

Unit 2

Software Development by Composition and Integration

Lecture 2-2

Workflow-Based Software Development Concepts

Dr. Yinong Chen

- 2-1 { ■ Enterprise Architecture and Business Process
- ➔ 2-2 { ■ **Workflow Foundation 1: Concepts**
- 2-3 { ■ Workflow Foundation 2: Case Study
- 2-4 { ■ BPEL (Business Process Execution Language)
 - WSDL in BPEL
 - BPEL constructs and BPEL Process Definition
- 2-5 { ■ A Case Study of BPEL Application
- 2-6 { ■ Stateful Services
- 2-6 { ■ Development Frameworks Supporting BPEL
 - Oracle SOA Suite and BizTalk
- 2-7 { ■ Message-Based Integration
- 2-8 { ■ Web Caching and Recommendation

Unit Test 2



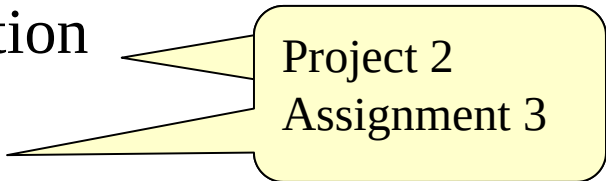
Lecture 2-2 Outline

- Previous lecture focused on the **architecture** design and **management**



■ Workflow for Architecture and Process Integration

- Cover the Gap between Architecture (CSE360) and Development (Programming Courses)
- Key Ideas of Workflow-based Application Development
- **Workflow Foundation** Constructs and Activities
 - Creating WF Workflow Application
 - Creating WF Workflow Service
 - Event-Driven Approach and State Machine in WF



Project 2
Assignment 3

What is Workflow?

4

- Workflow is a new solution to an old problem: Integrating, managing, and supporting business process.
- Workflow offers a new model for the division of labor between people and computer:
 - People do only those that computers cannot: What we want.
- Workflow better matches the business logic that customers require, and thus the customers can better understand the solution offered by the software engineers / architects.
- Workflow better separates the tasks of software **architects** and **programmers**.
- Service-Oriented Architecture makes workflow easier.



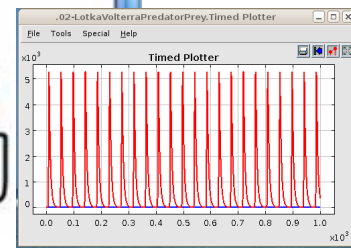
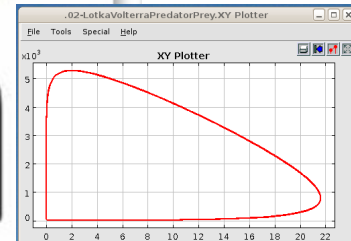
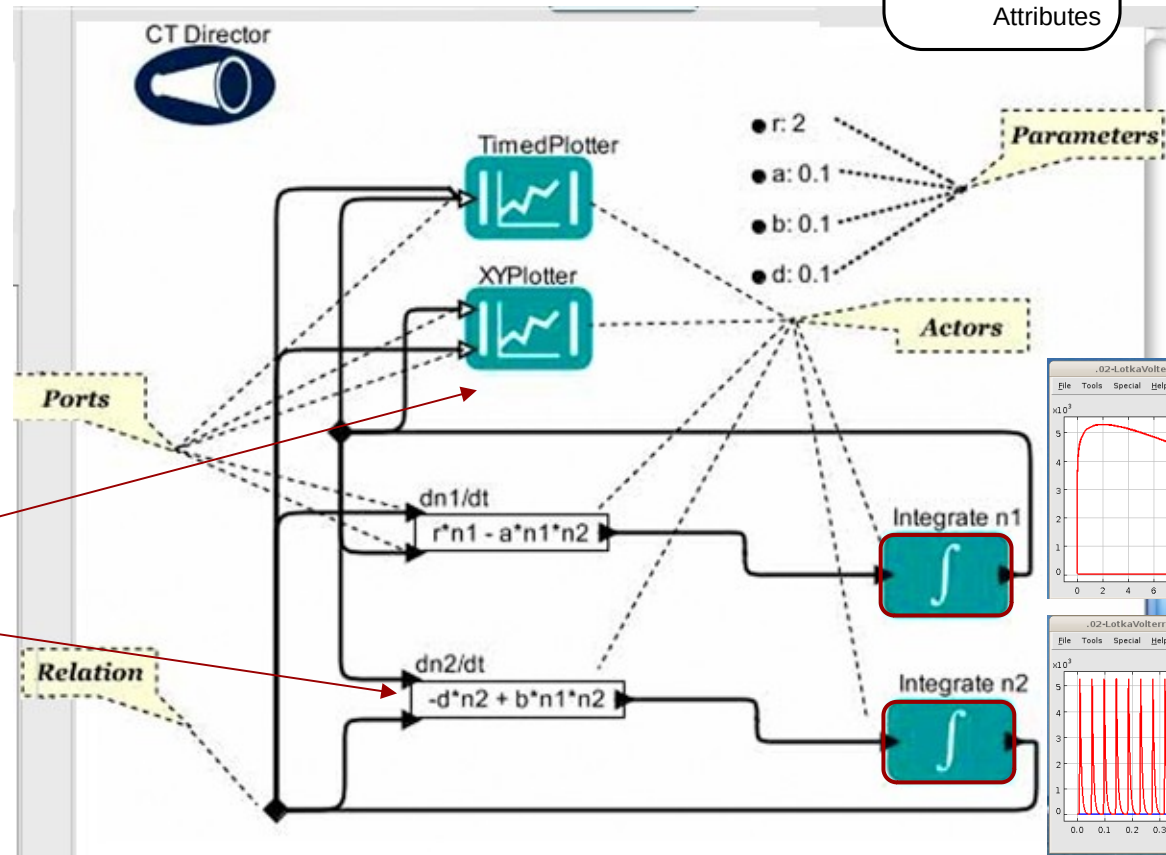
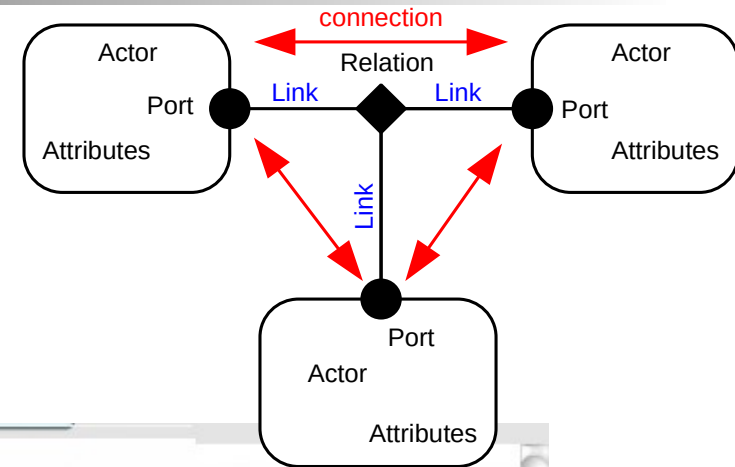
- Operate on **data** stored in a variety of formats, locally and over the internet;
- Integrating disparate **software components**, such as merging scripts with compiled code;
- Facilitating **remote**, distributed execution of models.
- Select and then connect relevant analytical components and data sources to create a "scientific workflow" — an **executable** representation of the steps required for generating results;
- Share and **reuse** data, workflows, and components developed by the community to address common needs.

Kepler Workflow Style

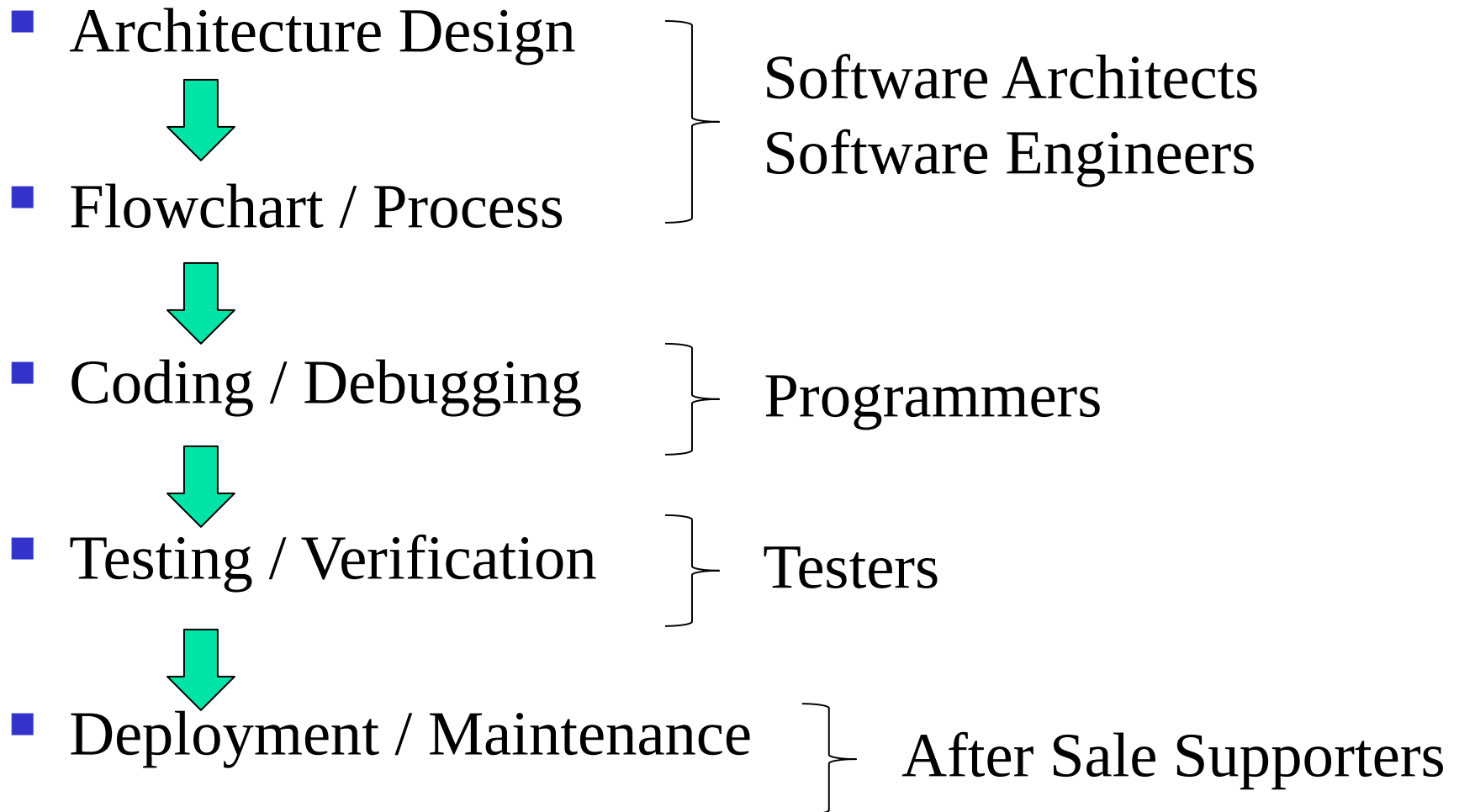
- Directors (define computing and communication models)
- Actors
 - Ports
 - Attributes
- Relations

- Example:
Integration of
a template,
a data set

Director



Traditional Software Development



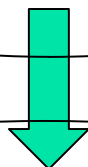
Architecture and Process Integration

Application Builders

- Architecture Design



- Flowchart / Process



Component / Service
Repository

Service
Providers

Service
Providers

Service
Providers

Integration Tools

- Kepler

- BPEL

- Workflow Foundation

- For integration

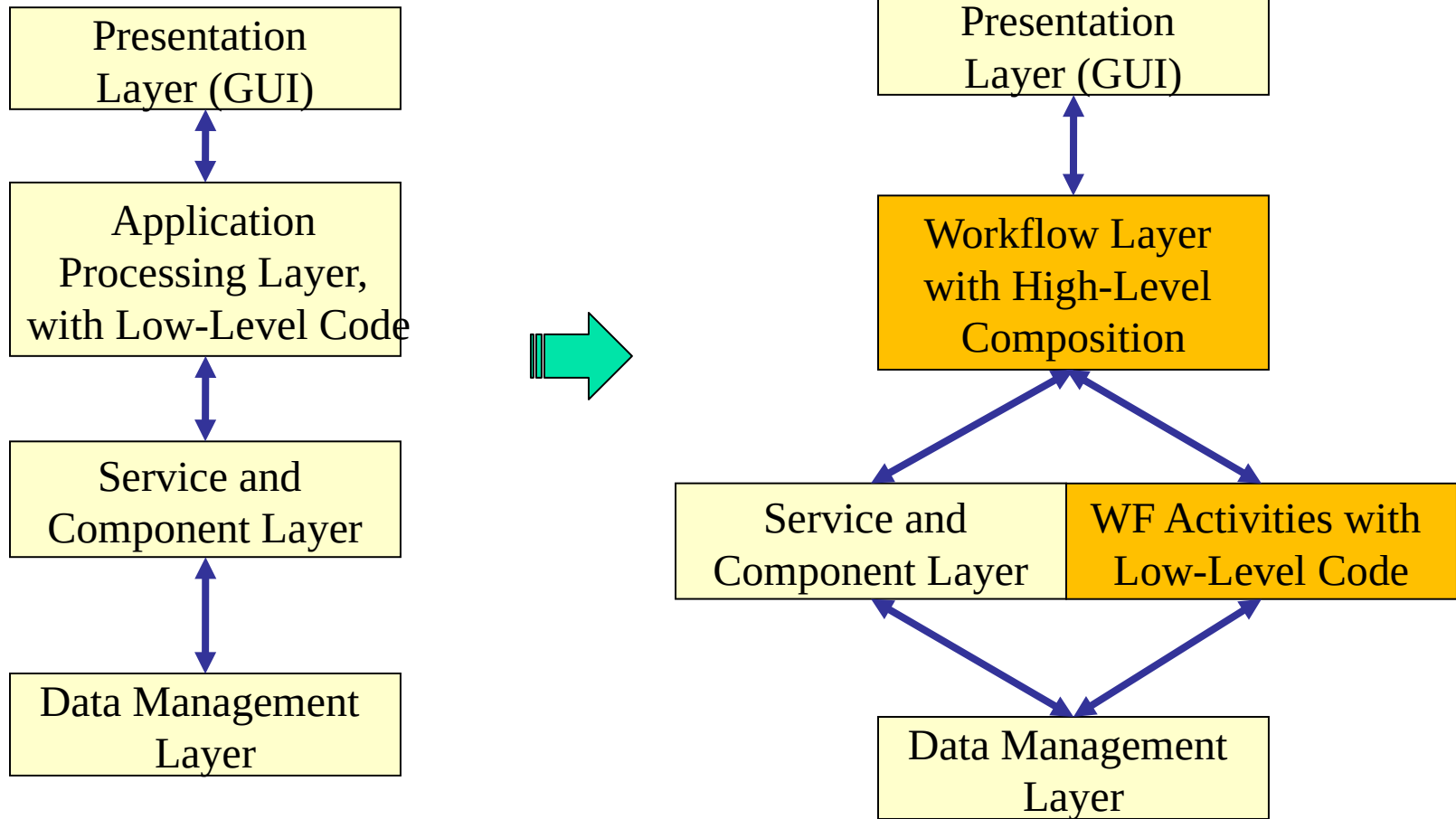
- For creating services

Why do we need
this?

Why Do We Need Workflow for Coding?

- Adding a layer of **abstraction** (graphical approach) to make application development faster and easier.
- It makes the application's main logic more visible.
- By providing a straightforward picture of what is going on, the architect can help developers more quickly to understand an application's structure.
- This can be especially helpful for the people who must maintain the deployed applications, since learning a new application well enough to change it can be a time-consuming process.
- Allows traditional programming style to seamlessly integrate with workflow

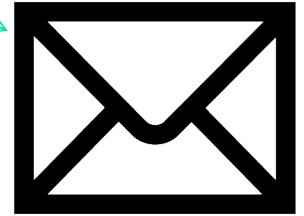
From Tiered Architecture Perspectives



Example: How Do I Write Code to Send a Mail/Message to a Client in My Web Application?

11

```
MService.ServiceClient TestSvcProxy = new MService.ServiceClient();
string EmailAddress = "account@gmail.com";
string Password = "password";
MService.CARRIER PROVIDER = MService.CARRIER.NAME;
string address, Message;
System.Console.WriteLine("Please enter an address: ");
Address = System.Console.ReadLine();
System.Console.WriteLine("Please enter the service provider: ");
System.Console.ReadLine();
System.Console.WriteLine("\nPlease enter a message: ");
Message = System.Console.ReadLine();
bool Result = TestSvcProxy.SMS(EmailAddress, Password, address,
PROVIDER, Message);
if (Result)
    System.Console.WriteLine("\n\nMessage has been sent.");
```



Mail Service

Example: Drag and Drop Designer in WF

<http://msdn.microsoft.com/en-us/library/ee342461.aspx>

Workflow Editor

Search

Basic

- Sequence
- SendMail

Sequence

Expand All Collapse All

CustomActivities.SendMail

Misc

DisplayName	SendMail
From	"donotreply@...
MailBody	Enter a VB ex...
Result	Enter a VB ex...
Subject	"Employee Ex...
ToAddress	Enter a VB ex...

To: mgrEmail

From: "donotreply@contoso.org"

Subject: "Employee Expense Report"

Body: emailBody

Name	Variable type	Scope	Default
mgrEmail	String	Sequence	Enter a VB expressi...
emailBody	String	Sequence	Enter a VB expressi...

Create Variable

Variables Imports

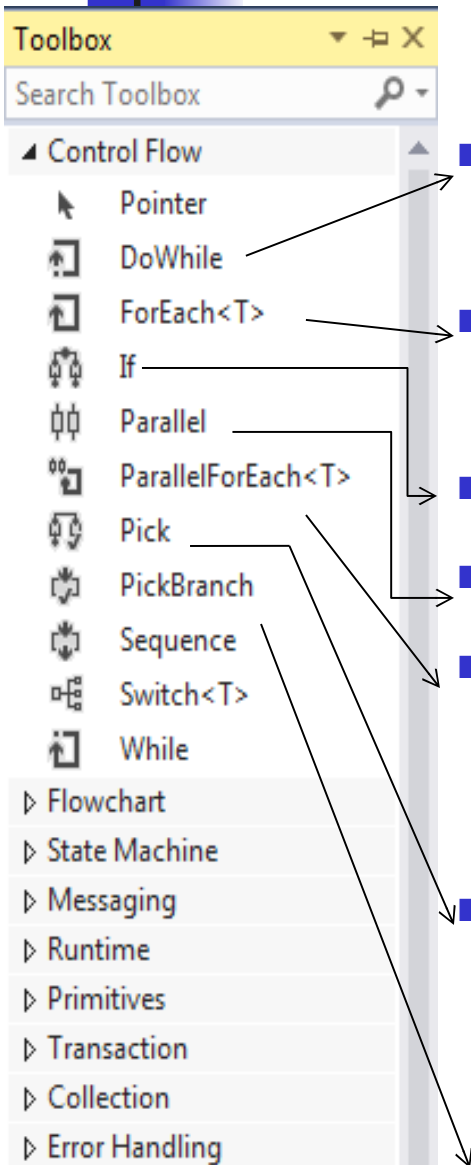
100%

Define parameter types

Configure your service

WF Built-in Activities: Control Flow (1)

<http://msdn.microsoft.com/en-us/library/dd647759.aspx>



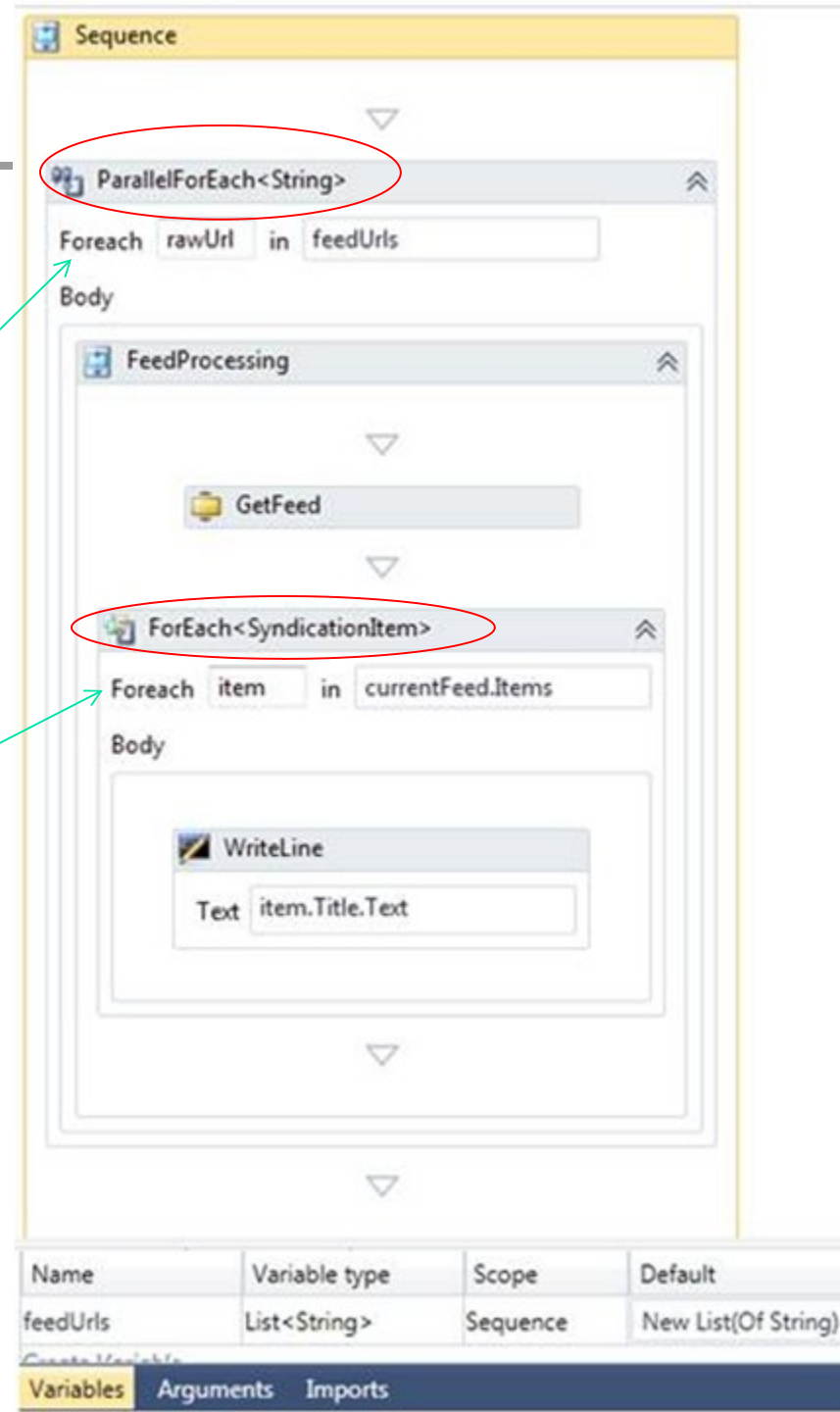
- **DoWhile:** executes an activity, then checks a condition, and repeats until the condition is false.
- **ForEach:** executes an activity for each object in a collection.
- **If:** creates a branch of execution.
- **Parallel:** forks multiple activities.
- **ParallelForEach:** Enumerates the elements of a collection and executes an embedded statement for each element of the collection in parallel.
- **Pick:** allows waiting for a set of PickBranch events, then executing only the activity associated with the first event to occur; **event-driven programming!**
- **PickBranch:** a member of Pick activity

ParallelForEach and ForEach

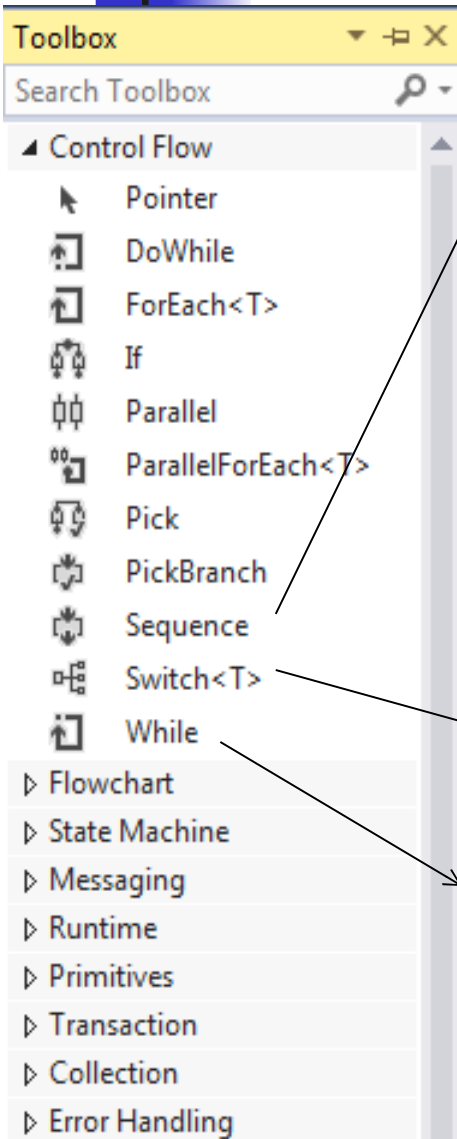
<http://msdn.microsoft.com/en-us/library/ee342461.aspx>

Example: Call many URLs in parallel:

- Take a list of URLs and asynchronously get all RSS feeds using `Foreach<URL>`.
- After the feed is returned, the `Foreach<item>` is used to iterate over the feed items and process them.

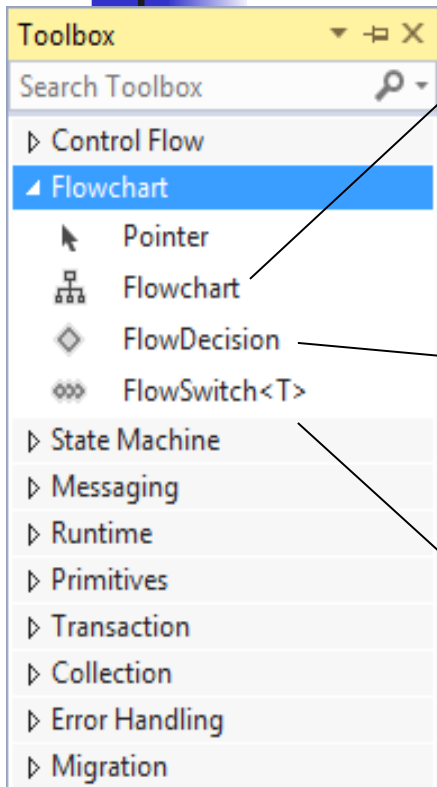


WF Built-in Activities: Control Flow (2)

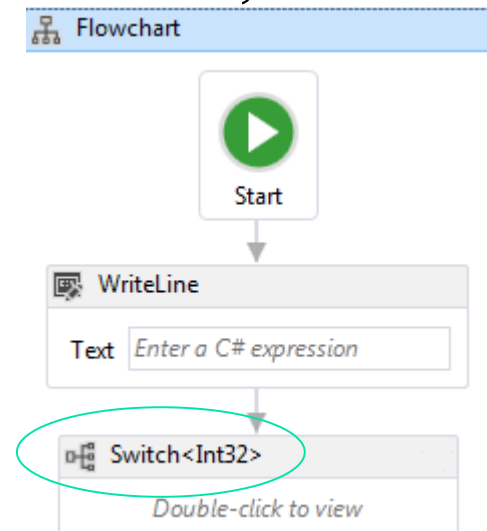


- **Sequence:** groups together a set of activities that are executed sequentially. Sequence is also useful inside workflows. For example, a While activity can contain only one other activity. If that activity is a Sequence, a developer can execute an arbitrary number of activities within the while loop.
- **Switch:** provides a multi-way branch of execution.
- **While:** executes a single activity as long as a condition is true.

WF Built-in Activities: Flowchart (3)



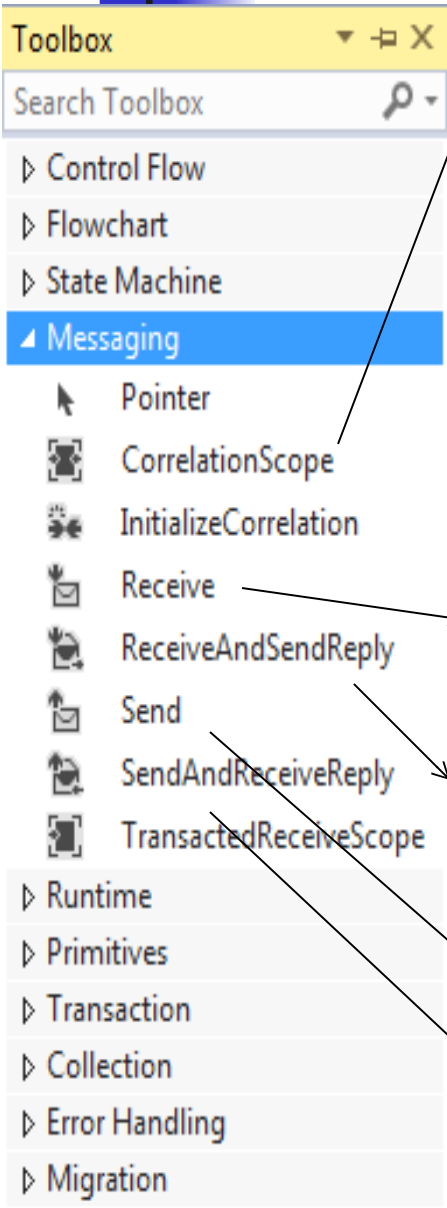
- **Flowchart:** groups together a set of activities that are executed sequentially, but also allows control to return to an earlier step. It creates a local component, similar to **CodeActivity**'s role.
- **FlowDecision:** Branches execution based on a Boolean condition, similar to **If**, but applied at the flowchart level.
- **FlowSwitch:** Branches execution based on an exclusive switch, similar to **Switch** in C#, but applied at the flowchart level.



WF Built-in Activities: Messaging (4)

<http://msdn.microsoft.com/en-us/library/dd487617.aspx>

17

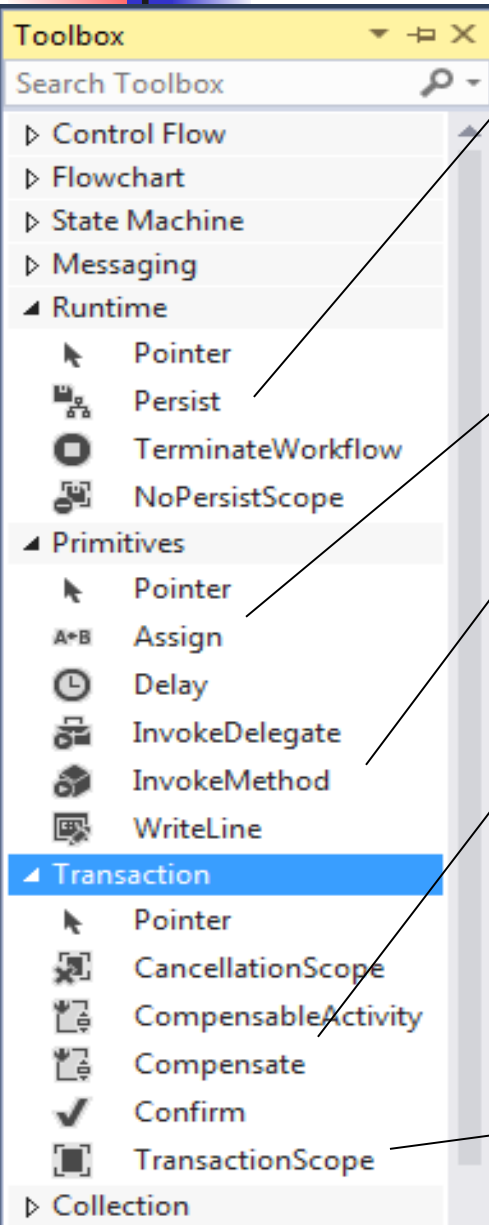


- **CorrelationScope**: Provides implicit CorrelationHandle for child messaging activities. The CorrelationHandle is only visible to child activities, which associates activities together in a correlation by representing a particular shared InstanceKey or transient context in the workflow. Used for asynchronous communication.
- **Receive**: An activity that receives a message via WCF service.
- **ReceiveAndSendReply**: Creates a sequence of two communications.
- **Send**: An activity that sends a message via WCF.
- **SendAndReceiveReply**: Creates an **asynchronous** communication sequence.

Creating Asynchronous Service in WF

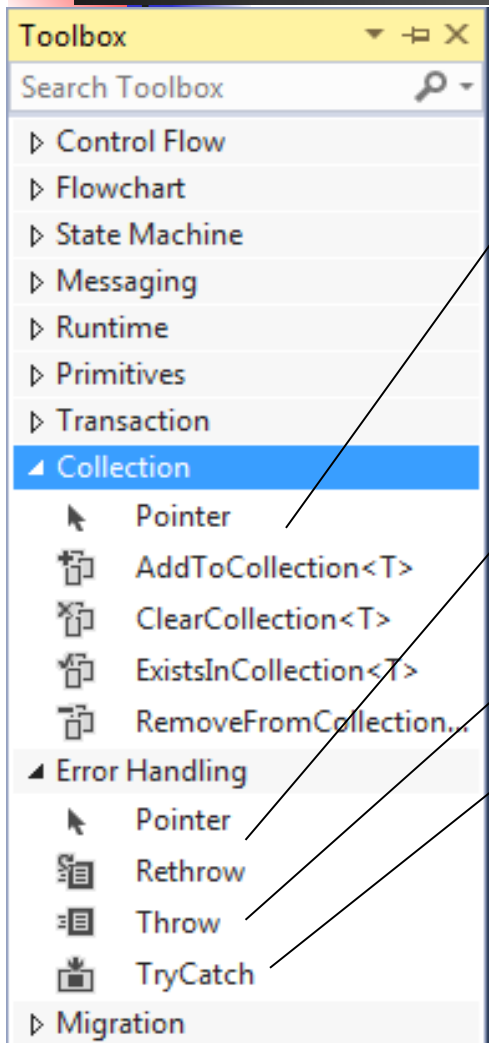
- **SendAndReceiveReply**: is a template used for creating a pair of pre-configured **Send** and **ReceiveReply** activities within a **Sequence** activity
- Send and Receive are **correlated** as part of an asynchronous request/response message exchange pattern on the client. **SendAndReceiveReply** also
 - Configures the **OperationName**, **ServiceContractName** properties of the **Send** activity.
 - Binds the Request property of the **ReceiveReply** activity to the Send activity.
 - Creates a **CorrelationHandle** as a variable in the parent activity.

WF Built-in Activities (5)



- **Persist**: explicitly requests the WF runtime to persist the workflow, to allow a workflow to be re-loaded later on the machine or even on another machine other, if necessary. Save all states.
- **Assign**: assigns a value to a variable in the workflow.
- **InvokeMethod**: Invokes a method on the object, synchronously or asynchronously.
- **Compensate**: An activity used to explicitly invoke the compensation handler of a `CompensableActivity`. It provides a way of handling a problem that occurs in a long-running transaction.
- **TransactionScope**: Makes a code block transactional. This class cannot be inherited.

WF Built-in Activities (6)

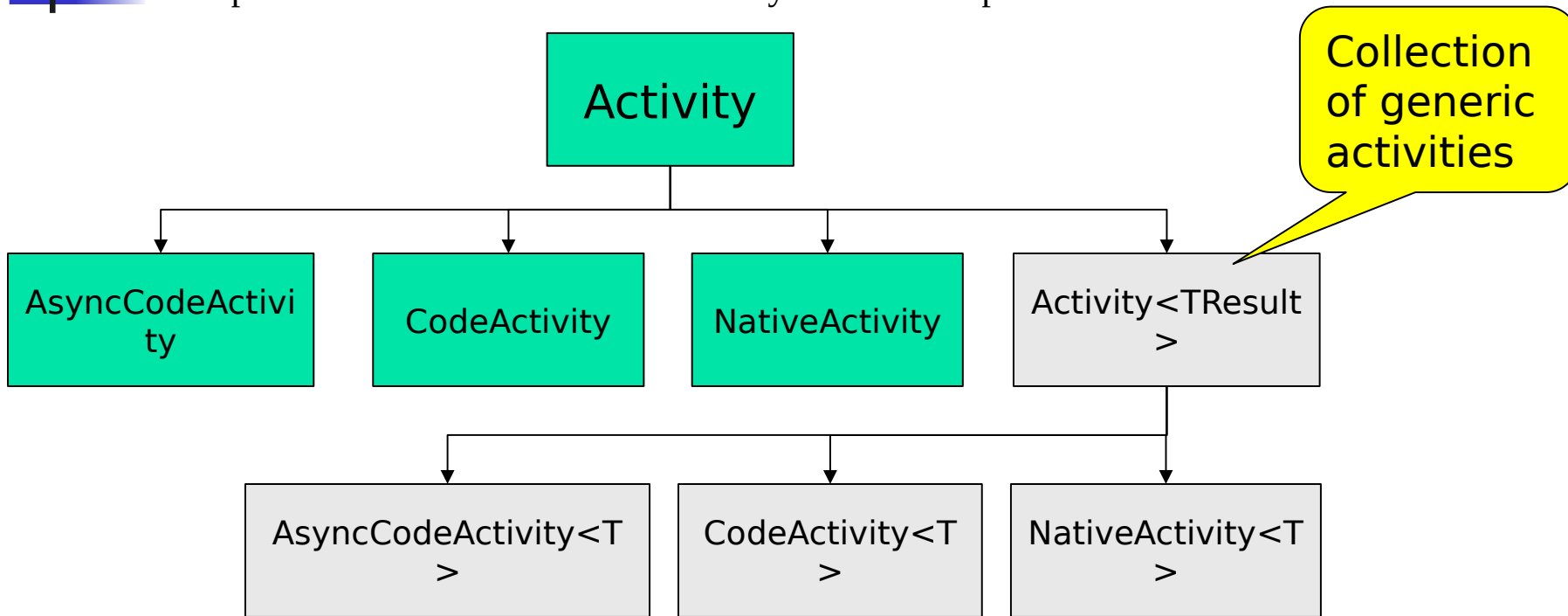


- **AddToCollection, ClearCollection, ExistInCollection, RemoveFromCollection:** allow to add, remove, clear, and check membership of a collection.
- **Rethrow:** throws a previously thrown exception from within a Catch activity.
- **Throw:** raises an exception.
- **TryCatch:** allows creating a try/catch block to handle exceptions. In the block, it contains workflow elements to be executed by the workflow runtime within an exception handling block.

- The developer can wrap any block of C#, VB, and WF code into a custom activity (local component) to integrate into the workflow;
- Custom activities can be simple, performing just one task, or they can be composite activities containing arbitrarily complex logic;
- A business application created using WF might implement application-specific logic as an activity;
- A software vendor using WF might provide a set of custom activities to make its customers' lives easier.

Custom Activity Class Hierarchy

<http://msdn.microsoft.com/en-us/library/ee342461.aspx>



- **Activity** – can compose of other activities.
- **AsyncCodeActivity** – used when your activity perform work asynchronously.
- **CodeActivity** – Creating a local component when you need to write code to do the job.
- **NativeActivity** –when your activity needs access to the runtime internals, for example to schedule other activities or create bookmarks.

A Typical Workflow of Client Service

<http://msdn.microsoft.com/en-us/library/dd851337.aspx>



WF Designer, Markup and Code Behind

Design
view

Code
behind
design

Code

```
class X { ... }
```

```
class Y { ... }
```

```
class Z { ... }
```

Sequence

State

ReceiveMessage

X

Y

SendMessage

ReceiveMessage

While

Z

SendMessage

XAML

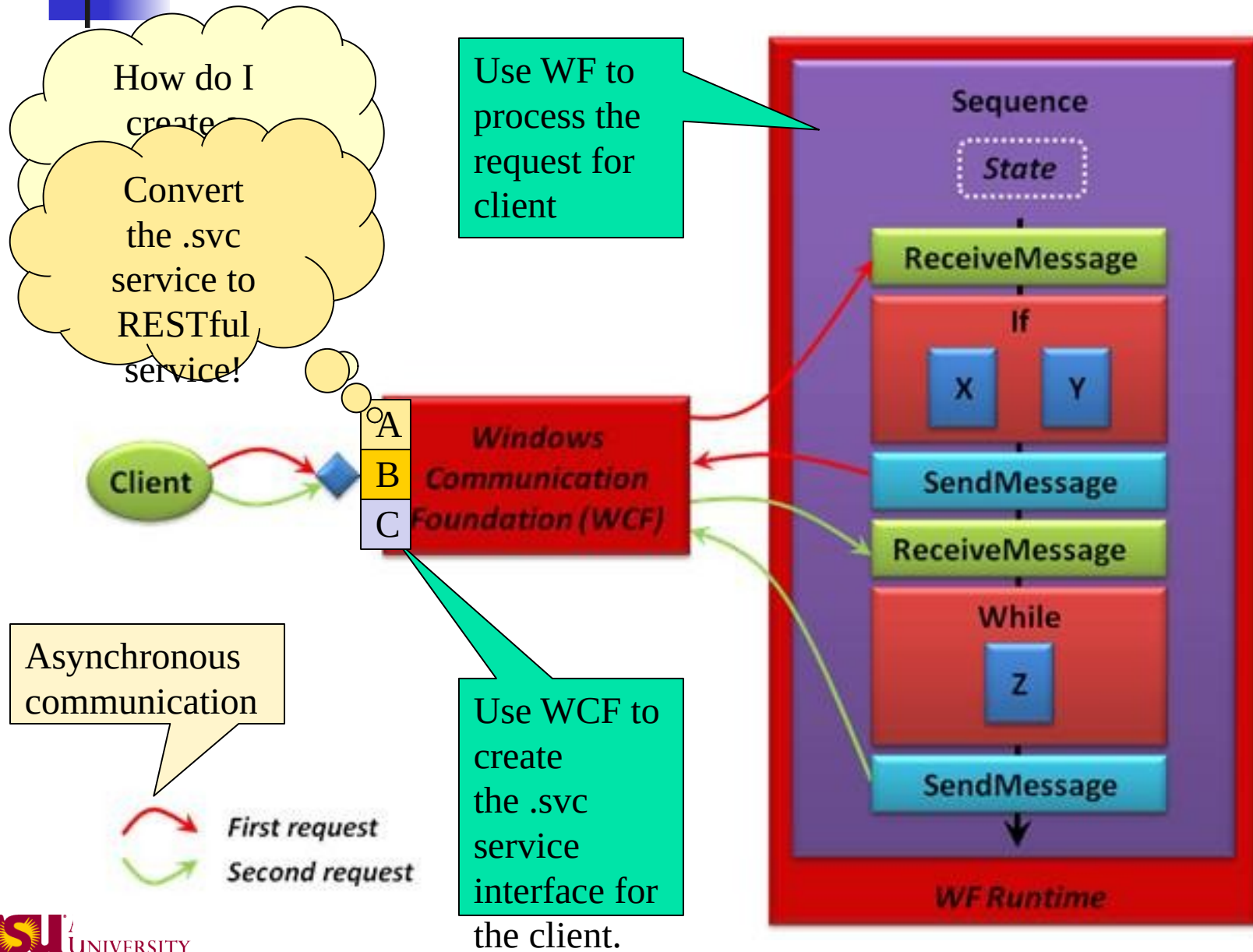
```
<Sequence ... >
  <Sequence.Variables>
    <Variable ... />
    <Variable ... />
  </Sequence.Variables>
  <ReceiveMessage ... >
    ...
  </ReceiveMessage>
  <If ... >
    ...
  </If>
  <SendMessage ... >
    ...
  </SendMessage>
  <ReceiveMessage ... >
    ...
  </ReceiveMessage>
  <While ... >
    ...
  </While>
  <SendMessage ... >
    ...
  </SendMessage>
</Sequence>
```

Markup
view in
XAML

Creating a Workflow Service

- Building business logic as a service often makes sense, for example, when the same set of operations must be accessed from a
 - Console application
 - Windows forms application;
 - Website application.
- Implementing this logic as a service consisting of a set of operations can make these applications to interoperate with the operations;
- WCF interface can be used for creating SOAP and REST interface for WF service.

WF Using WCF for Creating Services

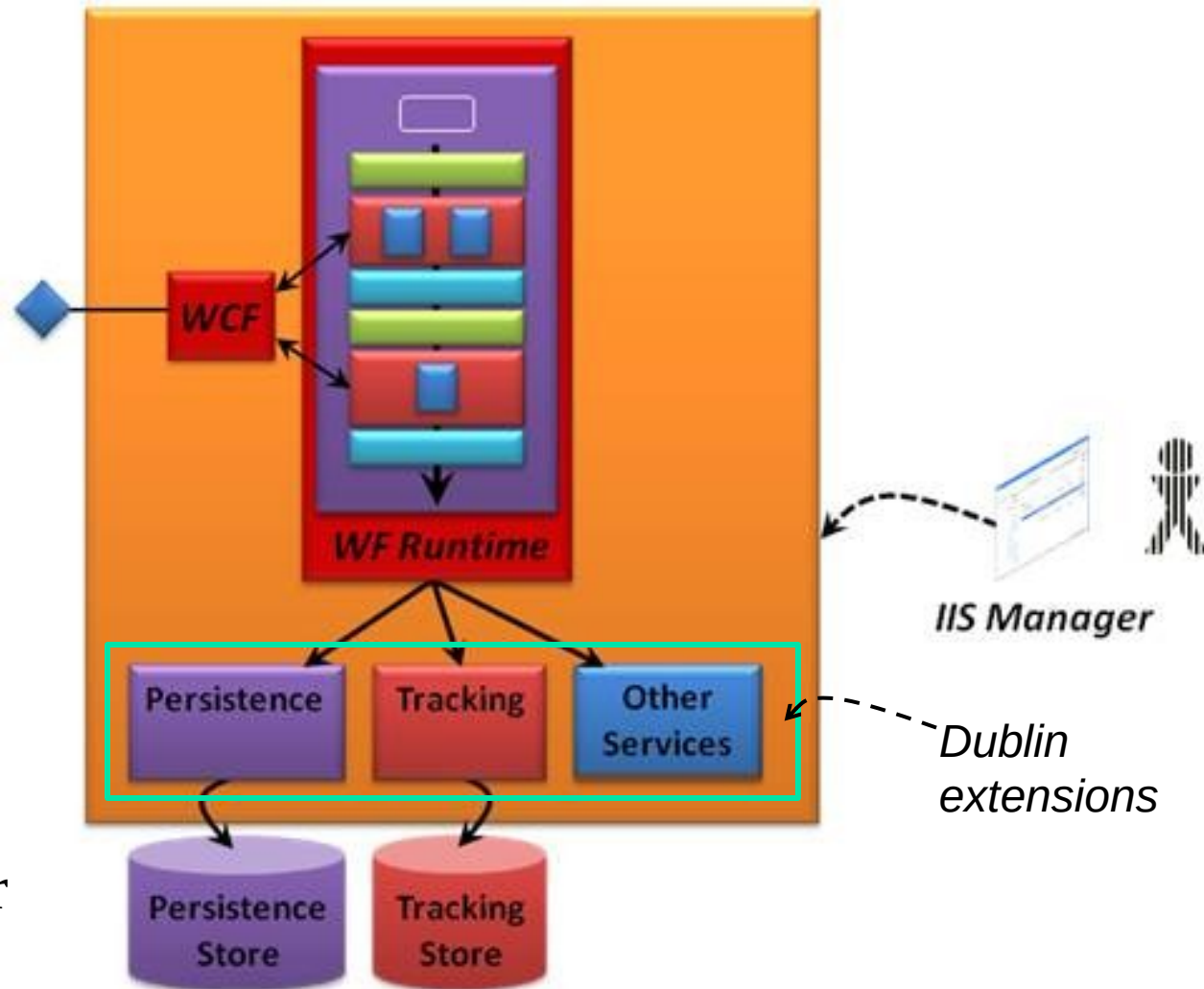


Hosting WF Service

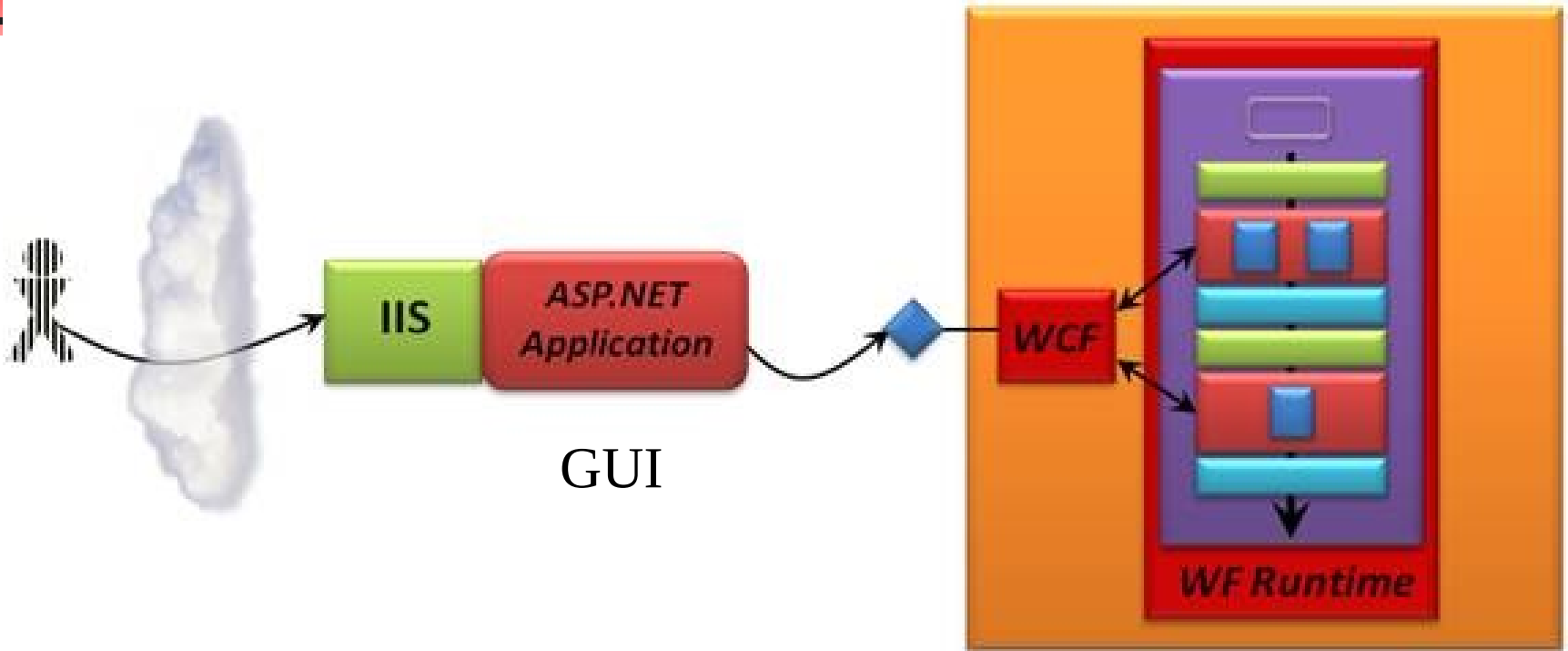
- Similar to WCF, WF services can be hosted in different ways:
 - Self-hosting
 - IIS hosting: running a thread of a worker process
 - “Dublin” hosting
- “Dublin” hosting is based on IIS with extensions for additional support:
 - Persistence management: Saving the process (code and states for later access) and waiting for the second part of service call persistently;
 - Tracking and monitoring
 - Other utility services

Hosting WF Service with Dublin

- IIS provide basic hosting service
- Dublin is the code name for extended IIS and WAS (Website Administration Services) for WF service hosting
- The primary goal of “Dublin” is to make IIS and WAS more attractive as a host for workflow services.



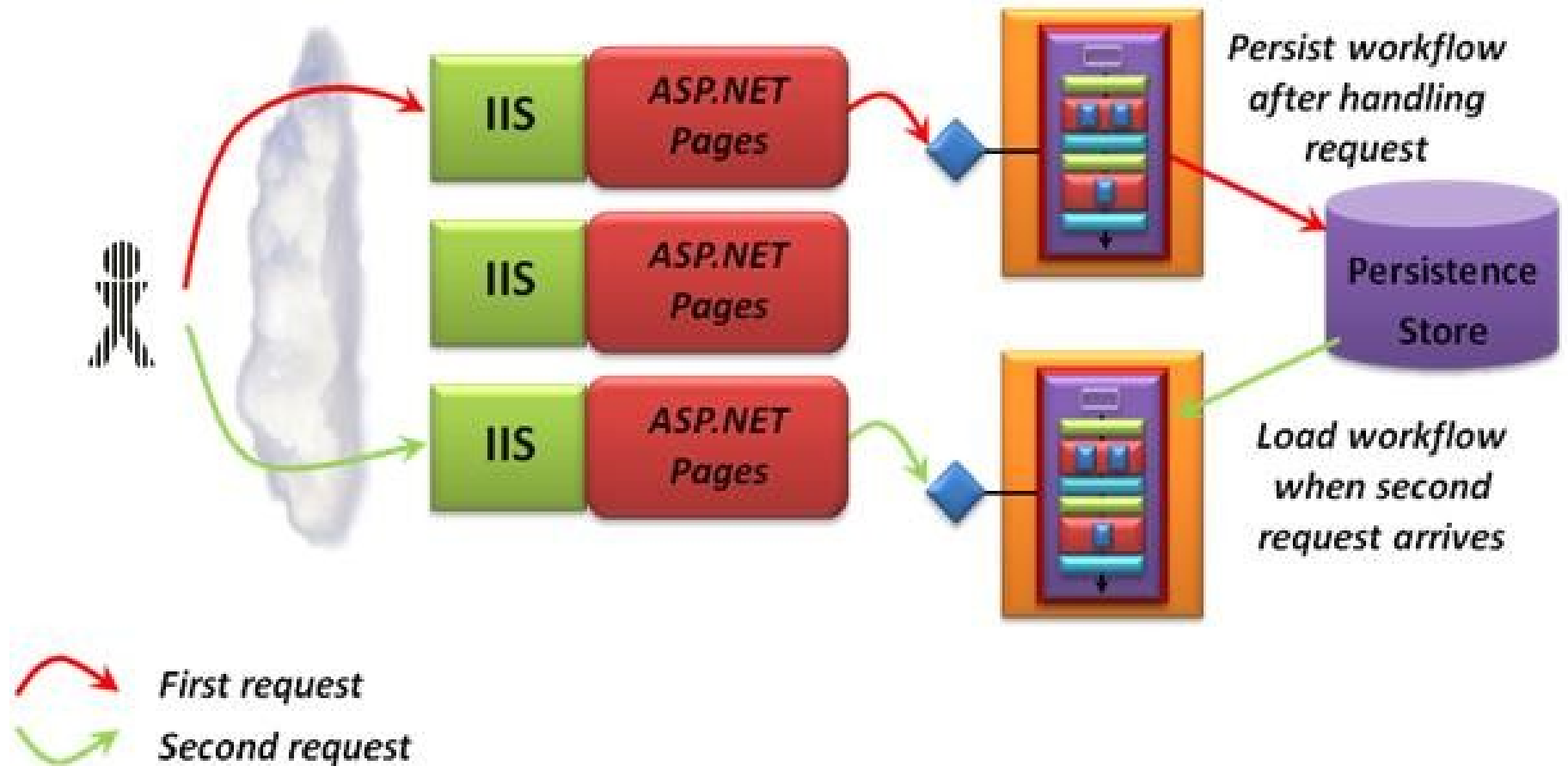
ASP .Net GUI for WF Services



Why do we use WF service behind ASP .Net? We could use code behind ASP .Net page to implement the functionality.


- Tier design: separating presentation layer from application logic layer;
- WF can manage states and **asynchronous communication easier** than ASP .Net
- With parallel construct and parallel threading, WF is **easier for implementing parallel computing**.

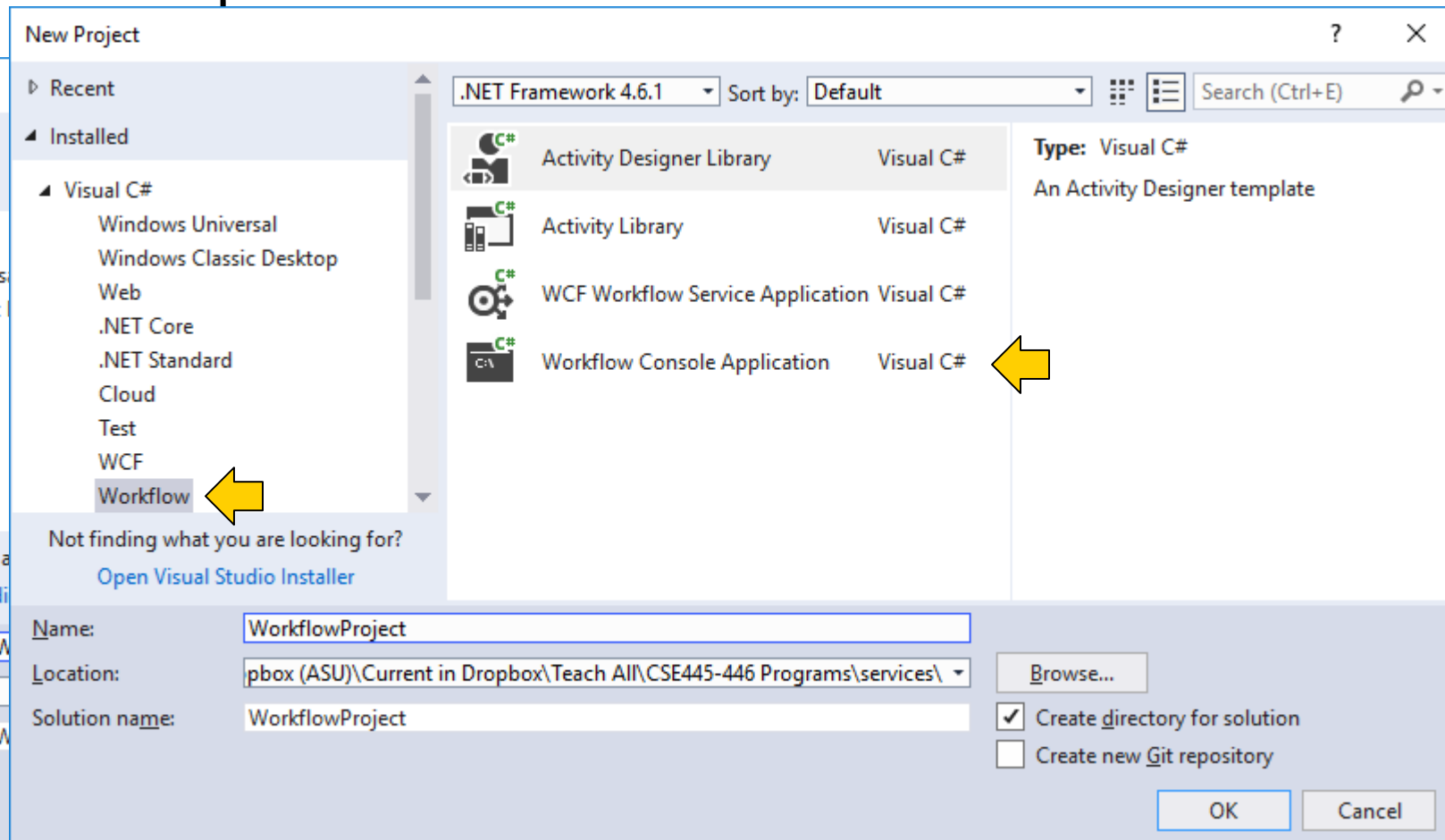
Concurrent Objects for WF Services



Multiple instances can be created for a single client's multiple requests

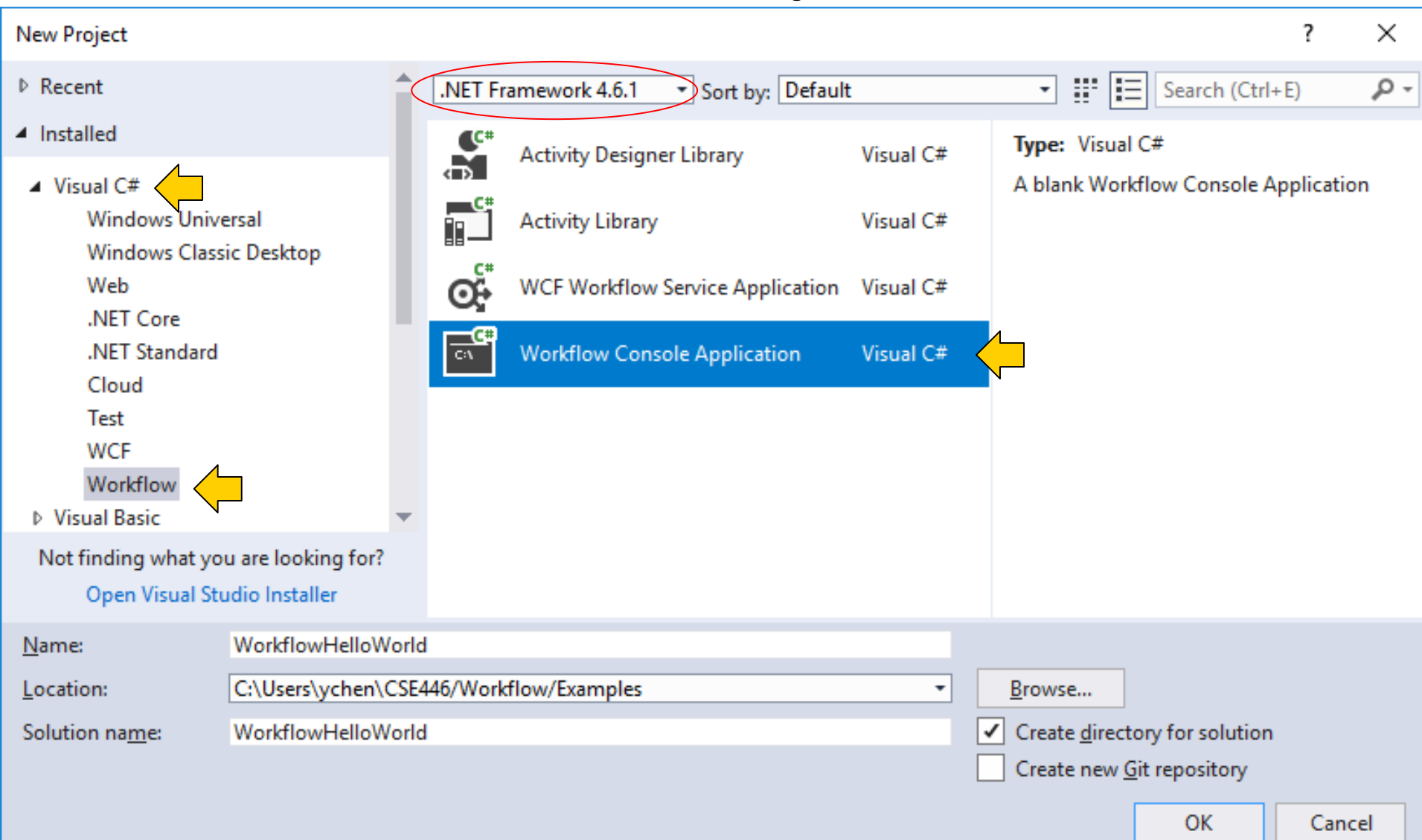
Getting Started With WF Development

- In VS 2017, if you do not see Workflow template, choose Open  Project, and open an existing **Workflow** project.
- Workflow template will be added



Getting Started With WF Development









- In VS, File → New → Project ...



Getting Started With WF Visual Studio 2019


Create a new project

Recent project templates


 Console App	C++
 ASP.NET Web Application (.NET Framework)	C#
 ASP.NET Core Web Application	C#
 Empty Project	C++
 Blank App (Universal Windows - C++/CX)	C++
 Console App (.NET Core)	C#
 WCF Service Application	C#
 Windows Forms App (.NET Framework)	C#

workflow


C# Windows

 Workflow Console Application
A blank Workflow Console Application


C# Windows Console

 WCF Workflow Service Application
A project for creating WCF Workflow Service Application th


Visual Basic Windows Office

 WCF Workflow Service Application
A project for creating WCF Workflow Service Application th


C# Windows Office

 Workflow Console Application
A blank Workflow Console Application

Visual Basic Windows Console

 Activity Library
A blank Workflow Activity Library

C# Windows Office

 Activity Library
A blank Workflow Activity Library

Choose Project Name and Location

Configure your new project

Workflow Console Application

C#

Windows


Console

Project name

WorkflowHelloWorld

Location

C:\Users\ychen10\Dropbox (ASU)\Current in Dropbox\Teach All\CSE445-446 Programs\2020\

Solution name 

WorkflowHelloWorld

☒ Place solution and project in the same directory

Framework

.NET Framework 4.7.2

Adding a Flowchart Item into Workflow

1. Add Flowchart into the Workflow
2. Right click project name “WorkflowHelloWorld” and Choose “Add New Item”

The screenshot displays the Microsoft Visual Studio interface for a project named "WorkflowHelloWorld". The main window shows a workflow diagram with a "Start" button. The "Toolbox" on the left lists various controls, with the "Flowchart" item highlighted. A red arrow points from the "Flowchart" item in the toolbox to the "Flowchart" item in the workflow diagram. The "Solution Explorer" on the right shows the project structure, with the "WorkflowHelloWorld" project selected. A red arrow points to the project name. The "Properties" window at the bottom right shows the properties of the selected "Flowchart" item.

WorkflowHelloWorld - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP

Yinong Chen VC

Quick Launch (Ctrl+Q)

Workflow1.xaml

Workflow1

Expand All Collapse All

Start

Toolbox

Search Toolbox

Control Flow

Flowchart

Pointer

Flowchart

FlowDecision

FlowSwitch<T>

State Machine

Messaging

Runtime

Primitives

Transaction

Collection

Error Handling

Migration

General

There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'WorkflowHelloWorld' (1)

C# WorkflowHelloWorld

Properties

References

App.config

Program.cs

Workflow1.xaml

Properties

System.Activities.Statements.Flowch...

Search: Clear

Misc

DisplayName Flowchart

ValidateUnconne... ☐

Error List

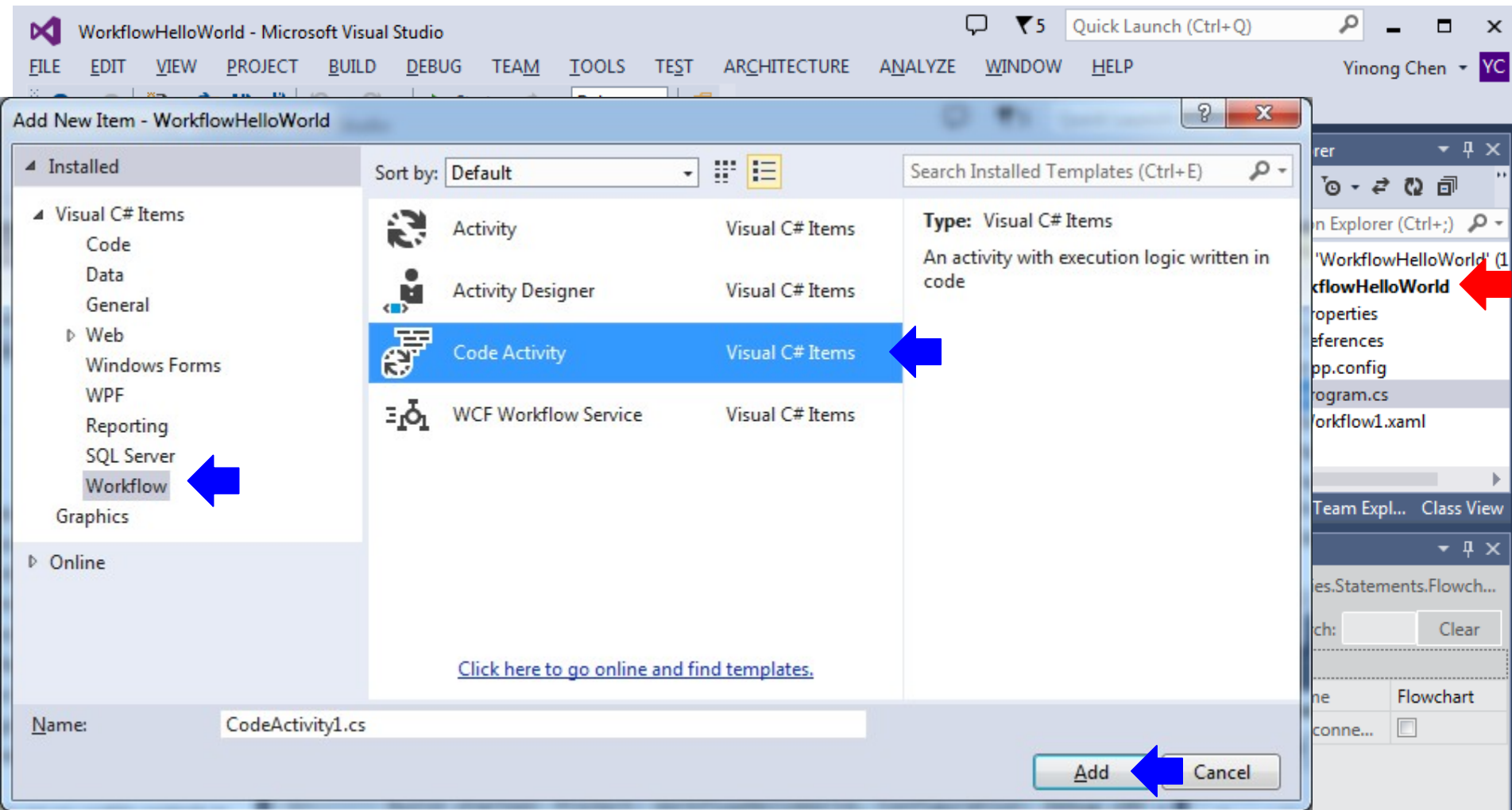
0 Errors 0 Warnings 0 Messages

Search Error List

Description	File	Line	Column	Project
-------------	------	------	--------	---------

Adding a Flowchart Item into Workflow

1. Right click project name “WorkflowHelloWorld” and Choose “Add New Item”
2. Choose CodeActivity and Add



Creating a Workflow

1. CodeActivity1 is added into the project
2. Build the project, the WorkflowHelloWorld project is added into the Toolbox
3. Drag CodeActivity1 from Toolbox into the workflow

The screenshot displays the Visual Studio IDE with the following components:

- Toolbox:** Located on the left, it shows a tree view of controls. The 'WorkflowHelloWorld' project is expanded, revealing 'CodeActivity1' under the 'General' group. A green callout bubble with the text 'Project added into Toolbox' points to this item.
- Workflow Designer:** The central area shows a workflow named 'Workflow1' in 'Flowchart' mode. It contains a 'Start' node (a green circle with a play button) connected by an arrow to a 'CodeActivity1' node (a rectangle with a computer icon). A green callout bubble with the text 'Drag and drop the item' points to the 'CodeActivity1' node in the workflow.
- Solution Explorer:** Located on the right, it shows the project structure for 'WorkflowHelloWorld'. The files listed are 'Properties', 'References', 'App.config', 'CodeActivity1.cs', 'Program.cs', and 'Workflow1.xaml'. A blue arrow points to 'CodeActivity1.cs'.
- Properties Window:** At the bottom right, it shows the properties for the selected 'Flowchart' element, including 'DisplayName' and 'ValidateUnconne...'.

At the bottom of the Toolbox, a message states: 'There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.'

Code Behind the CodeActivity1

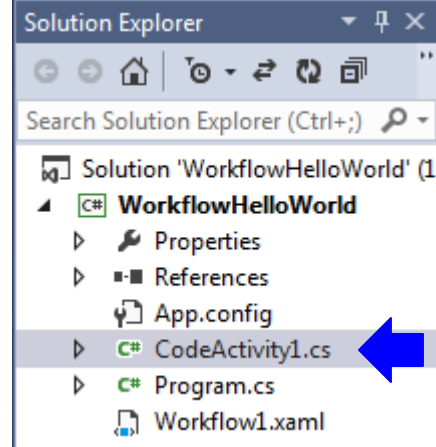
38

```
CodeActivity1.cs  Workflow1.xaml
WorkflowHelloWorld.CodeActivity1  Execute(CodeActivityContext context)

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Activities;

namespace WorkflowHelloWorld
{
    0 references
    public sealed class CodeActivity1 : CodeActivity
    {
        1 reference
        // Define an activity input argument of type string
        public InArgument<string> Text { get; set; }

        // If your activity returns a value, derive from CodeActivity<TResult>
        // and return the value from the Execute method.
        0 references
        protected override void Execute(CodeActivityContext context)
        {
            // Obtain the runtime value of the Text input argument
            string text = context.GetValue(this.Text);
            Console.WriteLine("Hello World"); // Add this line of code
        }
    }
}
```



Code Behind the Main Program

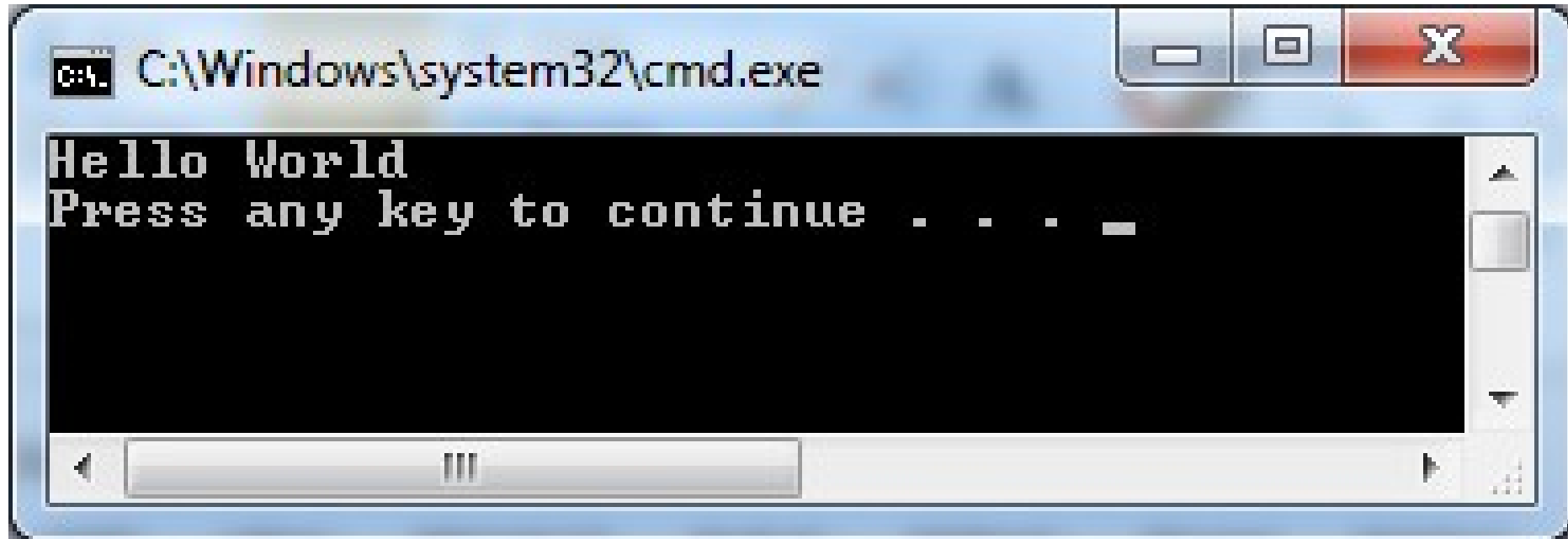
The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Displays `Program.cs` with the following code:

```
using System;
using System.Linq;
using System.Activities;
using System.Activities.Statements;

namespace WorkflowHelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Activity workflow1 = new Workflow1();
            WorkflowInvoker.Invoke(workflow1);
        }
    }
}
```
- Solution Explorer:** Shows the project structure for 'WorkflowHelloWorld'. The files listed are `Properties`, `References`, `App.config`, `CodeActivity1.cs`, `Program.cs` (highlighted with a blue arrow), and `Workflow1.xaml`.
- Properties Window:** Located at the bottom right, showing tabs for `Solution E...`, `Team Expl...`, and `Class View`.

Executing the Code



```
C:\Windows\system32\cmd.exe  
Hello World  
Press any key to continue . . . _
```


Workflow: Support Control Flow and Data Flow

- Control flow model: Assuming one processor will process the entire program, and thus, the code are organized in a **sequence**;
 - It is possible to use multithreading programming to perform parallel computing, but not easy.
- Data flow model: Assuming multiple processors will process different part of the program. A piece of code will be executed when the input **data** is ready.
- Workflow supports multiple computing models
 - Sequence construct
 - Parallel construct

Adding Constructs for Parallel Computing

The screenshot shows the Visual Studio IDE with a workflow diagram. The **Control Flow** toolbox on the left lists various constructs. The workflow diagram in the center shows a **Sequence** container containing a **Parallel** container, which contains two **CodeActivity1** and **CodeActivity2** elements. The **Solution Explorer** on the right shows the project structure for 'WorkflowHelloWorld'.

Control Flow

- Pointer
- DoWhile
- ForEach<T>
- If
- Parallel
- ParallelForEach<T>
- Pick
- PickBranch
- Sequence
- Switch<T>
- While

Workflow1 > Flowchart > Sequence

Sequence

- Parallel**

 - CodeActivity1**
 - CodeActivity2**

Solution Explorer

- Solution 'WorkflowHelloWorld' (1)
- C# WorkflowHelloWorld**
- Properties
- References
- App.config
- C# CodeActivity1.cs
- C# CodeActivity2.cs
- C# Program.cs
- Workflow1.xaml

```
Console.WriteLine("Printing from CodeActivity1"); // Thread 1
Console.WriteLine("Hello World"); // Add this line of code
```

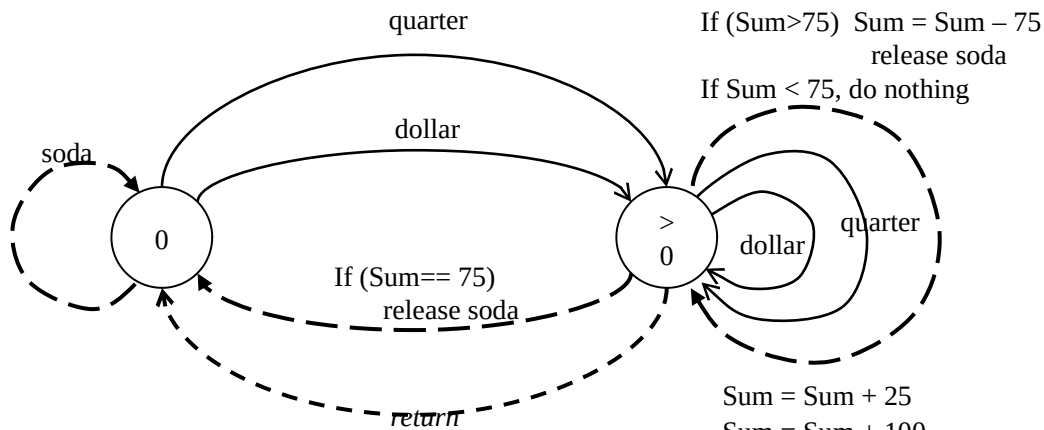
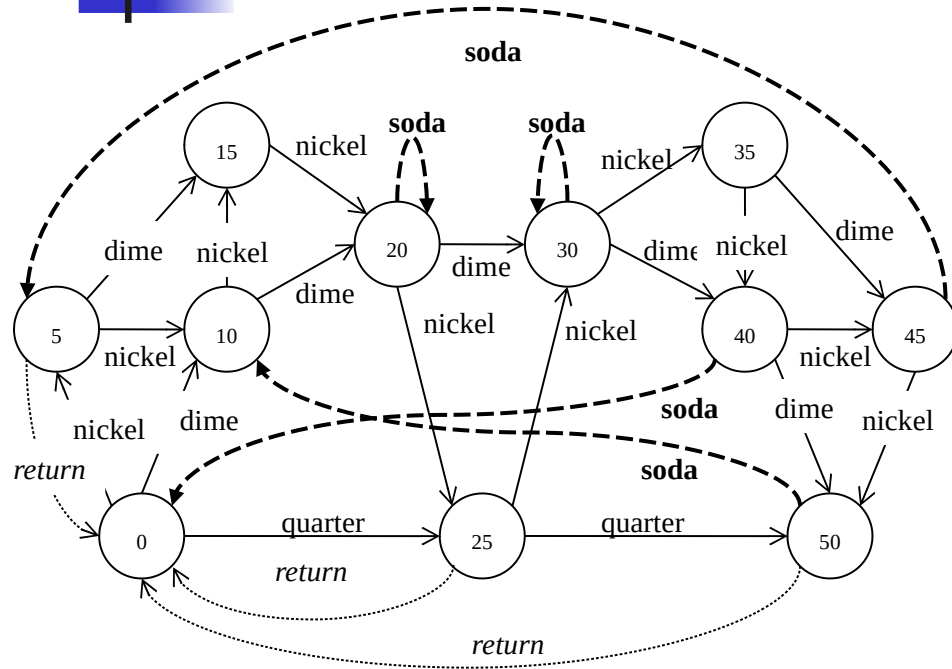
```
Console.WriteLine("Printing from CodeActivity2"); // Thread 2
Double piValue = System.Math.PI;
Console.WriteLine(piValue);
```

C:\Windows\system32\cmd.exe

```
Printing from CodeActivity1
Hello World
Printing from CodeActivity2
3.14159265358979
Press any key to continue . . .
```

Finite State Machine Model

<http://neptune.fulton.ad.asu.edu/WSRepository/CoffeeMachine/>



neptune.fulton.ad.asu.edu/WSRepository/CoffeeMachine/

Welcome to Coffee Vender. Each cup of Coffee cost 75 cents.
Print Your Name on the Cup:

John Doe

Insert a Quarter

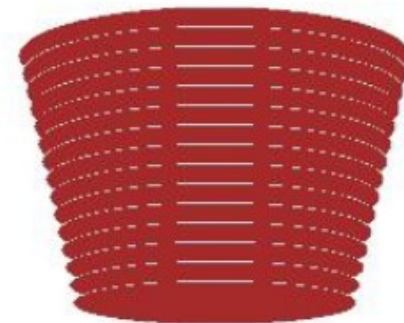
Insert a Dollar

The amount you have deposited: 75

Buy a Coffee

Return Deposit

Please Take Your Coffee



Implementing a State Machine

Workflow1.xaml

Workflow1

Expand All Collapse All

StateMachine

Start

0

add quarter

add dollar

return coffee

return changes

> 0

add quarter

add collar

return coffee

0

> 0

quarter

dollar

soda

return

If (Sum==75) release soda

If (Sum>75) Sum = Sum - 75
release soda

If Sum < 75, do nothing

dollar

quarter

Sum = Sum + 25

Sum = Sum + 100

There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Variables Arguments Imports

100%

Y. Chen

```

graph TD
    Start([Start]) --> S0[0]
    S0 -- "add quarter" --> S0
    S0 -- "add dollar" --> S0
    S0 -- "return coffee" --> S0
    S0 -- "return changes" --> S0
    S0 --> SGT0["> 0"]
    SGT0 -- "add quarter" --> SGT0
    SGT0 -- "add collar" --> SGT0
    SGT0 -- "return coffee" --> SGT0
    SGT0 --> S0
  
```

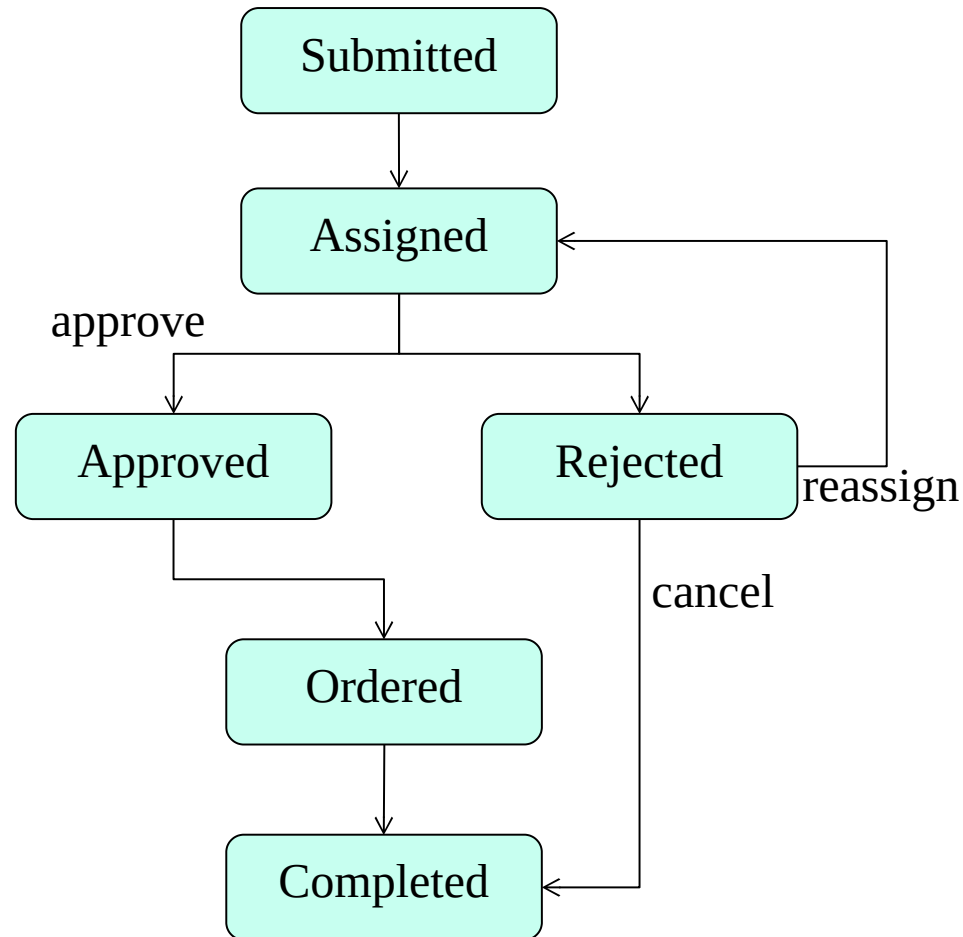
State transition diagram details:

- State 0:
 - Input: quarter → State > 0
 - Input: dollar → State > 0
 - Input: soda → State 0 (self-loop)
 - Input: return → State > 0
- State > 0:
 - Input: quarter → State > 0 (self-loop)
 - Input: dollar → State > 0 (self-loop)
 - Input: quarter → State > 0 (self-loop)
 - Input: Sum = Sum + 25 → State > 0 (self-loop)
 - Input: Sum = Sum + 100 → State > 0 (self-loop)
 - Input: If (Sum == 75) release soda → State > 0 (self-loop)
 - Input: If (Sum > 75) Sum = Sum - 75 release soda → State > 0 (self-loop)
 - Input: If Sum < 75, do nothing → State > 0 (self-loop)

Event-Driven and State Machine for eBusiness Application

<http://msdn.microsoft.com/en-us/magazine/cc163281.aspx>

An Ordering Process



State Machine of Ordering Process in Table View

State	Allowed Transitions
Submitted	Assigned
Assigned	Approved or Rejected
Approved	Ordered
Rejected	Assigned or Completed
Ordered	Completed
Completed	(None)

Using the State Machine Template

New Project

Project types:

- Visual C#
 - Windows
 - Web
 - Smart Device
 - Office
 - Database
 - Microsoft Robotics
 - Reporting
 - Test
 - WCF
 - Workflow**
- Other Languages
- Other Project Types
- Test Projects

Templates:

.NET Framework 3.5

Visual Studio installed templates

- Empty Workflow Project
- Sequential Workflow Library
- SharePoint 2007 State Machine Workflow
- State Machine Workflow Library**
- Sequential Workflow Console Applicat..
- SharePoint 2007 Sequential Workflow
- State Machine Workflow Console Appl...
- Workflow Activity Library

My Templates

- Search Online Templates...

A project for creating a state machine workflow library. (.NET Framework 3.5)

Name: WorkflowStateMachine

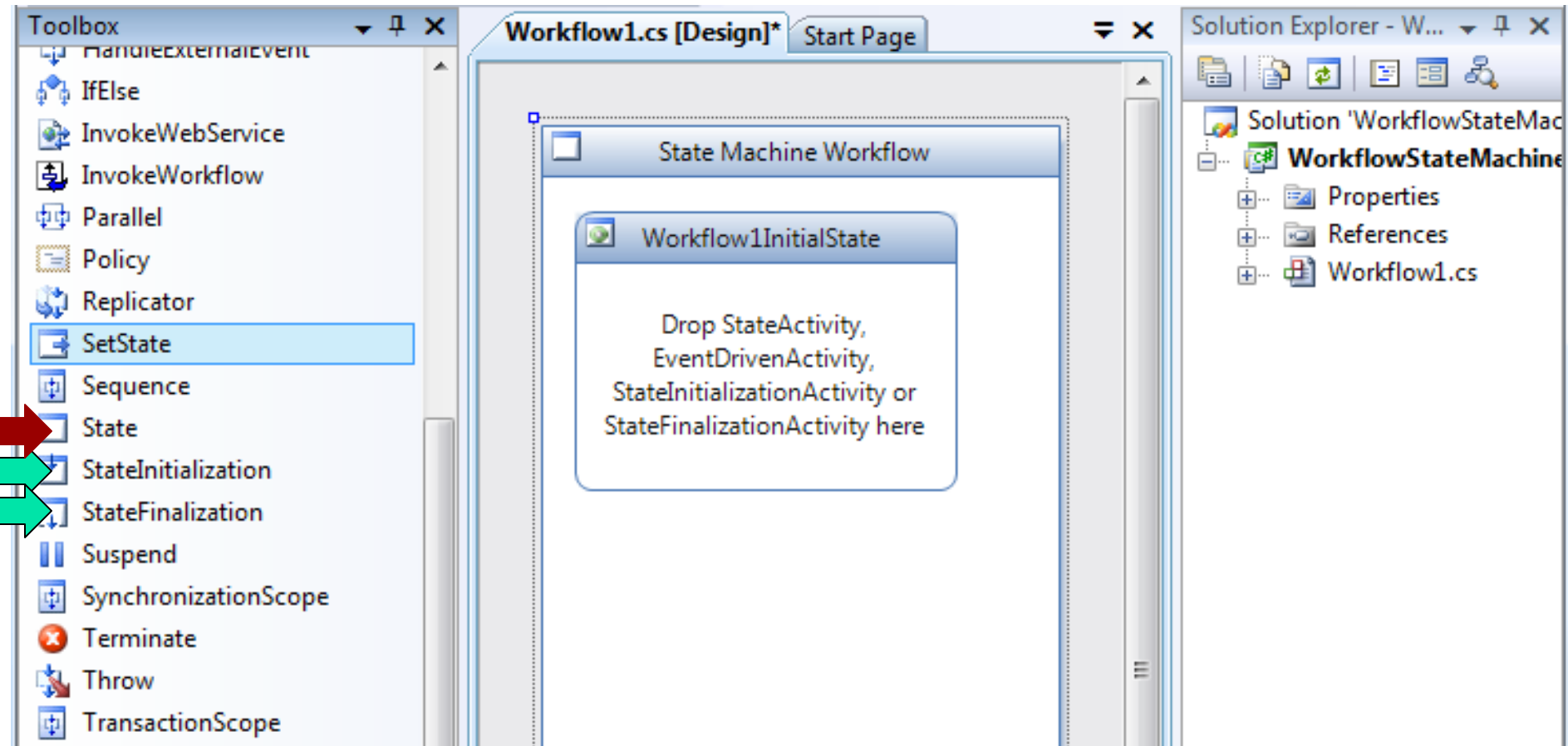
Location: C:\Users\yinong\Documents\Visual Studio 2008\Projects [Browse...](#)

Solution: Create new Solution ☒ Create directory for solution

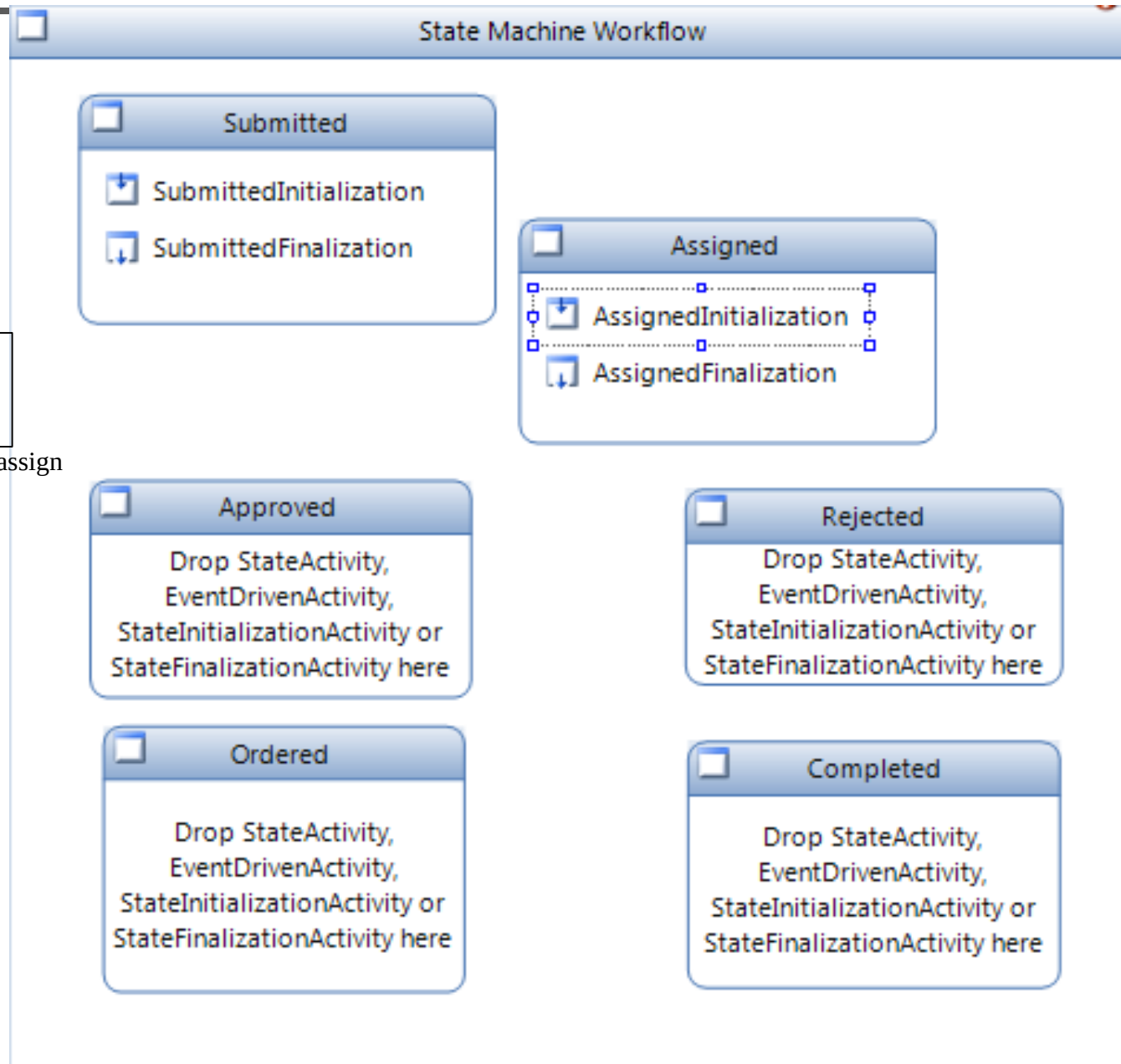
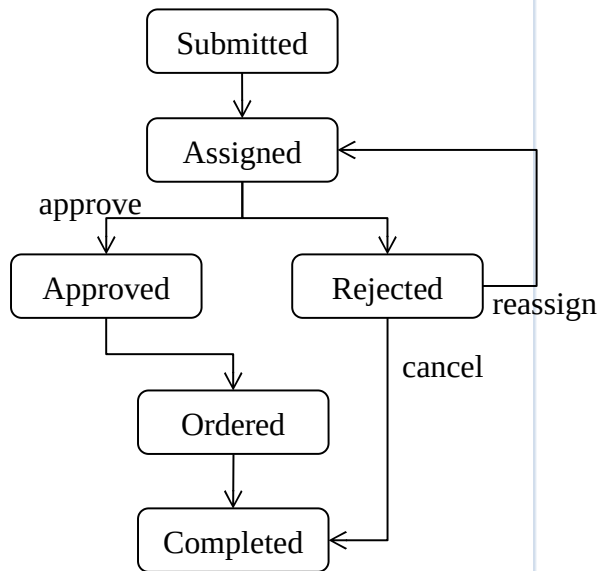
Solution Name: WorkflowStateMachine

OK Cancel

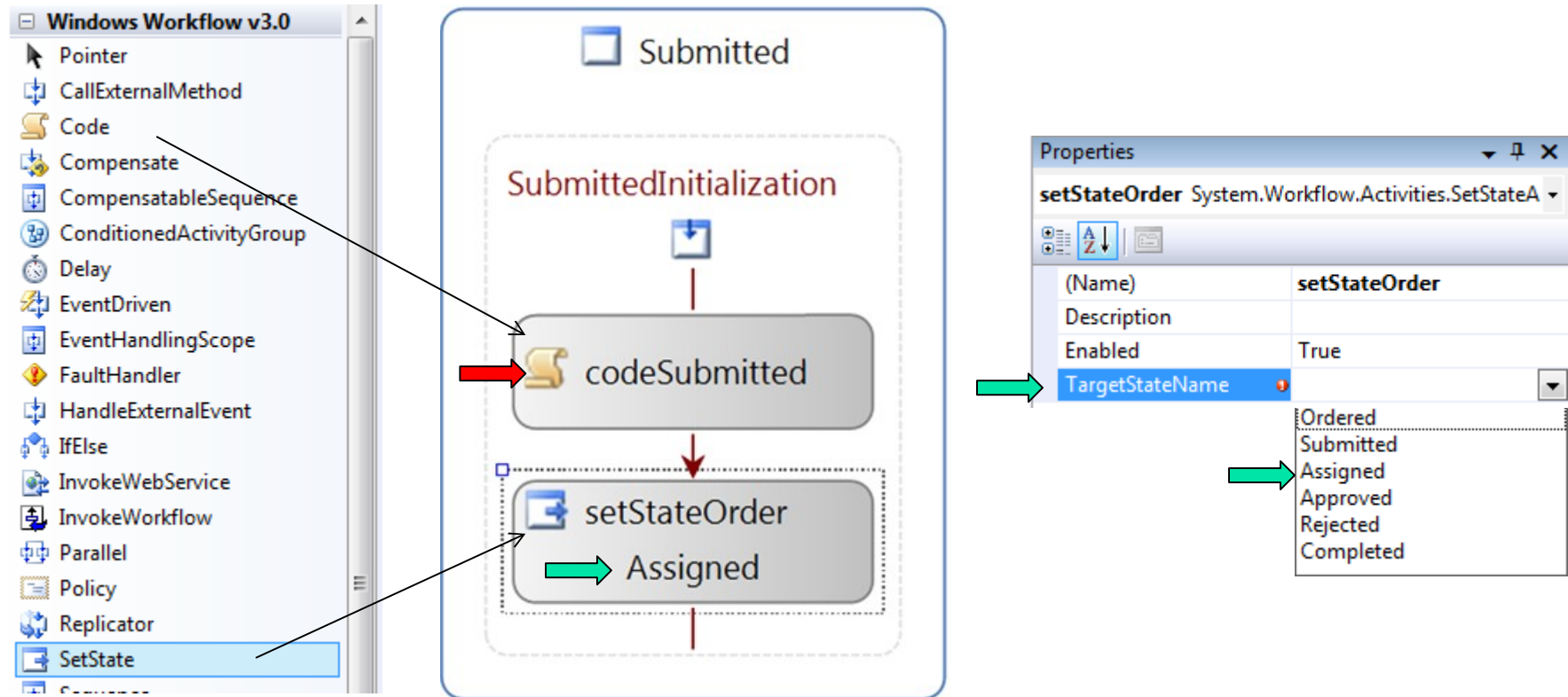
State Machine Template and Services



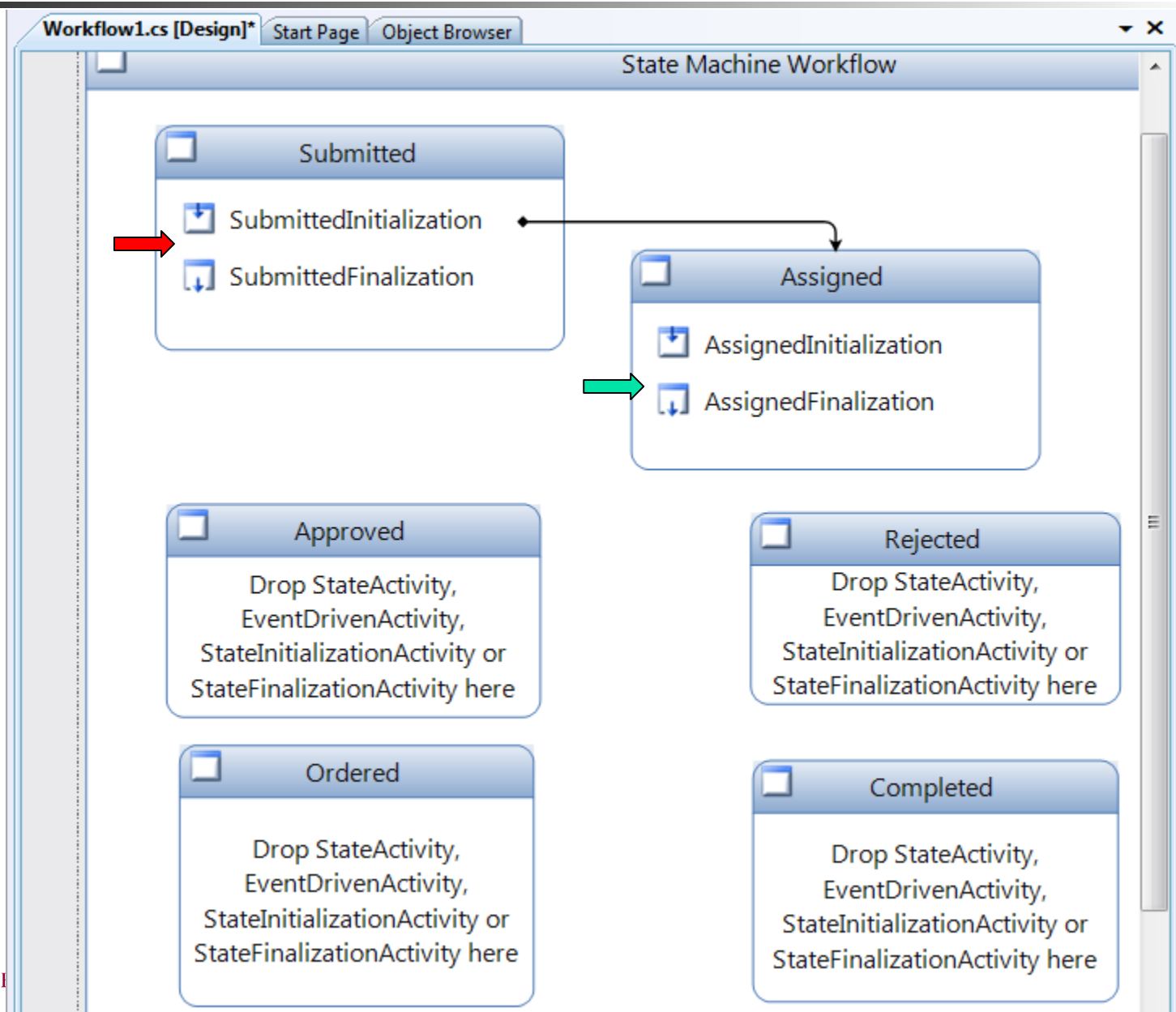
Define States



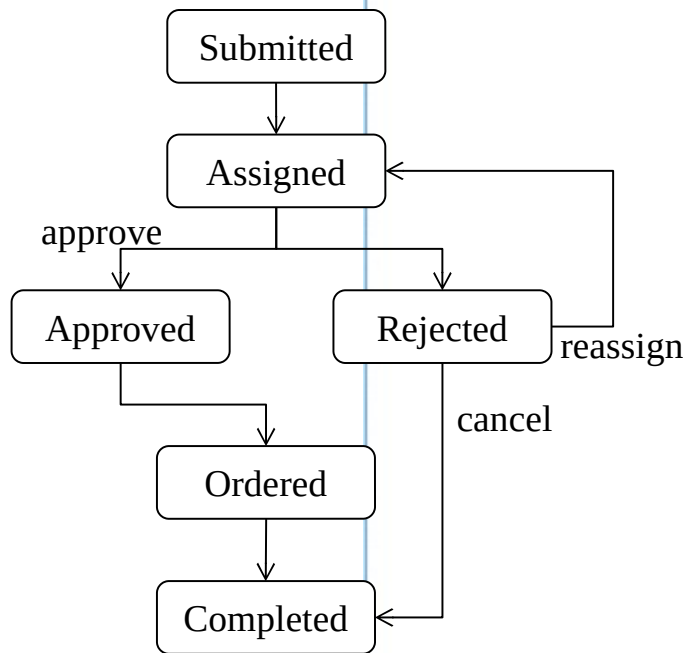
Connecting States Through Target State



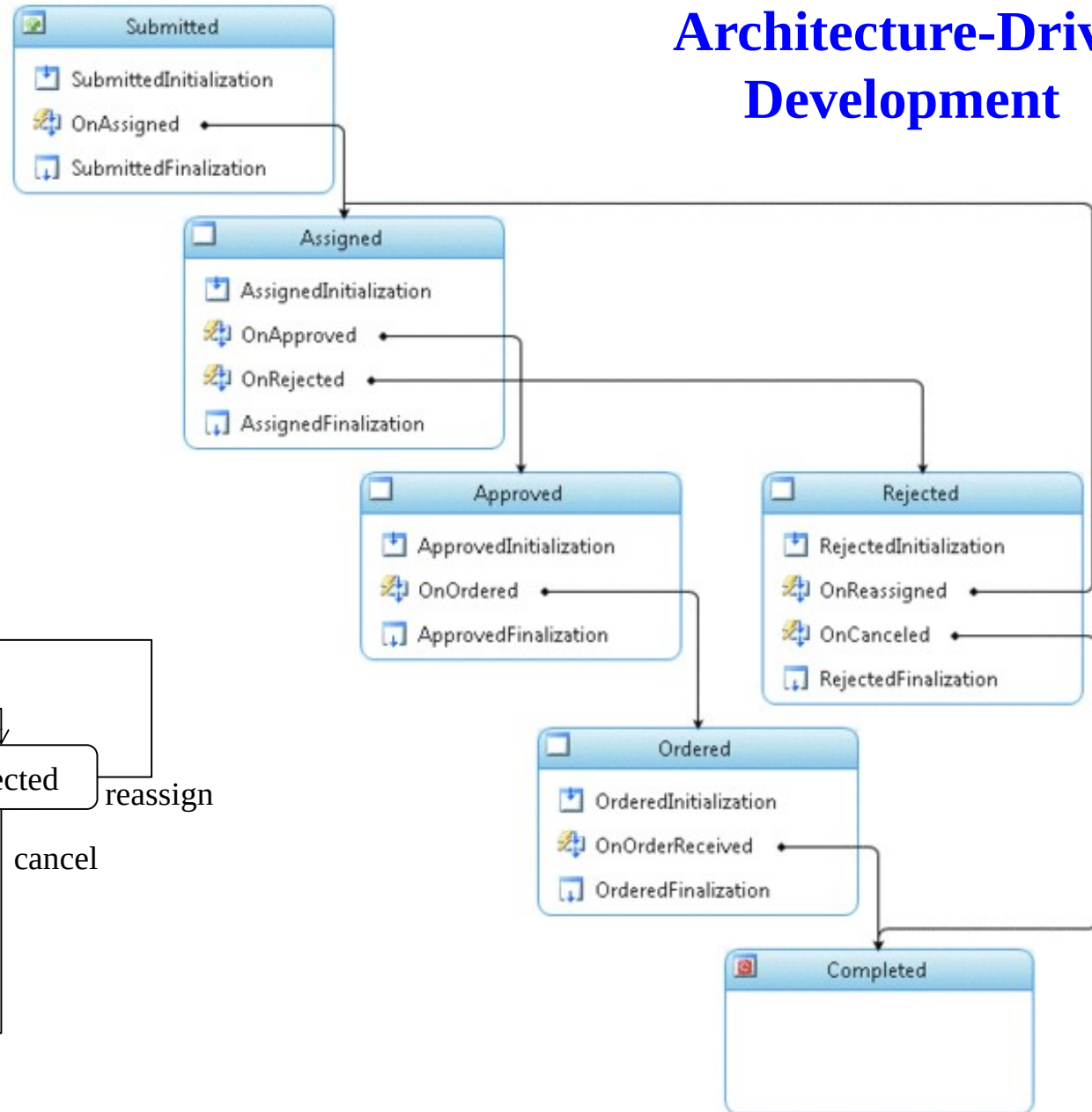
States are Connected



All Connected



Architecture-Drive Development



Steps Implementing the Application Logic

1. Determine the events required for data exchange and state transitions.
2. Define an External Data Exchange interface.
3. Add EventDriven activities to the state machine workflow.
4. Reference the external data exchange interface.
5. Implement the External Data Exchange interface.
6. Add the External Data Exchange to the workflow runtime.
7. Use the External Data Exchange to exchange data and transition states.

Code Behind: Event Interfaces

```
namespace ExternalDataExchange
{
    [ExternalDataExchange()] public interface IEventService
    {
        event EventHandler<SupplyFulfillmentArgs> Assigned;
        event EventHandler<SupplyFulfillmentArgs> Approved;
        event EventHandler<SupplyFulfillmentArgs> Rejected;
        ➡ event EventHandler<SupplyFulfillmentArgs> Reassigned;
        event EventHandler<SupplyFulfillmentArgs> Canceled;
        event EventHandler<SupplyFulfillmentArgs> Ordered;
        event EventHandler<SupplyFulfillmentArgs> OrderReceived;
    }
}
```

Implementing the Event Interface

```
→ public void RaiseAssignedEvent(System.Guid instanceId,  
    string assignedTo)  
{ // Check to see if event is defined  
    if (this.Assigned != null)  
    { // Create the EventArgs for this event  
        LocalService.SupplyFulfillmentArgs args = new  
            LocalService.SupplyFulfillmentArgs(instanceId);  
        args.AssignedTo = assignedTo;  
        // Raise the event  
        this.Assigned(this, args);  
    }  
}
```

Summary the Lecture

- Key Ideas of Workflow-based Application Development
- WF Constructs and Activities
- Creating WF workflow Application
- Integrating WF and WCF
- Creating WF Services
- Event-Driven Approach and State Machine in WF
- Workflow Supports
 - Architecture-Drive Development
 - Control flow and Dataflow
 - Synchronous and Asynchronous communications