



Search TeamOnBase Blogs

→ [Advanced Search](#)

Navigation

OnBase Product Blogs

Business Process Automation Modules

Vertical Solutions

Topics

- Module News and Events [1]
- Documentation [11]
- Presentations [19]
- Technical Information [16]
- FAQs - Technical [20]
- FAQs - Business/Sales [13]
- Business and Sales Info. [22]
- International Content [7]
- Archived Content [72]
- Discussions [22]

Links

Workflow / Technical Information / Workflow Patterns for teamonbase (part 1)

Workflow Patterns for teamonbase (part 1)

- Posted on Wednesday, Apr 4 2007 10:19PM by Carlos Chivardi

OnBase Implementations for Workflow Patterns

Introduction

I talked about workflow patterns in a previous post, and wanted to share some findings while trying to implement these patterns using OnBase workflow. This post describes the OnBase implementation of the widely accepted standard workflow design patterns.

Patterns are proven methods to engineer a solution or solve a particular problem; they are adopted over time when the approach taken has shown results.

Based on one of the most referenced workflow patterns studies by Wil van der Aalst and Arthur ter Hofstede, we implemented each one of the workflow patterns by using out-of-the-box OnBase workflow functionality and creative designs. For a full description of each pattern and business examples, please reference their whitepaper at <http://is.tm.tue.nl/research/patterns/download/wfs-pat-2002.pdf> or their website at <http://is.tm.tue.nl/research/patterns/patterns.htm>

This study focuses on the OnBase Implementation to provide the functionality needed in each workflow design pattern.

The designs described are by no means final, or the only ways to approach these problems, but we tried to solve them with a very simple and clean design.

Basic Control Patterns

Pattern 1: Sequence



















Description

An activity in a workflow process is followed by another activity, with dependency on completion of the first activity.

OnBase Implementation

Simple Queues with One-way transitions to Another Queue.

Queue A -----> Queue B -----> Queue C

 Product Overview 8.2 Sales Product Brief 8.2 OnBase.com Overview Text Module Reference Guide 8.2 - Workflow Module Reference Guide 8.2 - Workflow Timer Service Case Study: Lee County, FL Case Study: Bronson Healthcare Case Study: Hastings Mutual Case Study: SUNY Cortland Case Study: Noridian 8.2 Sales Release Training FLASH movie + audio Overview Session - 1 Minute FLASH movie + audio Overview Session - 7 Minute FLASH movie + audio 9.0 Release Notes RSS Feed 8.2 Release Notes RSS Feed 8.0 Release Notes RSS Feed 7.2 Release Notes RSS Feed RSS Subscribe to RSS Feed Download HSI RSS Reader HSI RSS Reader User Help Feedback

Please click here to provide feedback for the TeamOnBase Product Blog

Pattern 2: Parallel Split

Description

A process is split in multiple sub-processes, having each one of these run in parallel and could be independent of each other.

OnBase Implementation

This is normally achieved in workflow by having a splitting control queue which would add the item to 2 or more lifecycles serving as “sub-processes” or multiple threads, which can be executed in parallel. In the example below, Queue B would add the workflow item to Queue C and D simultaneously by adding them to Lifecycle 2 and 3.

Lifecycle 1
Queue A -----> Queue B
- Add to Lifecycle2
- Add to Lifecycle3

Lifecycle 2
Queue C

Lifecycle 3
Queue D

Pattern 3: Synchronization

Description

Multiple sub-processes merge into a single process thread. This is also called rendezvous and assumes there is only one instance of each sub-process.

OnBase Implementation

One way to implement this would be to add the document to another lifecycle and use the “Doc – This Document in Particular Lifecycle” rule for each sub process lifecycle.

Evaluating this rule in each lifecycle would grow in complexity with each sub process added. It makes more sense to evaluate this rule in the converging queue, or an external function lifecycle. In the example below this is solved by using an external function lifecycle. Once all sub-processes converge in Queue A, the document will be routed to the synchronized Queue B.

Lifecycle 1
Queue A -----> Queue B

Lifecycle 2
Queue C ---> Exit Queue 2
-Add to Lifecycle1
-Add to Function Lifecycle 4

Lifecycle 3
Queue D ----> Exit Queue 3
-Add to Lifecycle1
-Send to Function Lifecycle 4

Function Lifecycle 4
Queue Function_A
Task1 (break on success)
Is Doc in Queue C?

Is Doc in Queue D?
Send to Queue B

Pattern 4: Exclusive Choice

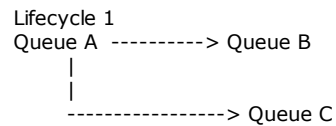
Description

An evaluation point where multiple paths in the process exists and only one path is taken.

OnBase Implementation

This is standard workflow functionality achieved by regular workflow rules, or a task breaking on success or failure, based on a group of rules. Since the document can exist on a given queue in the same lifecycle, this is always an exclusive choice, when configuring the routing in the same lifecycle.

The evaluation can occur on system work, user work, ad-hoc task, or timer work. The document would be routed based on the configured actions.



Pattern 5: Simple Merge

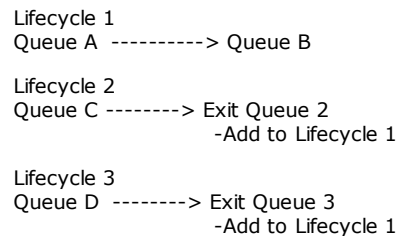
Description

A merging point in the workflow process when two or more activity branches join without having to synchronize. This pattern assumes that only one of the sub-processes is executed. For example, either a nurse checkup or a patient transfer needs to occur, before scheduling a doctor's appointment.

OnBase Implementation

This is also standard OnBase Workflow functionality if you implement each sub process in separate lifecycles. As you add the document to the next sub process (lifecycle), the document will either start the process in the initial queue, or it will remain in the current queue if another sub process has sent the document previously to this process.

For example when the process in Lifecycle 2 ends, it will send the document to lifecycle1, starting in Queue A. If the process in Lifecycle 3 ends sometime afterwards, the document will remain in whatever queue the document is in Lifecycle 1 (either Queue A or B)



Advanced Branching and Synchronization Patterns

Pattern 6: Multiple Choice**Description**

An evaluation point where one or more routes is chosen among multiple options.

OnBase Implementation

This functionality can be achieved by adding documents to other lifecycles through workflow rules. A document can exist on multiple lifecycles, so the multi-choice pattern can be applied for sending the document to one or more lifecycles.

The evaluation can occur on system work, user work, ad-hoc task, or timer work. The document would be routed based on the configured actions.

Lifecycle 1

Queue A

Rule1?

Add to Lifecycle 2

Rule2?

Add to Lifecycle 3

Rule3?

Add to Lifecycle 2 & Lifecycle 3

Pattern 7: Synchronizing Merge**Description**

This is a point in the workflow where one or more sub-process paths merge. If more than one path was taken in parallel, the process will stop in this point until all sub-processes have finished. If only one path was taken, the process moves forward as normal at this point after the only sub-process finishes. Only one instance of each sub-process can be executed at a time.

OnBase Implementation

This is a combination of the multi choice and synchronization patterns. The two pattern solutions can be combined to achieve this functionality. If both sub-processes need to be completed before the process in lifecycle moves forward, the rules in the function lifecycle will make sure there are no more documents in Queue C or D, otherwise, the document gets routed to Queue B without evaluating this.

The main difference when combining the previous patterns can be found in the exit queues in Lifecycle 2 and 3, which don't need to send the document to Lifecycle 1 since the document originated from Lifecycle 1.

Lifecycle 1

Queue A -----> Queue B

Rule1?

Add to Lifecycle 2

Rule2?

Add to Lifecycle 3

Rule3?

Add to Lifecycle 2 & Lifecycle 3

Lifecycle 2

Queue C ---> Exit Queue 2

-Add to Function Lifecycle 4

Lifecycle 3

Queue D ----> Exit Queue 3

-Send to Function Lifecycle 4

Function Lifecycle 4

Queue Function_A

Task1 (break on success)

Is Doc in Queue C?

Is Doc in Queue D?

Send to Queue B

Pattern 8: Multiple Merge

Description

This pattern is a synchronized merge, where the different branches converge into a single process and the same process is executed for every branch after the merge is completed.

OnBase Implementation

This is a similar pattern to the synchronized merge (workflow pattern #7), except that if more than one branch gets activated, possibly concurrently, the activity following the merge is started for every activation of every incoming branch.

The OnBase implementation involves a control lifecycle with the primary document (Lifecycle 1). Workflow rules are applied to the document to decide which branches it will take (Lifecycle 2, 3 or both). For each sub-process, an E-Form will be generated to control the sub-process. The creation of each E-Form will implicitly add the form to its corresponding lifecycle.

Once the sub-process is completed in each branch lifecycle, the E-Form will be routed to a final "post-merge activity" lifecycle (Lifecycle 4), which will execute the same activity for each one of the branches taken by the process. At the end of this lifecycle, the system will look for related E-Forms and advance the control document in Lifecycle 1 to Queue B.

Lifecycle 1

Queue A -----> Queue B

Rule1?

Generate E-Form (Add to Lifecycle 2)

Rule2?

Generate E-Form (Add to Lifecycle 3)

Rule3?

Generate E-Forms (Add to Lifecycle 2 & Lifecycle 3)

Lifecycle 2 – (Activity Branch 1)

Queue C ---> Exit Queue 2

-Add to Lifecycle 4

Lifecycle 3 – (Activity Branch 2)

Queue D ----> Exit Queue 3

-Add to Lifecycle 4

Lifecycle 4 – (After multiple merge activity)

Queue E -----> Queue F

Task1 (break on success)

Is Related E-Form in Queue C?

Is Related E-Form in Queue D?

Is Related E-Form in Queue E?

Send related primary document to Queue B

Pattern 9: Discriminator

Description

This is a merging point in the workflow where the first completed sub-process activity triggers the subsequent process. Subsequent processes are not triggered by the remaining completed sub-processes.

OnBase Implementation

This pattern can be implemented in a similar way as the synchronized pattern, having the function lifecycle act as the “discriminator”. The main change here is in the discriminator queue.

When a document arrives to this queue, the document will activate the subsequent activity by routing it to Queue B. When the second branch gets completed, the discriminator queue can check for the existence of the document in Queue B, or a status keyword that indicates the subsequent activity has been started. The second branch will get removed from the function lifecycle without affecting the rest of the flow.

```

Lifecycle 1
Queue A -----> Queue B
  Rule1?                               Set Status = subsequent
    Add to Lifecycle 2
  Rule2?
    Add to Lifecycle 3
  Rule3?
    Add to Lifecycle 2 & Lifecycle 3

Lifecycle 2
Queue C ---> Exit Queue 2
          -Add to Function Lifecycle 4

Lifecycle 3
Queue D ----> Exit Queue 3
          -Send to Function Lifecycle 4

Function Lifecycle 4
Discriminator Queue
  Task1 (break on success)
    Is Doc status = subsequent?
    Send to Queue B
  
```

Pattern 9a: N-out-of-M-join**Description**

N-out-of-M Join is a merging point where the workflow has M parallel sub-process paths and the subsequent process is activated once N sub-processes are completed.

OnBase Implementation

This pattern is similar to the discriminator with the main difference being that “n out of m” branches must be completed. To achieve this we can use the same model as the multiple merge with the addition of a counter that will indicate how many branches are still needed to discriminate the rest of the branches being completed.

This implementation can be adjusted to meet “n” number of branches, which could share a lifecycle or have a different lifecycle per branch, depending on the business process variations.

The “n” counter required must be pre-determined to use as a criteria to move the process forward. This can be hard-coded, retrieved from a variable (keyword or property), or dynamically assigned based from a database or web service source.

When the primary document arrives in Queue A, the counter is reset to zero before creating any of the E-Forms that will control each sub process branch.

Queue F in lifecycle 4 acts as the discriminator, and increases the "n" counter in the primary document for each completed branch. Once all required branches have been completed by comparing the "n" counter value to the pre-determined value, the primary document will be routed to Queue B in Lifecycle 1, flagging itself with a status equal to subsequent.

When subsequent branches are completed at this point, the counter will evaluate to true, but having flagged the primary document as "subsequent" we will use that information to discriminate these subsequent completed branches. The ignored branches will be reset by adding a status keyword at this point.

```

Lifecycle 1
Queue A ----->Queue B
    Set n = 0                               Set Status = subsequent
    Rule1?
        Generate E-Form (Add to Lifecycle 2)
    Rule2?
        Generate E-Form (Add to Lifecycle 3)
    Rule3?
        Generate E-Forms (Add to Lifecycle 2 & Lifecycle 3)
  
```

```

Lifecycle 2 – (Activity Branch 1)
Queue C ---> Exit Queue 2
    -Add to Lifecycle 4
  
```

```

Lifecycle 3 – (Activity Branch 2)
Queue D ----> Exit Queue 3
    -Add to Lifecycle 4
  
```

```

Lifecycle 4 – (After multiple merge activity)
Queue E -----> Queue F
    Task1 (break on success)
        Increase counter ( n++ ) in related primary doc.
        Is Related primary doc n < "predetermined n"?
        Is Related primary doc status = subsequent?
        Send related primary document to Queue B
  
```

Pattern 10: Arbitrary Cycles

Description

This pattern is a loop of one or more activities in the workflow process.

OnBase Implementation

This is standard OnBase Workflow functionality as it allows you to have documents routed in a loop pattern, based on the queue and transitions configured. The loops can be inside a lifecycle, or span multiple lifecycles. One example is displayed below.

```

Lifecycle 1
    -----
    |
    V
Queue A ---->Queue B----->Queue C----->Queue D----->Queue E
                                   |
                                   Rule x?
  
```

Send to Queue B

Pattern 11: Implicit Termination

Description

This pattern refers to be able to terminate a sub-process after completing the process activities.

OnBase Implementation

This is standard OnBase Workflow functionality when using the "Doc – Remove this document from this lifecycle" or "Doc – Remove this document from all lifecycles" actions, when a sub process or main process is completed.