Workflow Architecture

Objectives

- Why Workflow?
- Activities
- Workflows
- The Workflow Designer
- Persistence
- Messaging

Business Processes

- **Business processes model real world activities**
  - Hiring new starter
  - Processing insurance claim
  - Commissioning new server
  - Stock tracking and ordering
- **Business processes have general requirements**
  - Auditable
  - Understandable by business users
  - Potentially long running
  - Non-linear flow, particularly when humans involved

Software Solutions

- **Software solves technical problems**
  - Reading and writing from databases
  - Sending data across networks
  - Performing complex calculations
  - Reacting to user input
- **Software has general requirements**
  - Maintainability
  - Performance
  - Re-use
  - Secure

Applying software to business processes

- **Mismatch between business process and technical solution**
  - Requires orchestration of technical details into business solution
- **Traditional approaches fail to meet business general requirements**
  - Requires large amounts of plumbing code and development process discipline

!=

Applying Workflow to Business Problems

- **Workflow provides a framework to orchestrate technical solutions into a business solution**
  - Often associated with human involvement and work items
  - General model for all kinds of problems
- **Workflow can solve general business requirements**
  - Often graphical model allows audit and aids business user understanding
  - With persistence infrastructure can maintain state of long running process
  - Can model process flow based on external events

Windows Workflow Foundation

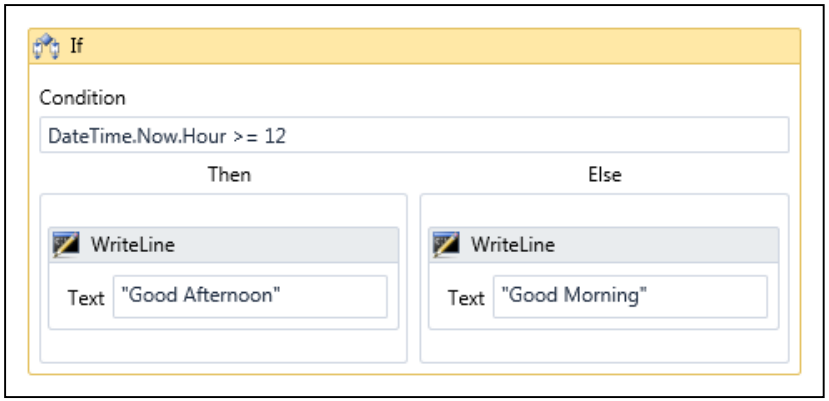
- **Windows Workflow Foundation (WF) is a framework for workflow based execution**
  - First introduced in .NET 3.0
  - Rewritten for .NET 4.0
  - Polished in .NET 4.5
- **Ships in a number of assemblies**
  - System.Activities
  - System.Runtime.DurableInstancing
  - System.ServiceModel.Activities

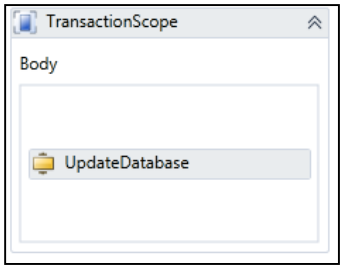
WF Concepts

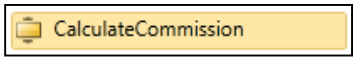
- **Several core concepts in WF**
  - Activities
  - Workflows
  - Persistence
  - Messaging

Activities

- **Activities are units of work**
  - Technical units to compose into business functionality
- **Activities come in a number of types**
  - Utility
  - Flow control
  - Infrastructure
  - Custom







Standard Activity Library

- **The standard activity library is the set of activities available by default**

Control Flow

```
DoWhile
ForEach<T>
If
ParallelForEach<T>
Pick
PickBranch
Sequence
Switch<T>
While
```

FlowChart

```
FlowChart
FlowDescision
FlowSwitch
```

Runtime

```
Persist
TerminateWorkflow
```

Messaging

```
CorrelationScope
InitializeCorrelation
Receive
ReceiveAndSendReply
Send
SendAndReceiveReply
TransactedReceiveScope
```

Primitives

```
Assign
Delay
InvokeMethod
WriteLine
```

Transaction

```
CancellationScope
CompensableActivity
Compensate
Confirm
TransactionScope
```

Collection

```
AddToCollection<T>
ClearCollection<T>
ExistsInCollection<T>
RemoveFromCollection<T>
```

Error Handling

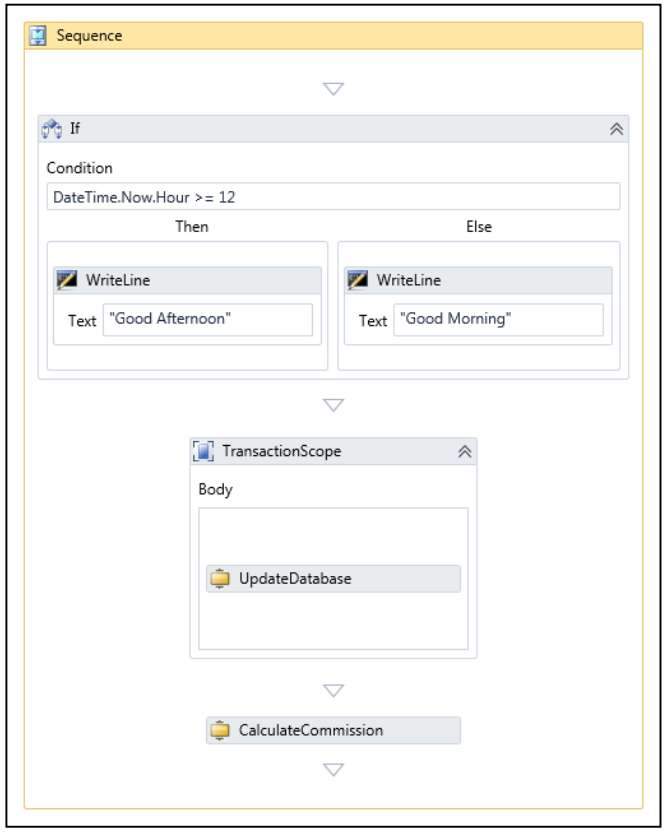
```
Rethrow
Throw
TryCatch
```

Migration

```
Interop
```

Workflows

- **Workflows are trees of activities that are executed as a unit**
  - Contains the units of work required to produce the business functionality

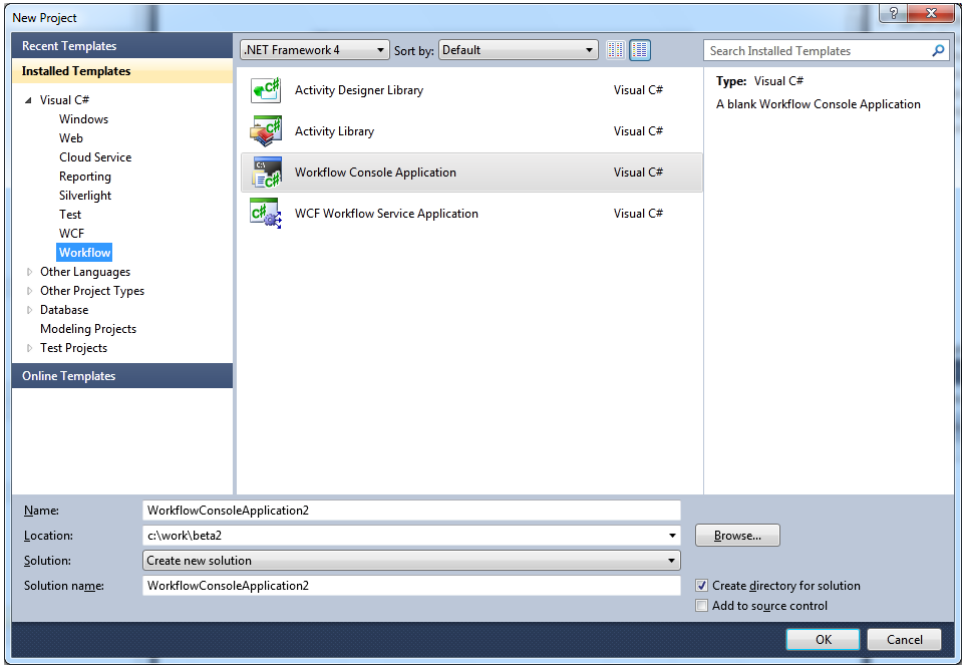


Workflow Execution Modes

- **Out-of-the-box three execution models:**
  - Sequential
  - Flow Chart
  - State machine
- **Sequential workflow**
  - Starts at top and execution proceeds downwards although loops and branches allowed
  - Good for processes with no human intervention
- **State Machine**
  - Modeled as states with transitions
  - Good for processes with defined states (often involving humans)
  - Out of the box with 4.5
- **Flow Chart**
  - Arbitrary switches of control
  - Good for full flexibility

Workflow Projects

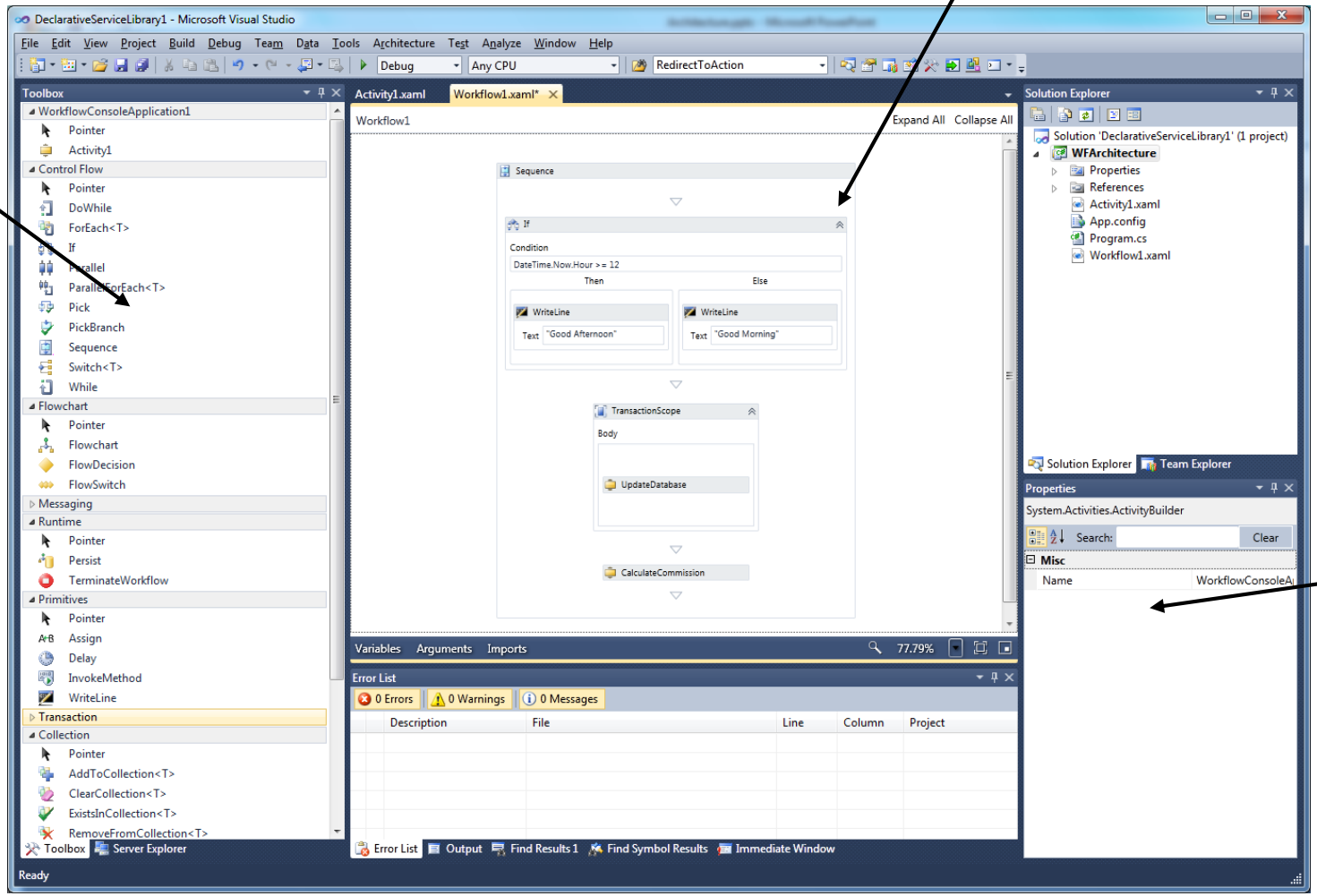
- **In Visual Studio there are four workflow project types:**
  - Activity Designer Library
  - Activity Library
  - Workflow Console Application
  - WCF Workflow Service Application



Workflow Designer

Visual Design Surface

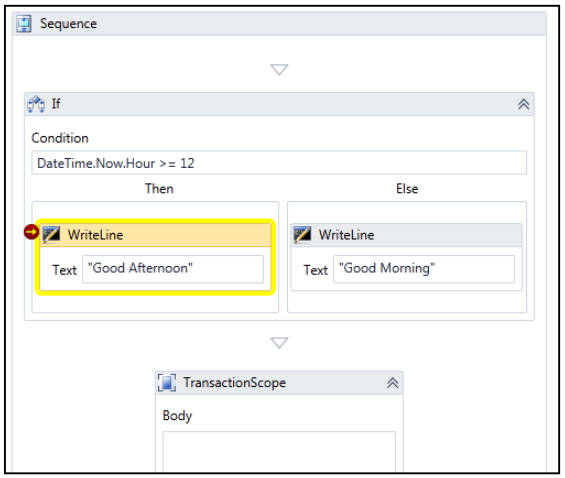
Toolbox



Property Grid

Rich Visual Designer

- **Workflow Designer is a rich graphical design environment**
  - Drag and Drop assembly of components
  - Zoom
  - Navigation
- **4.5 smooths some rough edges**
  - Multi-select
  - Auto-insert of sequence when dropping multiple activities
- **Designer also provides visual debugging**



Data Flow

- **Need a way to flow data between activities**
  - Outputs of one activity needed as inputs to another
- **Activities have arguments**
  - Explicitly define inputs and outputs
- **Parent activities have variables**
  - Named, typed data slots to store data
  - Available on any composite activity (activity with child activities)

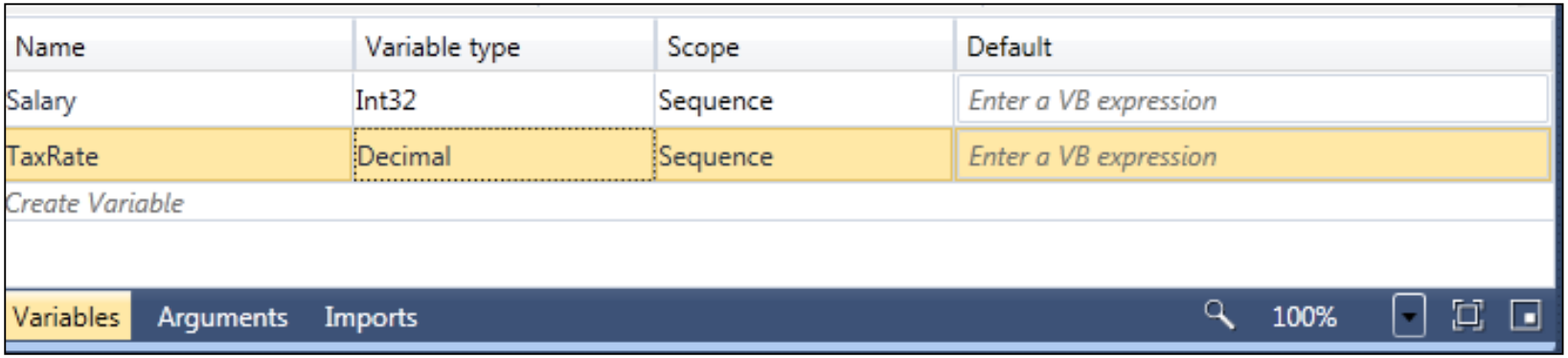
Arguments

- **Arguments have direction**
  - In, Out, InOut

```
public class TaxRateCalculator : CodeActivity
{
  public InArgument<int> Salary { get; set; }
  public OutArgument<double> Rate { get; set; }
  protected override void Execute(CodeActivityContext context)
  {
    int salary = Salary.Get(context);
    if (salary < 34500)
      Rate.Set(context, 0.22);
    else
      Rate.Set(context, 0.4);
  }
}
```

Variables

- **Arbitrary variables can be associated with any activity**
  - can be bound to arguments of any activities under the declaring one
  - New Variables dialog used for managing variables in scope

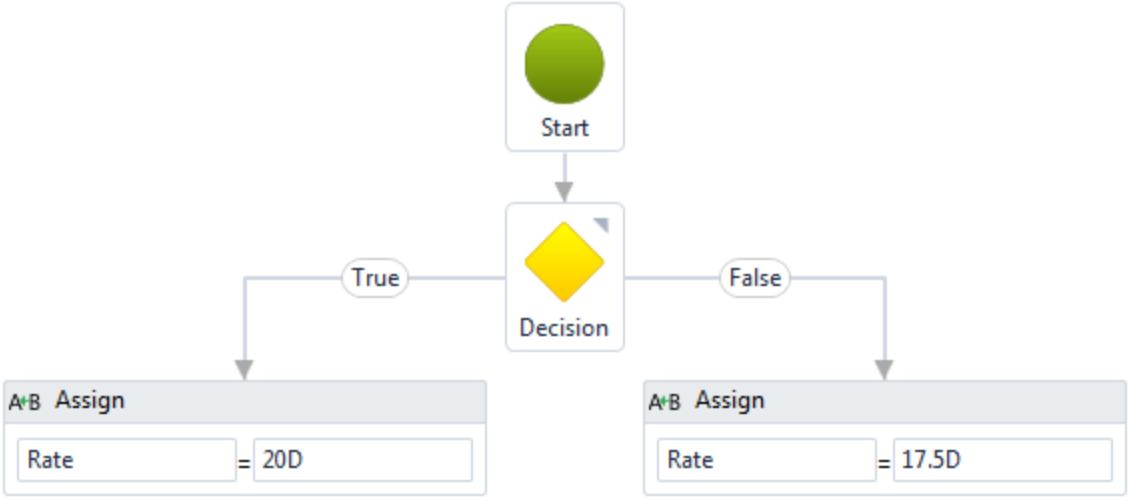


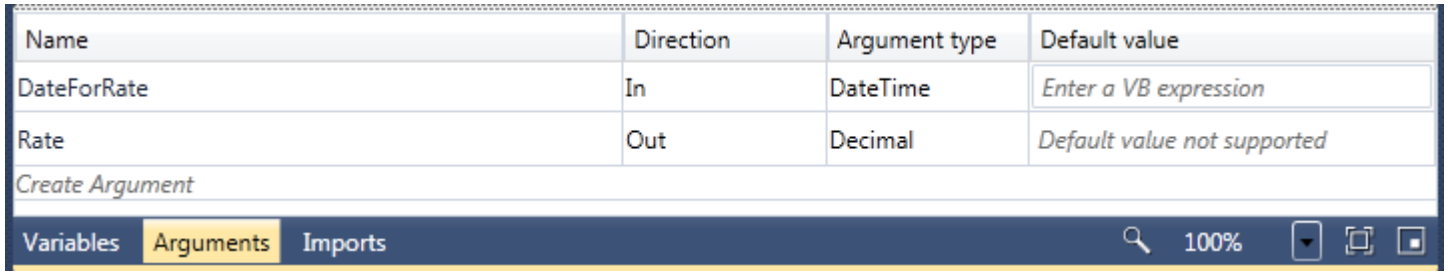
Expressions

- **Conditions and variable manipulation use expressions**
  - Must be VB.NET in 4.0
  - In 4.5 C# for C# projects and VB.NET for VB.NET projects
- **Give powerful mechanism for injecting snippets of logic**

Custom Activities

- **Custom activities normally built as composite activities in XAML**





Name	Direction	Argument type	Default value
DateForRate	In	DateTime	*Enter a VB expression*
Rate	Out	Decimal	*Default value not supported*
Create Argument			

Variables | Arguments | Imports | 100%

Creating “Building Block” Actvities

- **If the building block of a composite doesn’t exist three options**
  - derive from CodeActivity
  - derive from AsyncCodeActivity
  - derive from NativeActivity
- **CodeActivity**
  - simple synchronous custom activity
- **AsyncCodeActivity**
  - simple asynchronous custom activity
- **NativeActivity**
  - all features of workflow runtime available

Async Activities

- **Async crucial to long running execution**
  - Workflow spends most of its time waiting
- **Two async modes**
  - Bookmarks
  - AsyncCodeActivity
- **Bookmarks**
  - simplified access to queue based workflow communication
  - workflow will go idle and can be unloaded
- **AsyncCodeActivity**
  - lightweight async infrastructure
  - allows low latency async (e.g. Async IO)
  - workflow will go idle but cannot be unloaded

Bookmarks

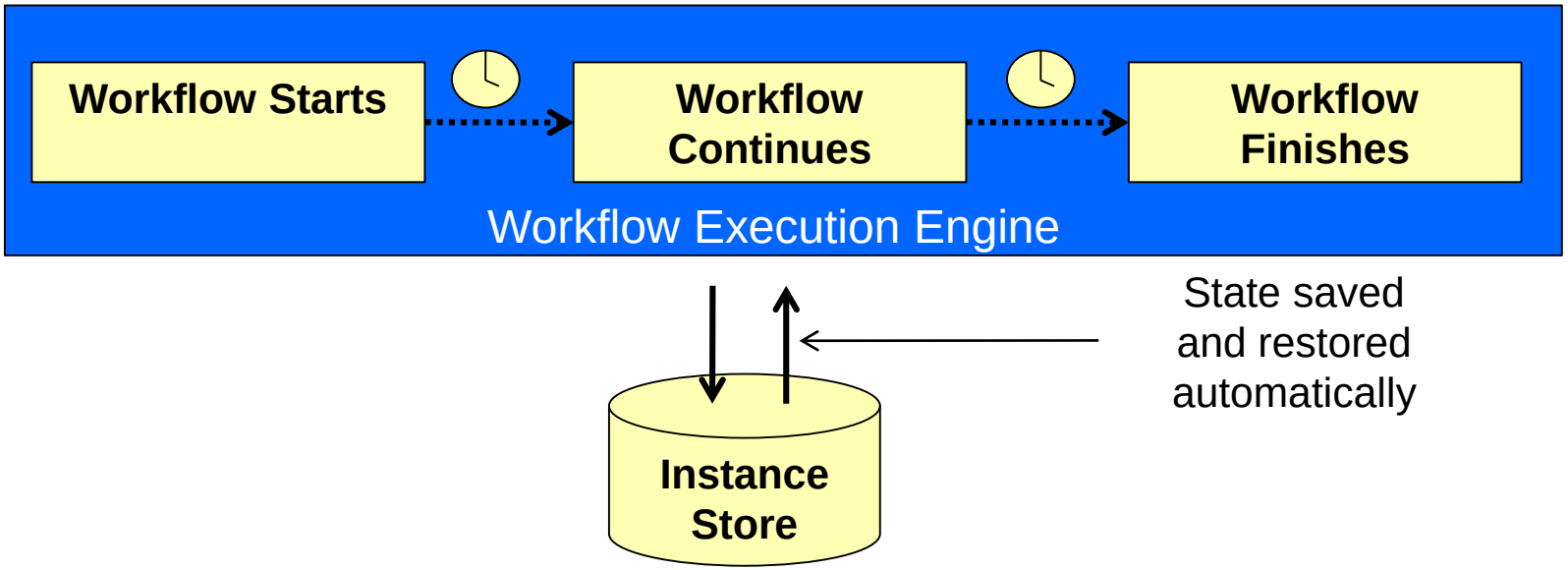
- **Used for long running async wait**
- **Only usable in NativeActivity**
- **Execute method can declare bookmarks**
  - Activity not complete until no outstanding bookmarks
- **Bookmarks complete asynchonously**
  - Workflow can go idle and be persisted with outstanding bookmark

AsyncCodeActivity

- **Derive from AsyncCodeActivity**
  - Implement BeginExecute to initiate async processing
  - Implement EndExecute to collate results of async processing
- **Performing more than one async operation requires custom implementation of IAsyncResult**
- **No support for Task based async in 4.5**

Long Running Execution

- **Technical issue for a workflow to support long running execution**
  - Must survive machine and process restart
  - Must be able to continue from a previous saved place
  - Must provide model for “waiting for information/event”
- **WF provides persistence infrastructure called InstanceStore**



Persistence

- **Persistence infrastructure uses an instance store**
  - must add the **store to the application**

```
string conn = "server=.;...";
WorkflowApplication app =
                  new WorkflowApplication(new Workflow1());
var store = new SqlWorkflowInstanceStore(conn);

app.InstanceStore = store;

app.Run();
```

When does persistence happen?

- **Host must take explicit control of when to persist**
- **PersistableIdle delegate invoked when idle and can persist**
  - return **PersistableIdleAction**

```
app.PersistableIdle = delegate
{
    return PersistableIdleAction.Unload;
}
```

Versioning

- **No built in versioning support in 4.0**
- **4.5 introduces versioning framework**
  - Based on WorkflowIdentity
- **Number of options**
  - Workflow can execute with definition it started with
  - Workflow can execute with new definition
  - Workflow can be loaded and activity tree manipulated at runtime

Messaging

- **Workflows need to receive data from outside world**
  - Can “seed” a workflow when starts executing
  - Need a way to pass data in during execution
- **Workflows need to send data to outside world**
  - Can produce results accessible after workflow complete
  - Need a way to send data out during execution
- **Messaging activities provide data in and out during execution**
  - Integrates with WCF

Visual Scripting

- **Designer can be re-hosted in your application**
  - Can constrain toolbox activities
  - Allows end users to create custom “scripted” execution of standard components
  - Provides powerful extensibility model

Summary

- Workflow bridges the technical world to the business world
- Includes a lot of out-of-the-box functionality
- Highly extensible
- Rich design environment
- Supports long running execution via persistence
- Arbitrary data flow enabled via WCF integration
- 4.5 polishes a number of rough edges