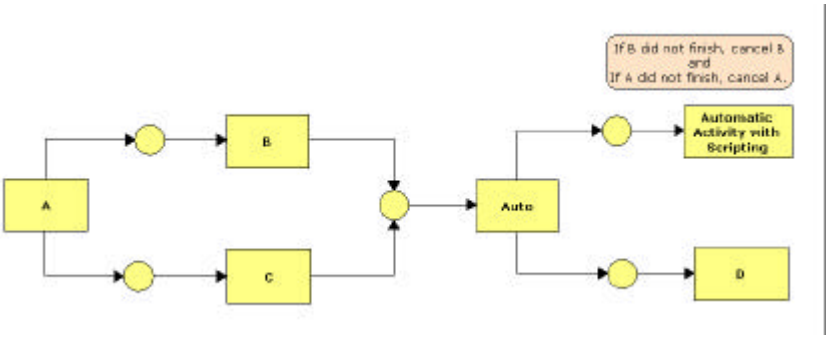


Description:

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

Examples:

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



Pattern 9: N-out-of-M Join

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

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

Description:

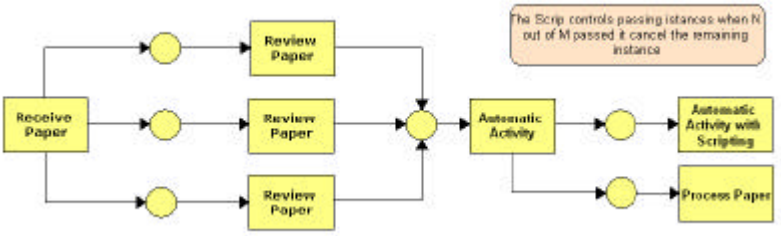
N-out-of-M Join is a point in a workflow process where *M* parallel paths converge int The subsequent activity should be activated once *N* paths have completed. Completio remaining paths should be ignored. Similarly to the discriminator, once all inc branches have "fired", the join resets itself so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

Synonyms:

Partial join, discriminator, custom join.

Examples:

1. A paper needs to be sent to three external reviewers. Upon receiving two revie 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 m if only one execution path is taken.

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

Description:

A point in the workflow process where multiple paths converge into one single thre more than one path is taken, synchronization of the active threads needs to take pla only one path is taken, the alternative branches should reconverge without synchroniza

Synonyms:

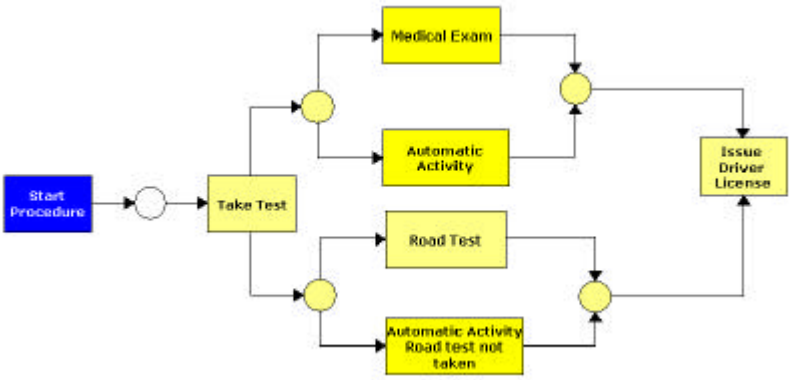
Synchronizing join.

Examples:

1. After one or both activities *medical exam* and *road test* have been completed (depe on if they have been executed at all), the activity issue_*driven_license* needs to be exe just once.

PECTRA Technology Inc.

Synchronization is made by using the identification number, the instances that were cr from an AND Split have this identification number equal, therefore in the synchroniz from the status before to an activity, all instances with the same identification numb selected. In a Join with synchronization the activity splits up when there are tokens wi 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 va syntactical constrains of the specific workflow enactment tool such as for example that should be only one entry and one exit point to the loop. In fact, to achieve p abstraction, the workflow engine should allow for execution of unconstrained mod typically they are much more suitable for the end-users to trace the execution o process.

Description:

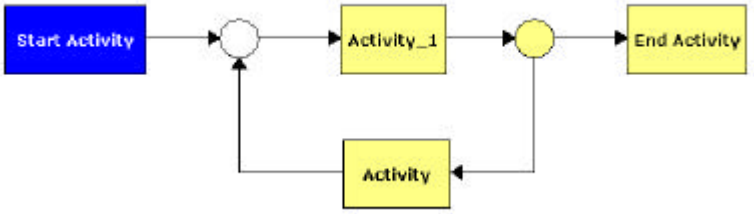
A point in a workflow process where one or more activities can be done repeatedly.

Synonyms:

Loop, iteration, cycle.

Examples:

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



Pattern 12: Implicit Termination

Terminate if there is nothing to be done.

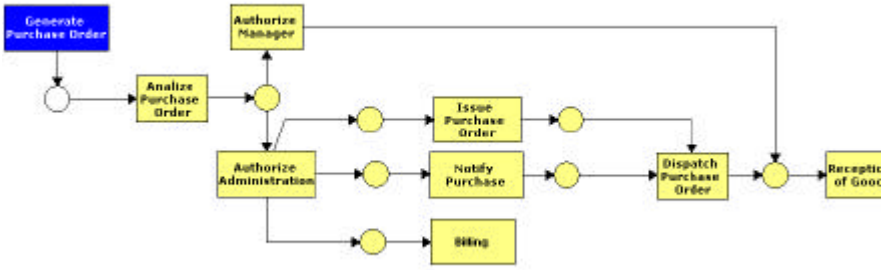
Another example of the requirement imposed by some of the workflow engines on a m is that the workflow model is to contain only one ending node, or in case of many e nodes, the workflow model will terminate when the first one is reached. Again, business models do not follow this pattern - it is more natural to think of a business p 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 words, there are no active activities in the workflow and no other activity can be active (and at the same time the workflow is not in deadlock).

Examples:

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



Problem Description:

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

Pectra Technology Inc. has a workflow model that can contain more than one e node. It is natural to think of a business process as terminated once the first ending r executed. But with PECTRA Technology, when a parallel instance has concluded, it do necessary imply that all others instances finish automatically.

Patterns Involving Multipe Instances

Pattern 13: MI with a priori known design time knowledge

Generate many instances of one activity when a number of instances is kno the design time.

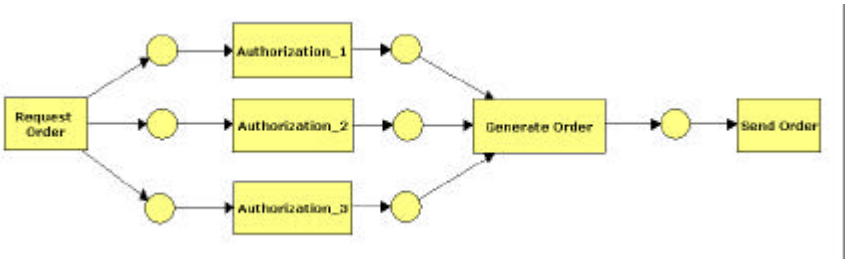
We would like to create many instances for one activity. The number of instances is at design time.

Description:

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

Examples:

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 ca determined at some point during the runtime (as in FOR loop).

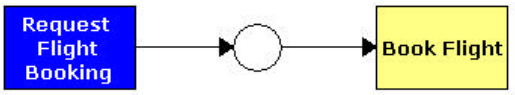
We would like to be able to generate many instances of an activity. The number of ins is dynamic, i.e. not known at the design time. It is known though at some point bef instances need to be executed. You can think about this pattern as a FOR loop instantiates an activity.

Description:

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

Examples:

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



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

Problem Description:

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

Pattern 15: MI with no a priori runtime knowledge

Generate many instances of one activity when a number of instances cann determined (as in WHILE loop).

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

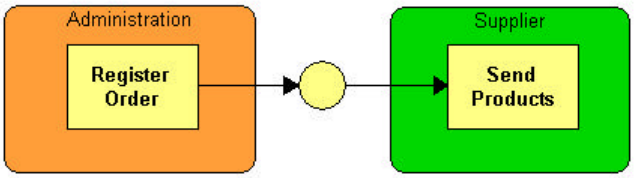
Description:

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

Examples:

The requisition of 100 computers involves an unknown number of deliveries. The num computers per delivery is unknown and therefore the total number of deliveries is not in advance. Once each delivery is obtained, it can be determined whether a next deli

to come by comparing the total number of delivered goods so far with the number goods requested.



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

In “Register Order” can generate instances as many as needed and continue ru allowing new instance to be created.

Problem Description:

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

Pattern 16: MI requiring synchronization

Generate many instances of one activity and synchronize them afterwards.

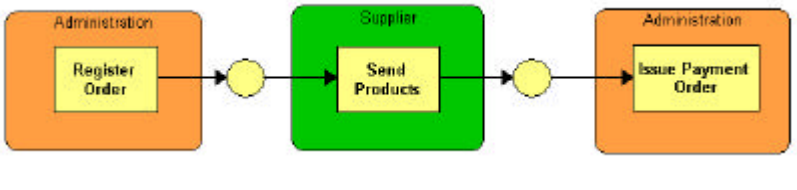
The other multiple-instances related patterns do not consider the synchronization of c instances. For example, spawning off a variable number of sub-processes from the process does only launch multiple instances without considering synchronization issue sometimes it is required to continue the process only after all instances are comp 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 r known at design time. After completing all instances of that activity another activity be started.

Examples:

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



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

Problem Description:

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

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

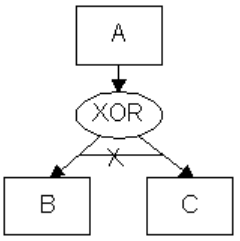
Stated Based Patterns

Pattern 17: Deferred Choice

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

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

thread is started, the other thread should be disabled (i.e. once activity A gets star
should disappear from the worklist).



Description:

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

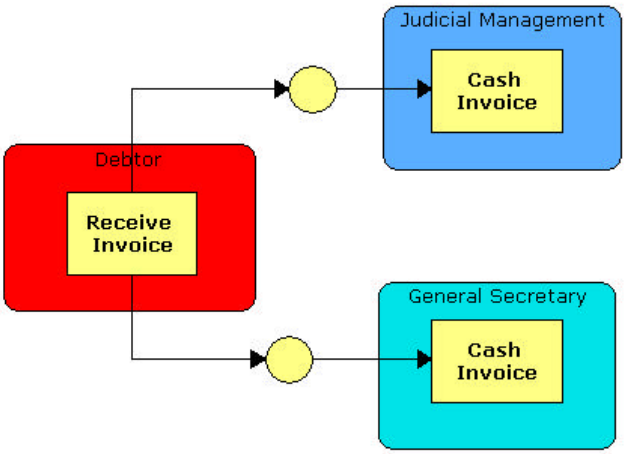
Synonyms:

External choice, implicit choice.

Examples:

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

PECTRA Technology Inc. handles this problem when canceling or blocking an instan
other activities are totally cancelled by scripting. A dispatcher of Notification can be
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 ro
Most workflow management systems support true concurrency, i.e. it is possible tha
activities are executed for the same case at the same time. If these activities share d
other resources, true concurrency may be impossible or lead to anomalies such a
updates or deadlocks.

Description:

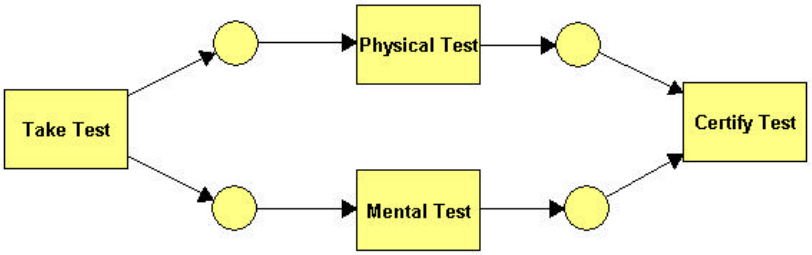
A set of activities is executed in an arbitrary order: Each activity in the set is execute
order is decided at run-time, and no two activities are executed at the same moment (
two activities are active for the same workflow instance at the same time).

Synonyms:

Unordered sequence.

Examples:

1. The Navy requires every job applicant has to take two tests: *physical_tes*
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 a *physical_test* is executing it, blocks the execution of the activity *mental_test*. When a *physical_test* has been executed, *mental_test* can be executed.

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

Problem Description:

Since most workflow management systems support true concurrency when using cons such as the Parallel Split and Synchronizing Join, it is not possible to specify interl 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 p Upon reaching some phase we would like to disable the activities that were prev enabled.

Description:

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

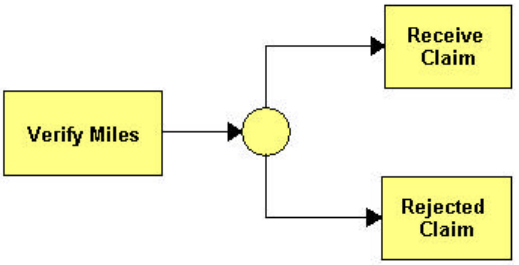
Synonyms:

Arc test, deadline, state co ndition.

Examples:

1. A customer can claim air miles until six months after the flight.

El reclamo se puede realizar hasta seis meses después del vuelo. Pasado ese perio tiempo el cliente no puede reclamar las millas aereas.



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

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

Cancellation Patterns

Pattern 20 y 21: Cancel Activity and Cancel Case

Cancel (disable) the process.

Description:

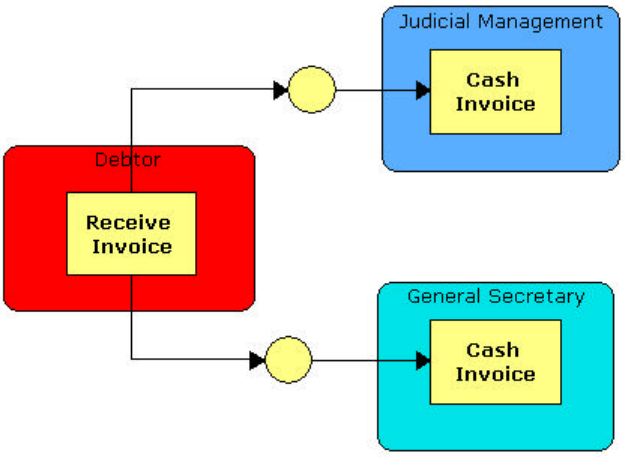
A case, i.e. workflow instance, is removed completely.

Synonyms:

Withdraw case.

Examples:

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



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

Problem Description:

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

Index