Creating While Loops with Microsoft SharePoint Designer Workflows Using Stateful Workflows

Version 1.00.

Nick Grattan Consultancy Limited specialise in providing design, consultancy, development and training services for Microsoft Office SharePoint Server 2007 and Microsoft SharePoint Services 3.0.

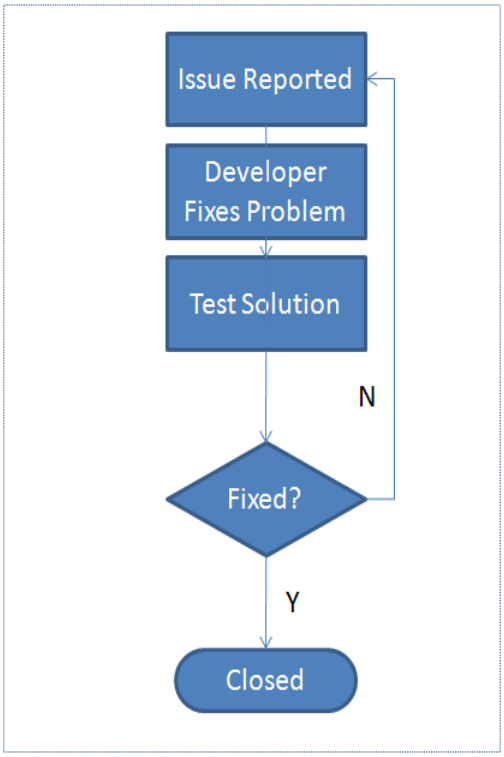
Nick Grattan Consultancy Limited,

Blog: http://nickgrattan.wordpress.com
Web: http://www.nickgrattan.net

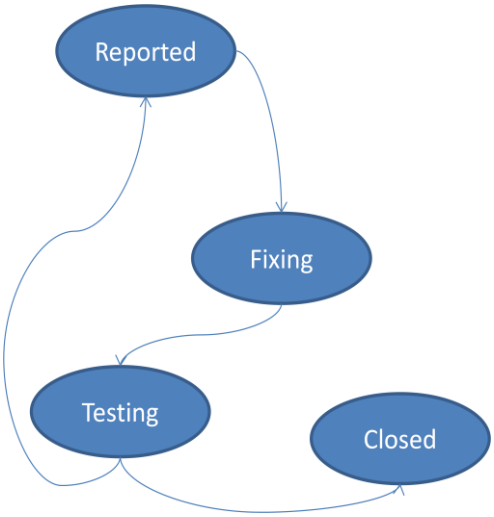
Creating Looping Workflows in SharePoint Designer

Amongst several significant limitations with SharePoint Designer workflows, perhaps the most important is the lack of looping. Therefore, expressing processes like “while not approved, rework” becomes difficult.

For this reason, workflows that look otherwise straightforward to implement become challenging. Here is an example:



One option is to express the workflow as a ***stateful workflow*** and simulate stateful workflows in Microsoft SharePoint Designer with the help of our Custom Activities Pack. The workflow shown above can be represented as the following stateful workflow:

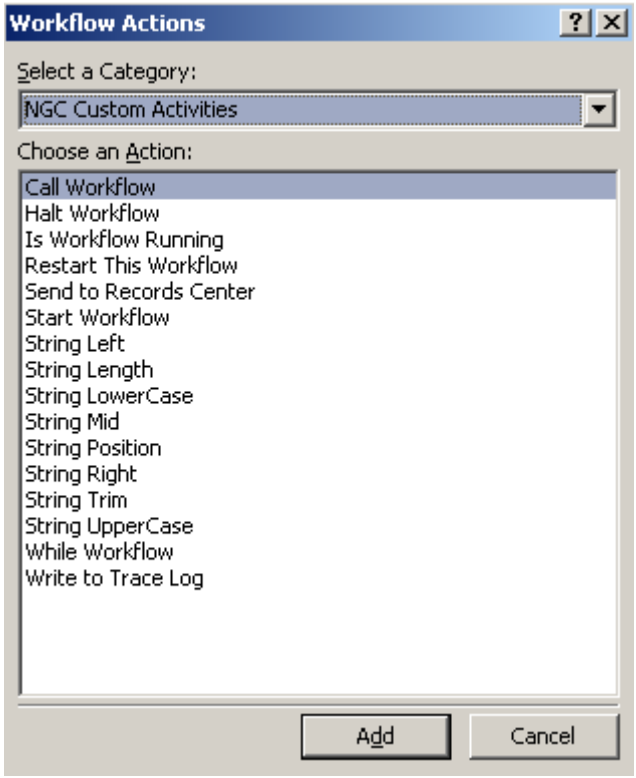


The Custom Workflow Activity Pack Actions

The "Custom Workflow Activity Pack" implements a number of custom activities which add new actions to a Microsoft SharePoint Designer workflow. Of special interest is the ability to start other workflows and restart the current workflow.

The custom activities can be used in Microsoft SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007, Standard and Enterprise edition installations.

You can download the Custom Workflow Activity pack for free from: http://www.nickgrattan.net/Downloads.aspx Once installed, the actions will be selectable from within SharePoint Designer:



The following actions are included in the Custom Workflow Activity Pack:

Custom Activity	Purpose
Start Workflow	Starts another workflow on the item or document. Note that a workflow cannot start itself and a workflow can only be started once for a given item or document.
Call Workflow	Starts another workflow and does not return until the called workflow finishes. This allows another workflow to be called like a sub-routine.
Halt Workflow	Halts (cancels) another running workflow. If the other workflow is not running the call is ignored.
IsWorkflowRunning	Returns a value indicating whether the workflow with the given name is running or not.
Restart This Workflow	Restarts the current workflow executing at the first step
Send To Records Center	Sends the current item/document to the configured records center
String Left	Returns the “n” left most characters from a string
String Length	Returns the length of a string
String LowerCase	Returns a string in lower case
String Mid	Returns a string starting at position “n” of length “l” characters from a string
String Position	Returns the location of one string in another string, or 0 if the string is not found
String Right	Returns the “n” right most characters from a string
String Trim	Returns a string trimmed of leading and trailing blanks (spaces)
String UpperCase	Returns a string converted to upper case
Write to Trace Log	Write a string to the SharePoint trace log, typically located in the folder \Program Files\Common Files\Microsoft Shared\web server extensions\12\LOGS

Implementing Stateful Workflows

In essence, creating a stateful workflow involves the following:

- Create a choice column in the list or library that will track the state of the executing workflow. This will have choice values such as “Reported”, “Assigned” etc.
- Create a SharePoint designer workflow. Each step will have an “If” condition that will check the current state and execute the appropriate code if the workflow is in that state.
- Restart the workflow once the current state’s code has been executed.

In SharePoint Designer Workflows, each step is executed in turn and when the last step is executed the workflow is terminated. When implementing a stateful workflow, the steps are coded like this:

```
If State = “Reported”
        Do Reported Actions
        Set State to “Assigned”
        Restart Workflow
```

```
If State = “Fixing”
        Do Fixing Actions
        Set State to “Testing”
        Restart Workflow
```

```
If State = “Testing”
        Do Testing Actions
        if (fixed)
                Set State to “Closed”
        else
                Set State to “Reported”
        Restart Workflow
```

```
If State = “Closed”
        Do Closed Actions
```

SharePoint Designer Workflow Steps

You will notice that for the “Testing” step, the state will be set to “Reported” if the issue has not been fixed. The effectively “jumps” control back to the “Reported” state at the top of the workflow. This is the key to writing looping workflows in SharePoint Designer.

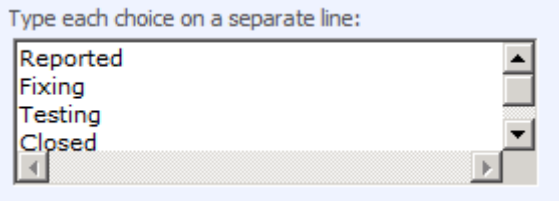
Creating the Workflow

This walkthrough explains how to create the workflow. Note that to simplify the process, steps that involve collecting information from users with tasks have been left incomplete. These would be coded in exactly the same way as with standard workflows.

In this case the workflow is applied to a list called “Test State Workflow”, but the same principles can be applied to a library.

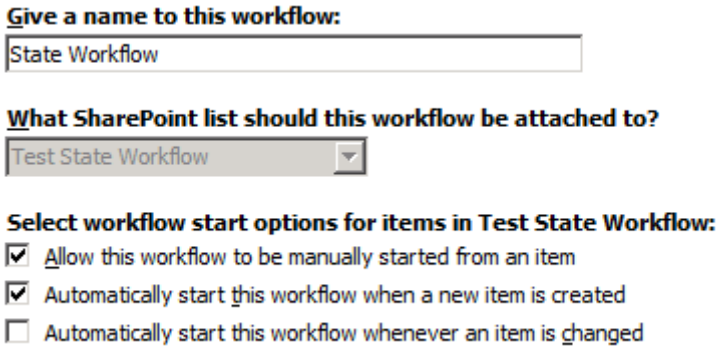
Create a Choice Column

First, create a choice column on the list to manage the state. The choice values will typically be the same as the states identified in the state diagram. In this case, the column is called **State** and the choice values are:

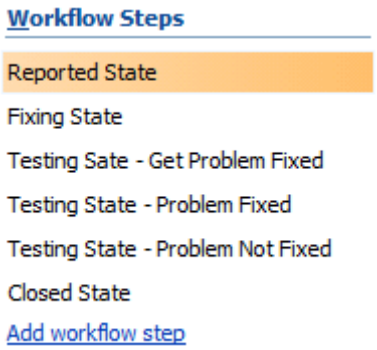


The SharePoint Designer Workflow

The SharePoint Designer workflow will be created in the normal way, and will probably be set to start automatically when an item is created:



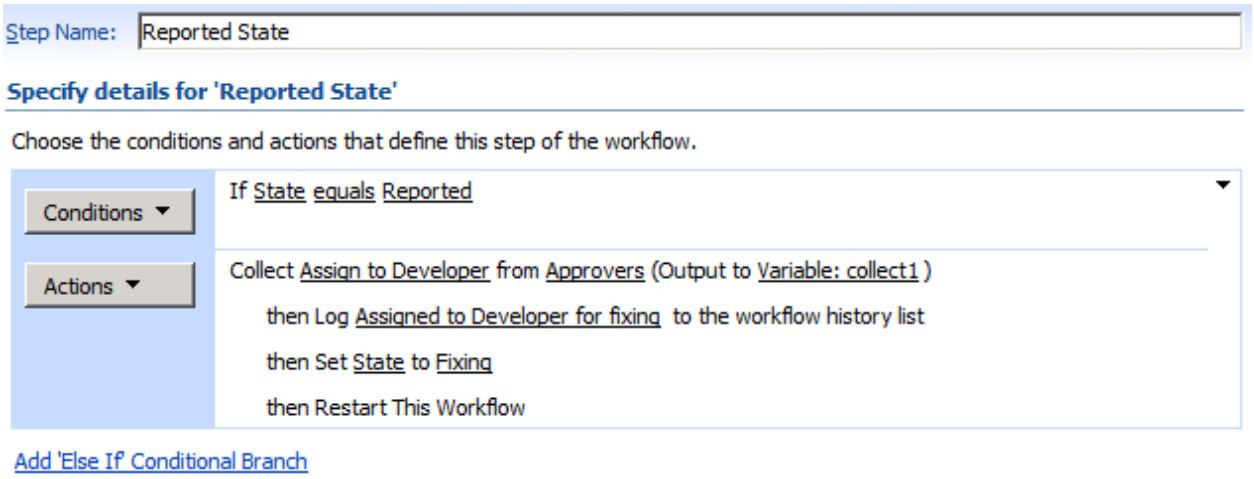
The following steps will be created in this workflow:



Note that "Testing State" has been split into three steps – the first step will collect data from the user in a task indicating whether the problem has been fixed or not, and the other two will take the appropriate course of action.

The "Reported" State Step

First, let's look at the "Reported State":

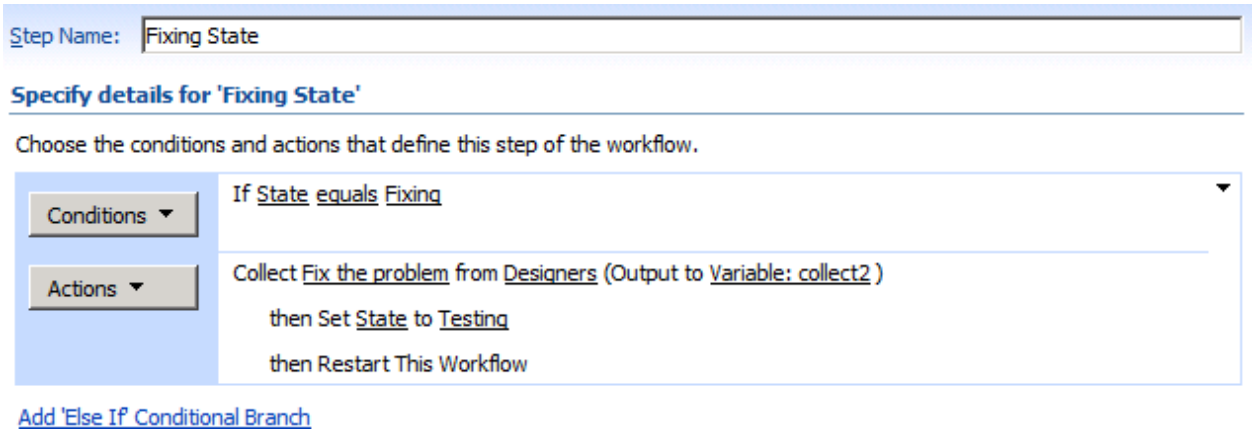


This step has a condition that checks if the state is equal to reported. If it does, it creates a task to assign a developer to fix, and then sets the state to "Fixing". Finally, the workflow is restarted using the custom action "Restart This Workflow".

The "Restart This Workflow" action cancels the current execution of the workflow and then restarts the workflow on the current item. Note that this process can take a few minutes.

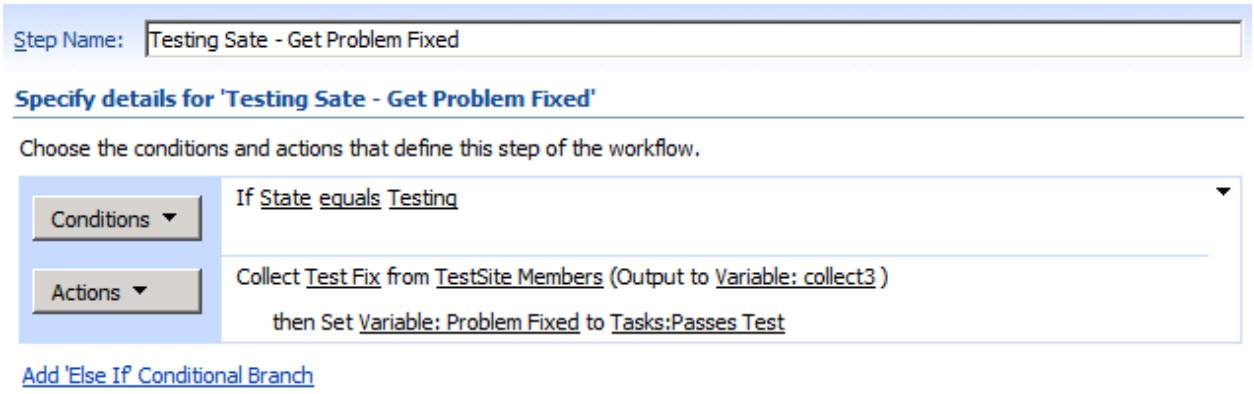
The "Fixing" State Step

The processing for this state is similar to "Reported": it creates a task that will be completed by the developer once the fix is finished. It updates the state to "Testing" and restarts the workflow:



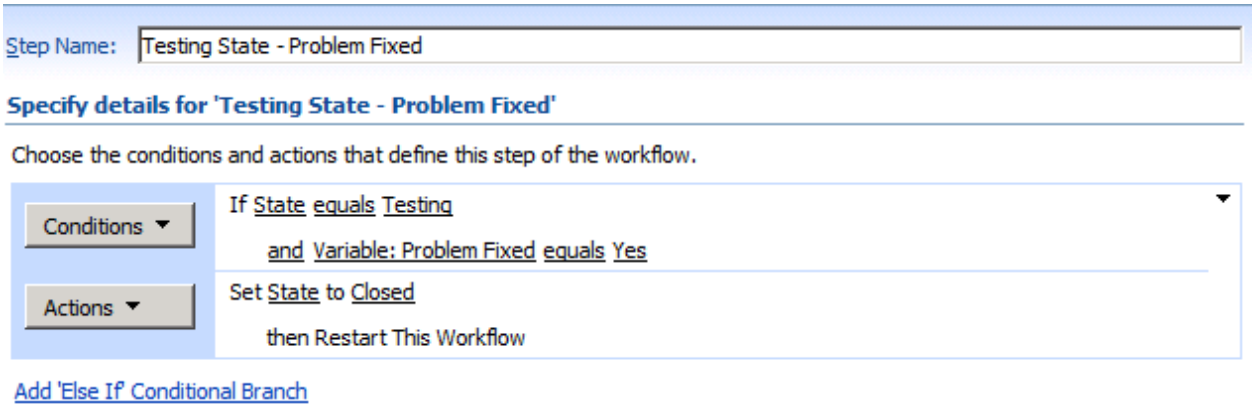
The "Testing" State Steps

The first step for "Testing" creates a task assigned to a tester which is used to indicate if the problem has been fixed. This takes a column value called "Problem Fixed" from the task and copies it to a workflow variable called "Passes Test":

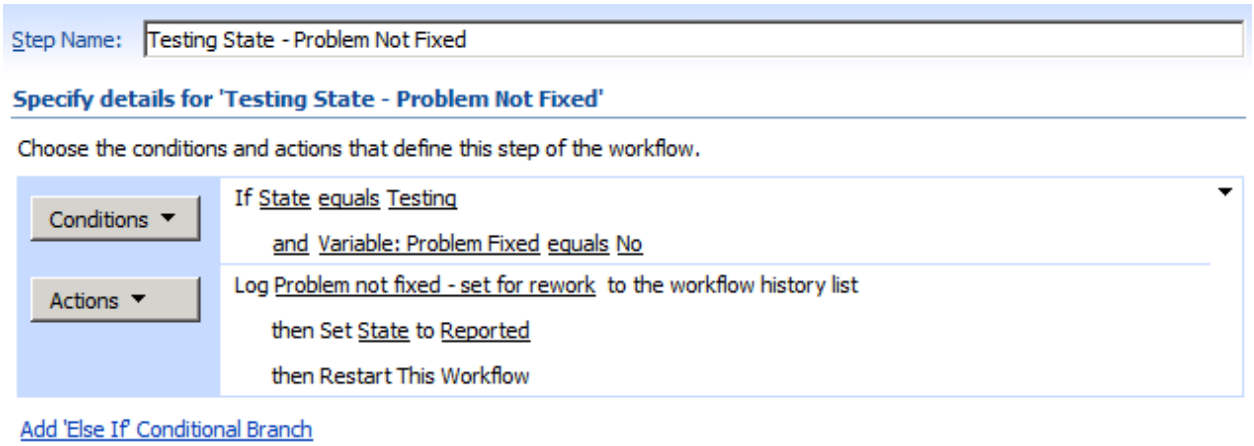


Note in this case the state is *not* updated and the workflow is *not* restarted. This causes the workflow to fall through to the next steps that will take action based on whether the problem was fixed or not.

The first of these steps takes the appropriate action for when the problem is fixed. Note that the condition tests if the state is “Testing” and the “Problem Fixed” workflow variable is “Yes”. The actions simply set the state to “Closed” and restarts the workflow:

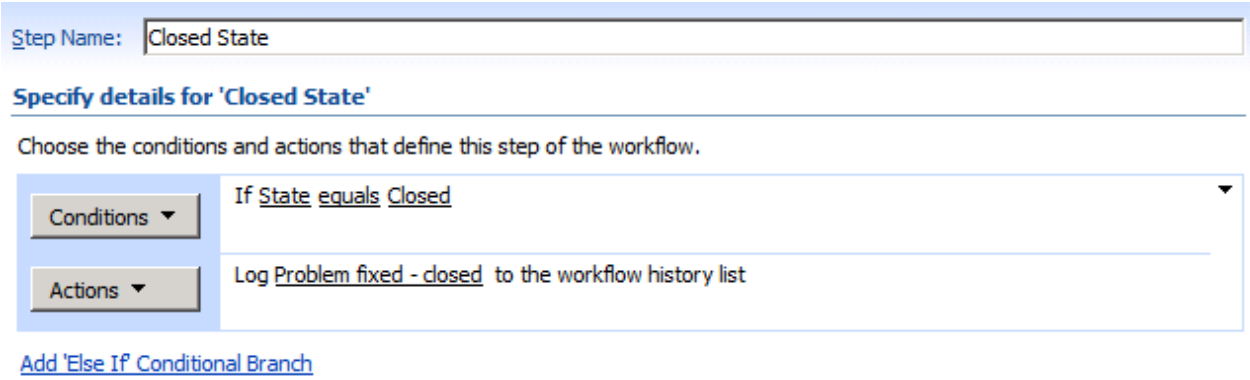


The step for when the problem has not been fixed sets the state to “Reported” and then restarts the workflow:



The “Closed” State Step

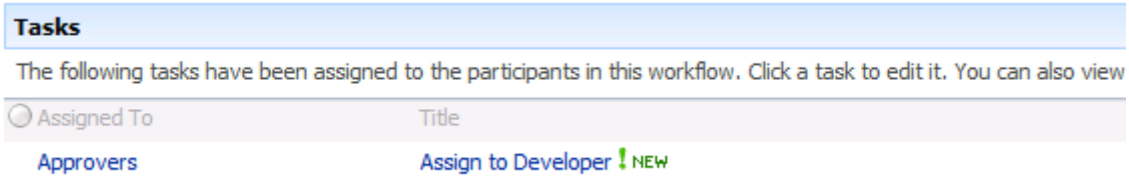
The closed step simply logs to the workflow history. The process will now be complete.



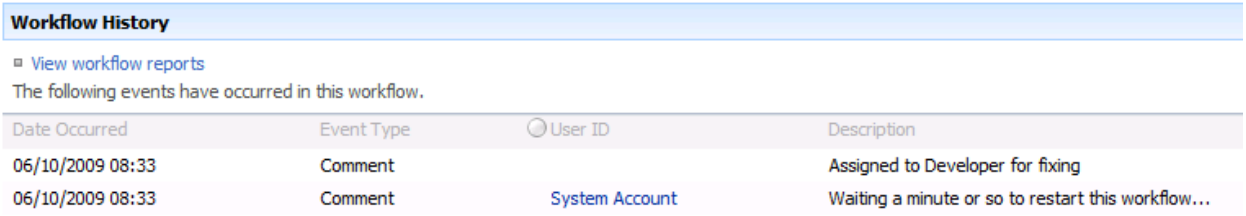
Testing the Workflow

Creating a new item in the list will initially set the state to "Reported" and the "State Workflow" status to "In Progress":

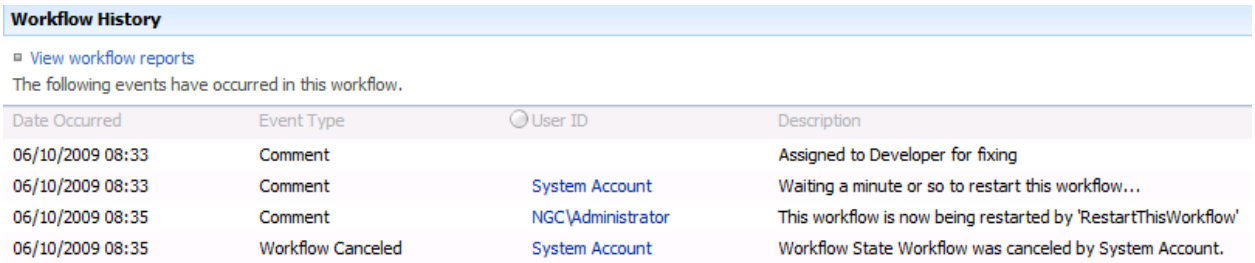
Clicking "In Progress" shows the task that is created that is used to assign the problem to a developer:



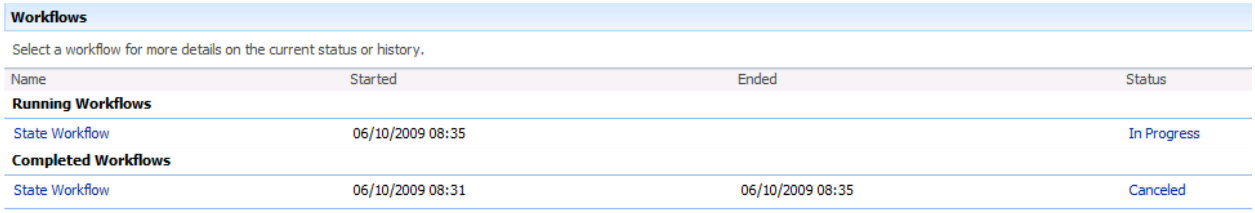
Once the task is completed the workflow history shows that the workflow is waiting a minute or two to restart the workflow:



After a period of time, the workflow will be "cancelled" and restarted. The original instance of the workflow history will now show:



You can see from the list of workflow instances for the item the two workflows: the first that was cancelled and the second that is in progress:



Workflows

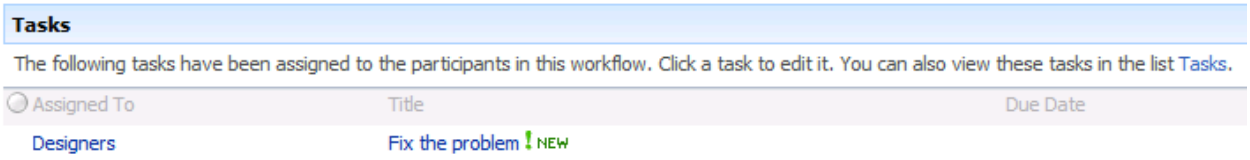
Select a workflow for more details on the current status or history.

Name	Started	Ended	Status
Running Workflows			
State Workflow	06/10/2009 08:35		In Progress
Completed Workflows			
State Workflow	06/10/2009 08:31	06/10/2009 08:35	Canceled

The state of the item is now “Fixing”:

Title	State	State Workflow
Test State Workflow ! NEW	Fixing	In Progress

The “In Progress” workflow has created a task used by developers to indicate that the problem has been fixed:



Tasks

The following tasks have been assigned to the participants in this workflow. Click a task to edit it. You can also view these tasks in the list Tasks.

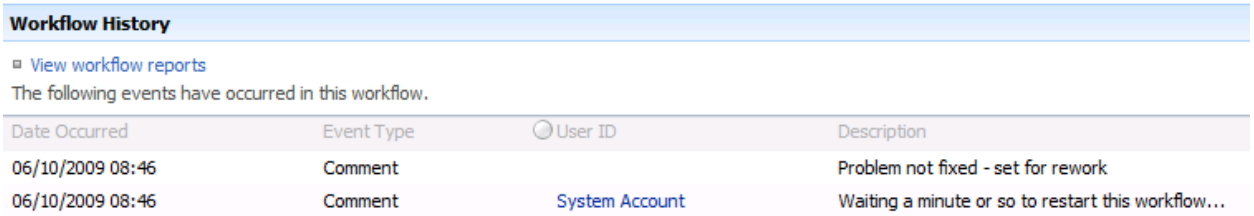
Assigned To	Title	Due Date
Designers	Fix the problem ! NEW	

Completing this task will cause this instance of the workflow to cancel and a new instance to be restarted. The state of the workflow will now be “Testing”:

Title	State	State Workflow
Test State Workflow ! NEW	Testing	In Progress

The new instance of the workflow will create a task used by the tester to indicate if the problem has been fixed. If the tester selects “No” the state will be set back to “Reported” and the process will start again”. If “Yes” is selected the state will go to “Closed” and no more instances of the workflow will be created.

For the situation where “No” is selected by the tester in the task the workflow history reports that the problem has not been fixed and the workflow is restarting:



Workflow History

View workflow reports

The following events have occurred in this workflow.

Date Occurred	Event Type	User ID	Description
06/10/2009 08:46	Comment		Problem not fixed - set for rework
06/10/2009 08:46	Comment	System Account	Waiting a minute or so to restart this workflow...

The state for the item has been changed:

Title	State	State Workflow
Test State Workflow ! NEW	Reported	In Progress

The entire process will now be repeated. If "Yes" is selected by the tester the state will go to "Closed":

Title	State	State Workflow
Test State Workflow ! NEW	Closed	In Progress

Viewing the workflows associated with an item can be used to inspect the history against any of the workflow instances used in this process:

Running Workflows

There are no currently running workflows on this item.

Completed Workflows

State Workflow	06/10/2009 09:00	06/10/2009 09:10	Canceled
State Workflow	06/10/2009 09:10	06/10/2009 09:10	Completed
State Workflow	06/10/2009 08:55	06/10/2009 09:00	Canceled
State Workflow	06/10/2009 08:50	06/10/2009 08:55	Canceled
State Workflow	06/10/2009 08:45	06/10/2009 08:50	Canceled
State Workflow	06/10/2009 08:35	06/10/2009 08:45	Canceled
State Workflow	06/10/2009 08:31	06/10/2009 08:35	Canceled

The workflow instance with the status "Completed" shows the last workflow instance that set the state to "Closed".

Conclusion

By using the "Restart Workflow" Custom Action stateful workflows can be created using Microsoft SharePoint Designer. This allows much greater flexibility in the workflows that can be created. Creating "While Loops" is now possible. However, you need to be aware of some downsides:

- A traditional flow diagram needs to be converted to a state diagram and the various states and transitions identified. It is important that this work is done before trying to implement the workflow in Microsoft SharePoint Designer.
- The workflows tend to be more complex to implement and debug as each step needs to have a condition based on state associated with it. A step that does not have such a condition will be executed *each time* the workflow is restarted.
- There is a slight pause when the workflow is restarted. In production this rarely creates an issue but does make testing more long winded.

The alternative is to use Visual Studio and the Workflow Foundation to create your workflows. This has many advantages, but requires code to be written and can have a steep learning curve.