Process Modeling Notations and Workflow Patterns

Stephen A. White
IBM Corporation

ABSTRACT

The research work of Wil van der Aalst, Arthur ter Hofstede, Bartek Kiepuszewski, and Alistair Barros has resulted in the identification of 21 patterns that describe the behavior of business processes. This paper reviews how two graphical process modeling notations, tShe BPMN Business Process Diagram from the Business Process Management Initiative (BPMI), and the UML 2.0 Activity Diagram from the Object Management Group (OMG), can represent the workflow patterns. The solutions of the two notations are compared for technical ability to represent the patterns as well as their readability.

INTRODUCTION

The research work of Wil van der Aalst, Arthur ter Hofstede, Bartek Kiepuszewski, and Alistair Barros has resulted in the identification of 21 patterns that describe the behavior of business processes. The rationale for the development of the patterns was to describe the potential capabilities that a workflow server may have during the performance of business processes. The patterns range from very simple to very complex and cover the behaviors that can be captured within most business process models. The researchers have developed a web site[1] that contains descriptions and examples of these patterns, plus supporting papers and evaluations of how workflow products support the patterns.

The purpose of this paper is to examine how two modeling notations, the BPMN Business Process Diagram from BPMI, and the UML 2.0 Activity Diagram from the OMG, can graphically represent the workflow patterns. For each of pattern, there will be a comparison of the two notations about how well they handled the pattern. The focus of the comparison will be both technical and how visually intuitive each notation represents the pattern.

BASIC CONTROL PATTERNS

The first five patterns are fairly simple examples of process behavior. They define the basic modeling patterns of business processes.

WORKFLOW PATTERN: SEQUENCE

The Sequence pattern is defined as being an ordered series of activities, with one activity starting after a previous activity has completed. The WfMC[2] defines this behavior as a “Sequential Routing.”

Business Process Diagram

A Business Process Diagram defines this pattern as a series of activities connected by Sequence Flow (see Figure 1). The direction of the Sequence Flow arrowheads determines the order of the Sequence. The behavior of this pattern can be described by the use of a conceptual “Token” that travels down a Sequence Flow from the source object to the target object—as shown by the directionality of the arrows. For this pattern, when an activity completes, a Token will travel through the Sequence Flow from that activity to the next activity in the Sequence. There is no conditionality or any other type of control put upon the Token. In BPMN, any control placed upon the flow of Tokens will be indicated by a diamond shape, with either a Gateway object or a conditional Sequence Flow. Examples of the flow control will be seen in the other workflow patterns.

Tokens will be used to describe the behavior of the patterns covered in this paper.

[1] http://tmitwww.tm.tue.nl/research/patterns/patterns.htm

[2] The Workflow Management Coalition Terminology & Glossary (1999)

Figure 1: WP #1: Sequence—Business Process Diagram

Activity Diagram

The UML Activity Diagram defines this pattern as a series of activities connected by control flow (see Figure 2). The direction of the control flow arrowheads determines the order of the Sequence. As with the Business Process Diagram, when an activity completes, a Token will travel through the control flow from that activity to the next activity in the Sequence.

Figure 2: WP #1: Sequence—Activity Diagram

Comparison

There is no major difference between BPMN and UML in diagramming the Sequence pattern. Both notations use rounded rectangles to notate activities. The exact degree of rounding for the corners of the rectangles is up to the modeler or tool vendor. Both use solid, directed lines to show the direction of flow. However, the arrowheads are different between the two standards. UML uses a line arrowhead for control flow and standard object flow, while BPMN uses a solid arrowhead for its Sequence Flow. UML does use a solid arrowhead, but this is used for streaming object flow edges—which means that multiple object Tokens can enter the activity without generating new instances of that activity. This type of behavior handled through Association links in BPMN (refer to the respective UML and BPMN specifications for more details on these types of behaviors). The arrowhead differences will apply to all the patterns described below.

WORKFLOW PATTERN: PARALLEL SPLIT

The Parallel Split pattern is defined as being a mechanism that will allow activities to be performed concurrently, rather than serially. A single path through the process is split into two or more paths so that two or more activities will start at the same time. The WfMC defines this behavior as an “AND-Split.”

Business Process Diagram

A BPMN Business Process Diagram provides three mechanisms for creating the Parallel Split pattern. The first mechanism allows that a flow object can have two or more outgoing Sequence Flow (see Figure 3). A special flow control object is not required to fork the path, since it is considered uncontrolled flow; that is, flow will proceed down each path without any dependencies or conditions—i.e., there is no Gateway that controls the flow. A Token will be generated for each of the outgoing Sequence Flow. Forking Sequence Flow can be generated from a Task, Sub-Process, or a Start Event.

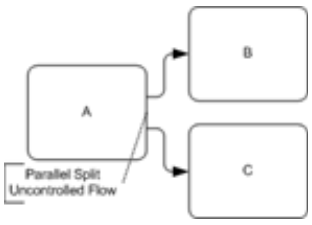

Figure 3: WP #2: Parallel Split—Business Process Diagram, Version 1

The second mechanism uses a Parallel Gateway (A diamond with a plus sign internal marker—see Figure 4). For situations as shown in Figure 4, a Gateway is not required, since the same behavior can be created through multiple outgoing Sequence Flow, as shown in Figure 3. However, some modelers and modeling tools may use a forking Gateway as a “best practice.” After the Token arrives at the Gateway, there will be a Token immediately sent down each of the outgoing Sequence Flow.

There are other situations where the Parallel Gateway would be required, such as a Parallel Split that follows an Exclusive Choice (see Figure 10 for an example of an Exclusive Choice).

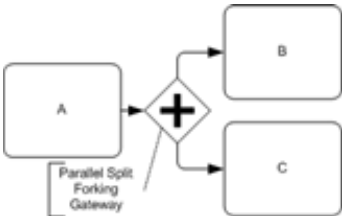

Figure 4: WP #2: Parallel Split—Business Process Diagram, Version 2

A third mechanism can be used to create the Parallel Split pattern. If a Process or a Sub-Process does not have a Start Event, which is optional, then any activity that does not have any incoming Sequence Flow will start when the Sub-Process starts. A Token will be provided for each of the starting activities. The general idea is that the modeler can create a "parallel box," as seen in Figure 5, and create a parallel situation with a compact and visually distinct technique.

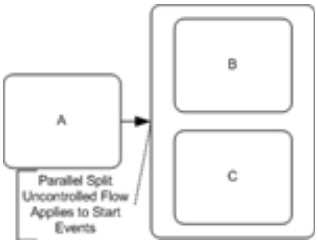

Figure 5: WP #2: Parallel Split—Business Process Diagram, Version 3

Activity Diagram

The UML Activity Diagram uses a fork node to create a set of parallel paths (a vertical or horizontal bar—see Figure 6). The bar shown in the figure indicates that all outgoing control flow from the bar will create a set of parallel flows. A Token will be generated for each of the outgoing control flow. The target activities for each of those flows will be available to start at the same time.

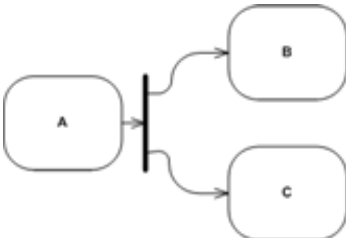

Figure 6: WP #2: Parallel Split—Activity Diagram

Comparison

The mechanisms for providing the Parallel Split pattern are different between the two notations. BPMN provides much simpler mechanisms with the multiple outgoing Sequence Flow and the "parallel box." If a control object is required or desired, then a parallel Gateway is available. UML requires that a graphical object, a fork node, create the Parallel Split. In general, the approach for controlling flow is different between the two notations. BPMN employs a single shape, a diamond, to represent any control or constraint on the flow of Tokens between activities. Internal markers indicate the exact type of control (e.g., alternative or parallel splits). The diamond shape was chosen since it is widely recognized as a decision object in flow charting. The BPMN approach provides for a smaller set of core modeling objects. UML employs two different types of objects (a diamond and a bar) to represent control of Token flow. This provides a easier distinguishing between the two basic types of control (alternative and parallel), but when more complex control is required, such as a N out of M Join or a Synchronizing Merge, which contain both alternative an parallel behavior, then the distinction becomes less clear.

WORKFLOW PATTERN: SYNCHRONIZATION

The Synchronization pattern combines the paths that were generated by a Parallel Split pattern. The final set of activities within the flows must be completed before the process can continue. This is the "synchronization" of the parallel paths. The WfMC defines this behavior as an "AND-Join."

Business Process Diagram

In BPMN, there are two mechanisms for the Synchronization pattern. The first mechanism involves the use of a Parallel Gateway (see Figure 7). This Gateway also functions for the Parallel Split pattern (see Figure 4), and can perform both functions simultaneously. The Gateway will accept multiple incoming Sequence Flow and wait for a Token to arrive from each flow before a single Token will continue past the Gateway. This mechanism is used for both of the first two types of Parallel Split patterns (see Figure 3 and Figure 4). A Gateway must be used in this case since the flow of the Process is being controlled (synchronized).

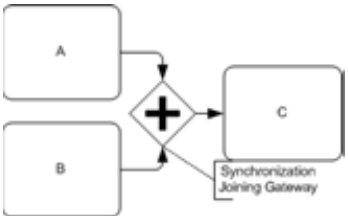

Figure 7: WP #3: Synchronization—Business Process Diagram—Version 1

The second mechanism is the closing half of the "parallel box" technique shown in Figure 5. If a Process or a Sub-Process does not have an End Event, then the Sub-Process ends when all the activities end that do not have any outgoing Sequence Flow. The Sub-Process will not be complete until all the internal activities have been completed. Then a Token will then continue past the Sub-Process within the overall Process. The "parallel box" mechanism is really a subset of the general mechanism for ending a Sub-Process. If there are parallel paths within a Sub-Process, then each of those paths can end in a separate, non-terminating End Event. The Sub-Process will not be complete until all the parallel paths have reached an End Event. This synchronizes the flow, which will then move up to the higher level process.

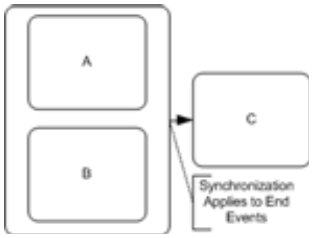

Figure 8: WP #3: Synchronization—Business Process Diagram—Version 2

Activity Diagram

The UML Activity Diagram uses a join node to combine a set of parallel paths (see Figure 9). The bar shown in the figure looks the same as the fork node (see Figure 6), and in fact, a single bar can perform both functions. The join bar indicates that Tokens must arrive from all the bar's incoming control flows. Then, a single Token will continue past the bar.

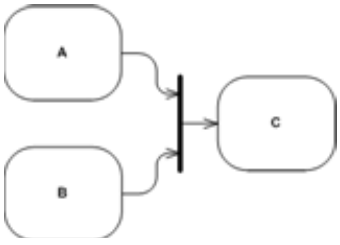

Figure 9: WP #3: Synchronization—Activity Diagram

Another mechanism that is available for an Activity Diagram utilizes a sub-activity. If there are parallel paths within a sub-activity, then each of those paths can end in a separate, non-terminating flow final node. The Sub-Process will not be complete until all the parallel paths have reached a flow final node. This synchronizes the flow, which will then move up to the higher level activity.

Comparison

The mechanisms for providing the Synchronization pattern are basically the same for the two notations, but they look different. A Business Process Diagram employs a parallel Gateway and an Activity Diagram employs a join node. Both notations also provide synchronizing behavior for the completion of a Sub-Process. The respective benefits of the flow control objects were discussed in the comparison of the Parallel Split pattern.

WORKFLOW PATTERN: EXCLUSIVE CHOICE

The Exclusive Choice pattern is defined as being a location in a process where the flow is "split" into two or more exclusive alternative paths. The pattern is exclusive in that only one of the alternative paths may be chosen for the Process to continue. The WfMC defines this behavior as an "OR-Split."

Business Process Diagram

Since additional control is needed to create the Exclusive Choice pattern, BPMN uses a Gateway. Specifically, an Exclusive Data-Based Gateway is used for this pattern (a diamond—see Figure 10). The outgoing Sequence Flow from the Gateway will have Boolean expressions that will be evaluated to determine which Sequence Flow should be used to continue the Process. When a Token arrives at the Gateway, the expressions will be evaluated (in a predetermined order) and for the first expression that is determined to be true, the corresponding Sequence Flow will be chosen and the Token will continue down that path. Only one Token will exit the Gateway for each Token that arrives at the Gateway.

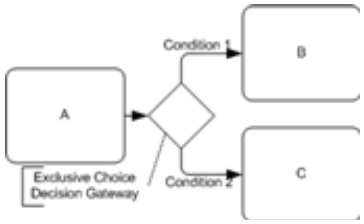

Figure 10: WP #4: Exclusive Choice—Business Process Diagram

Activity Diagram

The UML Activity Diagram uses a decision node to create a set of exclusive alternative paths (see Figure 11). The outgoing control flow from the decision node will have Boolean expressions that will be evaluated to determine which control flow should be used to continue the process. When a Token arrives at the node, the expressions will be evaluated (in a non-determined order) and for the expression that is determined to be true, the corresponding control flow will be chosen and the Token will continue down that path. Only one Token will exit the decision node for each Token that arrives at the node.

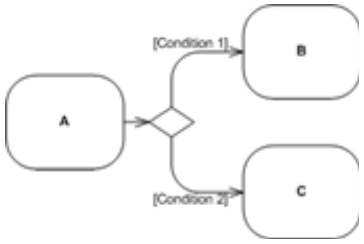

Figure 11: WP #4: Exclusive Choice—Activity Diagram

Comparison

This pattern is an example of where the two notations represent the behavior in virtually the same way. Both notations use a diamond to represent this type of decision.

WORKFLOW PATTERN: SIMPLE MERGE

The Simple Merge Pattern is defined as being a location in a process where a set of alternative paths is joined into a single path. The WfMC defines this behavior as an "OR-Join."

Business Process Diagram

The Simple Merge pattern is another example of uncontrolled flow within BPMN. That is, a Token can proceed between two activities without any intervening control mechanism (Gateway). Multiple incoming Sequence Flow are allowed (see Figure 12) since the expected behavior is that only one of the incoming Sequence Flow is expected to have a Token at performance time. When the Token arrives, the activity will immediately be started.

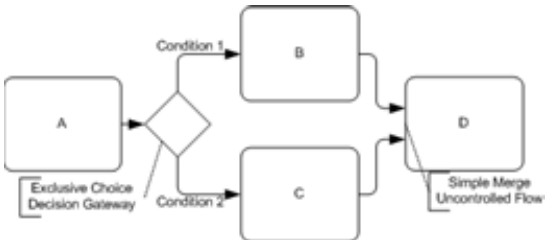

Figure 12: WP #5: Simple Merge—Business Process Diagram, Variation 1

An Exclusive Gateway can also be used to merge alternative paths (see Figure 13). Modelers may want to use them as a "best practice." When a Token arrives at the Gateway, and only one will be expected, it will immediately continue down the outgoing Sequence Flow. It should be noted that if the model creates multiple Tokens arriving at the Gateway, the Gateway will act as a Discriminator (see below).

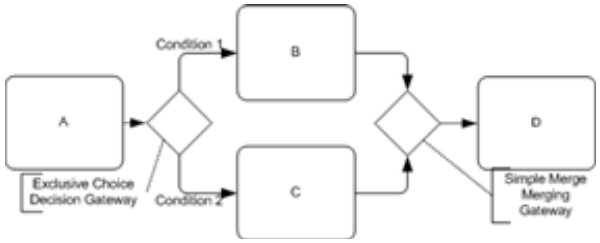

Figure 13: WP #5: Simple Merge—Business Process Diagram, Variation 2

Activity Diagram

The UML Activity Diagram uses a merge node to converge a set of alternative paths (see Figure 14). The diamond shown in the figure looks the same as the decision node, and in fact, a single diamond can perform both functions. When a Token arrives at the node, it will immediately continue down the outgoing control flow. Technically, a merge node is not required and multiple incoming control flow is allowed for an activity, but this is not considered the appropriate method to model Activity Diagram behavior.

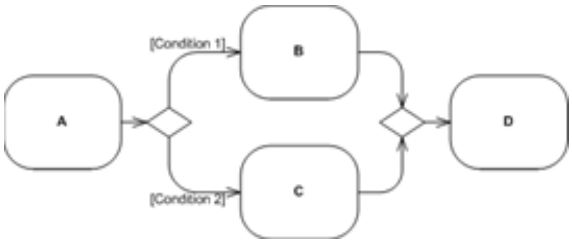

Figure 14: WP #5: Simple Merge—Activity Diagram

Comparison

As a standard practice, a Business Process Diagram provides a simpler mechanism for the Simple Merge pattern: multiple incoming Sequence Flow. An Exclusive Gateway can be used if desired, and this would make the Business Process Diagram equivalent to the Activity Diagram solution.

ADVANCED BRANCHING AND SYNCHRONIZATION PATTERNS

The next five patterns describe more complex ways of splitting and joining the flow of a business process.

WORKFLOW PATTERN: MULTIPLE CHOICE

The Multiple Choice pattern differs from the Exclusive Choice pattern in that the Multiple Choice pattern allows from one to all of the alternative paths may be chosen at performance-time. Technically, a Multiple Choice pattern may allow zero paths chosen, but this could be considered an invalid situation where the process flow stops unexpectedly.

Business Process Diagram

BPMN provides two mechanisms for handling the Multiple Choice pattern:

The first mechanism involves conditions placed on the outgoing Sequence Flow from an activity. A Gateway object is not used, but since there is control of the flow being affected by the conditions, mini-gateways (small diamonds) are shown at the beginning of the Sequence Flow lines. To ensure flow integrity of a model, BPMN requires that there be multiple outgoing Sequence Flow if any one of them is conditional and that there must be at least one valid Sequence Flow at performance-time. Tokens will be generated for each outgoing Sequence Flow whose condition proves to be true. Thus, in the diagram shown below, one or both of the outgoing Sequence Flow from Task “A” will be chosen.

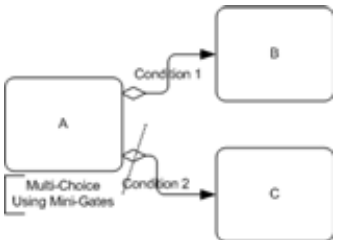

Figure 15: WP #6: Multiple Choice—Business Process Diagram -- Variation 1

The second mechanism uses an Inclusive Gateway (a diamond with a circle internal marker—see Figure 16). The outgoing Sequence Flow from the Gateway will have Boolean expressions that will be evaluated to determine which Sequence Flow should be used to continue the Process. When a Token arrives at the Gateway, all the expressions will be evaluated and for each expression that is determined to be true, the corresponding Sequence Flow will be chosen and a Token will continue down that path. For a single incoming Token, there may be a Token generated for one to all of the outgoing Sequence Flow.

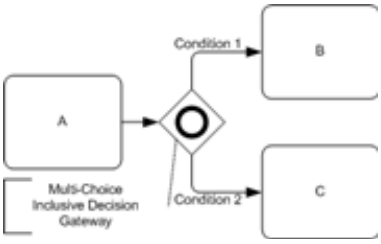

Figure 16: WP #6: Multiple Choice—Business Process Diagram—Variation 2

Activity Diagram

The UML Activity Diagram uses a fork node where the outgoing control flow have guards to create the Multiple Choice pattern (see Figure 17). The fork node creates a set of parallel paths. When a Token arrives at the node, it will be divided into multiple Tokens for only the outgoing control flow whose guards are evaluated to True. Technically, a fork node is not required and the guards can be placed on multiple outgoing control flow from an activity, but this is not considered the appropriate method to model a Simple Merge pattern.

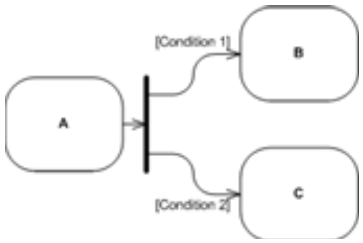

Figure 17: WP #6: Multiple Choice—Activity Diagram

Comparison

The basic difference between the two notations is that a Business Process Diagram uses a variation of the Gateway (diamond) for this pattern and the Activity Diagram uses a variation of the fork node for this pattern. For the Activity Diagram, it is a bit confusing, since the behavior involves decisions as well as parallelism, but a decision diamond is not used. The Business Process Diagram uses a variation of a decision diamond to show that it is a decision point, but also uses an internal marker to indicate that the decision is not exclusive.

WORKFLOW PATTERN: MULTIPLE MERGE

The Multiple Merge pattern is defined as a location in the process where there are multiple paths merging, but without any control of the flow to Tokens. When a Token arrives at an activity, that activity will be instantiated. Thus, if there are multiple paths converging on an activity without any Token control, it is possible that the activity will be instantiated one time for each of the paths that are merged. Furthermore, the Tokens will all continue independently through the remainder of the process (if any).

Thus, the Multiple Merge pattern differs from Synchronization or Simple Merge patterns in that for these patterns the target activity will only be instantiated once. Given this, the Multiple Merge pattern is potentially confusing for a business person to be able to understand the behavior of a section of the process being implicitly duplicated, but it does provide a potentially useful modeling technique for specific situations.

Business Process Diagram

Since the Multiple Merge pattern involves an uncontrolled flow situation, the BPMN solution will not employ a Gateway. BPMN allows multiple incoming Sequence Flow into an activity and this will provide the location of the Multiple Merge (see Figure 18). For each Token that arrives down an incoming Sequence Flow, a separate instance of the activity will be generated.

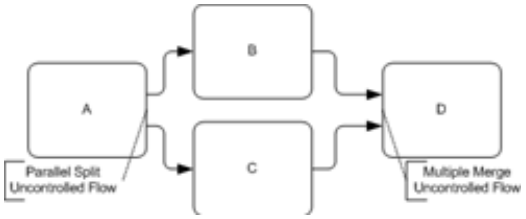

Figure 18: WP #7: Multiple Merge—Business Process Diagram

Activity Diagram

The UML Activity Diagram uses a merge node to converge a set of paths (see Figure 19). The diamond shown in the figure looks the same as the decision node, and in fact, a single diamond can perform both functions. When a Token arrives at the node, it will immediately continue down the outgoing control flow. Unlike the Exclusive Gateway in BPMN, the merge node is merely a graphical merging and it will not stop any other Tokens that may happen to arrive at the node. Thus, if the original source of the paths coming into the merge node was a fork node (as in the figure below), then the activity that follows the merge node will be instantiated multiple times. Technically, a merge node is not required and multiple incoming control flow is allowed for an activity, but this is not considered the appropriate method to model Activity Diagram behavior.

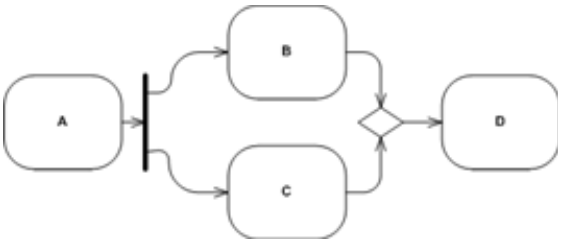

Figure 19: WP #7: Multiple Merge—Activity Diagram

Comparison

As a standard practice, a Business Process Diagram provides a simpler mechanism for representing this pattern. However, this pattern can be potentially confusing for either notation. Thus, the modeler should probably add annotations to the model explaining the behavior.

WORKFLOW PATTERN: DISCRIMINATOR

The Discriminator pattern is another way of combining the paths that were generated from a Parallel Split pattern. It is the point where a set of parallel flows come back together. It differs from the Synchronization pattern in that the process will continue pass the Discriminator location once the first Token arrives there. All remaining Tokens that were generated from the Parallel Split will eventually arrive at the Discriminator, but they will be blocked.

Business Process Diagram

A Business Process Diagram uses an Exclusive Gateway to create the behavior of the Discriminator pattern (see Figure 20). Since the Gateway is exclusive, it will exclude all but one of the Tokens that arrive at the Gateway. The Gateway is defined as allowing the first Token through and blocking (consuming) the remaining Tokens.

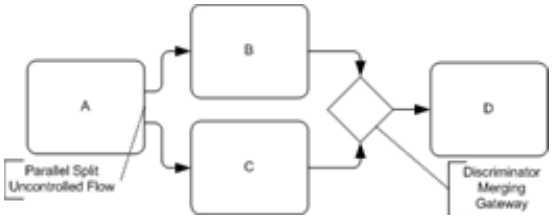

Figure 20: WP #8: Discriminator—Business Process Diagram

Activity Diagram

The UML Activity Diagram uses a combination of features to create the behavior of the Discriminator pattern (see Figure 21). A join node is used to combine the paths. A condition can be set for the join node to test whether any of the source activities have completed. When the first Token arrives at the join, a Token will continue on the outgoing control flow. Any other Tokens arriving at the join will be stopped.

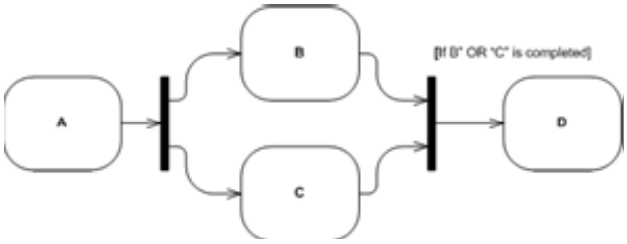

Figure 21: WP #8: Discriminator—Activity Diagram

Comparison

A Business Process Diagram automatically provides the behavior of this pattern with the defined merging behavior of an Exclusive Gateway. The Activity Diagram does not have a predefined mechanism for this pattern. However, a

modeler can create the behavior by the use of carefully constructed guard conditions on incoming control flow for a merge node. Given this, BPMN provides a much simpler and more natural way to represent this pattern.

WORKFLOW PATTERN: N OUT OF M JOIN

The behavior of the N out of M Join pattern lies somewhere between Synchronization and Discriminator patterns. In this case, instead of one or all of the incoming parallel Tokens being required to continue past the point of joining, the N out of M Join allows the modeler to define how many of the incoming Tokens are necessary to continue. All remaining Tokens will be stopped (like the Discriminator).

Business Process Diagram

A Business Process Diagram uses a Complex Gateway to provide the N out of M Join pattern behavior (a diamond with a "asterisk" internal marker—see Figure 22). In fact, the Complex Gateway was included in BPMN for this type of situation. The modeler can add an expression to the Gateway that will determine how many Tokens must arrive from its incoming Sequence Flow, before a Token will be sent down its outgoing Sequence Flow. Any additional Tokens that arrive later will be blocked at the Gateway.

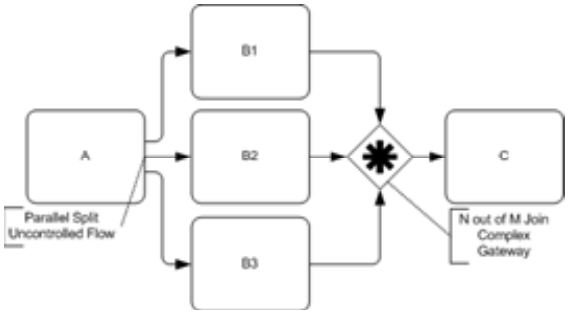

Figure 22: WP #9: N out of M Join—Business Process Diagram

Activity Diagram

The Activity Diagram uses a join node to combine a set of parallel paths (see Figure 23). As with the Synchronization pattern, the join bar indicates that Tokens must arrive from all incoming control flow for the bar. However, a condition can be added to the join that will determine how many Tokens are required before a single Token will continue past the bar. Any additional Tokens that arrive later will be blocked at the join node.

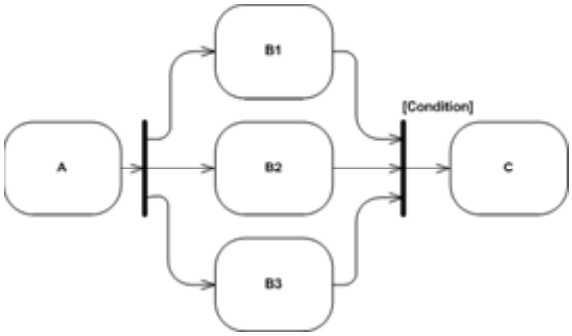

Figure 23: WP #9: N out of M Join—Activity Diagram

Comparison

The comparison between the two notations is basically the same as for the other types of flow control mechanism used in the workflow patterns. A Business Process Diagram uses variations of a diamond shape to indicate the appropriate behavior. An Activity Diagram uses either a diamond or a bar for flow control. The merits of these mechanisms have been discussed above.

WORKFLOW PATTERN: SYNCHRONIZING MERGE

This pattern is a variation of the Synchronization pattern. The intent is to synchronize the Tokens from all the parallel paths that are incoming to an activity. The challenge is that it will not be known ahead of time how many Tokens will actually be arriving. This situation is created by the Multiple Choice pattern. Thus, the Synchronizing Merge pattern must be able to determine how many Tokens were generated upstream, then synchronize those Tokens, but not wait for any other Tokens.

Business Process Diagram

The Business Process Diagram uses an Inclusive Gateway (see Figure 10) to merge the paths created by an upstream Inclusive Gateway.

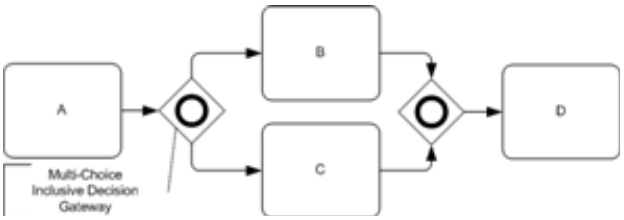

Figure 24: WP #10: Synchronizing Merge—Business Process Diagram

Activity Diagram

The UML Activity Diagram uses a join node for the Synchronizing Merge pattern (see Figure 25). The join node with a condition expression that controls how many Tokens must arrive from the incoming control flow before a Token will continue through the outgoing control flow.

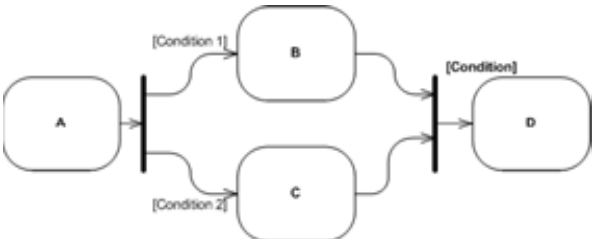

Figure 25: WP #10: Synchronizing Merge—Activity Diagram, Variation 1

Comparison

As with the previous pattern, the comparison between the two notations is basically the same as for the other types of flow control mechanism used in the workflow patterns. A Business Process Diagram uses variations of a diamond shape to indicate the appropriate behavior. An Activity Diagram uses either a diamond or a bar for flow control. The merits of these mechanisms have been discussed above.

STRUCTURAL PATTERNS

The two patterns in this group cover such behavior as looping and the independence of separate process paths.

WORKFLOW PATTERN: ARBITRARY CYCLES

The Arbitrary Cycle pattern is a mechanism for allowing sections of a process to be repeated—it is a loop. This pattern allows looping that is unstructured or not block structured. That is, the looping segment of the process may allow more than one entry or exit point. This pattern is important for the visualization of valid, but complex, looping situations in a single diagram. Notations that allow only block structured loops would not be able to display the entire process in a single diagram or process level or would distribute the behavior in a non-intuitive manner.

Business Process Diagram

It is possible to create an Arbitrary Cycle pattern within a Business Process Diagram by connecting Sequence Flow to upstream activities (see Figure 26).

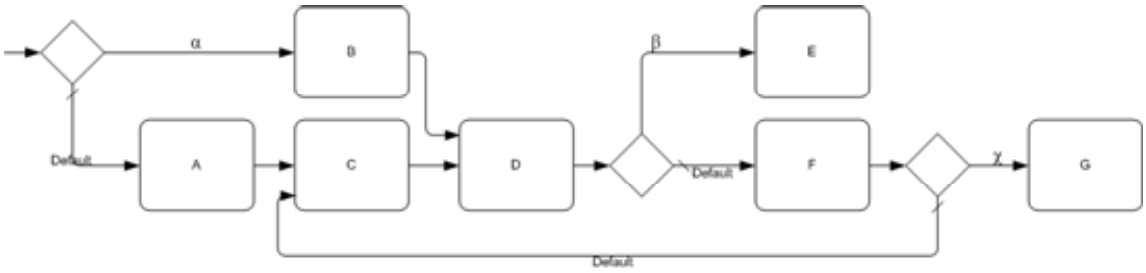

Figure 26: WP #11: Arbitrary Cycles—Business Process Diagram

Activity Diagram

It is possible to create Arbitrary Cycle patterns within an Activity Diagram by connecting control flow to upstream activities (see Figure 27).

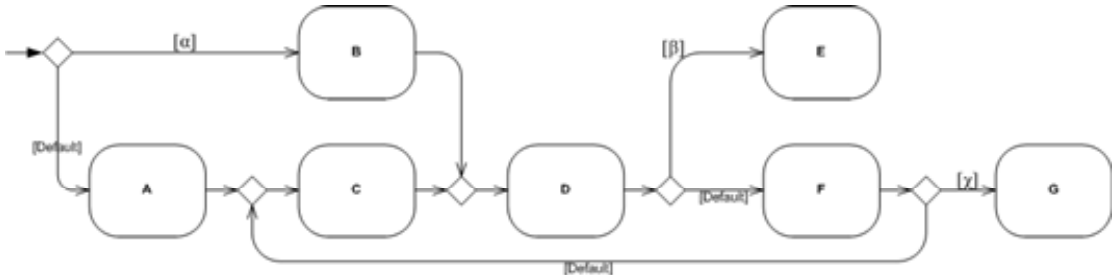

Figure 27: WP #11: Arbitrary Cycles—Activity Diagram

Comparison

There is no major difference between BPMN and UML in diagramming this pattern. Both notations allow unstructured looping by connecting downstream activities to upstream activities in a format that is not block-structured. A minor difference is that BPMN does not use merging Gateways for each location in the process where there are multiple alternative incoming Sequence Flow. The Activity Diagram uses a merge node in each of these situations. Thus, the Activity Diagram will use a few more objects and lines than the Business Process Diagram to depict the same behavior.

WORKFLOW PATTERN: IMPLICIT TERMINATION

This pattern allows that a specific path of a process be concluded without other parallel paths be required to end as well. This is opposed to some notations that required the whole process to end when any end node is reached. Many notations, such as the Business Process Diagram and Activity Diagram provide both functionalities.

Business Process Diagram

The End Event (a circle with a thick border) in a Business Process Diagram is an indicator that a particular path has completed (see Figure 28). BPMN has seven variations of the End Event and all of them, except one, provides for the Implicit Termination pattern. The variations of the End Event all have internal markers to indicate a particular result of ending the path. The Terminate End Event, as its name indicates, will cause the entire process to stop, even if there are activities that have not started or are still active.

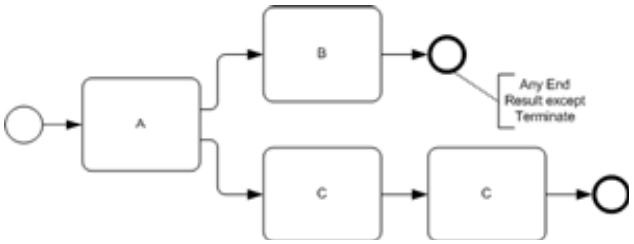

Figure 28: WP #12: Implicit Termination—Business Process Diagram

Activity Diagram

The Flow Final node (a circle with an "X" internal marker) in an Activity Diagram is an indicator that a particular path has completed and this provides for the Implicit Termination pattern (see Figure 29). The Activity Final node (a circle with a smaller filled circle internal marker) will cause the entire process to stop, even if there are activities that have not started or are still active.

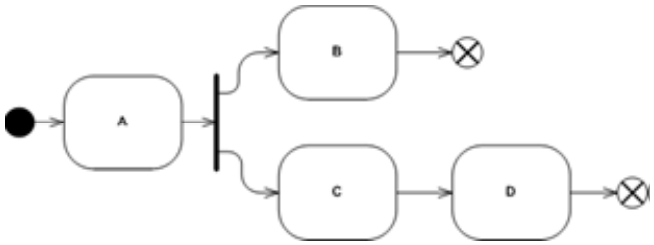

Figure 29: WP #12: Implicit Termination—Activity Diagram

Comparison

The two notations provide similar mechanisms for diagramming this pattern. Both notations allow the Implicit Termination pattern in a fairly straightforward way. A difference is that BPMN allows a modeler to provide additional business information about the result of completing a particular path in the Process. Different variations of the BPMN End Event will show that the end of the path results in a message, exception, transaction cancellation, compensation, link to another process, or a combination of results.

PATTERNS INVOLVING MULTIPLE INSTANCES

The following four patterns describe how multiple instances or copies of activities are created. In some cases, the patterns can be handled by standard looping patterns. However, many of the situations described are more complex than standard looping.

WORKFLOW PATTERN: MI WITH A PRIORI DESIGN TIME KNOWLEDGE

This pattern describes how an activity can be instantiated a known number of times, and in parallel. That is, the modeler knows how many times the activity should be performed and defines that number in the process model. The definition of the pattern does not describe exactly what happens after the copies of the activity have been started. If the process should synchronize all the copies of the activities (and their Tokens) before continuing, then the MI requiring Synchronization pattern should be used. If then process can continue separately for each copy of the activity (i.e., allow all Tokens to continue independently), then a variation of the MI with no A Priori Knowledge pattern should be used.

Business Process Diagram

An activity in a Business Process Diagram can have multiple copies by setting the LoopType attribute of the activity to Multi-Instance and the MI_InstanceGeneration attribute to Parallel (a pair of parallel lines will be shown in the bottom center of the activity—see Figure 30). The MI_Condition attribute of the activity can then be set to a static number to create the behavior of this pattern.

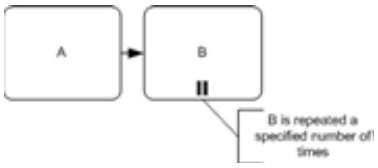

Figure 30: WP #13: MI with A Priori Design Time Knowledge—Business Process Diagram

Activity Diagram

The Activity Diagram uses an expansion region around an activity to create the multiple copies of the activity (see Figure 31). This is indicated by the input and output expansion nodes (the set of small rectangles) bounding the activity. Note that this is a more compact way of showing an expansion region with a single activity inside. The concurrency attribute will be set to parallel to create parallel performance of the activity copies. The number of

elements in the region's input collection will determine how many copies of the contents of the region will be performed. The number of elements can be defined statically to create the behavior of this pattern.

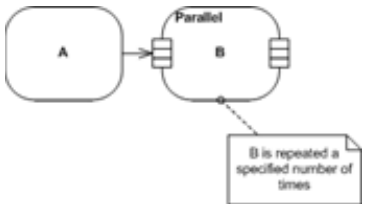

Figure 31: WP #13: MI with A Priori Design Time Knowledge—Activity Diagram

Comparison

Functionally, there is no major difference between BPMN and UML in diagramming this pattern. Both notations allow multiple instances for activities. The notation is different, but the behavior will be the same for both notations. It is possible, however, that the Activity Diagram expansion nodes might be confused with other pin configurations of an activity.

WORKFLOW PATTERN: MI WITH A PRIORI RUNTIME KNOWLEDGE

This pattern is similar to the MI with A Priori Design Time Knowledge pattern, except that the number of copies is not known until the process is being performed and cannot be set ahead of time. In addition, this pattern allows that the copes be performed sequentially as well as in parallel. The examples shown below will be for sequentially performed copies.

Business Process Diagram

An activity in a Business Process Diagram can have multiple copies by setting the LoopType attribute of the activity to Multi-Instance and the MI_InstanceGeneration attribute to serial (a looping symbol will be shown in the bottom center of the activity—see Figure 32). The LoopCondition attribute can then be set to determine dynamically the number of copies that will be performed. This behavior can also be created by using an exclusive Gateway Decision after the activity and looping back to merge with the same activity (e.g., Task "B" in the figure) for each copy of the activity.

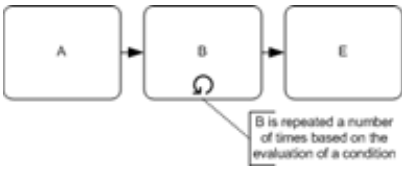

Figure 32: WP #14: MI with A Priori Runtime Knowledge—Business Process Diagram

Activity Diagram

The Activity Diagram uses an expansion region around an activity to create the multiple copies of the activity (see Figure 33). This is indicated by the input and output expansion nodes (the set of small rectangles) bounding the activity. The concurrency attribute will be set to iterative to create sequential performance of the activity copies. The number of elements in the region's input collection can dynamically determine how many copies of the contents of the region will be performed before the process continues. This behavior can also be created by using a decision node after the activity and looping back to merge with the same activity (e.g., activity "B" in the figure) for each copy of the activity.

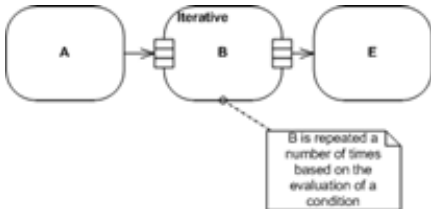

Figure 33: WP #14: MI with A Priori Runtime Knowledge—Activity Diagram

Comparison

As with the MI with A Priori Design Time Knowledge pattern, there is no major difference between BPMN and UML in diagramming this pattern. Both notations provide mechanisms for creating the variations of this pattern.

WORKFLOW PATTERN: MI WITH NO A PRIORI KNOWLEDGE

This pattern differs from the previous two patterns in that the number of copies of an activity cannot be determined before the copies are to be created. The exact number is actually determined during the performance of those copies. This means that a standard looping or multi-instance pattern will not be sufficient to create this behavior. A more complex form of looping is needed and shown in the two examples below. A specific pattern of activities and flow control mechanisms are required to create this pattern. The target activity (activity "B") must be paired with another activity (activity "C") that will determine if another copy of the target activity is required. These two activities are placed within a loop. The completion of the target activity is not required before the loop is iterated. However, all the copies of the target activity (activity "B") must be completed before the final activity (activity "E") can be started.

Business Process Diagram

A combination of Gateways and activities is used in a Business Process Diagram to create this pattern (see Figure 34). Tasks "B" and "C" are started in parallel by a Parallel Gateway, but an Exclusive Decision immediate follows Task "C." If another copy of Task "B" is required, then the Sequence Flow will connect upstream to an Exclusive Merging Gateway. The Merging Gateway is required in this situation to avoid the synchronization behavior of the Parallel Gateway.

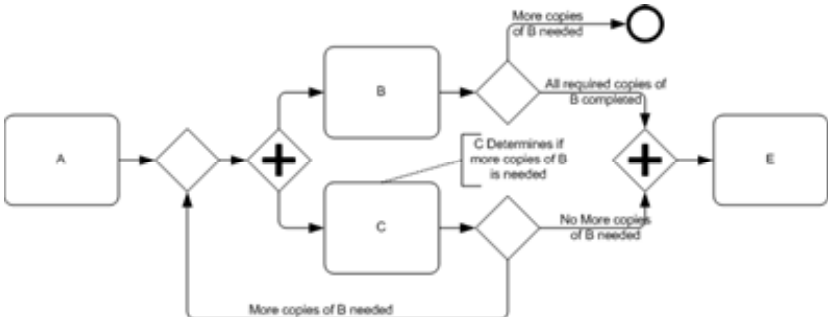

Figure 34: WP #15: MI with no A Priori Knowledge—Business Process Diagram

Activity Diagram

A combination of a decision node, merge node, fork node and activities is used in an Activity Diagram to create this pattern (see Figure 35). Activities "B" and "C" are started in parallel by a fork node, but a decision node immediate follows activity "C." If another copy of activity "B" is required, then the control flow will connect upstream to a merge node. The merge node is required to avoid the synchronization behavior of the fork node (which also can be a join node).

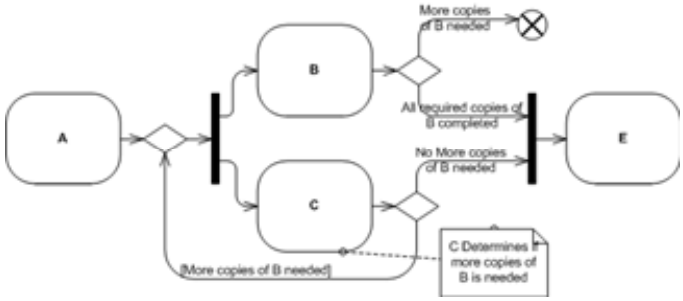

Figure 35: WP #15: MI with no A Priori Knowledge—Activity Diagram

Comparison

This pattern also handled similarly between the two notations. The pattern of objects is basically the same with only the shape of the forking mechanism being graphically different.

WORKFLOW PATTERN: MI REQUIRING SYNCHRONIZATION

This pattern is similar to the MI with A Priori Runtime Knowledge pattern except that it requires that all the copies of the repeated activity (performed in parallel) be completed before the process continues.

Business Process Diagram

An activity in a Business Process Diagram can have multiple copies by setting the LoopType attribute of the activity to Multi-Instance and the MI_InstanceGeneration attribute to parallel (a pair of parallel lines will be shown in the bottom center of the activity—see Figure 36). The LoopCondition attribute can then be set to determine dynamically the number of copies that will be performed. The MI_FlowCondition attribute must be set to “All” to insure that all the copies of the activities will be completed before the process continues (this means that only one Token will continue through the Process).

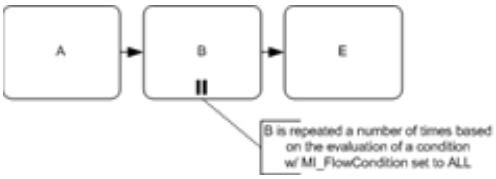

Figure 36: WP #16: MI requiring Synchronization—Business Process Diagram

Activity Diagram

The Activity Diagram uses an expansion region around an activity to create the multiple copies of the activity (see Figure 37). This is indicated by the input and output expansion nodes (the set of small rectangles) bounding the activity. The concurrency attribute will be set to parallel to create parallel performance of the activity copies The number of elements in the region’s input collection can dynamically determine how many copies of the contents of the region will be performed before the process continues. The defined behavior of the expansion region will always synchronize all the activity within the region before the process will continue past the region.

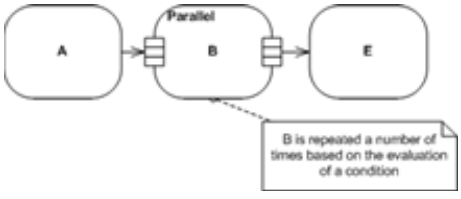

Figure 37: WP #16: MI requiring Synchronization—Activity Diagram

Comparison

The comparison for this pattern is basically the same as the comparisons for the other three multiple instance patterns.

STATE-BASED PATTERNS

The three patterns in this group define how the behavior of a business process may sometimes be affected by factors outside the direct control of the process engine.

WORKFLOW PATTERN: DEFERRED CHOICE

This pattern represents a type of exclusive decision, similar to the Exclusive Choice pattern. However, the basis for determining the path that will be taken is different. The Exclusive Choice pattern is based on the evaluation of process data. The Deferred Choice pattern is based on an event that occurs during the process. Once the event occurs (i.e., the choice is made), then the other alternative paths will be disabled.

Business Process Diagram

A Business Process Diagram uses an Exclusive Data-Based Gateway to (see Figure 38). This Decision represents a branching point where Alternatives are based on an Event that occurs at that point in the Process. This solution is unique in BPMN in that the Gateway is supported by additional objects to control the Sequence Flow. The Gateway is immediately followed by Intermediate Events. When a Token arrives at the Gateway, it will wait there. The specific Event, usually the receipt of a Message, determines which of the paths will be taken. Other types of Events can be used, such as Timer. Only one of the Alternatives will be chosen. The first Event that occurs will trigger the path that will be taken, to the exclusion of any other path from the Gateway (Note that Tasks of type Receive can also be used in place of Message Intermediate Events). Essentially, the Event “pulls” the Token from the Gateway, where it will continue down the Sequence Flow that leaves the Event.

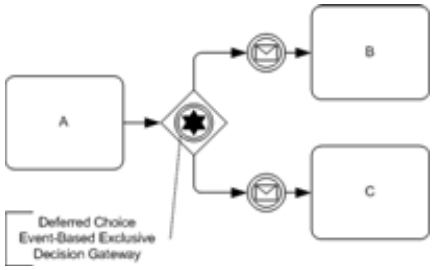

Figure 38: WP #17: Deferred Choice—Business Process Diagram

Activity Diagram

The Activity Diagram uses a special configuration of a fork node, an interruptible region, and receive signal action to provide the behavior of this pattern (see Figure 39). At the point where the choice should be made a fork node creates a number of parallel paths that equals the number of options in the choice; thus, multiple Tokens are created. These parallel paths cross into an interruptible region. Each of the parallel paths will lead to a receive signal action within the region. When the first signal arrives and interrupting edge will be used to exit the region, thereby disabling the receipt of any other signal. One of the Tokens will traverse the interrupting edge to its destination, while the other Tokens are consumed by the interruptible region.

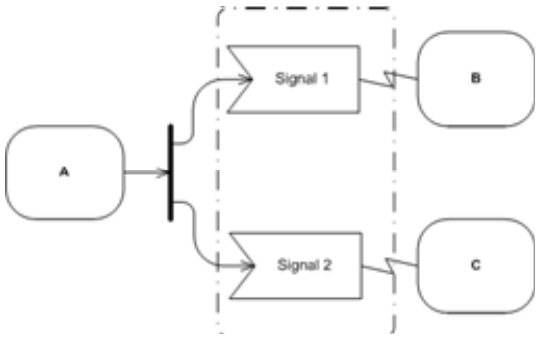

Figure 39: WP #17: Deferred Choice—Activity Diagram

Comparison

Both notations adequately provide for the behavior of this pattern. However, the mechanisms that are used are different. BPMN puts the behavior in the context of a decision and the events that drive the decision, while UML puts the behavior in the context of a fork node that generates multiple paths that are ignored except the one trigger by a signal. The use of a fork and an interrupting region does not intuitively convey the idea that a decision is being made in the process.

WORKFLOW PATTERN: INTERLEAVED PARALLEL ROUTING

The use of the word “parallel” in the name of this pattern can be a bit deceptive since the activities in the pattern must be performed sequentially, but in no specific order. The sequential performance of the activities is often due to the requirement that the activities share or update the same resources. The performers of the activities will decide the order of the activities.

Business Process Diagram

To handle this type of behavior, BPMN has the concept of an Ad-Hoc Process (a Sub-Process object with a “tilde” marker at the bottom center of the object). This type of Process is a collection of activities that can be performed in any order (see Figure 40). Tasks “B” and “D” are the part of the diagram depicting the interleaved parallel routing pattern. The Ad-Hoc Sub-Process has a modeler-defined condition that must be satisfied before it can be completed. In addition, the ordering attribute of the Ad-Hoc Process must be set to Sequential so that the activities will be performed one at a time.

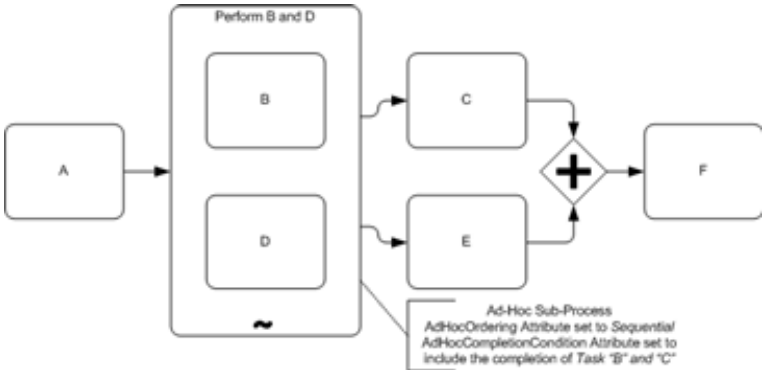

Figure 40: WP #18: Interleaved Parallel Routing—Business Process Diagram

Activity Diagram

The Activity Diagram does not have the concept of an Ad Hoc process and, thus, does not have an adequate graphical solution for the pattern. The simplest way to create this behavior would be to place the activities between a fork node and join node (see Figure 41). The activities look like they can be performed in parallel, but there must be an added constraint to each of the activities such that they are required to use the same resource. This mechanism will specify that the two activities cannot be performed at the same time, but there is no graphical indication of the constraint other than an annotation.

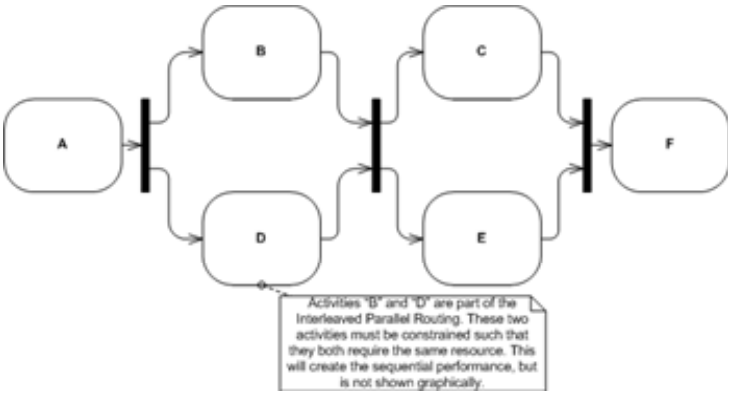

Figure 41: WP #18: Interleaved Parallel Routing—Activity Diagram, Version 1

If the modeler wants to make the behavior more explicit, it is possible to create a set of alternative paths that utilize a Deferred Choice and also uses duplicates of the activities (see Figure 42). Activities "B" and "D" are the part of the diagram depicting the interleaved parallel routing pattern. This type of configuration will, in fact, depict the desired behavior, but the diagram will be difficult to understand, especially if the number of activities in the pattern is more than two.

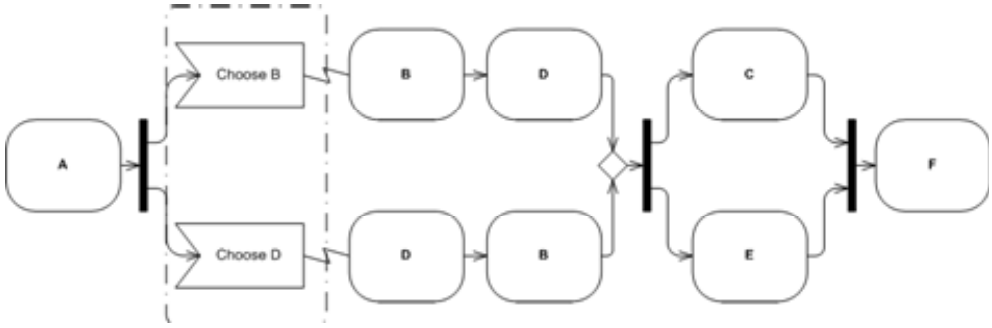

Figure 42: WP #18: Interleaved Parallel Routing—Activity Diagram, Version 2

There may be other, more complicated methods for creating Interleaved Parallel Routing with UML, especially if State Diagrams are used. However, these solutions would be very hard to understand for the average business modeler.

Comparison

This pattern shows a bigger difference between the notations than the other patterns. This is apparent since BPMN has built-in functionality for Ad-Hoc Processes and such functionality in UML must be explicitly laid out with all possible configurations and a duplication of activities. The result is that BPMN provides a compact and understandable diagram for this pattern and UML provides for the behavior, but not in a format that is readily understandable. As can be seen in Figure 41 it is difficult to tell the difference between the Interleaved Parallel Routing pattern and the Parallel Split pattern in the diagram.

WORKFLOW PATTERN: MILESTONE

There are points within a process where it is important to know whether a specific event as occurred or a condition has been met. These events or conditions can be referred to as milestones. The process model must be able to identify and react to the milestone. There is more than one way to realize this pattern, depending on the exact nature of how the milestone will be used by the process. We will review at two of the examples provided by the workflow researchers[3].

[3] http://tmitwww.tm.tue.nl/research/patterns/milestone.htm

Business Process Diagram

The example below shows how information can be passed from one part of a Business Process Diagram to another part when normal Sequence Flow cannot be used (see Figure 43). This information reflects the occurrence of a milestone. In the example, the completion of an activity (Task "B") in one Sub-Process is required before an activity (Task "B") in another Sub-Process can start. A Link End Event that follows Task "B" is used to trigger a corresponding Link Start Event that precedes Task "D." An optional Association link is used to reinforce this relationship.

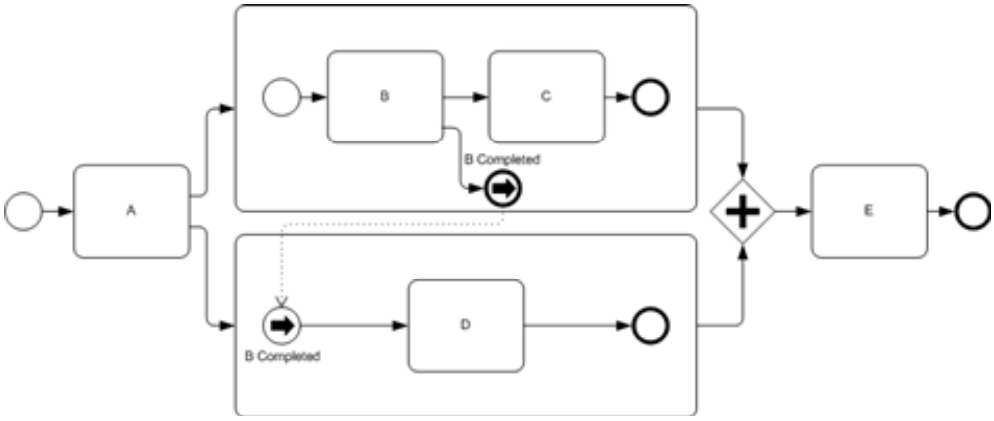

Figure 43: WP #19: Milestone—Business Process Diagram (Example 2)

Activity Diagram

The second example shows how information can be passed from one part of a process to another part when normal control flow cannot be used (see Figure 44). In the example, the completion of an activity (activity "B") in one sub-activity is required before an activity (activity "D") in another sub-activity can start. A broadcast of a signal (the rectangle with the point on the right side) that follows activity "B" is used to trigger a corresponding receipt of a signal (the rectangle with the notch on the left side) that precedes activity "D."

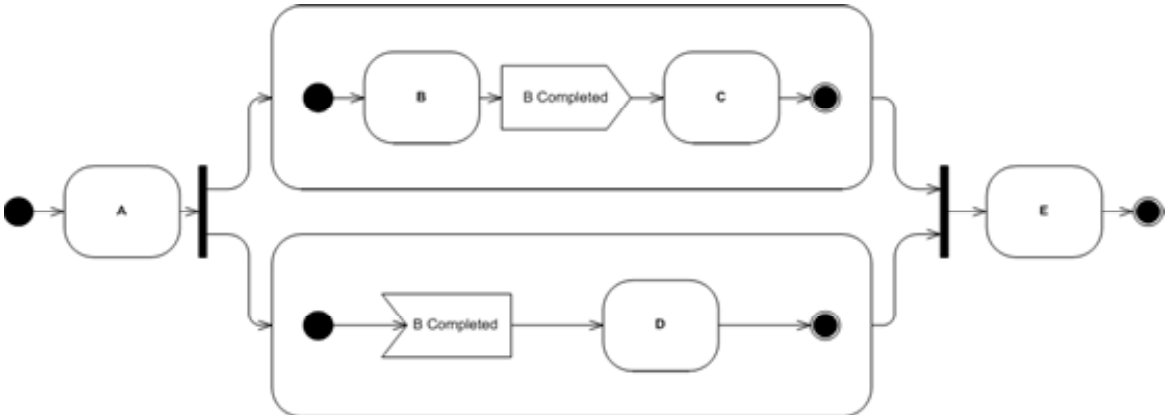

Figure 44: WP #19: Milestone—Activity Diagram (Example 2)

Comparison

Both notations can handle the behavior of this pattern adequately and with similar mechanisms. However, the first example does introduce the mechanisms for exception handling for the notations. Both mechanisms will provide the same behavior, but the BPMN mechanisms of attaching an Intermediate Event directly on the boundary of the activity is more visually descriptive than the interruptible region with receive signal actions within.

CANCELLATION PATTERNS

The following two patterns show how the completion of one activity may cause the cancellation of an activity or group of activities.

WORKFLOW PATTERN: CANCEL ACTIVITY

This pattern describes how it is possible that there are two competing activities. When one of the activities completes, it signals that the other activity should stop processing. Thus, a mechanism of signaling the cancellation and a mechanism for interrupting activities based on this signal are both required.

Business Process Diagram

A Business Process Diagram performs exception handling through Intermediate Events that are attached to the boundary of an activity (see Figure 45). If the trigger for the Intermediate Event occurs while the activity is being performed, the activity will be interrupted and the Token will exit the activity through the Intermediate Event and proceed down the Sequence Flow to the next object. In the figure below, the trigger for the exception Intermediate Event is another Intermediate Event that follows Task "B" and is in the general flow of the process (not attached to the boundary of an activity). This Intermediate Event "throws" the cancellation signal and the Intermediate Event attached to the boundary of Task "C" "catches" the signal.

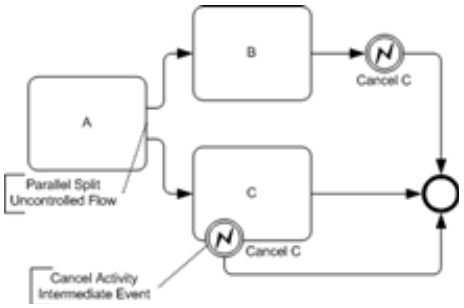

Figure 45: WP #20: Cancel Activity—Business Process Diagram

Activity Diagram

An Activity Diagram performs exception handling through an interruptible region that surrounds one or more activities (see Figure 46). If something, either the completion of an activity or the receipt of a signal, causes a Token to traverse an interrupting edge (the jagged line in the figure), then all the activities within the region will be stopped, and flow will only continue down the interrupting edge. In the figure below, the trigger for the exception is the receipt of a signal. The signal is broadcast immediately after the completion of activity "B." The send signal action "throws" the cancellation signal and the receive signal action in the interruptible region "catches" the signal.

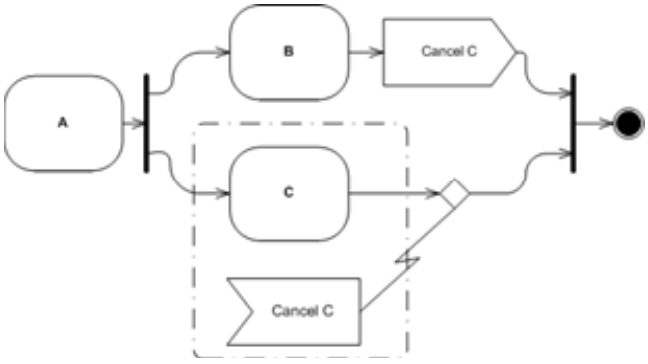

Figure 46: WP #20: Cancel Activity—Activity Diagram

Comparison

The two notations both handle the throwing and catching of exceptions in a similar manner. The BPMN mechanism of attaching Intermediate Events to the boundary of activities is a more focused and intuitive representation of where the exceptions are applied. In addition, the BPMN mechanism is only applied to complete activities, a Task or a Sub-Process. The UML interruptible region is a more risky construct in that it can interrupt activities that are a subset of a cohesive group of activities, such as those within a fork and join pattern.

WORKFLOW PATTERN: CANCEL CASE

This pattern is an extension of the Cancel Activity pattern. In this pattern, however, an entire process is cancelled.

Business Process Diagram

The BPMN solution for the Cancel Case pattern is basically the same as for the Cancel Activity pattern. In this situation however, an Intermediate Event is attached to the boundary of a Sub-Process that contains other activates, rather than a Task (see Figure 47). The figure shows the configuration for canceling a sub-case of the process.

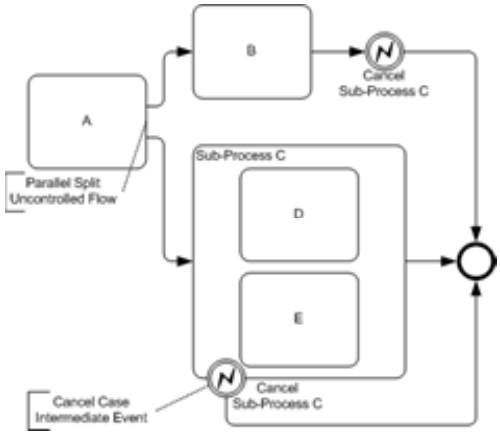

Figure 47: WP #21: Cancel Case—Business Process Diagram, Example 1

If the modeler would like to cancel the whole process, then the solution is actually simpler. All that is required is the use of a Terminate End Event at the location where the cancellation should be signaled (see Figure 48). When a Token arrives at a Terminate End Event, the entire process is stopped.

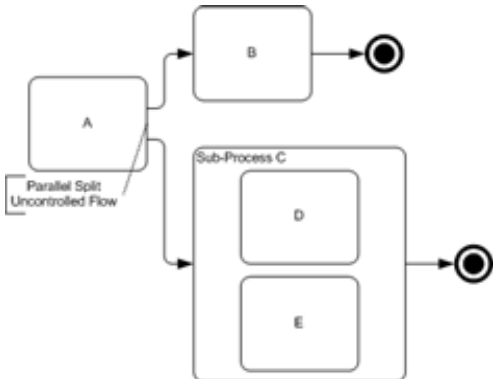

Figure 48: WP #21: Cancel Case—Business Process Diagram, Example 2

Activity Diagram

The UML solution for the Cancel Case pattern is basically the same as for the Cancel Activity pattern. In this situation however, an interrupting region surrounds a compound activity that contains other activates, rather than a simple activity (see Figure 50). The figure shows the configuration for canceling a sub-case of the process.

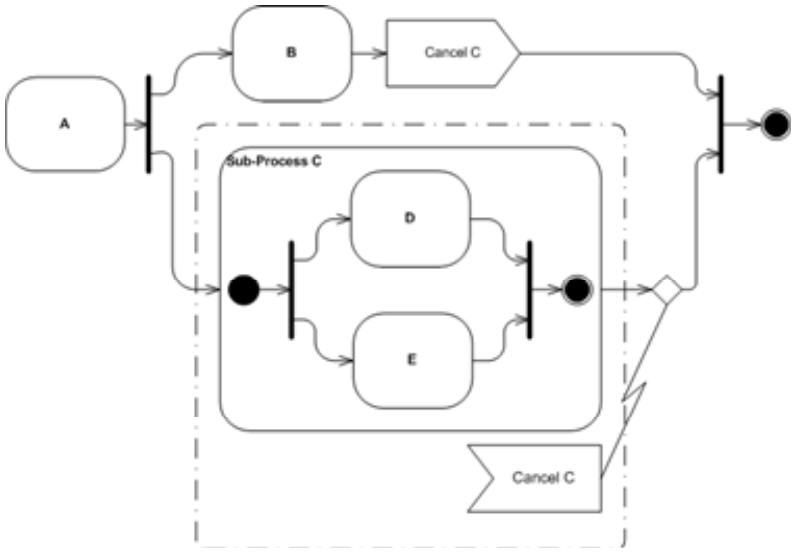

Figure 49: WP #21: Cancel Case—Activity Diagram, Example 1

If the modeler would like to cancel the whole process, then the solution is actually simpler. All that is required is the use of an activity final node at the location where the cancellation should be signaled (see Figure 50). When a Token arrives at an activity final node, the entire process is stopped.

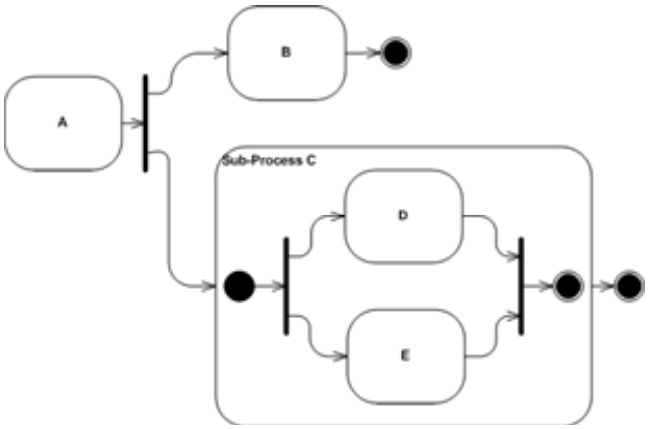

Figure 50: WP #21: Cancel Case—Activity Diagram, Example 2

Comparison

The comparison between the two notations is the same as the comparison for the Cancel Activity pattern. Both notations also have almost the same graphical object for modeling the cancellation of the whole process: a Terminate End Event for a Business Process Diagram and an activity final node for an Activity Diagram.

CONCLUSIONS

The examination of how the 21 workflow patterns can be modeled with a Business Process Diagram and an Activity Diagram demonstrated that both notations could adequately model most of the patterns. The only exception was that Activity Diagram did not have an adequate graphical representation of the Interleaved Parallel Routing pattern, even though the underlying Activity Diagram metamodel has the appropriate structure to create the pattern.

The fact that both notations provide similar solutions to most of the patterns indicates how close the notations are in their presentation. They both share many of the same shapes for the same purposes (e.g., rounded rectangles for activities, diamonds for decisions, etc.). There are some differences in modeling object shapes, with BPMN containing fewer core objects and then having variations on these objects to handle the complexities that might arise

in modeling processes, as shown with the above workflow patterns. Another difference between the two diagrams is often just terminology. For example, an Activity Diagram has a start *node* and a Business Process Diagram has a Start *Event*. Additional differences between the two notations were not demonstrated in this paper since the workflow patterns are focused on the flow of control between process activities. The other differences exist when the flow a data between activities is considered.

The similarities between the two diagrams are present because they are designed to solve the same basic problem: the diagramming of procedural business processes. The differences between the two diagrams are present because of target users of the diagrams. The BPMN effort started two years ago with a charter to create a notation for business people to use. The UML effort started several years ago to standardize modeling for software development and the Activity Diagram was included as part of that effort. Although the UML 2.0 development included a more focused effort to upgrade the Activity Diagram in terms of its use for business people, it is still more technically oriented—including an influence of the very technical EDOC business process modeling effort that was also work done within the OMG.

The two diagrams also share the characteristic of being a view (a diagram) for the Business Process Definition metamodel being developed through an RFP process in the OMG. The Activity Diagram is naturally a view since the Business Process Definition metamodel is an extension to the UML 2.0 metamodel. For BPMN, the Business Process Diagram has been shown to map very closely to Business Process Definition metamodel in a response the metamodel RFP.

Since the Activity Diagram and Business Process Diagram are very similar and are views for the same metamodel, it is possible that they will converge in the future. The OMG is making a concerted effort to address the concerns of higher-than-software-development-levels of business modeling, including business process and business rules. Even though originally developed within BPMI, a Business Process Diagram may find its future development as a part of the high-level business modeling infrastructure being developed within the OMG. The mapping of BPMN to the Business Process Definition metamodel and this paper's comparison of a Business Process Diagram and an Activity Diagram in the context of the workflow patterns are the initial activities of that process.

Stephen A. White is currently working in the standards and strategy area for BPM within IBM's Software Group. His background includes 20 years of process modeling experience, ranging from modeling pilot workload to commercial business processes. He has been involved in most aspects of business process modeling software, from product management, design, consulting, training, and technical writing. Stephen is active with BPMI as a member of the Board of Directors and as chair of BPMI's Notation Working Group, which is developing BPMN.

