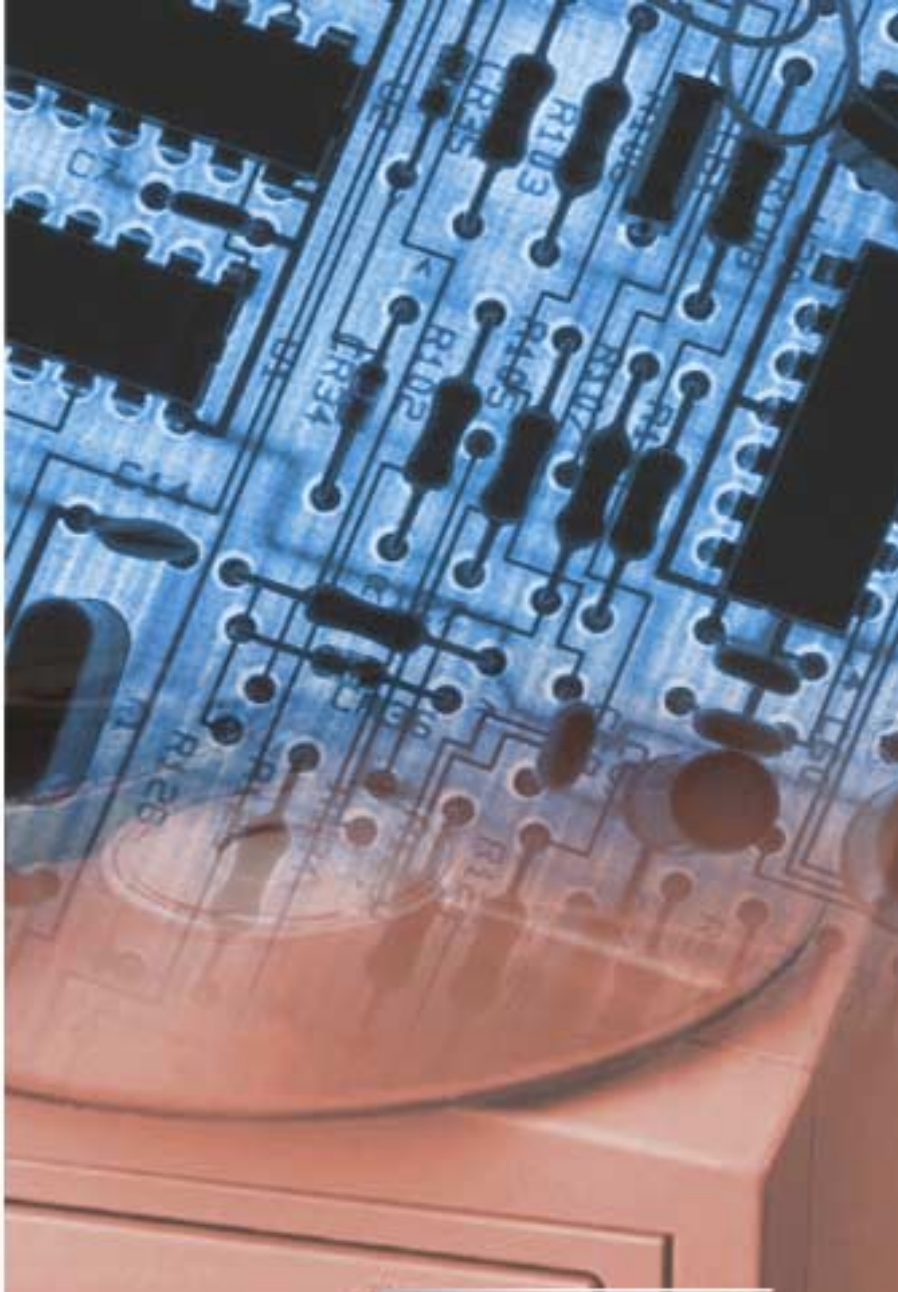
Business Agility!



BIZ AGI

Who we are

Vision Software has excelled for more than fourteen years through its creativity, quality and dedication to client service.

We focus on collaborative software development and enterprise integration services.

Our customers are medium and large organisations and we are particularly well positioned in the financial sector.

The trust that our customers have shown in Vision Software, as is reflected in the renewal of service contracts over various years, motivates us to improve day by day.

We are known for converting our passion for leading edge technology into solutions that add business value for our customers.

BizAgi ® is our award winning BPM solution.

What is Business Process Management

BPM is a management theory that is based on two fundamental paradigms:

- The process view of the organization is the underlying theory that provides management with the adequate information for the correct allocation of resources.
- Management decisions are based on reliable real-time indicators that provide a comprehensive view of the actual situation and future trends.

• By 2005, at least 90 percent of large enterprises will have BPM. Enterprises that continue to hard-code all flow control, or insist on manual process steps and do not incorporate BPM's benefits, will lose out to competitors that adopt BPM."

Gartner Group

BizAgi ® is a Business Process Management solution which optimizes performance in process based organisations by providing the necessary operational and management tools to visualize, control and improve all of the firm's processes in real time.

BIZAGI'S BENEFITS FOR YOUR ORGANISATION

BizAgi ® provides the following business benefits:

- BizAgi ® improves visibility because it allows real-time monitoring and analysis of what is really going on in each and every business process. You will always be able to see through you organisation and know how each team and individual is performing.
- BizAgi ® gives governance back to management because it guarantees that the processes are carried out according to the specified times and conditions. The execution and performance reports are produced automatically, showing the reality that you need to see and not what other people want you to see.
- BizAgi ® offers a true focus on business processes in your organisation, in such a way that the performance analysis and execution indicators always lead towards continuous improvement.

- BizAgi ® increases the agility of your applications, so that they adjust to your reality and not all the way around. The independent handling of the business rules will give you the flexibility necessary to make the application evolve according to the requirements of the organisation, without the actualization becoming traumatic.
- BizAgi ® is standardizing, centralizing the procedures in one unified model which contains all those processes vital to your business. This scheme facilitates evaluation and gives true control of your organisation.
- BizAgi ® guarantees a permanent synchronization between business processes and their supporting technological tools.

HOW DOES BIZAGI ® WORK

BizAgi provides the tools to create an application that models the process, while at the same time extracting and providing process information to management.

- The first step to create the application is to draw the process. BizAgi ® extends Microsoft ® Visio by creating its own BPM stencil to offer a standard, ready available graphical modelling tool.
- Then the data of the process is modelled within BizAgi ® (in Entity-Relationship diagrams).
- This data is grouped in forms that are composed and linked to the activities of the process.

- The fourth step consists of assigning the resources to the activities according to skills, positions, geographical areas and other variables. BizAgi even offers advanced work distribution algorithms.
- Afterwards the business rules (for example, the definition of a big loan application that needs to be approved by a committee) are defined and linked to the flowchart.

- Finally the intranet application generated by BizAgi ® is extended with interfaces to communicate with existing systems and by tools that provide very specific functionality within an activity.

BizAgi ® will automatically extract and process key information and provide it (through an integrated process data warehouse) to management.

By generating an intranet application in this way, BizAgi ® offers strategic advantages to traditional software approaches:

- Unlike a generic software package, BizAgi ® will model your organisation's processes and not the processes that someone else thought your organization should comply to. It is the software adapting to your processes, not the other way around.
- Unlike custom made solutions, advanced components (such as Infotuation™ , our Data-Process-User Interface automation technology) offer huge productivity and flexibility gains, so that your organisation gets processes automated ***faster*** and changes to the processes can be implemented in a more ***flexible*** manner.
- By knowing the process, BizAgi ® is able to display the right information to the right people at the right moment, independent of in which legacy systems this information resides.

CASE STUDY

The main objective in the implementation of BizAgi® at a leading national Bank with over 300 branches nationwide was to have a system to track, control and analyze *the bank's entire* processes trough a single BPM solution.

Just in the customer service department, over 350.000 cases have been processed (by more than 1.300 users) by BizAgi ® in a year of non stop operation (that is, 100% system availability).

As of June 2003 more than 70 processes (including complex core processes such as loan application) have been modelled with BizAgi ®.

Key benefits

According to the bank's president, the main benefits have been in "drastically increasing the processes efficiency while at the same time improving customer service".

To provide an example, the personnel assigned to one credit card process were reduced from over a hundred to forty five, while at the same time reducing process time in 80%.

VISION SOFTWARE

www.visionsoftware.biz

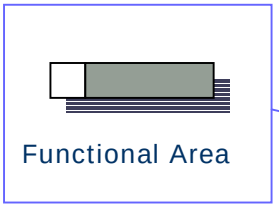
moreinfo@bizagi.com

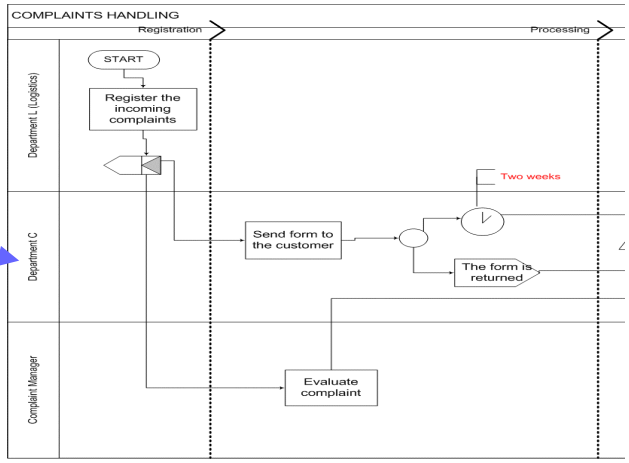
MODELLING ELEMENTS

These are BizAgi's modelling elements (or drawing shapes)

FUNCTIONAL AREAS

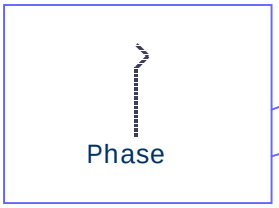
Functional areas define working teams for activities. Every activity in the flow chart has one functional area responsible to execute it.

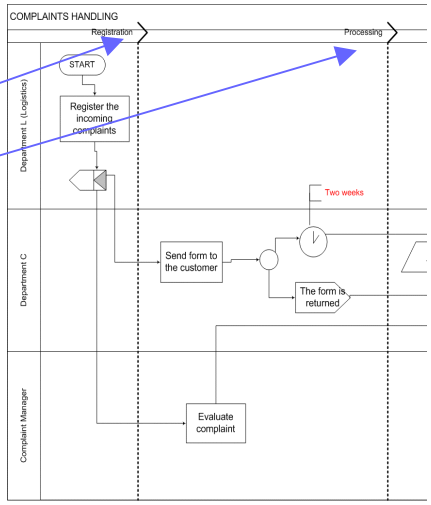




PHASES

Phases define the macro business states. Every phase represents a process state.

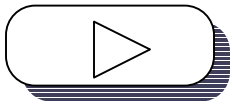




Start

It indicates the beginning of a process. Only a transition exits (No entry transitions).

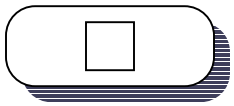
There must be only one per process.



Start

End

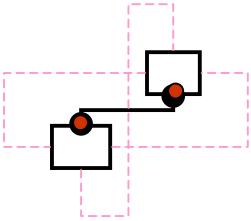
It indicates the end of a process. One or more transitions enter the shape. There are no outgoing transitions. There must be only one per process.



End

Transition

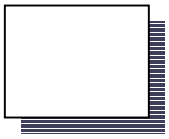
The transition shape connects two shapes in the flow chart. This shape indicates the path from one shape to another. In some cases it is linked to a Business Rule (on a Decision, Multi decision, Xor Split and Conditional Join shapes).



Transition

Activity

An activity represents a task performed by a user displaying a form. Each activity has its own characteristics like duration, alarms, event rules (business rules on enter, on save and on exit to the activity) and assignees, among others.



Activity

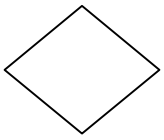
Automatic Task

An automatic task indicates a task performed by the system without human intervention, for example interfaces with other systems, computational activities, etc.

Automatic Task

Decision

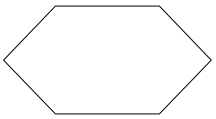
This shape is used in a point in the workflow process where, based on a decision or workflow control data, one of two branches is chosen.



Decision

Multi-Decision

This shape is used in a point in the workflow process where, based on a decision or workflow control data, several branches are chosen.



Multi decision

Sub process

This modelling element indicates the start of a sub process in the flow chart. Every process in BizAgi can also be used as a Sub process.

There are two types of sub processes:

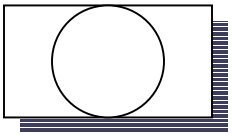
Integrated: The parent process wait until the sub process finalizes to continue with the following activities.

Standalone: The parent process launches the sub process and continues with the following activities.

Sub process

Module

A module represents a group of tasks connected between them that can be included in any process. Unlike a sub process, a module can't be a process itself, although it is created in then same way.



Module

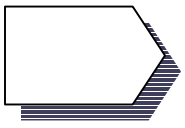
Wait

Then wait shape represents a delay within the process (wait for a period of time). This time can be fixed at design time or specified during runtime. During runtime a period can be specified for a specific date.

Wait

Event

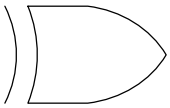
An event is an activity that can be performed at any time (that is, after the event has been set). Instead of being activated by the user responsible for the event, the event depends on external triggers for its activation. It can be activated manually or automatically by one business rule. It can also be used for inter-process synchronization.



Event

XOR Join

This shape is used when there is a point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen.



Xor Join

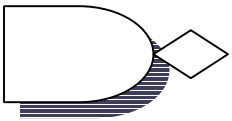
AND Join

This shape is used to synchronize all the incoming transitions (threads).

And Join

Conditional Join

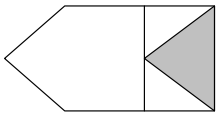
This shape evaluates a condition in some point in the process. It is a synchronization shape, that is, the transition exits only when the condition is valid.



Conditional Join

AND SPLIT

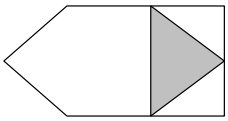
This shape is used when multiples activities have to be performed in parallel and in any order



And Split

XOR SPLIT

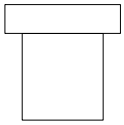
This shape is used in a point in the workflow process where, based on a decision or workflow control data, one of several branches is chosen.



XOR Split

Token Collector

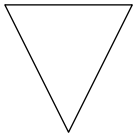
It is used when nothing should be performed after an activity or decision has been executed. The token collector has no outgoing transitions. If there are no pending Activities the case will be closed when the transition arrive to this shape.



Token Collector

Synchronizing Join

This modelling element waits until all the activated branches arrive to the shape. In the *Synchronizing Join* the corresponding *Multiple Decision* shape to synchronize has to be specified.

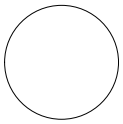


Synchronizing Join

Choice

All the activities following this shape will be pending until one of them is executed (Finished). This means that only one of these branches will be done completely.

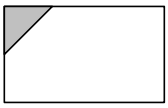
The valid shapes after the choice are: *Activity*, *Singleton Activity*, *Wait*, *Event*, *And Join* and *Conditional Join*



Choice

Singleton

It is similar to an activity; the difference is that when many instances are possible, only one is active at any point in time.

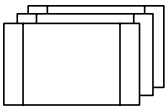


Singleton

Multiple Sub process

This shape allows the creation of multiple instances of a process. Each instance represents an item in a 1-N relationship with the process. In BizAgi the Entity Relationship data model is deeply connected with the process because the process has to handle complex, reusable, scalable information structures. In order to illustrate its functionality let's assume that the process is handling an order that might contain several items (1-N relationship). If an activity (s) has to be performed for each item we use the Multiple Sub process element specifying the 1-N relationship that defines during runtime the appropriate creation of the sub processes. N processes will be created, each one with the context that corresponds to each item.

It is also possible to specify to synchronize all the instances, to synchronize a constant or variable number of instances or to do no synchronization at all or to.

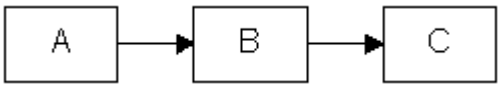


Multiple Sub process

WORKFLOW PATTERNS

1. 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.

BizAgi Support: An activity is enabled only after the completion of the previous activity.

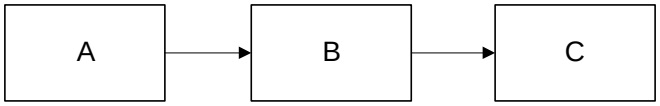
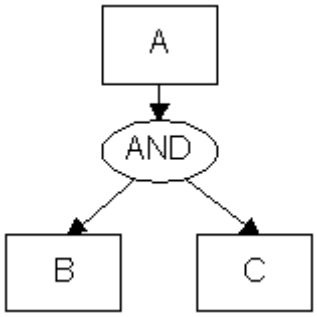


Figure 1. Sequence

Example: Activity B is enabled after the completion of A and activity C is enabled after the completion of B.

2. PARALLEL SPLIT

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 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 treads of control which can be executed in parallel, thus allowing activities to be executed simultaneously or in any order.

BizAgi Support: When one or more activities must be executed in parallel, an And Split is used to enable those activities. The And Split enables always all the activities.

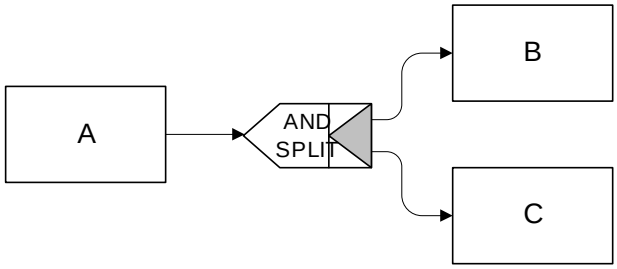
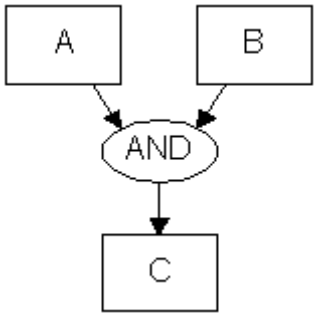


Figure 2. Parallel Split

Example: After the completion of activity A, activities B and C are enabled to be executed in parallel.

3. SYNCHRONIZATION

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, thus synchronizing multiple threads.

BizAgi Support: The And Join is used to synchronize all the incoming transitions (threads). The And Join requires the completion of all the threads even if they are not activated.

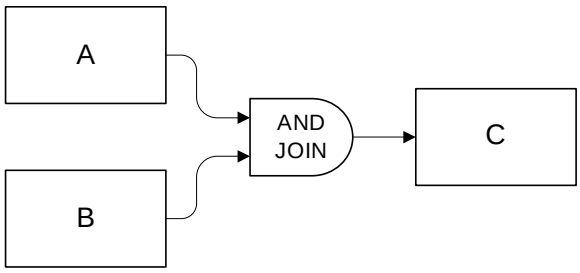
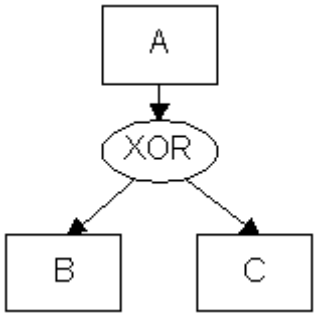


Figure 3. Synchronization

Example: Activity C starts only after the completion of A and B.

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.

BizAgi Support: If one of two branches must be chosen, a "Decision" can be used. A business rule is defined if is true YES Branch is chosen otherwise the other one will be activated.

If one of several branches must be chosen an "XOR Split" must be used. For every outgoing transition a business rule is evaluated and only one can be true.

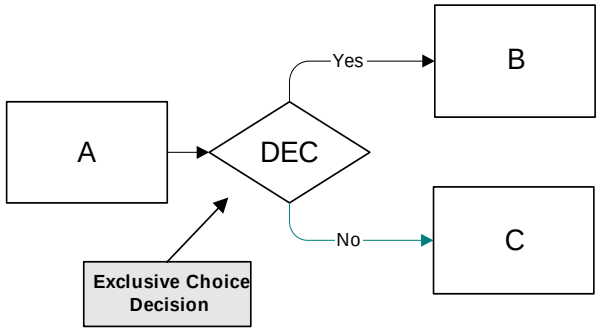


Figure 4.1 Exclusive choice using *decision*

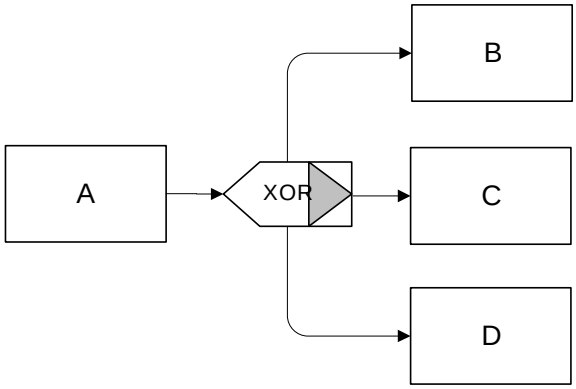


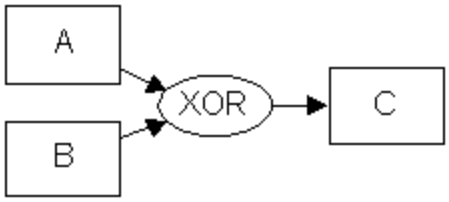
Figure 4.2 Exclusive choice using *xor split*

Example: Figure 4.1 If the business rule evaluated in the "decision" is true B is activated otherwise C is activated.

Figure 4.2. If only the evaluation of the business rule for the transition to D is true, D is activated.

5. SIMPLE MERGE

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 activated.

BizAgi Support: *XOR Join* is used to merge two alternative execution paths into one.

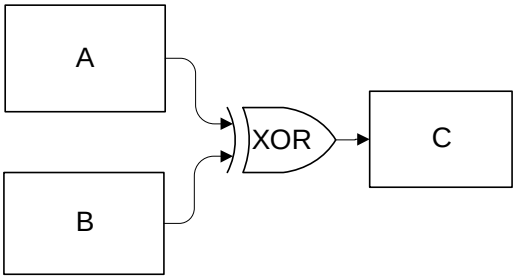
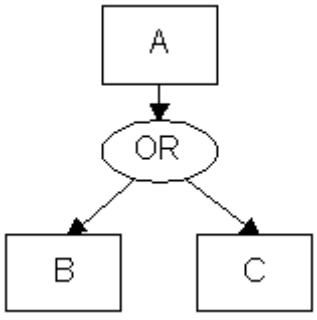


Figure 5. Simple Merge

Example: C is activated after completion of A or B.

6. MULTIPLE CHOICE

The Exclusive Choice pattern 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.

BizAgi Support: A *multi decision* allows taking one or more workflow paths if one or more business rules are fulfilled.

Example: After activity A the *multi decision* evaluates the business rules for the outgoing transitions to B, C and D, activating all the valid ones.

These are some examples for business rules:

Transition to B: "<varGotoB>==true"

Transition to C: "<loanValue> > 50000"

Transition to D: "BusinessRule.IsNewClient(<idClient>)"

Transitions are valid when Rule evaluation returns true.

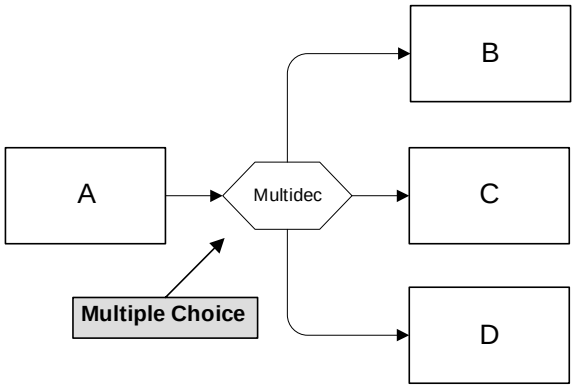
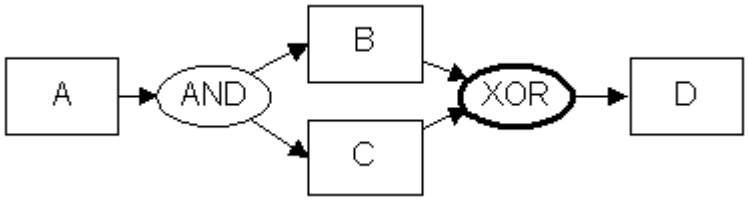


Figure 6. Multiple Choice

7. MULTIPLE MERGE

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: Multi-merge is 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 once for every incoming branch that gets activated.

BizAgi Support: An activity could be activated for every incoming branch.

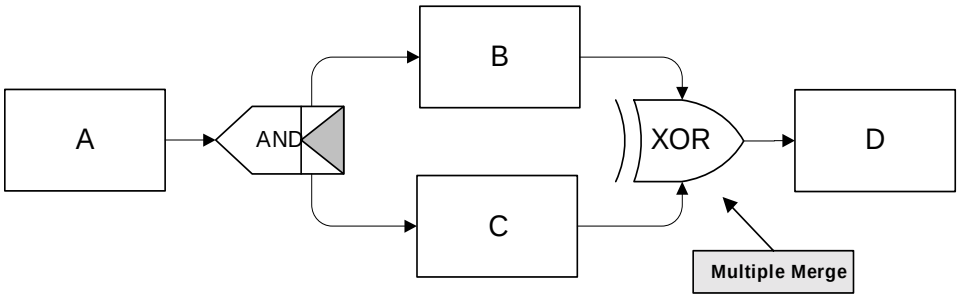
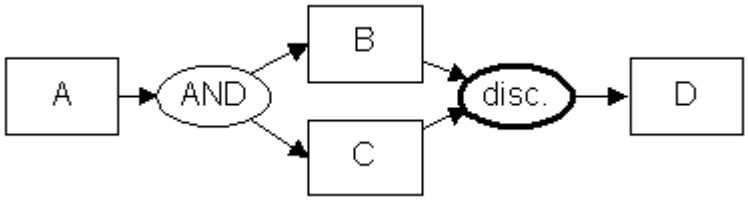


Figure 7. Multiple Merge

Example: In the diagram above, D will be instantiated twice.

8. DISCRIMINATOR

This pattern can be seen as the opposite of the multi-merge, that is, the subsequent activity should be instantiated only one time, once the appropriate number of incoming transitions has been activated, while ignoring any later ones.



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 and "ignores" them. Once all incoming branches have been triggered, it resets itself so that it can be triggered again.

BizAgi Support: In order to wait for a number of incoming branches a *Conditional join* is used.

It's necessary to use a script on the *OnEnter* of the *Conditional Join* and a Rule for the outgoing transition.

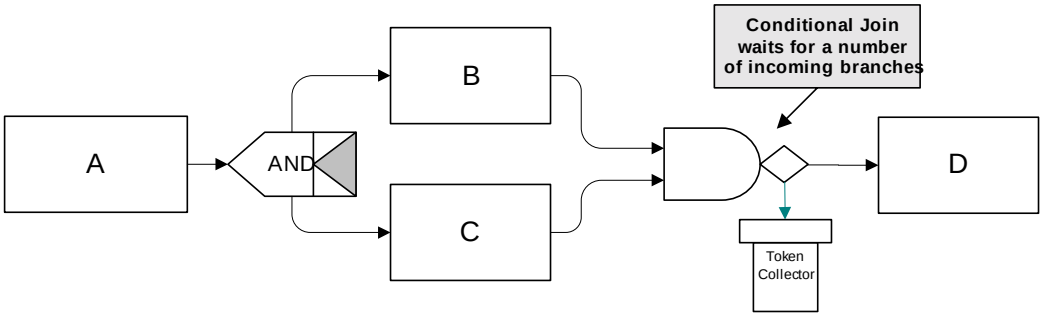


Figure 8.1 Discriminator

Example: The script for the *OnEnter* (in the *Conditional Join*) is: "<counter>++"

The Condition for the transition to D is: "<counter>==<waitNumber>"

When the counter reaches the specified number, D is activated otherwise the token collector will be activated.

In order to reset the counter once all incoming branches have been triggered, this script is added in the *OnExit* (*Conditional Join*):

"if (<counter>==<incomingbranches>) <counter>=0"

9. N OUT OF M JOIN.

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 be fired again.

BizAgi Support: When N branches are activated, in order to join these branches a *Conditional join* is used. For each activation of an incoming branch, a counter is increased in the conditional join. When the counter reaches the value M, D is activated otherwise the token collector will be activated.

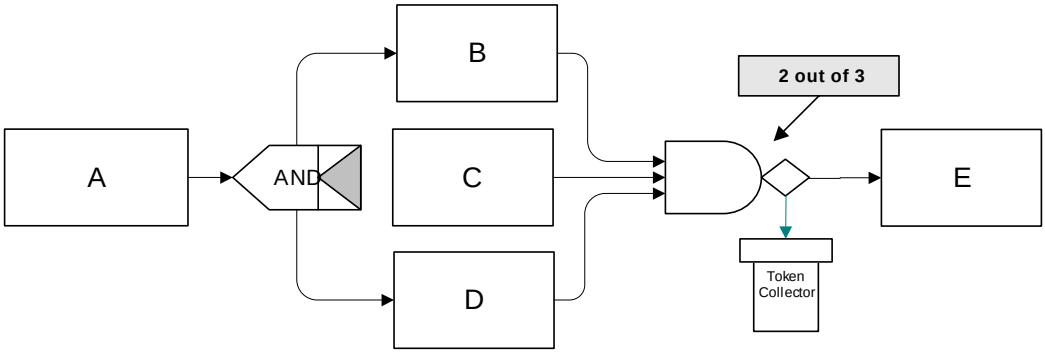


Figure 9. N out of M Join

Example: In the Conditional Join we use this script <counter>++; D is activated when <counter>==2

10. SYNCHRONIZING JOIN

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 support. The OR-join should have the capability to synchronize parallel flows and to merge alternative flows. The difficulty resides in being able to identify when to synchronize and when to merge. Synchronizing alternative flows leads to potential deadlocks and merging parallel flows may lead to the undesirable effect of executing activities multiple times.

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.

BizAgi Support: After a *multi decision* shape, one or more paths can be activated; the *Synchronizing Join* will wait for the activated braches. In the *Synchronizing Join* the *Multiple Decision* shape to synchronize with has to be specified.

Between the Multi decision and the Synchronizing Join, all shapes and loops are supported as long as they keep thread consistence. For instance, if there is an *And Split* there must be a corresponding Join.

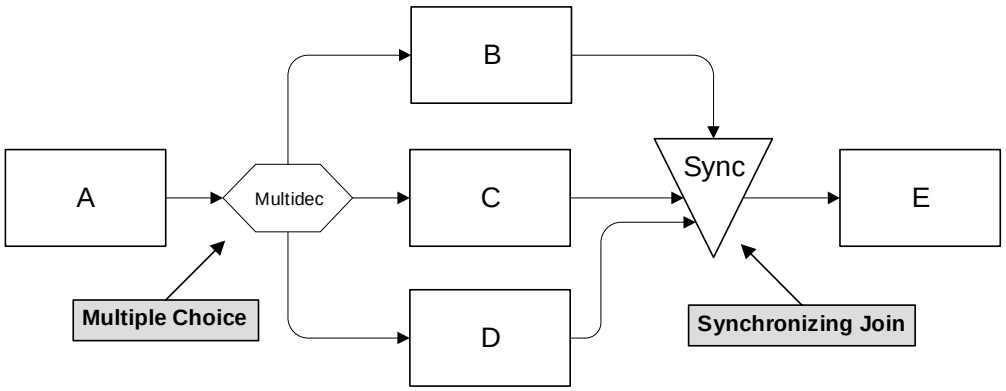
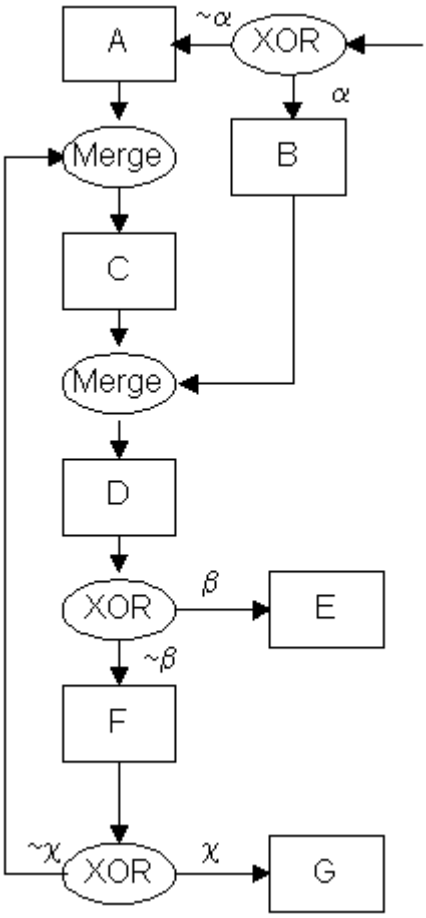


Figure 10. Synchronizing Join

Example: *Synchronizing Join* is set to synchronize the multi-decision, therefore if multi-decision activates C and D, E is activated after C and D are completed.

11. ARBITRARY CYCLES

During the workflow analysis/design time it is undesirable to be exposed to various syntactical constrains of the specific workflow enactment tool such as for example 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 when one or more activities can be done repeatedly.

BizAgi Support: In BizAgi one or more activities can be done repeatedly with no restriction for a single entry point.

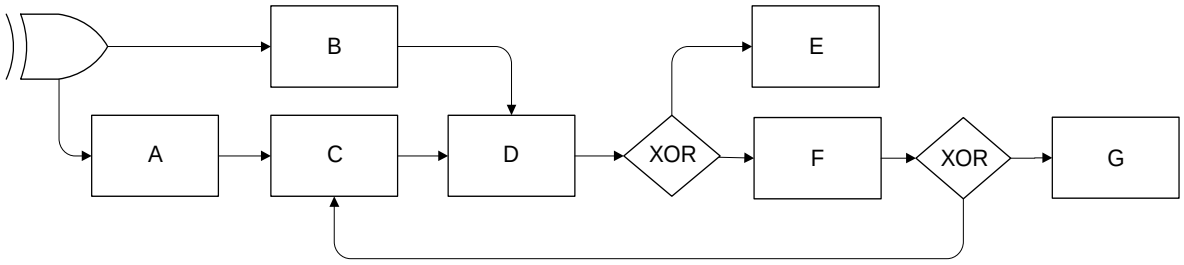


Figure 11. Arbitrary Cycles

Example: Activities C, D and F can be done repeatedly.

12. IMPLICIT TERMINATION

Another example of the requirement imposed by some of the workflow engines on a modeller 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).

BizAgi Support: In BizAgi there is only one End, but BizAgi supports the concept of terminating the process once there is nothing else to be done. This is using the *token collector*.

If there is nothing else to do in a branch, this one finishes in a token collector. Once the token collector is activated and there aren't pending activities, the process is closed automatically.

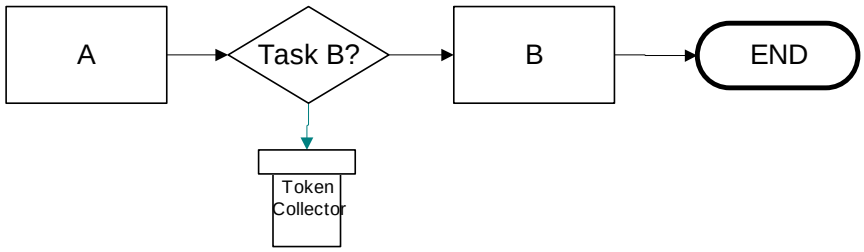


Figure 12. Implicit Termination

Example: If B doesn't have to be performed token Collector is activated; since there are no more pending activities the process is closed.

13. MI WITH A PRIORI KNOWN DESIGN TIME KNOWLEDGE

This pattern supports the creation of many instances of one activity. The number of instances is known at the 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.

BizAgi Support: The activity will be included in the workflow as many times as needed, preceded by an *AndSplit.*

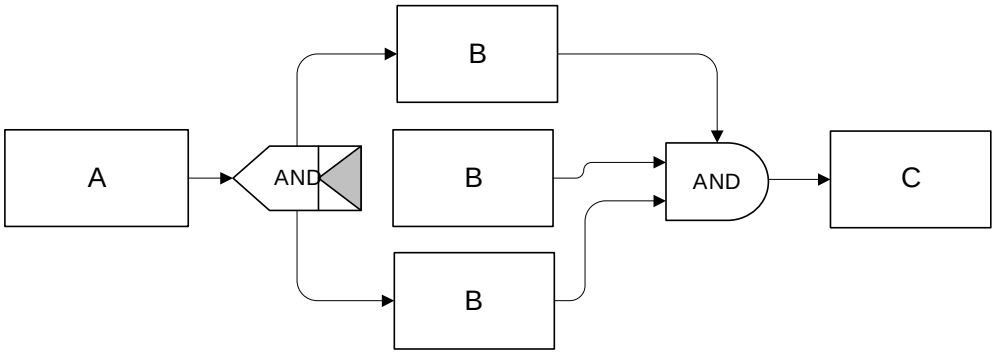


Figure 13. MI with a priori known design time knowledge

Example: B is enabled 3 times.

14. MI WITH A PRIORI RUNTIME KNOWLEDGE

This pattern supports the creation of many instances of one 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. This pattern can be thought of 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.

BizAgi Support: in BizAgi the process, information and user interface are managed integrally (the algorithms that glue all this together are called Infotuation™); business data represents the process context. BizAgi manages this pattern using data information extracted from the 1-N relationship between the process and a data entity. This relationship is specified in the multiple sub-process element.

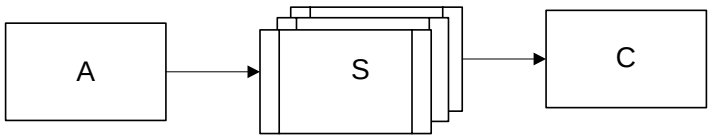


Figure 14. MI with a priori runtime knowledge

Example: Inside activity A the products to be delivered to a given client are captured. In this case we have N products for each order. In the *multiple sub process* shape it is defined to work with the Process-Products relationship. BizAgi is in charge of creating and managing the necessary sub-processes for this relationship. It's possible to specify whether to wait for all sub-processes to finish or to wait for a

given number of them. This number could be defined in design or execution time. Additionally, it is possible to group the items of the relationship (example: departments in the products).

15. MI WITH NO A PRIORI RUNTIME KNOWLEDGE

This pattern supports the creation of many instances of one 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. This pattern can be thought of 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.

BizAgi Support: In BizAgi, a sub process can be started as many times as needed. The process can wait the sub process to finish or not.

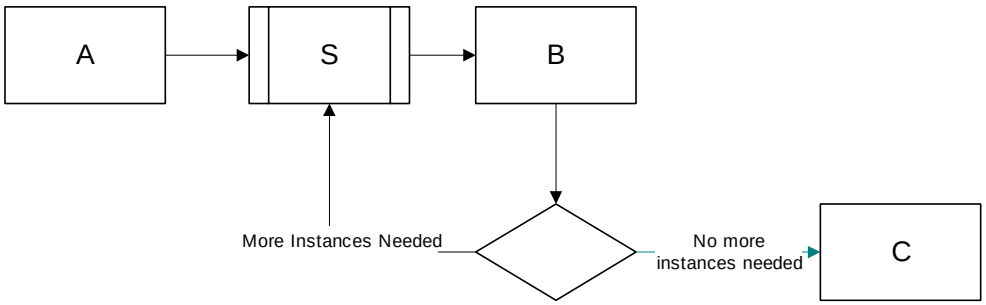


Figure 15. MI with no a priori runtime knowledge

Example: Activity A is performed and sub process S is enabled. In activity B it is defined whether it is necessary to keep enabling S and so on, until B defines that S should not be enabled.

16. MI REQUIRING SYNCHRONIZATION

There are other multiple-instances related patterns that do not consider the synchronization of created instances. For example, spawning off a variable number of sub process from the main process, as supported by Visual WorkFlo and I-Flow, 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.

BizAgi Support: The *multiple sub process* shape is used in the same manner than in pattern 14. In BizAgi it is possible to determine how to make the synchronization of the created instances or even to wait for none or all of them, in design or run time.

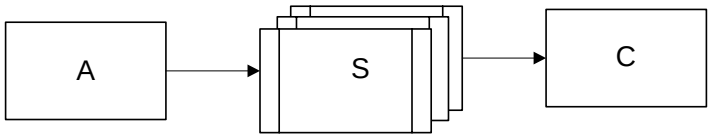


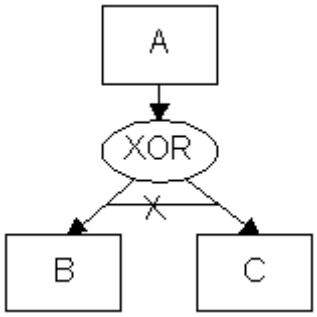
Figure 16. MI requiring synchronization

Example: In activity A it is defined that N instances should be created. Activity C is executed when the number of instances to be synchronized is reached.

17. DEFERRED CHOICE

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 work list). Once one of the threads is started, the other thread should be disabled (i.e. once activity A gets started, B should disappear from the work list).



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.

BizAgi Support: the shape *Choice* enables all paths leaving from it. When one of these is activated the other ones are disabled. The valid shapes after the choice are: *Activity, Singleton Activity, Wait, Event, And Join* and *Conditional Join*

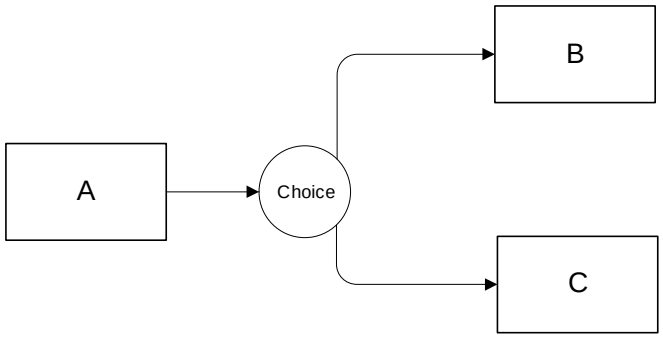


Figure 17 Deferred choice

Example: After executing activity A, the Choice enables activity B and when B is executed C is disabled (or vice versa).

18. INTERLEAVED PARALLEL ROUTING

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).

BizAgi Support: Not supported graphically, it's necessary to program this behaviour in the activity events.

19. MILESTONE

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.

BizAgi Support: This pattern can be modelled with the Choice shape.

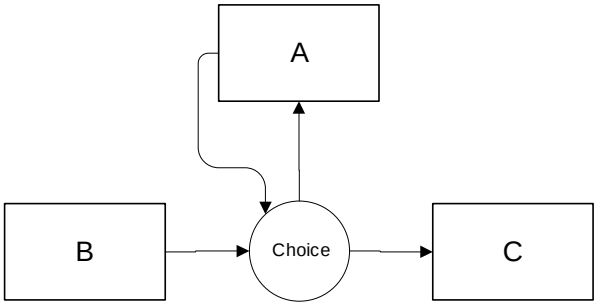


Figure 19. Milestone

20. CANCEL ACTIVITY

Typical processing after the activity is completed is to enable another activity. But some business processes require a different action to be taken. Cancel Activity pattern recognizes that it may be valid to disable another activity as a result of an activity being completed.

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

BizAgi Support: This pattern can be modelled using the *Choice* and *Event* shapes. The *Choice* enables the activity and the *event* cares for its cancellation.

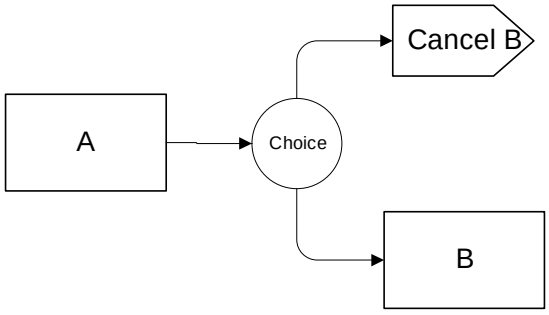


Figure 20. Cancel Activity

Example: The *Choice* enables activity B and the *Cancel* event B. Only the activity or the event can be executed. If the cancellation event is given, the activity is not executed.

21. CANCEL CASE

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

BizAgi Support: In BizAgi, when the End shape is reached the case is finished. This way, it's possible to have a cancellation event that leads to the end. If this event is triggered the case is finished.

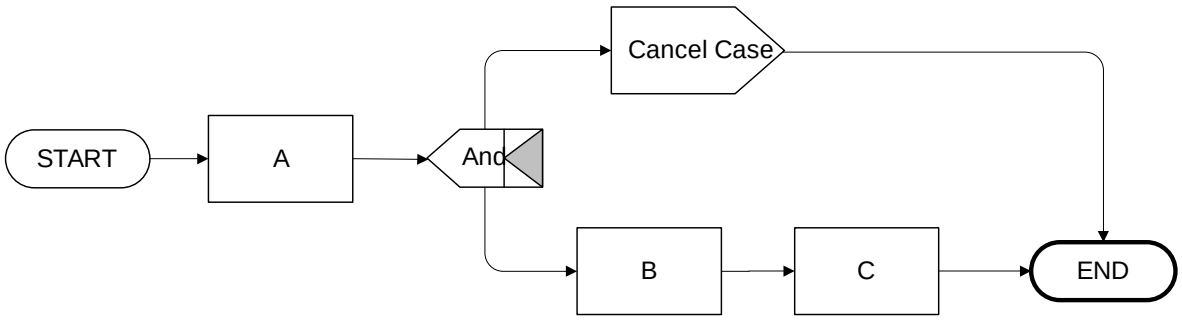


Figure 21. Cancel Case

Example: When the "Cancel case" event is thrown the case leads to "End" and then the case is finished.

Contact us:

VISION SOFTWARE

www.visionsoftware.biz

moreinfo@bizagi.com