Microsoft

Introducing Windows Workflow Foundation

An Early Look

David Chappell, Chappell & Associates

August 2005

Contents

Describing Windows Workflow Foundation

Virtually all software used in enterprises today has the same goal: supporting business processes. Some processes are entirely automated, relying solely on communication among applications. Others—probably the majority—also rely on people to initiate the process, approve documents the process uses, resolve any exceptional situations that arise, and more. In either case, it's often possible to specify a discrete series of steps known as a *workflow* that describes the activities of the people and software involved in the process. Once this workflow has been defined, an application can be built around that definition to support the business process.

Creating and executing a workflow in software poses unique challenges. Some business processes can take hours, days, or weeks to complete, for example. How can a developer maintain information about the workflow's current state for this length of time? This kind of long-running workflow will also typically communicate with other software in a non-blocking way. How can the challenges of asynchronous communication be made easier for developers? And while modeling fixed interactions among software is relatively straightforward, people tend to want more flexibility, including things such as the ability to change a business process on the fly. How can the workflow handle the diverse and unpredictable behavior of humans? Without the right foundation to build on, meeting requirements like these is hard. Yet if technology explicitly designed to support workflows is available, creating this useful kind of software can be straightforward.

Microsoft's Windows Workflow Foundation (WWF) was created to address these requirements. A core component of WinFX, it will be a fundamental part of the Windows platform for developers. WWF provides a common framework for building workflows into Windows applications, whether those workflows coordinate interactions among software, interactions among people, or both. Scheduled for release in mid-2006, WWF will run on the forthcoming Windows Vista and will also be available for Windows XP and Windows Server 2003[1].

1 This paper describes the first beta release of WWF. Be aware that some changes are likely before the technology's final release.

What Workflow Applications Require

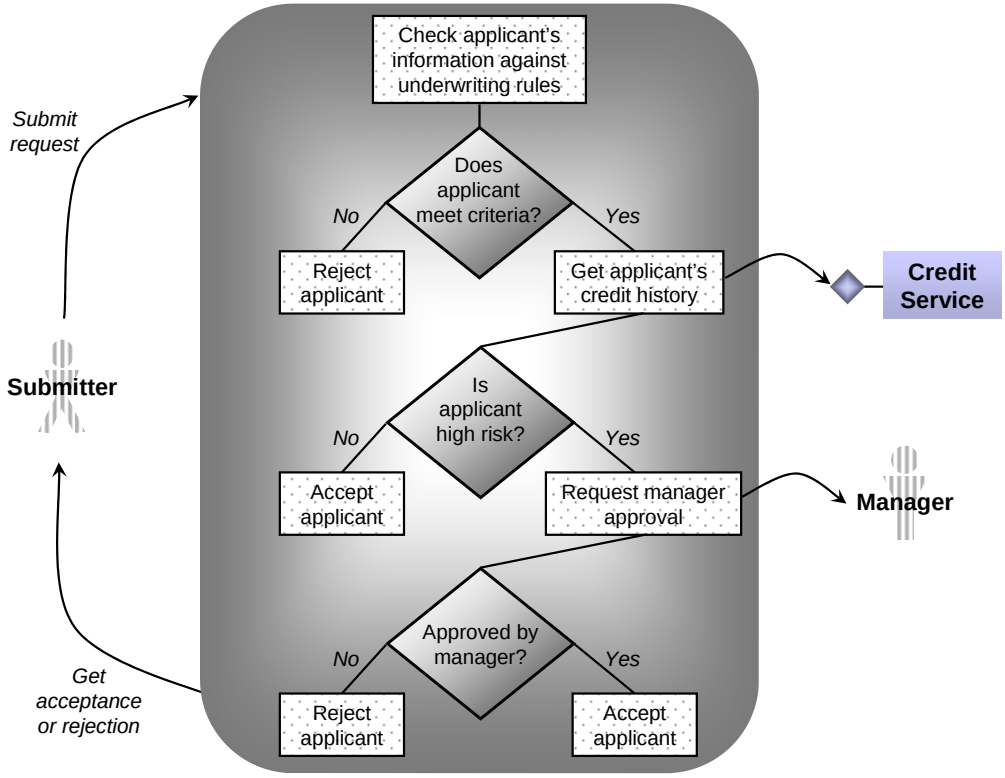
To get a sense of what's required from a workflow framework like WWF, it's useful to think about different kinds of applications that might use workflows. Here are some examples:

An ASP.NET application that displays pages to its users might use a workflow to control the order in which those pages are shown. Doing this can make it easier to change the page flow without changing the pages themselves, as well as cleanly separating the application's user interface from its controlling logic.

A composite application in a service-oriented environment might implement its core behavior using a workflow. As more and more applications expose their logic through web services, creating business processes built on those services becomes easier. A workflow technology such as WWF provides a foundation for the logic that will invoke those services, knitting them together into a composite application.

An application targeting a specific problem, such as customer relationship management (CRM), or a specific vertical market, such as financial services, might be built around a workflow. This kind of application commonly implements a number of different business processes. Building the logic that drives those processes on a common workflow foundation such as WWF can make the application faster to build, quicker to change, and easier to customize.

To get a better sense of the requirements that a workflow framework needs to address, it's worth looking at an example workflow application in a bit more detail. Suppose that an independent software vendor (ISV) wishes to create a workflow-based application for insurance companies. The figure below illustrates a simple example of how an application for requesting an automobile insurance policy, one of the business processes this product supports, might look.



The process begins when a submitter sends in an application. This submitter might be an employee in a call center, an insurance agent in the field, or even a customer directly submitting an application over the Internet. However it's done, the arrival of a new application creates a new instance of this workflow. The workflow begins by checking the information supplied in the request against this company's rules for issuing policies. If the applicant fails to meet the company's underwriting criteria, the applicant is rejected. Otherwise, the workflow requests the applicant's credit history from an external credit service. A satisfactory credit score results in immediate acceptance, but high-risk applicants with bad credit histories require a manager's approval. If this approval is granted, the applicant is accepted. If not, the applicant is rejected.

This simple example illustrates many of the requirements that a typical workflow-based application presents. Those requirements include:

> The ability to make decisions based on business rules. This includes simple rules, such as a yes-or-no decision based on the result of a credit check, and more complex rules, like the potentially large set that must be evaluated to make an initial underwriting decision.

Ways to communicate with other software and other systems outside the workflow. In this example, the initial request is received from some other part of the application, while some aspects, such as contacting the credit service, require communication using web services or other technologies.

Ways to interact with people. Here, a manager must approve some applicants, and so the workflow must be able to display a user interface itself or interact with human beings through other software.

The ability to maintain state throughout the workflow's lifetime. Especially when people are involved, as with the manager in this example, a workflow can take a long time to complete. (What if the manager has left for the day, or is on a two-week vacation?) Building scalable systems requires a way to deactivate the workflow and store its state persistently, then reactivate it and load that state when the next step can be executed.

All of these are fundamental requirements for workflows. Most of them are also challenging to do without building on software designed explicitly for supporting workflows.

A workflow framework can offer more than this, too. It might, for example, offer things such as:

A component-like approach to workflows, where each step can be implemented by a specific chunk of software. Like other component technologies, it's possible to declaratively specify behaviors for these steps, such as transactions, rather than writing code to implement those behaviors. It might also be possible to create pre-defined steps for workflows that are useful in a particular domain, such as insurance applications or systems management, then use these in many different workflows.

Tools to create and modify workflows graphically. Since a workflow consists of a defined number of steps, it can be built using tools that illustrate those steps and the relationships among them.

The ability to monitor a running workflow, examining its execution in real time.

A way to change a running workflow instance by, say, adding a step. While this isn't typically needed when only software is involved, this kind of flexibility is a common requirement for workflows that interact with people.

Workflow is a distinct technology, one with its own unique benefits and requirements. While this approach is applied today in several different areas, a generally-available foundation for creating workflows could make it even more widely used. And the rise of a service-oriented world, driven by the global vendor agreement on web services, is all but certain to give this approach a more central role.

Workflow technology has been waiting in the wings of mainstream software development, a supporting player but never a star. The odds are good that this is about to change.

What Windows Workflow Foundation Provides

As a general foundation for workflows, WWF targets a number of goals. Three stand out, however, as most important: providing a common workflow technology for Windows, offering a workflow framework for diverse applications, and unifying system and human workflow. This section examines each of these.

A Common Workflow Technology for Windows

Whether they're created by Microsoft or by others, many Windows applications include some kind of workflow support. Looking only at Microsoft products, workflow technology is available today in BizTalk Server, Exchange Server, and others. This broad use certainly illustrates how useful workflow can be, but having multiple workflow technologies for Windows doesn't make much sense. Like other mainstream development technologies, workflow support should be part of the Windows platform itself.

This is exactly what WWF provides. By implementing workflow as a part of WinFX, this approach to creating software will be available for any Windows application that needs it. This includes applications running on clients or on servers, as well as applications created by end users, by independent software vendors (ISVs), and by Microsoft itself. Although it will take some time, WWF will become the common foundation for workflow in Microsoft products and technologies. A primary example of this is the next release of Windows SharePoint Services (WSS), which will provide workflow services based on WWF, and the next release of server technologies from the Microsoft Office System, which will provide a variety of solutions based on those services. Future releases of other Microsoft products, including BizTalk Server and Microsoft Business Solutions, will also implement their workflow services using WWF. And since all of these applications will eventually use the same workflow technology, the advent of WWF should make it easier to implement business processes that rely on multiple Windows applications.

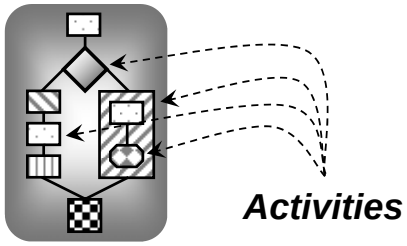
It's important to understand that WWF is a framework targeting developers, not a workflow application intended for immediate use by end users. Accordingly, it doesn't provide tools for information workers to interact with workflows (although the next release of Microsoft Office, code-named Office "12", will include tools for working with WWF workflows deployed to SharePoint sites, as described later in this paper). WWF also doesn't include full-featured management and monitoring tools, although it exposes the information needed to create them. The purpose of WWF is not to be a complete workflow solution for Windows. Instead, the goal is to make it easier for software developers to create workflow-based Windows applications.

A Workflow Framework for Diverse Applications

Creating a common workflow technology for Windows raises an obvious and challenging problem: given the diverse ways that Windows applications use workflow technology, how can any one solution successfully address all of them? Traditional workflow products,

which typically implement a single language for expressing workflows and a single graphical tool for defining them, aren't general enough to meet this broad need. Some other approach is required.

Rather than offering a single language and a single tool, WWF instead provides a general framework for creating and executing workflows. In WWF, a workflow consists of a group of *activities,* as the figure below shows, each of which executes some action in the workflow.

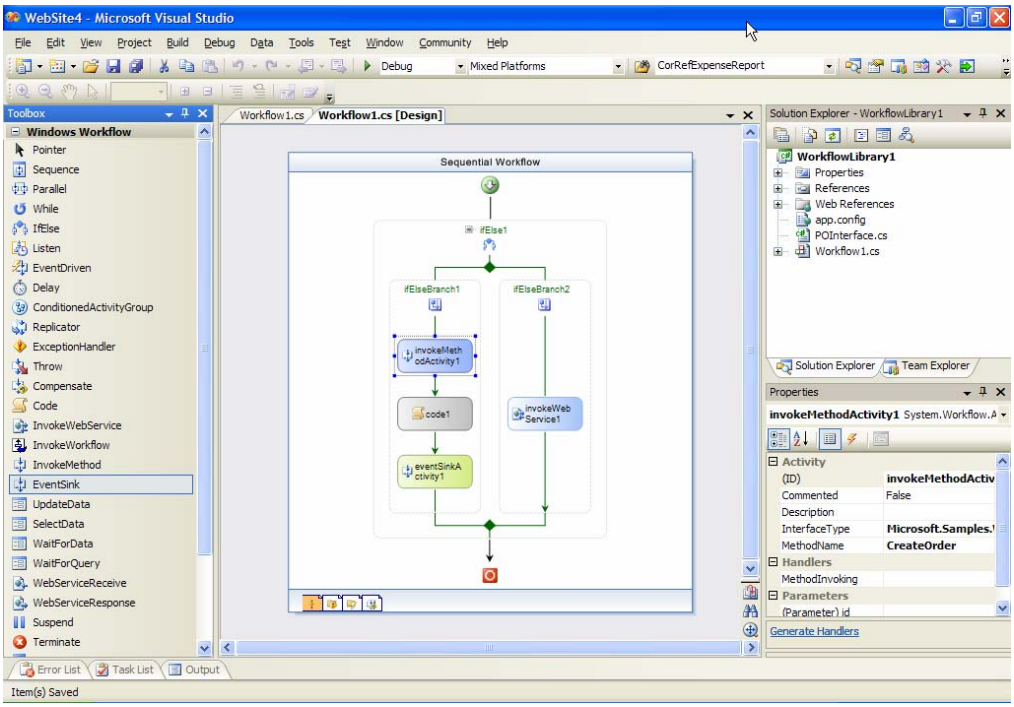


Workflow

WWF ships with a set of general purpose activities for defining workflows. This *base activity library* provides the ability to define control flows using familiar constructs such as if/else and while loops. It also includes a rules engine, support for communicating with other software using web services, and more.

Yet while these built-in activities offer a good deal of workflow functionality, developers are also free to implement custom activities focused on a particular problem space. In effect, activities provide something much like a component model for workflows. Microsoft Office "12" servers, for instance, will provide a set of activities targeting the creation of document-oriented workflows. Other applications, whether they're built by Microsoft or others, are free to create their own groups of activities. A workflow application that supports an approval process might include activities such as RequestApproval, Approve, and Reject, for instance, while one focused on health care applications might provide a completely different set. Workflows can combine custom activities with those in the base activity library, since custom activities use the same public interfaces as the activities that ship with WWF. The creators of custom activities are also free to completely ignore WWF's base activity library—there's no requirement to build on top of them. The goal is to let a larger group of developers create workflows that use domain-specific activities created by a smaller number of more technical developers.

WWF workflows can be written directly in code. Workflows can also be defined graphically, with code added where required. To make this possible, WWF provides the *Workflow Designer*, a tool that allows developers to create and modify workflows. As the figure below shows, the Workflow Designer can be hosted in Visual Studio 2005. The toolbox on the left contains icons for the base activity library, which can be dragged and dropped onto the design surface to create a workflow. Each activity has properties and events that can be set by a developer in Visual Studio.



The Workflow Designer can also be hosted in and customized for other environments. An ISV that wishes to include workflow within its own offering might host this tool directly inside that product, giving it whatever look and feel is appropriate. Similarly, an organization that wished to provide a way for information workers to create and modify workflows could host the Workflow Designer within a more business-friendly environment than Visual Studio. ISVs are also free to create other graphical design tools for working with WWF workflows—using the Workflow Designer isn't required.

Another challenge to providing a general workflow foundation for Windows is hosting: what kind of application should be able to run a workflow? The answer provided by WWF is to allow the runtime engine it provides for executing workflows to be hosted in pretty much any Windows process, ranging from simple console or Windows Forms applications up to large, complex server software written within an enterprise or provided by an ISV. The intent is to let WWF be used in a broad spectrum of applications.

A primary goal of WWF's creators is to foster an ecosystem around this workflow framework. Specialized activities focused on distinct problem domains, customized environments that host the Workflow Designer, and unique hosting environments for the WWF runtime can all be created to address particular requirements. But no matter how WWF is used, the core programming model and runtime engine remain the same, and the behavior of the graphical development tool will be similar. While the WWF user experience

is likely to become quite diverse over time, the foundation on which this experience is built will remain common.

Unified System and Human Workflow

Even though business processes commonly involve both people and applications, automating interactions among software is quite different from automating interactions among people. Accordingly, workflow technologies have typically emphasized one or the other. One of the major goals of WWF is to end this dichotomy by providing a unified solution that addresses both problems.

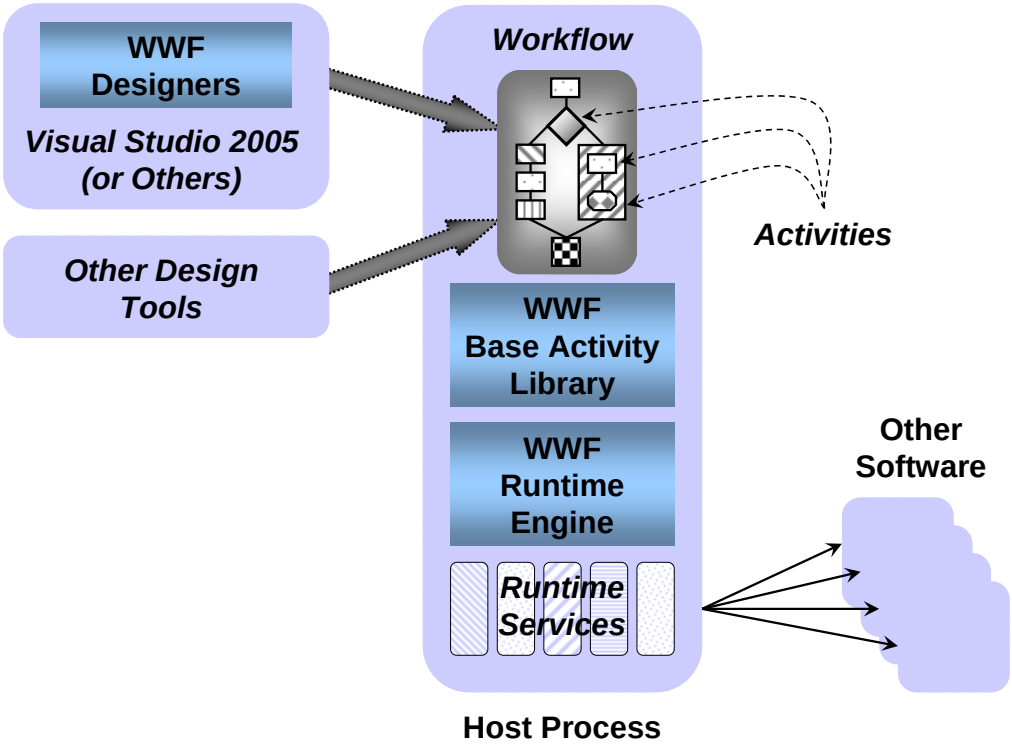
To see why this is a challenge, think about the typical characteristics of automated interactions among applications. Sometimes known as *system* workflow[2], application interactions are usually predictable and relatively static. The logic that directs this interaction can be defined once, then used over and over without changes. System workflows also typically exchange structured, well-defined data, such as XML documents, that can be effectively processed by software without human intervention.

Contrast this with *human* workflow, which coordinates interactions among people. Applications interacting on behalf of people tend to need much more flexibility than those that interact purely with other software. People change their minds, introduce new ideas and exceptions, decide to cancel a process unexpectedly, and do other things that make effective human workflows significantly more dynamic than system workflows. Accordingly, software supporting human workflows should allow a more event-driven, flexible approach. Human workflows also typically work with less-structured, human-readable data, such as email and text documents, rather than the structured information used in system workflows.

Given the different requirements of these two kinds of workflow, it's not surprising that addressing both in a single technology has been challenging. Yet WWF is explicitly focused on addressing both kinds of workflow in a unified way. To do this, it provides both sequential and event-based approaches to defining workflows, the ability to add steps to or otherwise modify a running workflow, and other things, all of which are described later in this paper.

---

[2] The term *orchestration* has often been used to refer to coordination of work done purely by software, while *workflow* typically implies that people are also involved. Since business processes today commonly involve both software and people, it makes sense to use the more general name *workflow* to refer to the intelligence that coordinates the interaction of both.



The diagram above illustrates the fundamental components of WWF. They are:

Activity: a unit of work. The work an activity implements can range from very simple to quite complex.

Workflow: a group of activities that implements all or part of a business process.

WWF designers: graphical tools that can be used to create and modify workflows and activities.

WWF base activity library: provides a fundamental group of activities that developers can use to create workflows.

WWF runtime engine: a library that executes workflows. The runtime engine also provides other services, such as mechanisms for communicating with software outside the workflow.

Host process: a Windows application that hosts the WWF runtime engine and any workflows it executes. The host process provides supporting runtime services for persisting a workflow's state, handling transactions, and other functions.

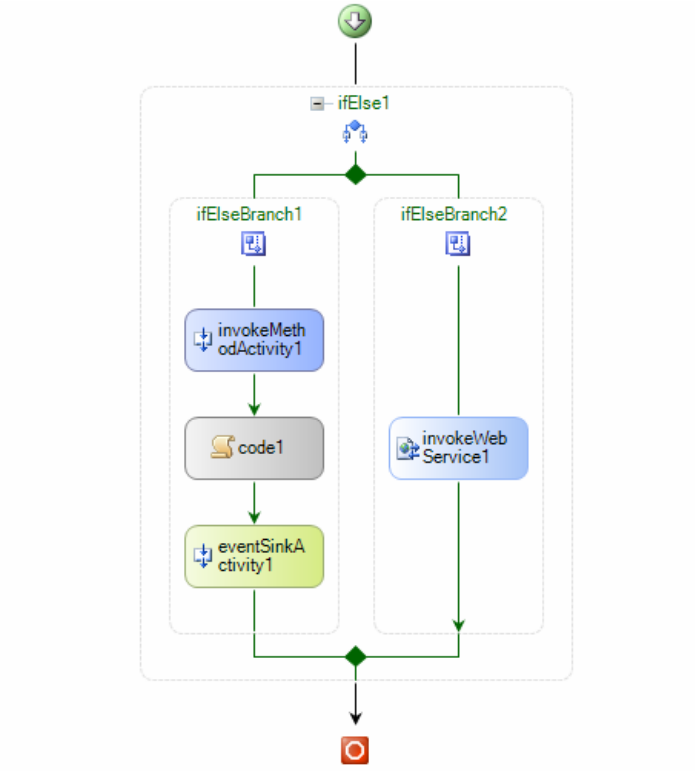
Understanding WWF requires getting a grip on all of these components. The place to start, though, is with the reason everything else exists: workflows.

Understanding Workflows

Every WWF workflow contains some number of activities, each of which performs some aspect of that workflow's function. The workflow acts as a container for these activities, providing a way to control their lifecycles and order of execution. WWF aspires to support both system and human workflows in a unified fashion, which presents a challenge. As described earlier, system workflows tend to execute activities in well-defined, predictable ways, while human workflows do not. To address both of these requirements, WWF provides two built-in workflow types: *sequential* workflows, capable of executing activities in a pre-defined pattern, and *state machine* workflows, capable of responding to external events as they occur. Both rely on the same runtime environment, and both can use the same custom activities. The sequential approach is a natural fit for system workflow, while state machines provide a way to model the more loosely-defined nature of human workflow. A single workflow can combine elements of both styles, allowing a combination of the two. If necessary, a developer can also create custom workflow types, but most WWF applications will use one of these two standard options. This section describes both, starting with sequential workflows.

Using Sequential Workflows

The diagram below illustrates a simple sequential workflow created using the WWF Workflow Designer. Sequential workflows are intended for applications where the workflow's activities are executed in some kind of sequence. This sequence can include loops and branches and other kinds of control flow, but the workflow nonetheless has a defined path from beginning to end.



The base activity library that ships with WWF contains a group of activities that can be used in sequential workflows. Those activities include:

IfElse: executes the activities contained in two or more possible paths based on whether a condition is met.

While: repeatedly executes one or more activities as long as a condition is true.

Sequence: executes a group of activities one at a time in a defined order.

Parallel: executes two or more sequences of activities in parallel, waiting for all sequences to complete before continuing.

Code: executes a defined chunk of code.

Listen: waits for a specific event, then executes one or more activities when that event is received.

Delay: suspends a workflow's execution for a specified amount of time.

InvokeMethod: calls a method in an object that's in this application but outside of the workflow.

EventSink: waits for a call from another method that's in this application but outside of the workflow.

InvokeWorkflow: causes another workflow to begin executing.

InvokeWebService: calls a web service.

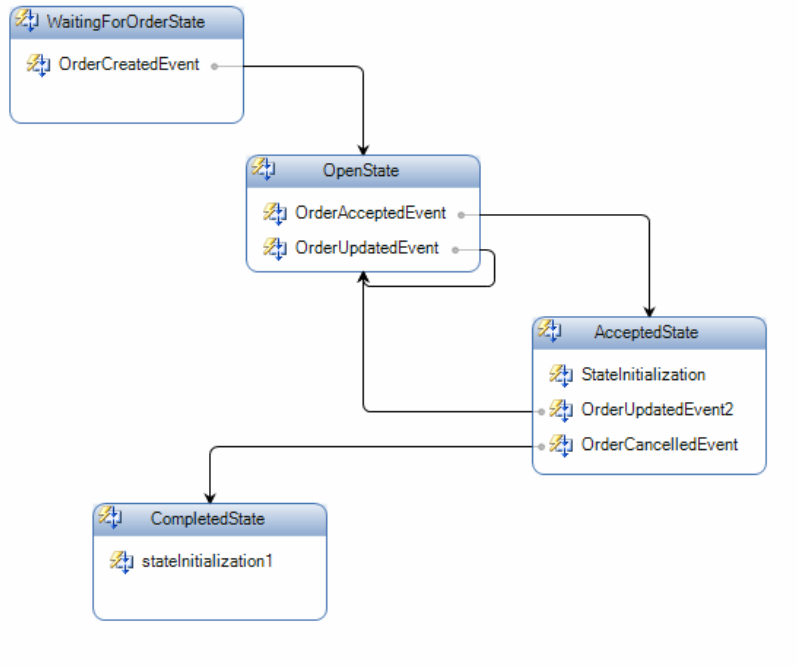
Terminate: ends a workflow's execution.

A common workflow requirement is to group activities, then define some action that should be taken across the entire group. One example of this is transactions, where all of the work performed by the grouped activities should either completely succeed or completely fail. To allow this, the base activity library includes a TransactionalContext activity that can contain other activities. A transaction context can be set to require an *atomic* transaction, which wraps all of the work in this context inside a traditional ACID transaction. A context can also be set to require what's known as a *long-running* transaction. Unlike an ACID transaction, a long-running transaction doesn't assume that all of the data modified inside it is locked for the transaction's duration. Given that workflows often coordinate long-running business processes, this makes sense. (Locking, say, your checking account for a week while waiting for a vacationing manager's approval of a funds transfer isn't an appealing prospect.) Since all of a long-running transaction's data isn't locked at once, it's not generally possible to atomically roll back all of the changes made within this transaction. Instead, some kind of *compensation* is typically used that attempts to undo the work that has already been completed. To allow this, a TransactionalContext activity can contain compensating activities that run if a long-running transaction fails.

Anyone with some experience in this area might have noticed that many of the default activities used in a sequential workflow are similar to statements in a conventional workflow/orchestration language such as the Business Process Execution Language (BPEL). Originally defined by Microsoft and IBM, BPEL is now in the process of being standardized by the Organization for the Advancement of Structured Information Standards (OASIS). BPEL is a language for defining system workflows, which is a subset of the more general approach taken by WWF. For developers who wish to use BPEL, WWF includes a BPEL Activity Library that implements the constructs defined by version 1.1 of the BPEL specification. Using this library, it's possible to create workflows that can be exported to BPEL or to import a BPEL definition into WWF.

Using State Machine Workflows

Unlike a sequential workflow, which structures its activities into a pre-defined pattern, a state machine workflow organizes its activities into a finite state machine. The diagram below shows a simple state machine workflow created using the WWF Workflow Designer. As this figure suggests, the developer defines a group of states and events, which trigger transitions between these states.



Organizing a workflow's activities like this is useful when the exact sequence of events isn't known in advance, such as for event-driven business processes, or when the number of possibilities makes defining all possible paths impractical. A primary example of this is a workflow that coordinates the work of people rather than just applications. State machine workflows make it easier to skip steps on the fly (perhaps a credit check can be bypassed for a good customer), jump to any step in the business process (maybe a high-priority request must be handled right now) or cancel the business process at any time (if, say, the customer cancels the order part way through). State machines can also provide a way to define workflows that matches how some organizations think about and document their business processes. In cases like this, modeling workflows in this fashion can help developers and business people work together more effectively.

The base activity library includes activities explicitly designed for creating state machine workflows. They are:

State: represents a state in a workflow's state machine.

EventDriven: defines a transition containing one or more activities that should be executed when a specific event is received while in a particular state.

SetState: changes the state of the workflow's state machine. A transition might or might not change the workflow's state.

StateInitialization: defines one or more activities that should be executed by default whenever a particular state is entered.

Although it's not shown in the example above, it's also possible to nest states. An event defined in an outer state applies both to that state and to all of the states it contains. This provides a straightforward way to express events that can occur at any time in any of these states, such as cancellation of an order.

Because a state machine workflow can also use the activities described earlier for sequential workflows, the actions taken within each transition can contain sequences of activities and other logic. This combination of WWF's two default workflow styles provides a more unified way to address the different styles required for human and system workflows.

Creating and Modifying Workflows

The simplest way to create and modify a WWF workflow is by using the Workflow Designer. As shown earlier, the Designer's default host is Visual Studio 2005, where it adds project templates for creating a sequential workflow, a state machine workflow, and more. By dragging and dropping various activities onto the design surface and setting their properties, developers can create new workflows and modify existing ones. The approach is similar to Windows Forms, where a developer drops controls onto a form to create a graphical user interface. With the Workflow Designer, activities are dropped onto a workflow to create business logic. In both cases, the goal is the same: higher developer productivity.

The Workflow Designer supports development only in C# and Visual Basic. A developer who wants to use the services of WWF but prefers to work directly in code is free to do so, however; using the Workflow Designer isn't required. In this case, a developer can use any language that's compliant with the .NET Framework's Common Language Specification, including C#, Visual Basic, C++, and others.

Under the covers, a workflow is really just a class. Three namespaces—System.Workflow.Activities, System.Workflow.ComponentModel, and System.Workflow.Runtime—provide the types needed to create and execute workflows and the activities they contain. To create a new sequential workflow, for example, a C# developer can use a skeleton like this:

```
using System.Workflow.Activities;

public class ExampleWorkflow : SequentialWorkflow

{

    …

}
```

When a new instance of a workflow class is created, its constructor creates and configures the activities in that workflow, much like the constructor of a form defined using Windows Forms initializes the controls that form contains.

Workflows can also be defined using XML. The basic skeleton for a workflow class is:

```
<?Mapping XmlNamespace="Activities"
   ClrNamespace="System.Workflow.Activities"
   Assembly="System.Workflow.Activities" ?>
<SequentialWorkflow x:Class="ExampleWorkflow"
   xmlns="Activities" xmlns:x="Definition">
    …
</SequentialWorkflow>
```

The **Mapping** processing instruction that begins this simple example is analogous to the **using** statement that begins the code example, since both specify a namespace that will be used. The **SequentialWorkflow** element then defines a class called **ExampleWorkflow** that uses this namespace.

Why provide this markup-based option for defining workflows? One reason is that some developers prefer to work in this style, at least for certain kinds of applications. Another reason is that many tool builders find it easier to create tools that generate and parse XML rather than generating and parsing code. In fact, WWF's Workflow Designer generates XML, so developers can always see the XML version of workflows created using this tool. A workflow can be built from any combination of Workflow Designer output, developer-written code, and XML, and however it's created, it's ultimately compiled into a standard .NET assembly.

Creating Activities

Activities are the building blocks of workflows. An activity can be small and technically-focused, such as Delay and Terminate in the base activity library, or it can be larger and more business-focused, such as a ProcessHelpDeskRequest activity that might be defined by an ISV building on WWF. Whether it's simple or complex, created by Microsoft or by others, every activity is built on the same set of WWF interfaces.

A new activity can be built from scratch, or it can extend an existing activity. A new activity can also be created that groups existing activities together to form a *composite* activity. Many of the activities in the base activity library, such as IfElse, Sequence, and While, are actually composite activities. In fact, a workflow itself is just a special kind of composite activity.

The simplest way to create an activity is by using the graphical Activity Designer tool. Activities can also be created directly in code, since like a workflow, an activity is just a class. Each activity can have events that it responds to and properties containing its state, and its C# definition looks something like this:

```
using System.Workflow.ComponentModel;

public class ExampleActivity : Activity

{

  …

}
```

However it's built, an activity's creator can control how much of its internals are visible to other developers who use that activity. This allows some control over the visibility of proprietary information. Activity creators can also define custom themes that determine the look and feel of activities, including their graphical representation in the Workflow Designer. This can make it easier for ISVs to build WWF into their applications while still retaining the style of their own product.

Using Conditions and Rules

Business processes depend on the rules of the business that uses them. Maybe an organization allows purchasing managers to approve POs only up to fifty thousand euros, for example, and gives customers over 55 a 10% discount, or perhaps the hotels employees stay in can't cost more than $250 a night. These simple statements are expressions of business rules. Handling business rules well is an important part of creating a workflow, and so WWF provides several approaches to this problem.

Defining Simple Conditions

In a typical workflow, each condition in an IfElse or While activity is actually a business rule. WWF provides two options for defining these conditions. One choice is to express the condition directly in code, creating what's called a *code condition*. To do this, the developer writes a method that evaluates the relevant data and returns a Boolean value. When the condition needs to be evaluated, this method is called, and the result it returns is used.

The other option is to define a *rule condition*, either directly in code or using a tool called the Rule Condition Editor. Rule conditions are stored in an XML format in a separate file. When a workflow reaches a rule condition, the condition's expression is evaluated and a Boolean value returned. Unlike code conditions, rule conditions can be changed on the fly for a running workflow. The same rule condition can also be used by multiple activities in the same workflow, allowing the business rule to be expressed (and thus maintained) in only one place.

Grouping Conditions and Activities: The CAG Activity

Activities such as IfElse and While provide a straightforward way to control which activities a workflow executes. For more dynamic scenarios, such as workflows that involve people, WWF's Conditioned Activity Group (CAG) activity can be useful. A CAG activity contains other activities, each of which is associated with what's known as a *when condition*. Each when condition can be specified using either a code condition or a rule condition. When a CAG activity is encountered in a workflow, all of its when conditions are evaluated. Every when condition that returns true has its associated activity executed. Since these activities can be composite activities, this execution can be arbitrarily complex.

The CAG's activities are executed in parallel, and once any of them has completed, the when conditions on all of the activities in the CAG that aren't currently executing are evaluated again. Any when conditions that evaluate to true once again have their associated activities executed. This cycle continues, checking when conditions and executing the activities associated with those that evaluate to true, until one of two things happens: either no more when conditions are true, or an *until condition* associated with the CAG activity itself becomes true.

In some ways, a CAG activity can be thought of as a mini-workflow, one driven by business rules. It allows execution of a group of activities based on which conditions are true. This is similar to, but not quite the same as, using a rules engine. For applications that need a full-fledged rules engine, however, WWF also provides this option, as described next.

Using a Rules Engine: The Policy Activity

For evaluating complex sets of business rules, building and maintaining an intricate group of conditions can be problematic. In the insurance application process described earlier, for example, each applicant must be checked against the potentially complex set of rules that make up the firm's underwriting policies. While expressing these rules using the WWF options described so far is certainly possible, it's not always the best choice. Instead, the right approach for handling complex groups of rules can sometimes be to use a rules engine.

To make this possible, WWF provides a rules engine that can be accessed via the Policy activity. Using this activity, a developer can define a group of rules called a *rule set*. Each rule has the form IF <condition> THEN <action> ELSE <action>. For example, an insurance company might create a Policy activity with a rule set containing all of its underwriting qualifications. Drivers under 21 might be ranked as high-risk, for example, unless they have good grades in school, while married drivers might be given a lower risk ranking. A workflow that needs to determine whether a particular applicant conformed to these rules could then invoke this Policy activity. The WWF rules engine would determine which rules in this rule set are true, then execute those rules. This execution might change the state of the workflow in such a way that other rules in the rule set have now become true. To handle this, the rules engine examines and, if necessary, executes any affected

rules in the rule set again, a technique known as *forward chaining*. This process continues until either no new rules become true or a predefined limit is reached.

Using the Policy activity requires writing some code. There's currently no default graphical representation for this activity in the Workflow Designer toolbox, for example, and so a developer must explicitly create a specific Policy activity. A developer must also write code to define the rules in a rule set.

Hosting the Runtime Engine

Every workflow depends on the WWF runtime engine. The engine executes each workflow and manages workflow state throughout the workflow's lifetime. The WWF runtime is a library, however, and so it must run in some host process. Rather than provide a single required host, WWF allows the runtime (and any workflows it executes) to be hosted in pretty much any Windows process, ranging from a simple console or Windows Forms application on up to a complex server designed with workflow in mind. WWF ships with a set of services that allow the runtime to execute within ASP.NET, but ISVs and end users are free to create their own hosts or to host the WWF runtime in existing applications. Other Microsoft products will also eventually host WWF, including BizTalk Server and Windows SharePoint Services.

Different hosts will have different characteristics. For example, a workflow running in a simple Windows Forms application on a desktop will be less scalable and less reliable than a workflow hosted in WSS or BizTalk Server. Because WWF is a workflow framework rather than a standalone product, supporting this kind of diversity is an explicit goal of its creators. And even though every host uses the same runtime engine, each host must provide a set of runtime services, as shown in the earlier diagram. These services provide support for persisting the state of a workflow, tracking a workflow's execution, using transactions, and more. WWF provides a default implementation for all of these services that can be used by any host. To customize a host to meet unique requirements, however, that host's creator can replace one or more of these defaults if desired.

To get a sense of how the WWF runtime works with the host-provided services, think about one of the fundamental aspects of a workflow: it can be long-running. Because a workflow might run for hours, days, or weeks, the WWF runtime will automatically shut down a running workflow and persistently store its state if it has been inactive for a period of time. The decision to unload[3] the workflow, perhaps because it's blocked waiting for an external event, is typically made by the runtime. To actually write the workflow's state to disk, however, the runtime depends on the persistence service provided by its host process. The ASP.NET-based host that ships with WWF relies on SQL Server or the freely-distributable SQL Express for persistence. Another host might prefer to persist

[3] Unloaded workflows are sometimes said to have been *dehydrated*.

workflows using some other technology, such as a proprietary database used by an ISV application.

Whatever services the host provides, the basics of hosting the WWF runtime are simple. Here's an example:

```
using System.Workflow.Runtime;

class ExampleHost

{

 static void Main()

 {

   WorkflowRuntime runtime = new WorkflowRuntime();

   runtime.StartRuntime();

   runtime.StartWorkflow(typeof(ExampleWorkflow));

   …

 }

}
```

As this sample shows, the runtime is just another class. Once an instance of this class is created, it can be initialized by calling its StartRuntime method, then given a workflow to execute via its StartWorkflow method.

The host process is a fundamentally important part of the environment in which a workflow runs. Because the runtime can be hosted in a range of Windows processes, WWF workflows can be used in a variety of different scenarios.

Communicating with Software Outside the Workflow

Given that the role of a typical workflow is to coordinate the work of people and/or systems, WWF must provide some way for a workflow to communicate with other software. Yet recall that an idle workflow will be unloaded, with its state persisted to disk. A workflow in this condition can't directly receive any inbound communication—it's not even running. Because of this possibility, the WWF runtime acts as an intermediary for all communication with all workflows. When an incoming request arrives, the runtime receives it, then determines which workflow instance this request is destined for, a process known as *correlation*. The runtime then delivers the request to the target instance, first reloading that instance if it is dehydrated. In effect, the WWF runtime acts as a proxy for all communication with software outside the workflow.

To communicate with other objects in the same Windows process, a workflow can use two activities in the base activity library. The InvokeMethod activity allows calling a method in an object outside the workflow, while the EventSink Activity allows receiving a call from an

object outside the workflow. A developer defines an interface to specify the names and parameter lists for each of these calls.

WWF also provides built-in activities for communicating via web services. Built on ASP.NET web services, better known as ASMX, these activities are:

InvokeWebService: calls a web service and synchronously returns the response.

WebServiceReceive: accepts an incoming web service call.

WebServiceResponse: sends a response to an incoming web service call that was received via WebServiceReceive.

Using WebServiceReceive and WebServiceReponse, a workflow can expose itself to clients via web services. This allows workflows to publish web services as well as consume them.

Tracking Workflow Execution

Workflows are built from activities, each of which is a well-defined unit of execution. This standard structure makes it possible to track the execution of any workflow in a standard way. WWF can provide tracking information for any workflow, including things such as when the workflow begins execution and when it ends, when each activity within the workflow is entered and exited, and more. This can be useful for gathering statistics about technical or business aspects of workflows, such as measurements of order fulfillment time, or maintaining a database with information about every running workflow, allowing real-time querying of this data, or other things. Exactly what tracking information is produced is controlled by developer-defined profiles, and the information is by default written to a SQL Server database. (This can be changed, however, since tracking is partially implemented by a replaceable runtime service in a workflow's host application.)

WWF's tracking facility is relatively simple. While it does provide the interfaces needed to define and collect tracking data, no tools are provided to display this information. There's also no standard mechanism to send notifications based on tracking data. Rather than providing a complete service akin to the Business Activity Monitoring component in BizTalk Server, WWF provides a foundation that ISVs and others can build on.

Modifying Running Workflows

If WWF were focused solely on system workflows that coordinate interactions among software, the ability to define a workflow and execute it unchanged would probably be sufficient. People tend to want their business processes to be more dynamic, however, and so supporting human workflow requires the ability to make changes on the fly. Suppose one of the participants in a currently executing workflow decides to add a new step to the business process, for example. Maybe the manager in the insurance application process described earlier wishes to get a second opinion about a particular

high-risk applicant, a change that requires adding another step to a particular running workflow instance. Whether the developers who create a workflow like it or not, the people who use automated business processes expect to be able to make these kinds of changes.

To allow this, WWF includes *dynamic update*. Using this option, a running instance of any workflow can be modified within safe, well-defined boundaries specified by its creator, then potentially saved as a new workflow. A new activity can be inserted into the workflow, for example, or a rule condition changed for an IfElse or CAG activity. While this kind of in-flight update isn't typical for purely system-oriented workflows, it can be essential for workflows that coordinate the activities of fickle human beings.

Supporting Human Workflows

As already mentioned, supporting both human and system workflows is a primary goal of WWF. Most of the WWF components that developers are likely to use to create workflows involving people have already been described. Those technologies include:

State machine workflows, which let the steps in a workflow be defined in a more dynamic, event-driven way than with sequential workflows.

The CAG activity, allowing rules-based execution of a group of activities.

Dynamic update, which lets a user make changes to a running workflow.

The Replicator activity, another member of the base activity library, is also worth mentioning in this context. As its name suggests, this activity is capable of replicating whatever activity is placed into it a specified number of times. A Replicator might be used, for example, in a workflow that allows a manager to assign a particular task to all of her direct reports, with the Replicator activity creating an instance of that task's activity for each one. By allowing the number of replicas to be specified at runtime, the Replicator activity provides a dynamic way for a workflow to assign work to the people involved in a business process.

One more aspect of WWF that's relevant to how people use workflows is its support for *roles*. WWF implements a general infrastructure for controlling which users are authorized to execute a particular activity based on that user's role. The creator of a workflow can define the roles used in that workflow, then determine which activities can be executed by which roles. Perhaps an insurance application can be submitted by anybody in the Employee role, for example, but that report can be approved only by someone in the Manager role. And rather than implement its own system for managing roles, WWF instead provides a mechanism that lets workflows use external role management systems. WWF ships with some built-in support for roles defined using Windows accounts, Active Directory, and ASP.NET, but other role management systems can also be integrated into this workflow framework.

Windows Workflow Foundation and Other Microsoft Technologies

WWF provides a general framework for workflow, one that can be used in many different kinds of applications. Once it ships, it will be a standard part of the Windows platform. This raises an important question: how does WWF fit with other parts of the Microsoft environment?

This section takes a look at how this new technology will be used in three other Microsoft offerings: BizTalk Server, Windows SharePoint Services, and the Microsoft Office System. Each one applies WWF in a specific way, using it to support workflows in a particular domain. This section also describes how WWF fits with Windows Communication Foundation, another part of the WinFX developer platform.

Windows Workflow Foundation and BizTalk Server

Perhaps the most well-known Microsoft implementation of workflow today is in BizTalk Server. BizTalk Server lets developers create system workflows[4] for business process management (BPM), enterprise application integration (EAI), and business-to-business (B2B) integration. The release of the product that follows BizTalk Server 2006 will add support for creating WWF workflows targeting these areas.

BizTalk Server and WWF have some obvious similarities. To a developer, for example, the BizTalk Server Orchestration Designer looks much like WWF's Workflow Designer. This commonality is no accident—the same group at Microsoft is responsible for both technologies. For the most part, however, deciding which one to use for a particular problem isn't hard. What follows are some guidelines to help in making this decision.

Use WWF when:

An application will itself host workflows. WWF lets workflow be built into an application, allowing the workflow to be deployed and managed as a native part of the application. Because it's focused on integrating diverse applications rather than providing a general workflow framework, BizTalk Server always runs orchestrations within the BizTalk Server process.

The business process being implemented requires human workflow. BizTalk Server addresses system workflow, and so it lacks WWF's support for things such as state machine workflows and dynamic update. A scenario that requires both human workflow and more complex system integration services could be addressed by using WWF and BizTalk Server together, however. For example, the Office "12" support for

[4] BizTalk Server uses the term *orchestration* rather than workflow.

document-centric workflows, based on Windows SharePoint Services, might be used for the human aspects of the problem, while BizTalk Server handles the system integration aspects. The two can interoperate using the BizTalk Server Adapter for SharePoint.

The workflow will execute on a client system. BizTalk Server is a server-focused product, and so it's less well-suited to run on desktop machines.

Use BizTalk Server when:

Solving an EAI problem that requires communication with diverse applications on diverse platforms. Because of its focus on cross-platform integration, a large set of adapters is available for BizTalk Server that allows communication with a range of other software. WWF is focused solely on workflow, not EAI, and so it doesn't provide these things.

B2B services are required. WWF doesn't address this area, while BizTalk Server provides tools for working with trading partners, accelerators for RosettaNet, SWIFT, and other industry standards, and more.

BPM services, such as Business Activity Monitoring (BAM) are needed. While the WWF tracking infrastructure can be used to create these services, BizTalk Server provides important extras, such as tools that let information workers define their own BAM views.

A complete management infrastructure and support for increased scalability are necessary. Unlike WWF, BizTalk Server includes a full set of tools for administering and scaling a production environment.

Windows Workflow Foundation and Windows SharePoint Services

Windows SharePoint Services, a standard part of the Windows Server operating system, aims at helping people and software collaborate more effectively. This makes support for workflow a natural extension to the technology. Accordingly, version 3 of WSS, scheduled for release in 2006, will host the WWF runtime. Using standard WWF tools, such as the Workflow Designer, developers will be able to create workflow applications that address document collaboration and other kinds of information sharing. Hosting WWF in WSS also provides the foundation for workflow support in the Microsoft Office System, as described next.

Windows Workflow Foundation and the Microsoft Office System

No Windows application is more widely used today than Microsoft Office, and so Office's adoption of WWF is an important step. As Office has evolved from a group of desktop

applications into the broader combination of client and server technologies that make up the Microsoft Office System, the problems it addresses have also evolved. Office "12", scheduled to be released in 2006, will include support for creating document-centric workflows. That support will depend on WWF.

Workflow in Office "12" server and client products will rely on and extend the WWF-based workflow support in Windows SharePoint Services. Supporting workflows covering formal document processes, Web content management, and more, Office "12" will also allow less technically-oriented people to create document-centric workflows. Using Microsoft FrontPage, web designers and others will be able to use a rule-based approach rather than the graphically-oriented technology of the Workflow Designer. Office "12" servers will also include a set of Office-specific WWF activities for working with tasks and documents, along with support for creating forms using InfoPath.

Like BizTalk Server and Windows SharePoint Services, Office "12" specializes WWF's generic workflow support for a particular kind of application. The objective is to make it easier for developers and others to create document-centric workflows for information workers. Given the popularity of Office desktop products, it's not surprising that Microsoft wants customers to rely on its familiar interfaces to interact with workflows. Incorporating WWF into Office "12" will help make this possible.

Windows Workflow Foundation and Windows Communication Foundation

Windows Communication Foundation (WCF) is Microsoft's forthcoming framework for creating service-oriented applications. Implementing a large group of industry-standard web services specifications, WCF allows developers to create applications that communicate securely and reliably both within the Windows world and across vendor boundaries. Given this, WCF and WWF are a natural combination. While WCF focuses on building services, WWF allows creating workflows that use those services to support business processes. Although the web services support in WWF's first beta release relies on ASMX, WWF activities that let workflows use this new communications infrastructure will be available in the future.

Conclusion

Without the right foundation, writing workflows is hard. Meeting the needs of a long-running business process, supporting the dynamic behavior that people require, and handling the other challenges that workflows present all require more time and effort than most developers can invest. Yet if the right supporting technology is available, creating workflows becomes straightforward, and this useful type of software can be much more widely used.

The goal of WWF is to provide this foundation. By building a general framework that supports both human and system workflow, and by making it a standard part of the Windows environment, Microsoft is laying the groundwork for the pervasive use of workflow technology. After years of being relegated to specialized uses and applications, workflow is about to go mainstream.

About the Author

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps technology professionals around the world understand, use, market, and make better decisions about enterprise software.