

# Contextual Workflow Modeling

Vinh Nguyen  
August 12, 2003

## Introduction

Workflows are commonly modeled as activity diagrams, i.e., sequences of actions to be performed to achieve a business objective. This approach to workflow modeling is called Activity-Based Workflow Modeling (AWM). While activity diagrams are a common and useful modeling technique, I will demonstrate in this whitepaper that AWM has fundamental drawbacks that limit its effectiveness in modeling complex, highly nuanced workflows. Then I will describe Contextual Workflow Modeling (CWM), a technology invented at Macronetics to overcome the limitations of AWM.

## AWM and Its Limitations

An activity diagram comprises of discrete actions and the relationships among them. The transition from one action to the next is automatic and predicate on the completion of the previous action. When there are multiple transitions coming out of an action, a split occurs, resulting in the concurrent execution of multiple branches that can join up later at synchronization points. Alternatively, decision points can be used to select one among many alternate branches.

Actions in an activity diagram are grouped into swimlanes to denote the responsibilities for these actions. Each swimlane is associated with a particular role and typically shown as a vertical track. All actions in a swimlane are to be performed by the role associated with the swimlane.

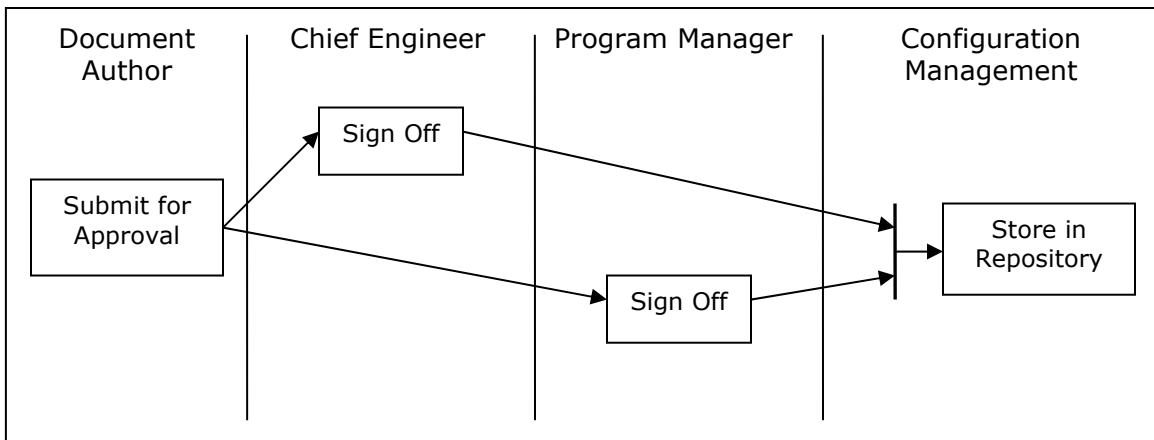
One fundamental characteristic of activity diagrams is that all actions are treated equal regardless of real differences in their nature. For example, some actions are

user-initiated while others may be system-generated notifications or triggers. Yet other actions may be business rule executions, often referred to in the AWM literature as automatic actions. A direct consequence of this semantic flattening is that activity diagrams often become complicated very quickly. It is not uncommon to see activity diagrams requiring hundreds of actions to model even moderately complex workflows. Attempting to understand these activity diagrams is often a daunting task.

We now look at two case studies that illustrate some specific limitations of AWM.

**Case Study 1 – Action Interchangeability**

Figure 1 shows a workflow fragment where a document is to be signed off by both the Chief Engineer and the Program Manager, but in no particular order. The split following the *Submit for Approval* action causes two separate branches to execute concurrently. The synchronization point prior to the *Store in Repository* action forces a wait for the completion of both *Sign Off* actions before proceeding.



**Figure 1. Example Activity Diagram**

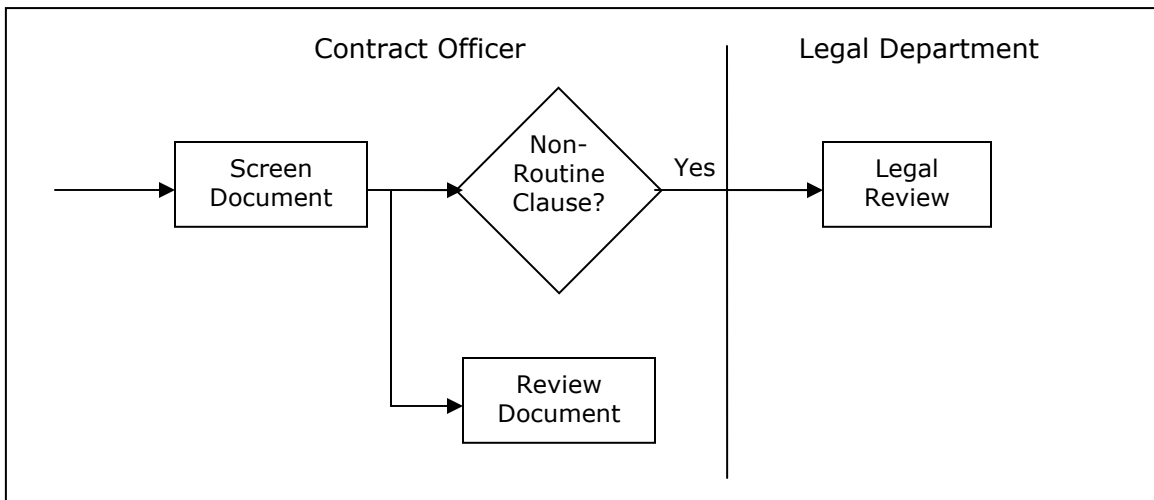
Now we add some nuance to this workflow fragment. Suppose that either the Chief Engineer or the Program Manager, but not both, needs to sign off on the document. In other words, the Chief Engineer role and the Program Manager role are interchangeable when it comes to signing off on the document. Suppose further that the swimlanes for the two roles have to remain separate because these roles need to perform role-specific actions in other parts of the workflow. How should the activity diagram in Figure 1 be modified to capture this nuance?

It turns out that this cannot be done in AWM without bastardization of the methodology. We can attempt to remove the synchronization point so that the completion of either *Sign Off* action will trigger the *Store in Repository* action. However, the remaining *Sign Off* action still appears on the to-do list of the other role unless some custom code is written to remove this action explicitly. Custom code usually translates to high maintenance costs and potential problems when upgrading to a new software release. AWM does not handle action interchangeability gracefully.

### Case Study 2 – Mid-Action Transitions

Consider a workflow fragment where a contract document is reviewed. If the Contract Officer encounters a non-routine legal clause, he requests assistance from the Legal Department and then continues with his review.

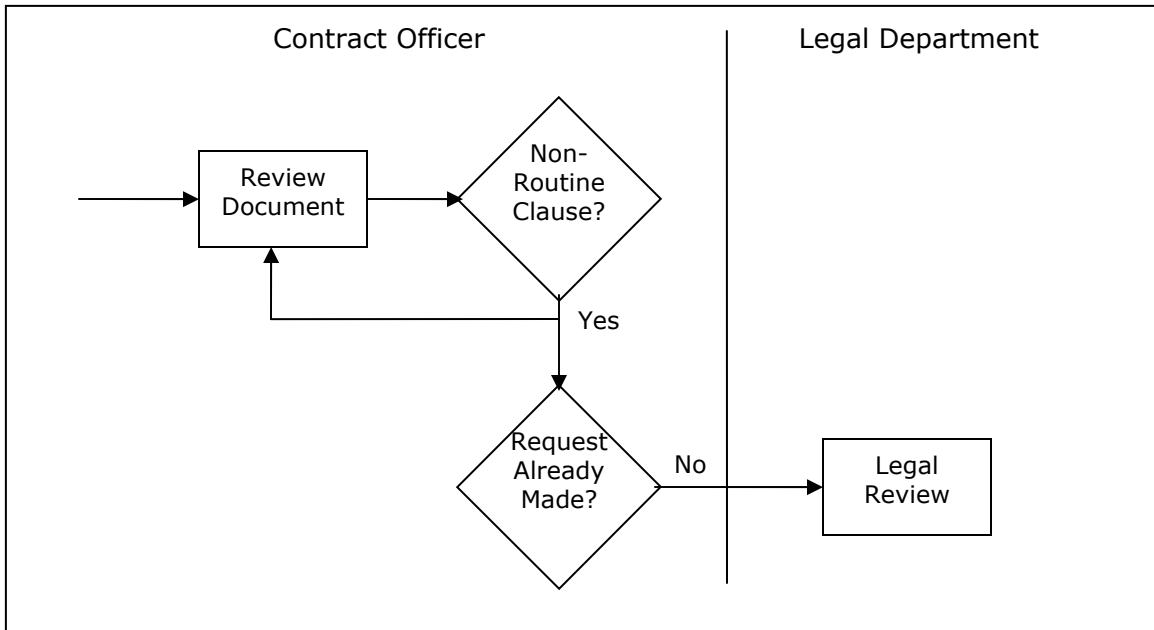
Figure 2 shows an attempt at modeling this workflow fragment. While the diagram is a reasonable approximation, it doesn't quite capture the fact that the check for non-routine clauses occurs inline, as the Contract Officer reviews the document, and not in a separate screening step.



**Figure 2. Contract Document Review – First Attempt**

Figure 3 shows another attempt at modeling the workflow fragment. At first glance, this model appears to work. However, if we were to examine the action history, we will see multiple instances of the *Review Document* action that are caused by the loop-back between the *Non-Routine Clause* decision point and the *Review Document* action. What we want instead is a single *Review Document* action instance that continues after the request is made for assistance from the Legal Department.

AWM cannot capture the semantics of this workflow accurately due to the fundamental requirement in the methodology that an action be completed before a transition can occur. This case study is simply beyond the expressive power of AWM.



**Figure 3. Contract Document Review – Second Attempt**

## CWM Overview

CWM is centered on a modified finite state machine running strictly in the context of a data environment. The data environment isolates the state machine such that the only way to effect a change in the running status of the state machine is through modifications to the data environment. Instead of focusing on the actions as in the case of AWM, CWM is primarily concerned with how the outcome of an action impacts the data environment.

CWM is a rule-based system where all workflow events, ranging from state activation to participant assignments, are driven by user-defined business rules. These rules operate on data drawn from the data environment and form the bridge connecting the data environment to the state machine. On the other hand, the combination of the current workflow status and the participant roles determines who has the privilege and responsibility to modify which part of the data environment.

The CWM approach provides a uniform mechanism through which external actors, whether human users or information systems, interact with the workflow model. There is no artificial distinction of user actions versus system actions in how they impact the data environment, and consequently the workflow status. As a result, integration and interchangeability of the different actor types are much more seamless than in the AWM approach.

The following sections describe the major components of CWM in some detail.

## Data Environment

The data environment consists of two types of data: system parameters and workflow variables. System parameters are generated automatically as the workflow is executed and include information such as the workflow runtime statistics, the currently active states, the users assigned to each role, etc. System parameters are read-only; they cannot be modified.

Workflow variables are user-defined and can be bound to different data sources including: data fields on a form, external databases, scripting functions, external method invocations, or constant values. Variables can also be computed from system parameters as well as from other variables, forming as large an enclosure as necessary to support the workflow business logic. In addition to supporting the business rules, variables also serve as a convenient mechanism for programmatic injection of data into a workflow and for transferring data between different parts of a workflow.

Variables observe scoping rules, enabling them to be defined globally or locally (to a particular form, role, or state). Local variables are used to prevent pollution of the global namespace and to allow reuse of variable names in different contexts. In case of name conflicts, local variables override global variables with the same names.

## Forms

Users interact with a workflow by filling out forms. The data fields on a form can be bound to variables and hence made available to the data environment. In CWM, forms are dynamic in that the data fields on a form can be turned on and off, i.e., made visible and/or writable, conditioned on the current status of the workflow as well as the roles of the participant.

Each data field on a form can be optionally declared as a required field. CWM checks for the presence of required, writable data fields to determine if a user is required to provide data to the workflow. If all required, writable data fields have been filled out then no further action is required from the user. Thus, a user's to-do list becomes a list of forms he is required to fill out. The content of this list is generated dynamically from the workflow data. If another user or information system happens to supply the requisite data then the to-do item will disappear from the user's list automatically. There is never any left-over action that has to be removed with custom code.

## Rules and Triggers

Business rules in CWM are micro-rules, i.e., (usually small) pieces of code that evaluate to either *true* or *false*. If a rule evaluates to *true*, it is said to execute successfully and some action is carried out as a result. For example, if a transition rule executes successfully then the associated state transitions are activated. Similarly, if an assignment rule for a participant role executes successfully then users are drawn from the associated resource pools and assigned to the role. Rules can be parameterized with system parameters and workflow variables.

Rules drive all aspects of workflow execution. CWM supports the following rule types:

- Abort Rules – determine when a state is aborted
- Activation Rules – govern state activation
- Assignment Rules – govern the assignment of users to participant roles
- Completion Rules – determine when a state becomes completed
- Escalation Rules – define escalation actions associated with time-sensitive states, e.g., sending email reminders or overdue notifications
- Transition Rules – govern the transition from one state to the next

Triggers, on the other hand, are external method invocations that occur as a result of some workflow event. For example, a trigger can be attached to the activation of a particular state so that it is invoked whenever the state is activated. At times, one may be interested in not only that a state has been activated but also how it was activated. In this scenario, triggers would be attached to the individual activation rules and invoked when the rules execute successfully. As with rules, triggers can be parameterized with system parameters and workflow variables.

## Participant Roles

Each participant in a workflow takes on one or more roles that, together with the current workflow status, determine the participant's responsibilities for providing data to the workflow as well as his ability to modify select portions of the data environment.

Assignments can be made manually, or users can be automatically drawn from resource pools as defined in the assignment rules. By default, assignments are made automatically when the workflow instance is created. You can, however, override this default behavior by specifying the assignment points in the workflow where automatic assignments to a particular role are to commence. This capability allows just-in-time resource allocation, enabling a more accurate view of enterprise resource utilization in real-time.

## State Machine

Each state in the state machine follows a well-defined lifecycle; the state's progression through its lifecycle is governed by business rules. When a state is instantiated, such as when a transition occurs, it also becomes activated by default. However, activation rules can be written to override this default behavior and activate the state only when certain conditions hold true. An active state becomes aborted or completed when the respective abortion rules or completion rules execute successfully.

Transitions out of a state can occur only when the state is active. Conversely, as long as a state remains active, outward transitions can occur at anytime. As a result, multiple transitions can occur out of a state throughout its active life. Note that unlike AWM, the occurrence of a transition out of an active state does not automatically lead to the completion of that state, and vice versa.

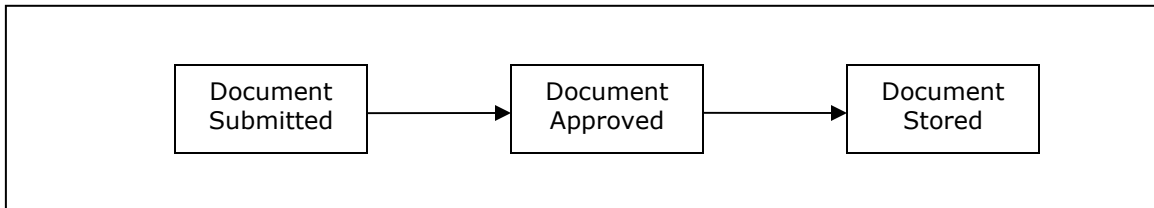
If a state has no outward transition then it is considered an end state. When all active states in a workflow instance are end states, the workflow instance is considered to be completed.

## CWM in Action

Now we will see how CWM handles the case studies outlined in the first section of this whitepaper.

### **Case Study 1 – Action Interchangeability**

Figure 4 illustrates how the workflow fragment would be modeled in CWM. The boxes in Figure 4 denote workflow states rather than actions as in Figure 1. A form would be created to allow both the Chief Engineer and the Program Manager to sign off on the document. The sign-off field on this form is marked as required and bound to a variable to make it available to the data environment. Both the transition rule and completion rule for the *Document Approved* state will check the value of the sign-off variable and allow successful completion when the sign-off is done. As soon as the document is signed off by either role, the data requirement is fulfilled for both roles and the transition to the *Document Stored* state occurs.

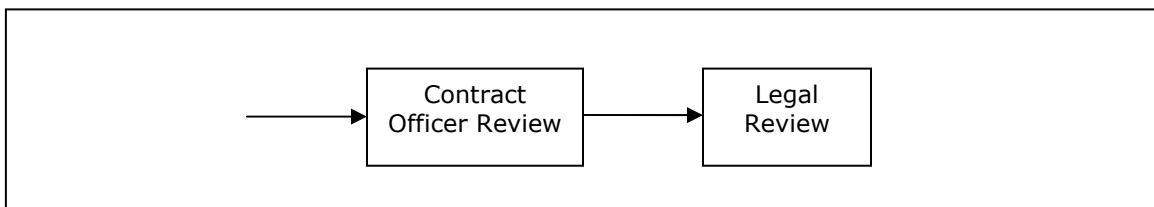


**Figure 4. CWM Solution for Case Study 1**

### **Case Study 2 – Mid-Action Transitions**

This case study becomes trivial in CWM. A form is designed to include two data fields, the first allowing the Contract Officer to request assistance from the Legal Department and the second signaling the completion of the contract review. Both data fields are bound to variables to make them available to the data environment.

A transition rule is then written keyed on the first variable (request assistance) to activate the *Legal Review* state while keeping the *Contract Officer Review* state active. The completion rule for the *Contract Officer Review* state would be keyed on the second variable (review completion). This way, the activation and completion of the *Contract Officer Review* state is kept independent of the *Legal Review* state.



**Figure 5. CWM Solution for Case Study 2**

It should be noted that in both case studies, the diagrams are more streamlined in the CWM approach as compared to the AWM approach. The reasons for this simplification are two-fold. First, CWM allows business rules and triggers to be tucked away under the respective states and thus do not clutter up the main diagram. Second, CWM provides much greater expressive power than does AWM, allowing fine nuances to be captured within the modeling framework more naturally and with much less effort.

## **Conclusion**

CWM represents a radical departure from the typical AWM approach. By switching the modeling focus from actions to states and their driving business rules, and by deriving action requirements dynamically from the workflow data content, CWM puts tremendous expressive power in the hands of the business process architect, allowing him to capture nuances that are simply beyond the reach of AWM.

As a bonus, workflow models in CWM tend to be streamlined, with the different action types organized logically according to their semantics. The rich semantic structure of CWM enhances comprehension of complex processes by giving analysts a top-level overview of a workflow and allowing them to drill down into particular areas of the workflow as necessary.