

## Unit 2

### Software Development by Composition and Integration

---

## Unit 2-3

# Workflow Composition and Case Studies

Dr. Yinong Chen

- **Creating a Workflow Console Application**
  - Adding a Flowchart Component into a Workflow
- Adding Other Components into a Workflow
  - Adding a CodeActivity into a Workflow
  - Adding an External **Web Service** into a Workflow
- Creating Workflow Services
  - Contract-First Approach, resulting a service.svc file
  - Workflow-First Approach, resulting a service.xamlx file
- Case Studies:
  - Image Verifier in Workflow
  - Mortgage Application Integration

# Creating a Workflow Console Application

In Visual Studio, Start a New Project

## New Project

Recent

Installed

Visual C#

Windows Universal

Windows Classic Desktop

Web

.NET Core

.NET Standard

Cloud

Test

WCF

Workflow

.NET Framework 4.6.1

Sort by: Default



Activity Designer Library

Visual C#



Activity Library

Visual C#



WCF Workflow Service Application

Visual C#



Workflow Console Application

Visual C#



# Workflow Console Application

In Workflow Console Application template,  
The Main() method in Program.cs is the entry point  
The workflowInvoker invokes the Workflow1.xaml file

The screenshot displays the Visual Studio IDE with the Workflow Console Application template. The **Program.cs** file is open, showing the **Main** method. A green arrow points to the `using System.Activities;` line. Another green arrow points to the **Workflow1.xaml** file in the **Solution Explorer**. A red oval highlights the `Workflow1()` argument in the `WorkflowInvoker.Invoke(new Workflow1());` line. A yellow callout box explains that the workflow is considered a thread and started by the Main method.

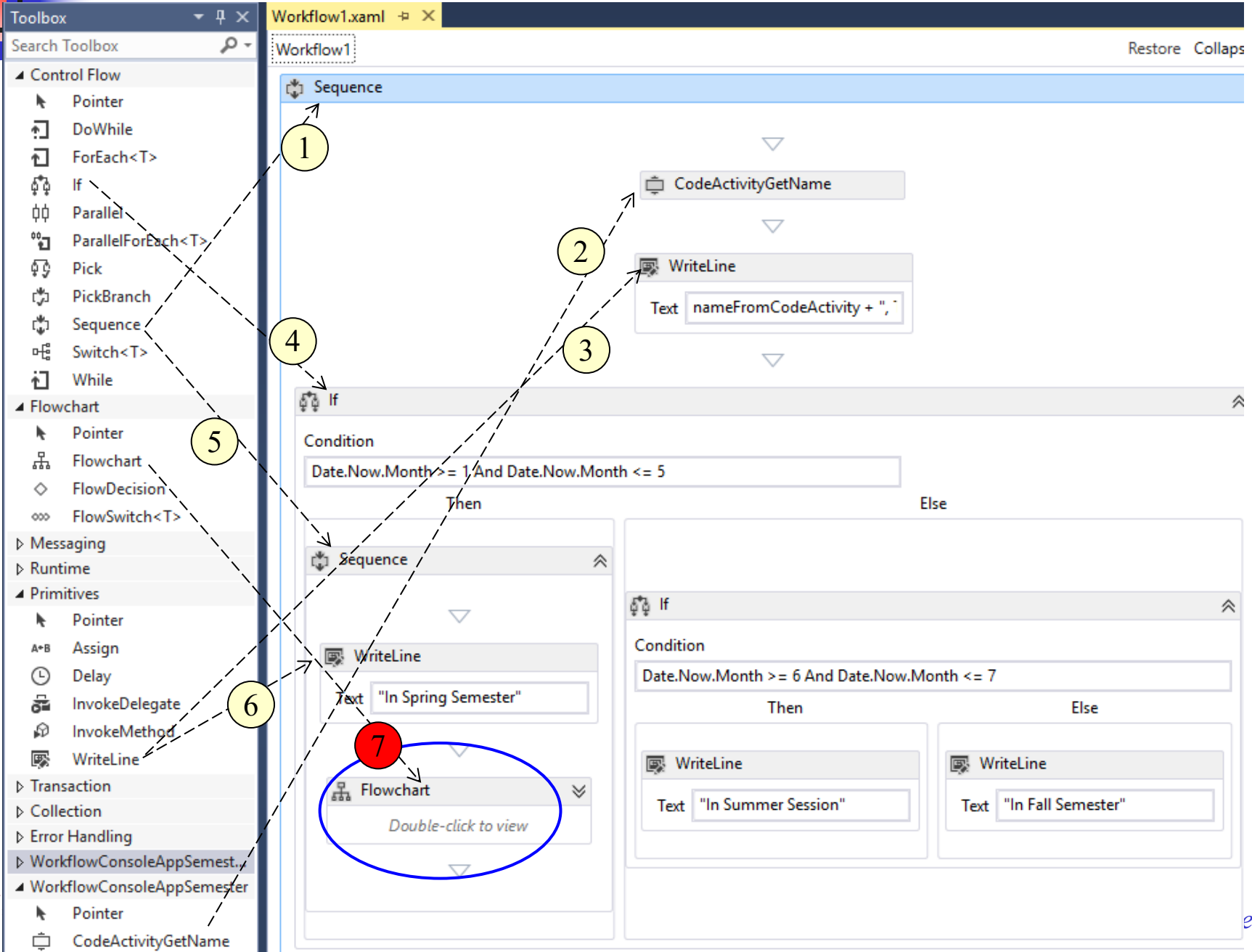
```
using System;
using System.Linq;
using System.Activities;
using System.Activities.Statements;

namespace WorkflowConsoleAppSemester
{
    class Program
    {
        static void Main(string[] args)
        {
            WorkflowInvoker.Invoke(new Workflow1());
        }
    }
}
```

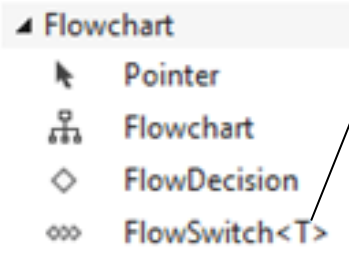
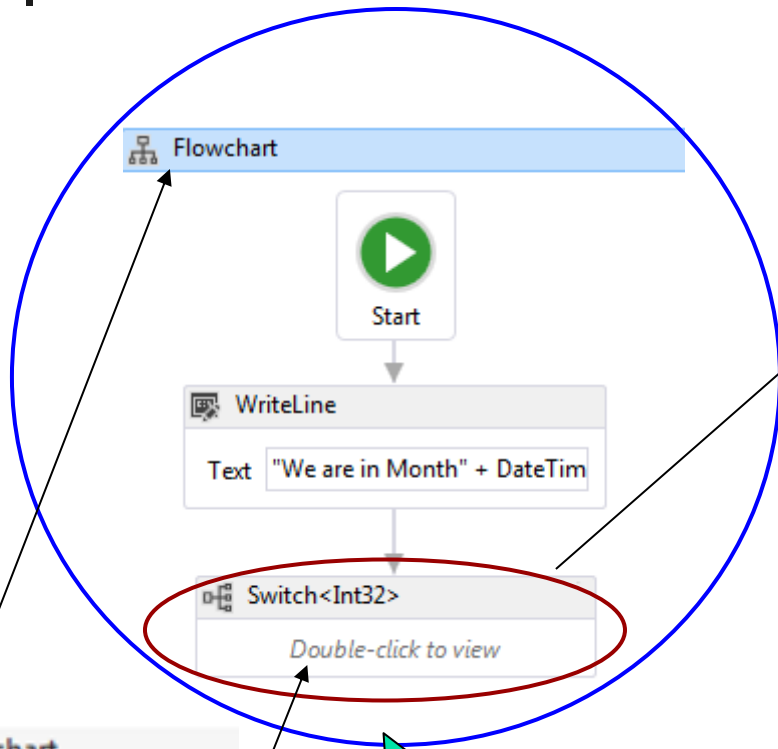
The workflow is considered a thread and started by the Main method.

# Graphic Development of Workflow.xaml

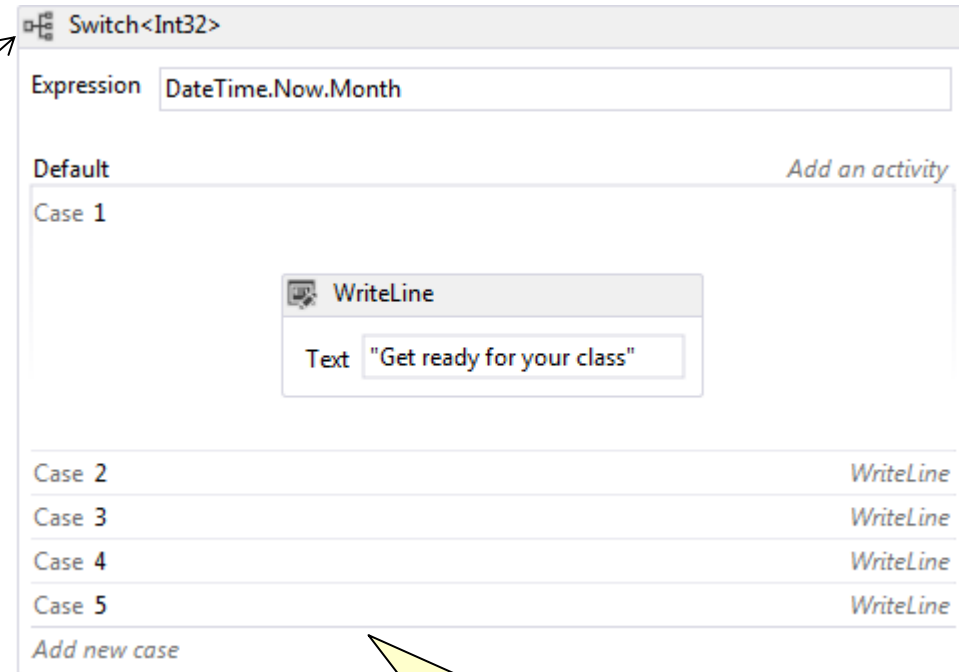
5



# A Flowchart is an Instant Component



A **Flowchart** creates an instant component in a workflow! The custom component does not appear in toolbox



A **Switch** is a construct only. It has a fixed structure.

# Outline of the Lecture

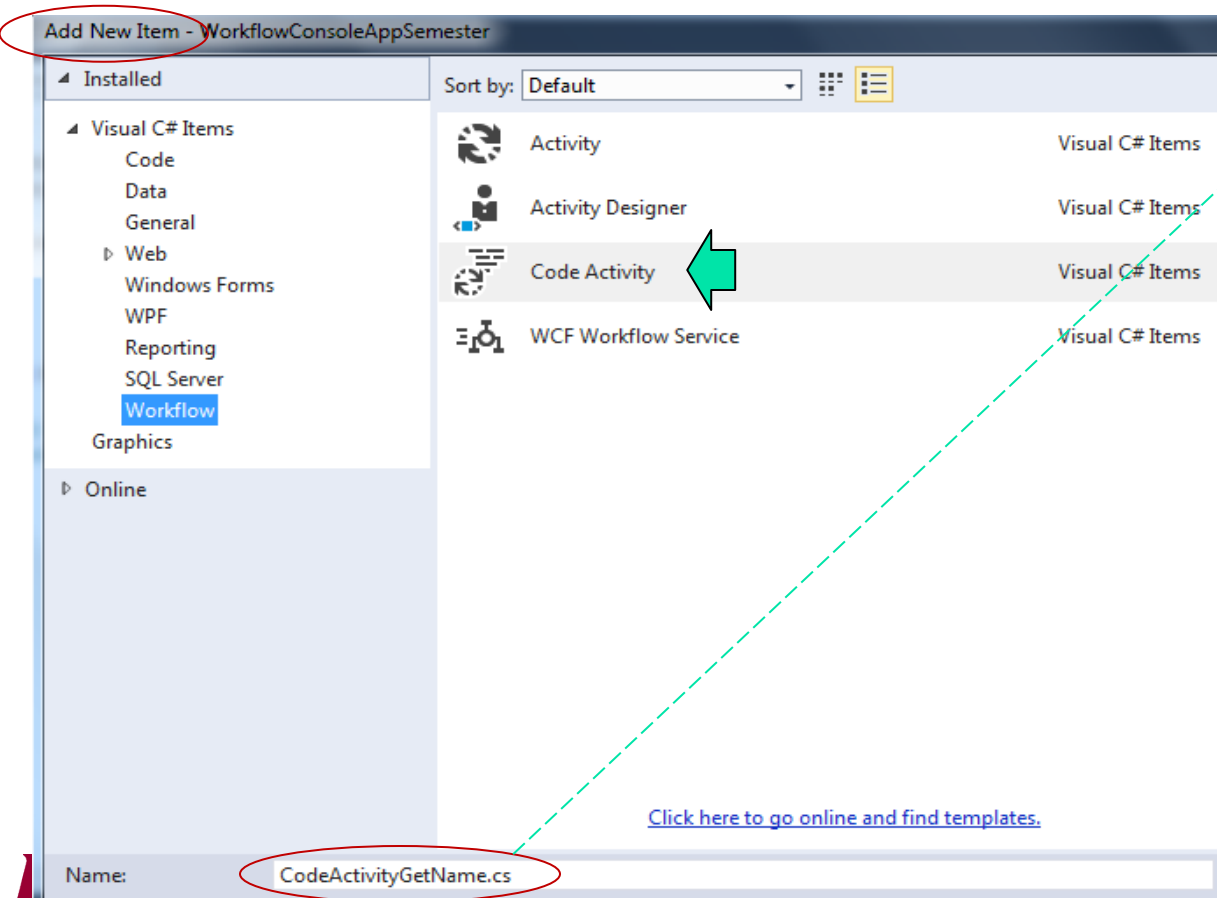
- Creating a Workflow Console Application
  - Adding a Flowchart component into a Workflow
- **Adding Other Components into a Workflow**
  - **Adding a CodeActivity into a Workflow**
  - Adding an External Web Service into a Workflow
- Creating Workflow Services
  - Contract-First Approach, resulting a service.svc file
  - Workflow-First Approach , resulting a service.xamlx file
- Case Studies:
  - Image Verifier in Workflow
  - Mortgage Application Integration

- Using pre-built components to compose applications are convenient and fast;
- We often cannot find the components we need: We need the capacity of building our own components.
- A Flowchart creates an instant component in the workflow. It creates a module, but it is not reusable.
- Furthermore, a flowchart is mainly for composition. It does not do basic computing well.
- We need the capacity of adding code (e.g., C#) activity: **CodeActivity**.
- WF makes this capacity easy.

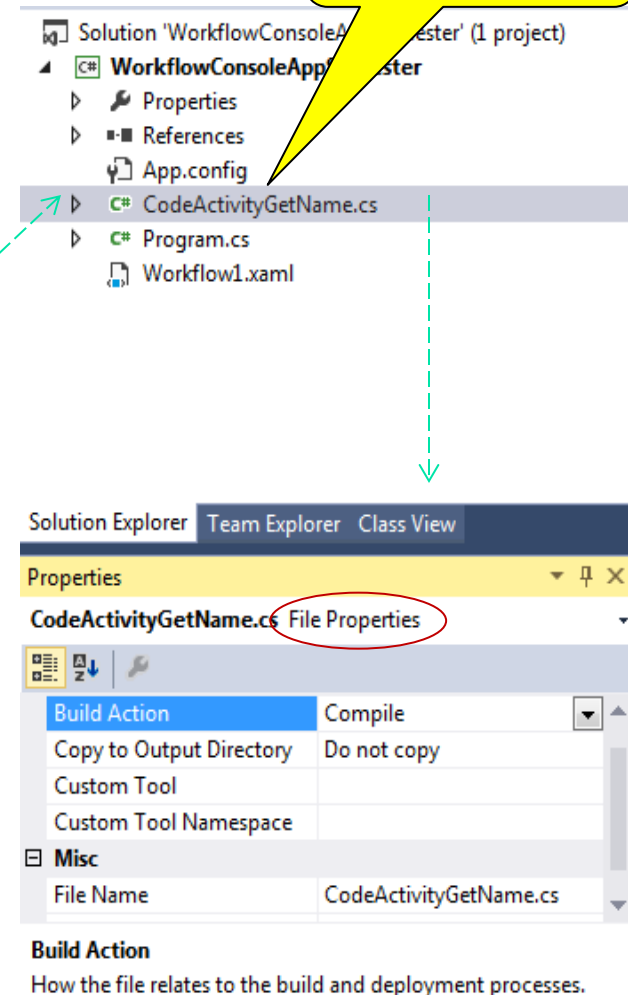


# Adding a CodeActivity into Workflow

- From VS we use Add New Item, and then it goes into the toolbox.



Type C# code in this file

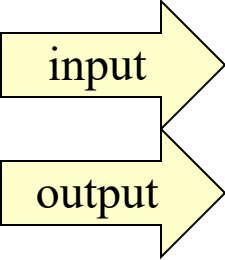


# Template Code in the Code Activity

```

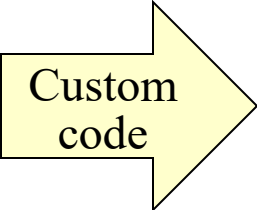
namespace WorkflowConsoleAppSemester {
    public sealed class CodeActivityGetName : CodeActivity {
        // Define an activity input argument of type string
        public InArgument<string> defaultName { get; set; }
        // Define an activity output argument of type string
        public OutArgument<string> enteredName { get; set; }
        // If your activity returns a value, derive from CodeActivity<TResult>
        // and return the value from the Execute method.
        protected override void Execute(CodeActivityContext context) {
            // Obtain the runtime value of the Text input argument
            Console.WriteLine("Please enter your name");
            string yourName = Console.ReadLine();
            if (yourName == "") {
                string dName = context.GetValue(this.defaultName);
                yourName = dName; }
            string helloName = " Hello " + yourName;
            context.SetValue(this.enteredName, helloName);
        }
    }
}

```



input

output



Custom  
code

# Adding CodeActivity into Workflow

Workflow1

Sequence

CodeActivityGetName

WriteLine

Text nameFromCodeActivity +

If

Condition

DateTime.Now.Month >= 1 & DateTime.Now.Month <= 5

Then

Sequence

WriteLine

Text "In Spring Semester"

Else

If

Condition

DateTime.Now.Month >= 6 & DateTime.Now.Month <= 7

Then

WriteLine

Text "In Summer Session"

Else

WriteLine

Text "In Fall Semester"

Properties

System.Activities.Presentation.View.DesignTimeVariable

Misc

Default "Yinong Chen";

Modifiers None

Name nameFromCodeActivity

Scope Sequence

Type String

Set value to activity arguments

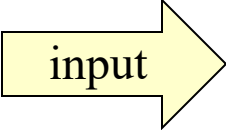
Added code activity is reusable

Variable created

Name	Variable type	Scope	Default
nameFromCodeActivity	String	Sequence	"Yinong Chen";

Create Variable

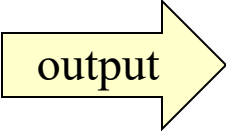
# Template for CodeActivity



input

- `public InArgument<string> defaultName`  
`{ get; set; }`

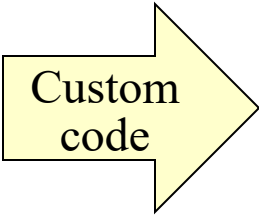
// which takes input from the workflow. In this  
// example, we could pass a string “John Doe” to the  
// code activity.



output

- `public OutArgument<string> enteredName`  
  
■ `{ get; set; }`

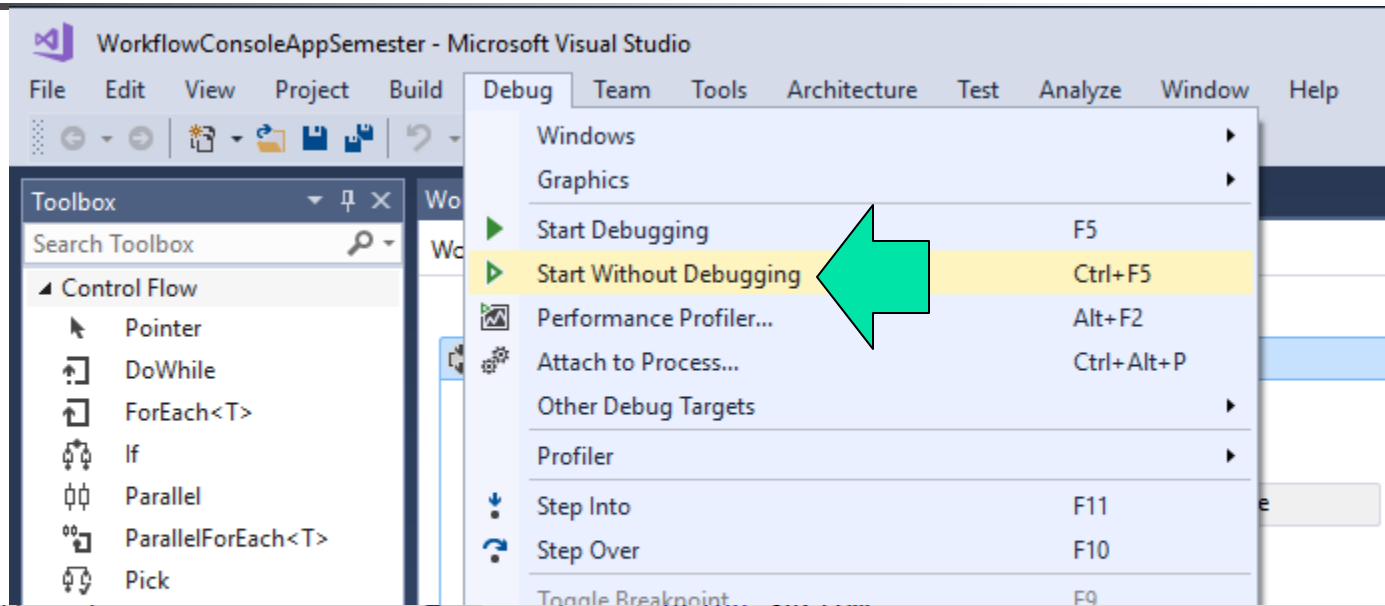
// which returns a value to the calling workflow.



Custom  
code

- `protected override void`  
`Execute(CodeActivityContext context) {`  
`// we add C# code that we want to execute here.`  
`}`

# Test the Code



C:\WINDOWS\system32\cmd.exe

Please enter your name

Yinong Chen

Hello Yinong Chen, Today's date is 2/4/2019 5:10:41 PM

In Spring Semester

We are in Month 2. What should I do?

Enjoy your classes

Press any key to continue . . .

# Outline of the Lecture

- Creating a Workflow Console Application
  - Adding a Flowchart component into a Workflow
- **Adding Other Components into a Workflow**
  - Adding a CodeActivity into a Workflow
  - **Adding an External Web Service into a Workflow**
- Creating Workflow Services
  - Contract-First Approach, resulting a service.svc file
  - Workflow-First Approach , resulting a service.xamlx file
- Case Studies:
  - Image Verifier in Workflow
  - Mortgage Application Integration

# Add and Call a Remote Web Service

<http://neptune.fulton.ad.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc>

**Add Service Reference**

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

Services:

Service
Service
IService

Operations:

Operation
Decrypt
Encrypt

1 service(s) found at address  
'http://neptune.fulton.ad.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc'.

Namespace:

**Toolbox**

Search Toolbox

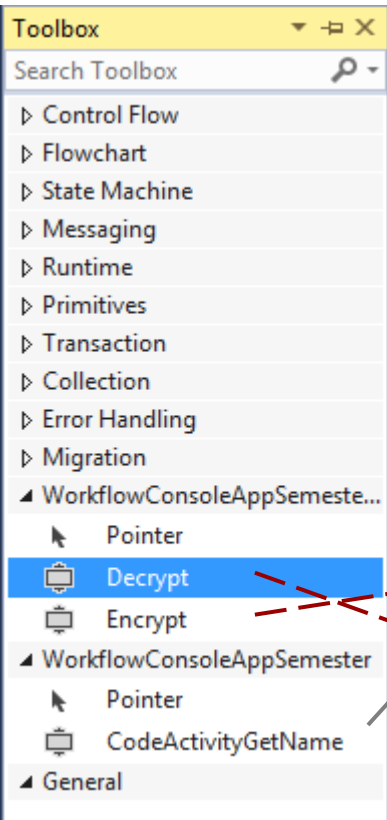
- Control Flow
- Flowchart
- State Machine
- Messaging
- Runtime
- Primitives
- Transaction
- Collection
- Error Handling
- Migration
- WorkflowServiceEncryptDecrypt
  - Pointer
  - CodeActivity1
- WorkflowServiceEncryptDecrypt.Cryptoservice.Activities
  - Pointer
  - Decrypt
  - Encrypt
- General

**Microsoft Visual Studio**

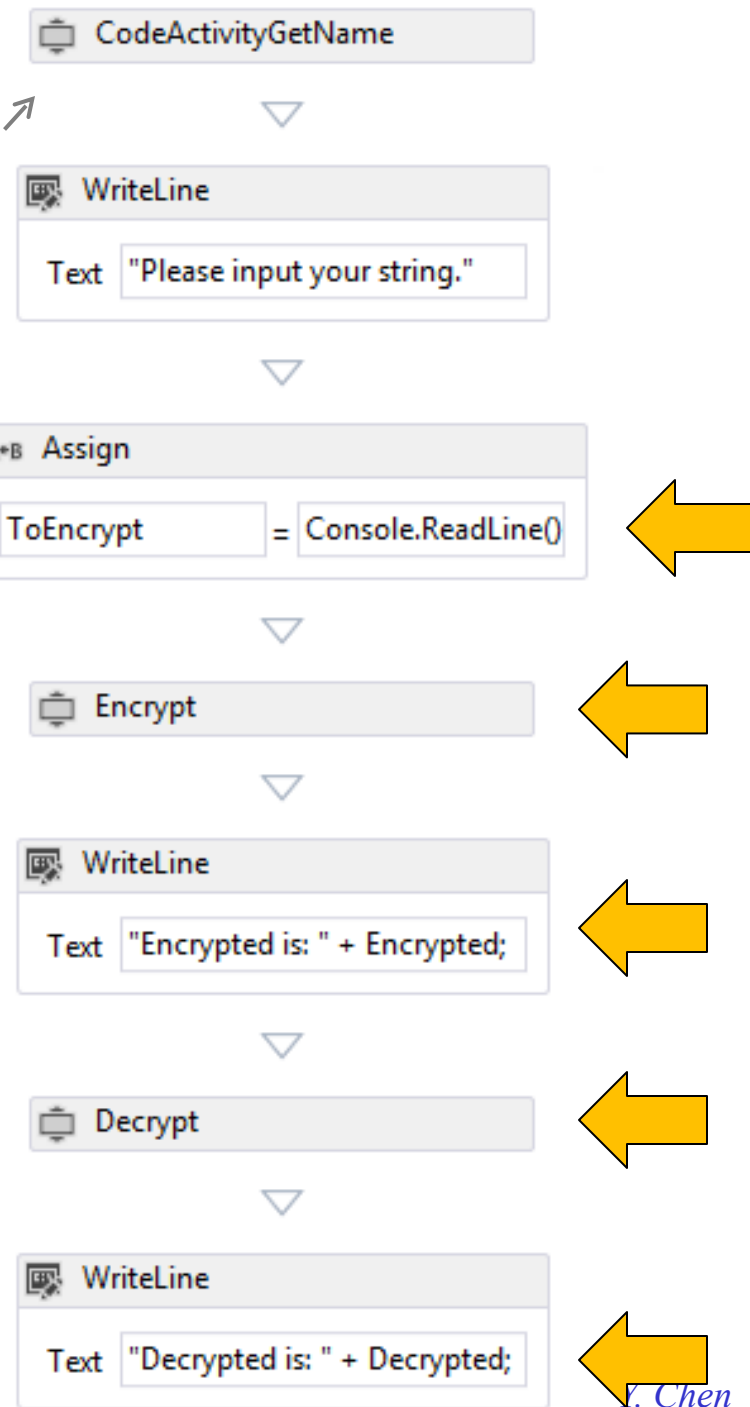
The operation is completed successfully. You will see the generated activities in the toolbox after you rebuild the project.

☐ In the future, do not show this message.

# Add Services



You can use a remote Web service just like a local code activity, by dragging and dropping!





# Defining Arguments for Services

CodeActivityGetName

WriteLine

Text "Please input your string."

A\*B Assign

ToEncrypt = Console.ReadLine()

Encrypt

WriteLine

Text "Encrypted is: " + Encrypted;

Decrypt

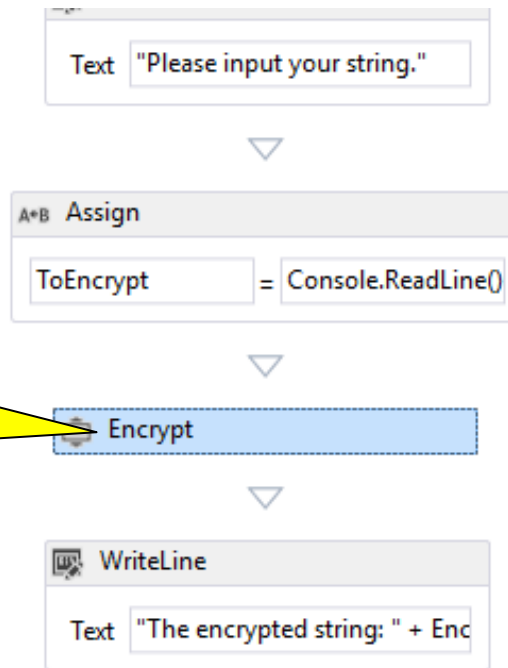
WriteLine

Text "Decrypted is: " + Decrypted;

Name	Direction	Argument type	Default value
ToEncrypt	In	String	<i>Enter a C# expression</i>
Encrypted	In/Out	String	<i>Default value not supported</i>
Decrypted	Out	String	<i>Default value not supported</i>
Create Argument			
Variables Arguments Imports			

# Associate the Arguments to the Service

1. Right click to open Properties



2

Search Solution Explorer (Ctrl+;)

- Solution 'WorkflowConsoleAppSemester' (1 project)
  - C# WorkflowConsoleAppSemester
    - Properties
    - References
    - Service References
    - Web References
    - App.config
    - ClassDiagram1.cd
    - CodeActivityGetName.cs
    - Program.cs
    - Workflow1.xaml

Solution Explorer Class View

Properties

WorkflowConsoleAppSemester.Cryptoservice.Activities.Encrypt...

Search:  Clear

Misc

Property	Value
DisplayName	Encrypt
EncryptResult	Encrypted
EndpointConfigurationName	BasicHttpBinding IService
text	ToEncrypt

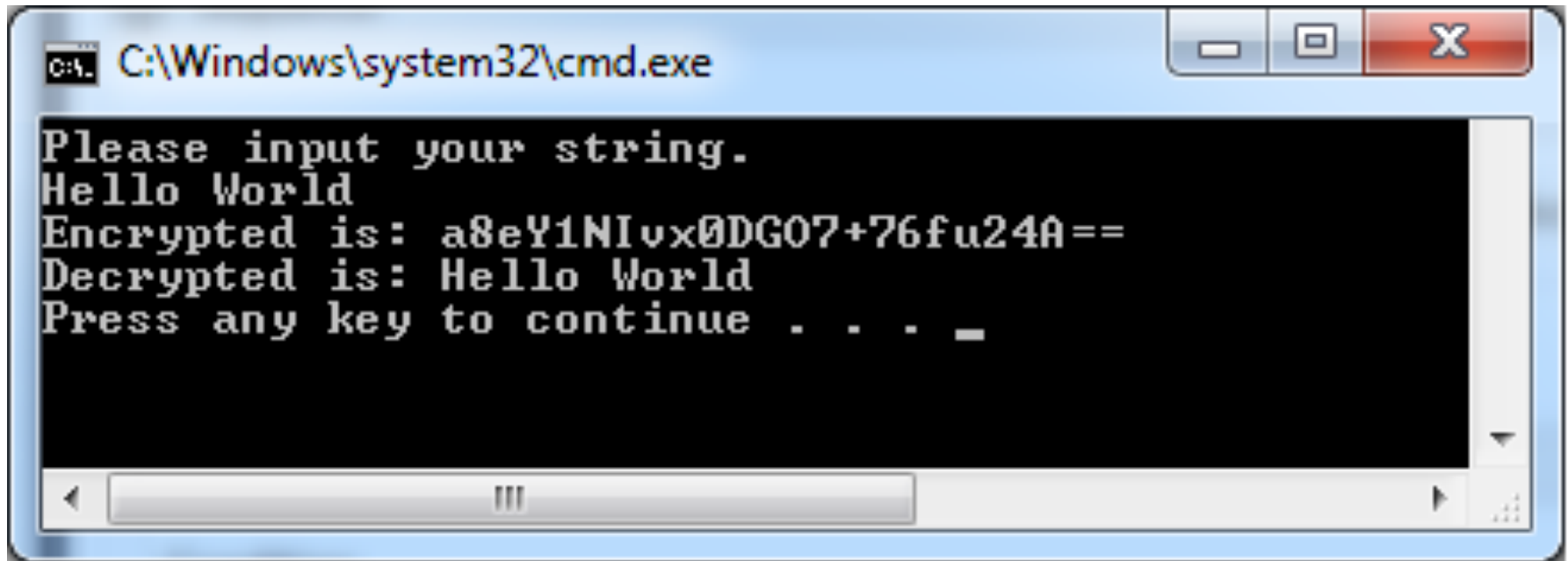
3

4

Name	Direction	Argument type	Default value
ToEncrypt	In	String	Enter a C# expression
Encrypted	In/Out	String	Default value not supported
Decrypted	Out	String	Default value not supported
Create Argument			

# Test the Code

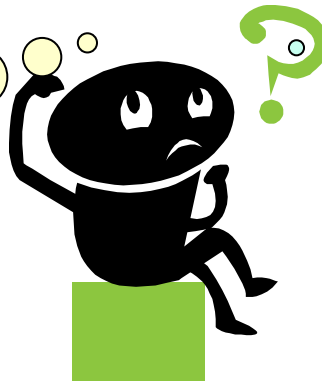
The input number is encrypted and the decrypted



```
C:\Windows\system32\cmd.exe

Please input your string.
Hello World
Encrypted is: a8eY1NIvx0DG07+76fu24A==
Decrypted is: Hello World
Press any key to continue . . . _
```

How do we  
call a RESTful  
service?



Using  
CodeActivity

# Outline of the Lecture

- Creating a Workflow Console Application
  - Adding a Flowchart component into a Workflow
- Adding Other Components into a Workflow
  - Adding a CodeActivity into a Workflow
  - Adding an External Web Service into a Workflow
- Creating Workflow-based Web Services
  - **Contract-First Approach, resulting a service.svc file**
  - **Workflow-First Approach , resulting a service.xamlx file**
- Case Studies:
  - Image Verifier in Workflow
  - Mortgage Application Integration

- Workflow supports architecture-driven approach of software development. It can be used for developing applications as well as services.
- WF is designed for working with WCF to
  - Consume WCF services, as we just discussed
  - Implement WCF services
- There are two approaches in applying workflow in a WCF **service**:
  - Contract-first
  - Workflow-first.

# Two Approaches of Workflow Services

## ■ Contract-First Service Development:

- Use the “WCF Service Application” template to start an ordinary WCF service project.
- The service will be exposed as a **.svc** service
- Add a workflow activity in the project to perform the high-level code organization work.
- Create a **synchronous** service

## ■ Workflow-First Service Development:

- Use a specifically designed template: “WCF Workflow Service Application” to create a new type of WCF service, with extension **.xamlx**
- The service interfaces are created in workflow development.
- Create a **synchronous** service or an **asynchronous** service

# Contract-First Service Development

- Start a new project and choose the WCF template “**WCF Service Application**”;
- Right-click project file and choose “Add” and then “New Item...”;
- Choose the **Workflow** template and then choose the “**Activity**” item; Name the item “myServiceActivity.xaml”;
- Follow the same steps to define IService.cs interface file;
- In your Service.svc.cs file, you add the directive: **using System.Activities**; and then, you can simply call the activity in the service operation:  
**WorkflowInvoker.Invoke(new myServiceActivity());**

# Contract-First Service Development

24

The image is a composite screenshot of Visual Studio illustrating the steps for Contract-First Service Development. It includes the 'New Project' dialog, the 'Add New Item' dialog, the 'Solution Explorer', and the code editor.

**Step 1: New Project**

The 'New Project' dialog is shown with the 'WCF Service Application' selected under 'Visual C#' > 'Web'. A green arrow points to this selection.

**Step 2: Add New Item**

The 'Add New Item' dialog is shown with 'ContractFirstWfService' selected under 'Visual C#' > 'WCF'. A green arrow points to this selection.

**Step 3: Solution Explorer**

The 'Solution Explorer' on the right shows the project structure. A green arrow points to 'myServiceActivity.xaml'.

**Step 4: Code Editor**

The code editor shows the implementation of the service. A red box highlights the workflow invocation code:

```
WorkflowInvoker.Invoke(new myServiceActivity());
```

A red arrow points from this code to the 'myServiceActivity.xaml' file in the 'Solution Explorer'.

**Annotations:**

- A yellow speech bubble says: "You can convert it into a RESTful service by following the steps we discussed." with a green arrow pointing to the 'WCF Service Application' in the 'New Project' dialog.
- A red box contains the text: "Finally, we can follow the previous example to write the workflow code for 'myServiceActivity.xaml'" with a red arrow pointing to the workflow invocation code.

```
1 using System;
2 using System.Activities;
3 using System.ServiceModel;
4 using System.ServiceModel.Web;
5
6 namespace ContractFirstWfService
7 {
8     // NOTE: You can use the "Rename" command on the "Refactor" menu to change the name of the service
9     // NOTE: In order to launch WCF Test Client for testing this service, please
10     // Orefences
11     public class Service1 : IService1
12     {
13         1 reference
14         public string GetData(int value)
15         {
16             WorkflowInvoker.Invoke(new myServiceActivity());
17             return string.Format("You entered: {0}", value);
18         }
19     }
20 }
```



# Two Approaches of Workflow Services

## ■ Contract-First Service Development:

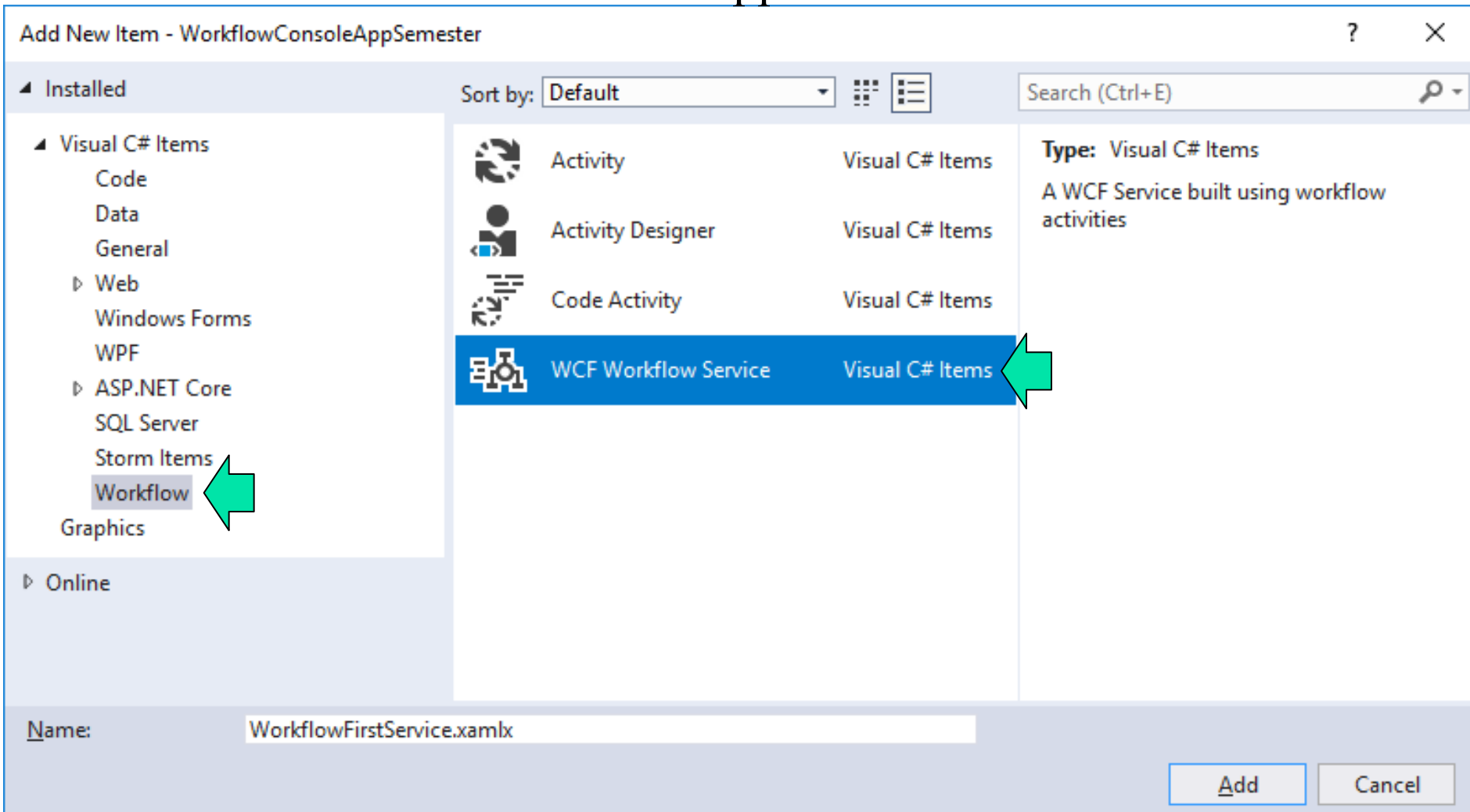
- Use the “WCF Service Application” template to start an ordinary WCF service project.
- The service will be exposed as a .svc service
- Add a workflow activity in the project to perform the high-level code organization work.
- Create a synchronous service

## ■ Workflow-First Service Development:

- Use a specifically designed template:  
“WCF Workflow Service Application” to create a new type of WCF service, with extension .xamlx
- The service interfaces are created in workflow development.
- Create a **synchronous** service or an **asynchronous** service

# Workflow-First Service Development

- Start a new project, select Workflow template, and select the “WCF Workflow Service Application”.



# Workflow-First Template

These activities can be used for synchronous and asynchronous communications

To build different types of interfaces

Use **Receive** activity to listen to the service call;

Use **Send** activity to send data back to client;

The screenshot displays the Visual Studio IDE with the Workflow-First Template project. The **Toolbox** on the left shows the **Messaging** category expanded, listing activities like **Receive** and **Send**. The **Solution Explorer** on the right shows the project structure, including **App\_Data** and **Service1.xamlx**. The **Properties** window at the bottom right shows the **Development Server** settings.

**Toolbox**

- Control Flow
- Flowchart
- State Machine
- Messaging**
  - Pointer
  - CorrelationScope
  - InitializeCorrelation
  - Receive
  - ReceiveAndSendReply
  - Send
  - SendAndReceiveReply
  - TransactedReceiveScope
- Runtime
- Primitives
- Transaction
- Collection
- Error Handling
- Migration
- General

**Sequential Service**

- ReceiveRequest**
  - OperationName: GetData
  - Content: View message...
- SendResponse**
  - Request: ReceiveRequest
  - Content: View message...

**Solution Explorer**

- Solution 'WorkflowFirstService' (1 project)
  - WorkflowFirstService**
    - Properties
    - References
    - App\_Data
      - Service1.xamlx**
    - Web.config

**Properties**

**WorkflowFirstService** Project Properties

**Development Server**

Always Start When Debugging	True
Anonymous Authentication	Enabled
Managed Pipeline Mode	Integrated
SSL Enabled	False

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



# Roadmap of the Lecture

---

- Creating a Workflow Console Application
- Adding Other Components into a Workflow
- Creating Workflow Services
- **Case Studies of Persistence Service:**
  - **Image Verifier in Workflow**
    - Text Section 7.4.5 with full detail
    - Service in ASU Service Repository
  - Mortgage Application Integration
    - Using Workflow-First approach
    - Code downloadable at:

<http://msdn.microsoft.com/en-us/magazine/ff646977.aspx>

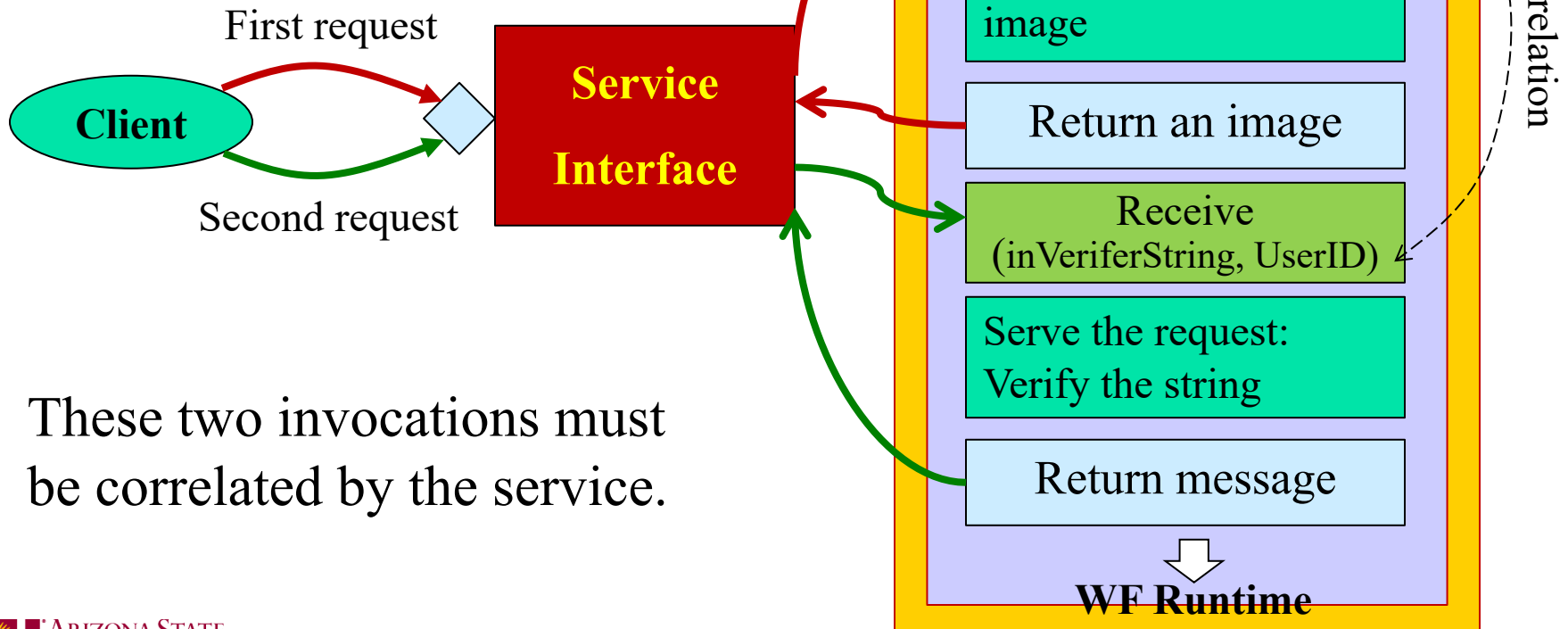
# What is a Persistence Service?

- A persistence service requires more than one access.
- It is similar to asynchronous service, where two calls are required, but it is different in the ways that a persistence service
  - may allow the second call to come after a long time, and thus it must save the partially finished service in a data store;
  - requires a correlation ID to relate all the calls to each other;
  - can accept independent inputs from all the calls.
- Use Image Verifier as an Example.

# Creating a Persistence Service With Correlation

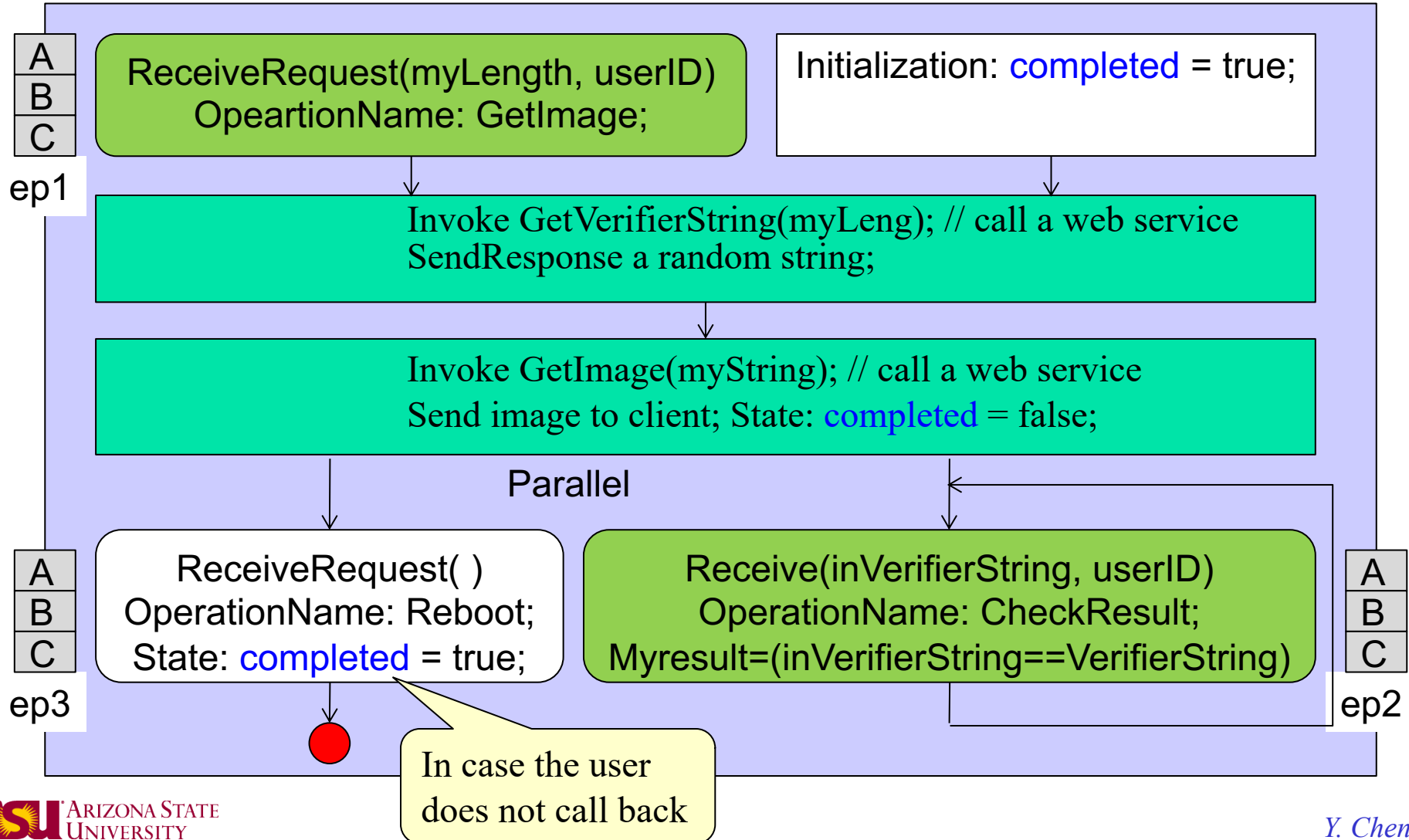
The client of an Image Verifier service must use two invocations to the service:

- (1) send string length,
- (2) send user-entered string



These two invocations must be correlated by the service.

# Workflow of Image Verifier Service as a Finite State Machine, with a variable “completed”



# Converting SVC Image Service to Persistence Service

**Add Service Reference** ? X

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

Services:

- Service
- IService**

Operations:

- GetImage
- GetVerifierString

1 service(s) found at address  
'http://neptune.fulton.ad.asu.edu/WSRepository/Services/ImageVerifierSvc/Service.svc'.

Namespace:

We will create a new persistence service with three correlated endpoints.

This services has two independent synchronous endpoints



# A New Two-Call Service

## that you can Add to other applications

Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

Services:

- Service1
  - Image
  - IReboot
  - IResult

Operations:

Endpoint 1: Start a new request

Endpoint 3: Reset the service in case the previous caller does not make the second call. Hide this from normal users.

Endpoint 2: Compare the user-entered string with the generated string.

1 service(s) found at address: 'http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFImage/WFservice/service1.xamlx '.

Namespace:

# Workflow of Image Verifier Service

## After .svc image service is added

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

- Toolbox:** Shows the 'ImageValidationWF.MyServ...' group with controls: Pointer, GetImage, and GetVerifierString. A message at the bottom states: "There are no usable controls in this group. Drag an item onto this text to add it to the toolbox."
- Service1.xaml:** The main workflow editor showing a Sequential Service with the following steps:
  - ReceiveRequest:** OperationName: GetImage, Content: View parameter...
  - GetVerifierString:** (Intermediate step)
  - GetImage:** (Intermediate step)
  - SendResponse:** Request: ReceiveRequest, Content: View message...
  - Assign:** completed = False
- Solution Explorer:** Shows the project 'ImageValidationWF' (2 projects) with files: Properties, References, Service References, App\_Data, Service1.xaml, and Web.config.

Yellow callout boxes provide additional context:

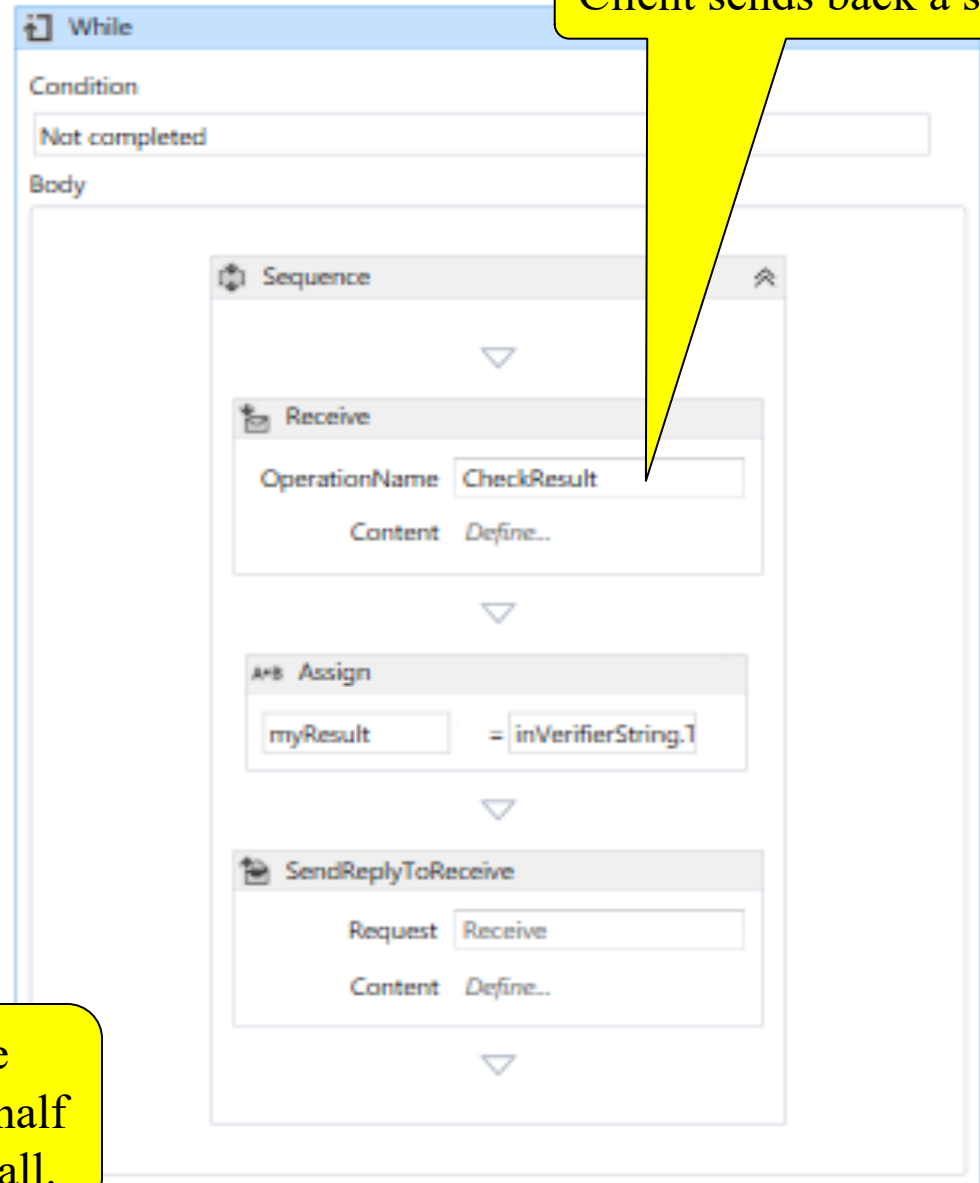
- Receive string length and UserID:** Points to the 'ReceiveRequest' step.
- Send image to client:** Points to the 'SendResponse' step.
- Waiting for client entered string:** Points to the 'Assign' step.

# The Main Part of the Workflow

Why do we need this endpoint?

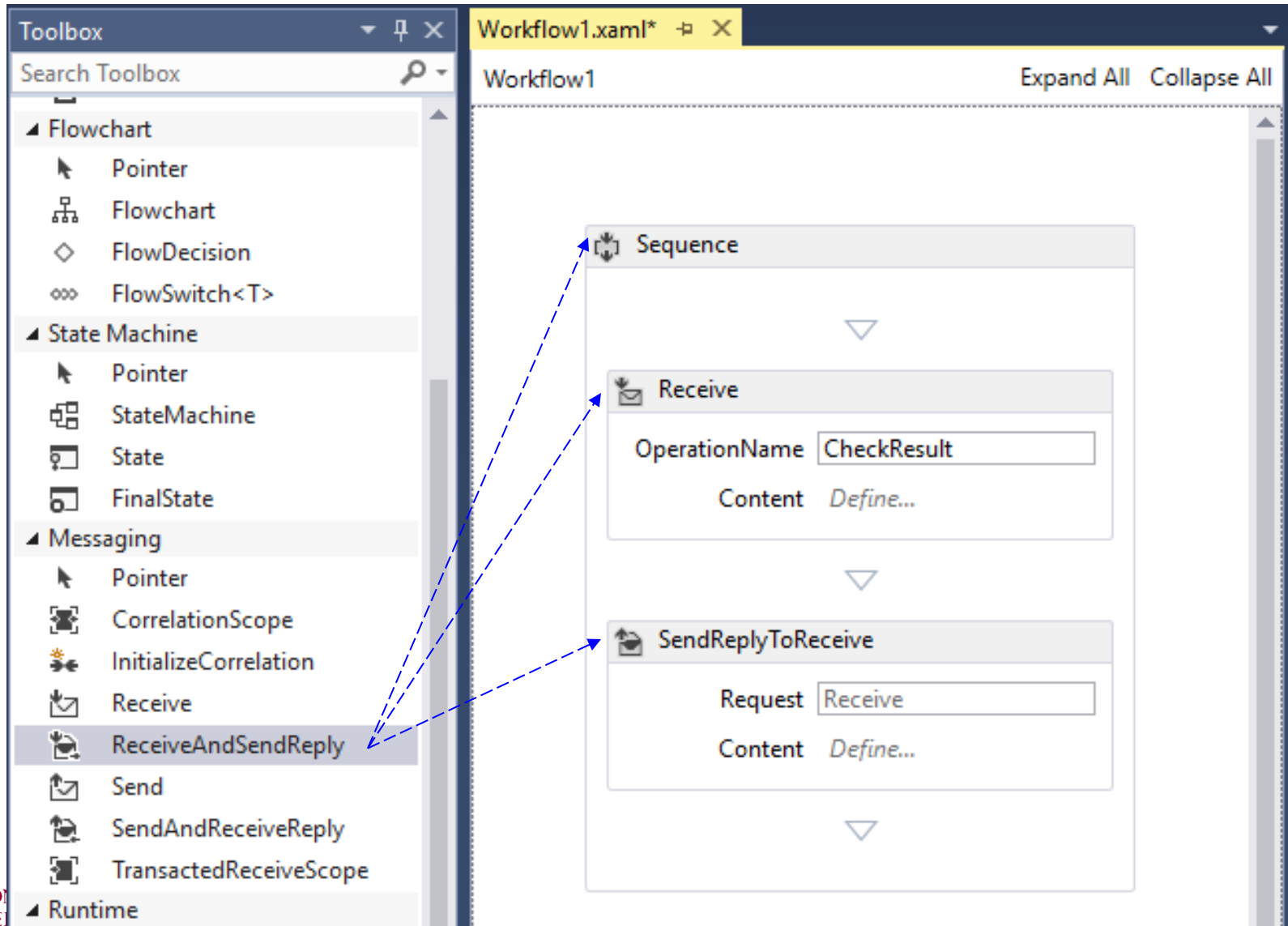
Client sends back a string

My server does not support persistence calls. Call it to terminate the previous half call, assuming it never makes the 2<sup>nd</sup> call.



# ReceiveAndSendReply Composite Activity

- It consists of Sequence, Receive and SendReplyToReceive



# “View parameter...” and “View message...”

Content Definition

☐ Message

☒ Parameters

Name	Type	Assign To
myLength	String	myLength
UserID	String	Enter a C# expression

Add new parameter

OK Cancel

Content Definition

☒ Message

Message data myImage

Message type System.IO.Stream

☐ Parameters

OK Cancel

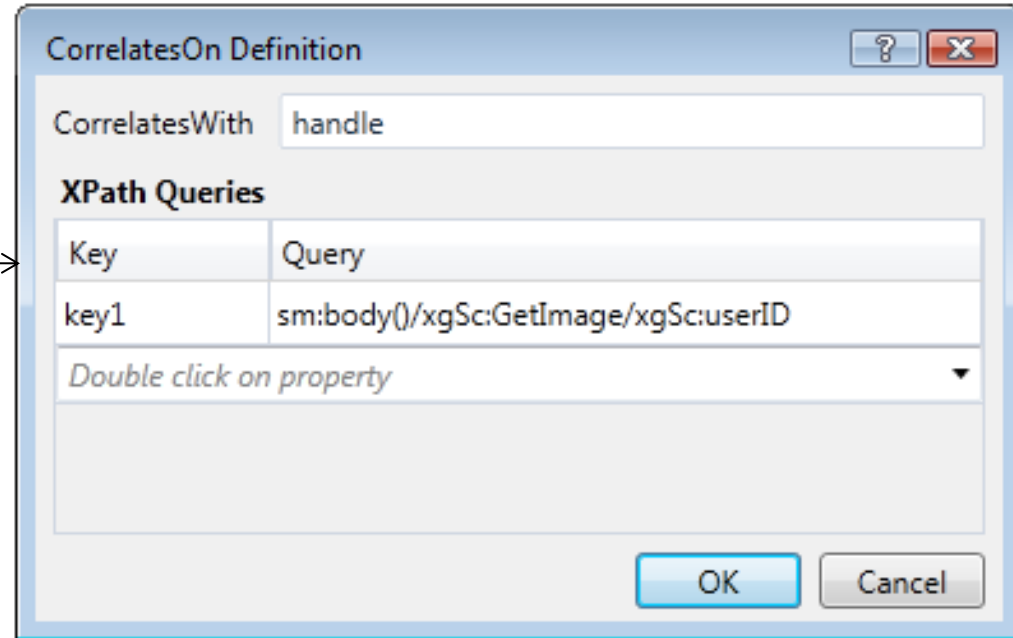
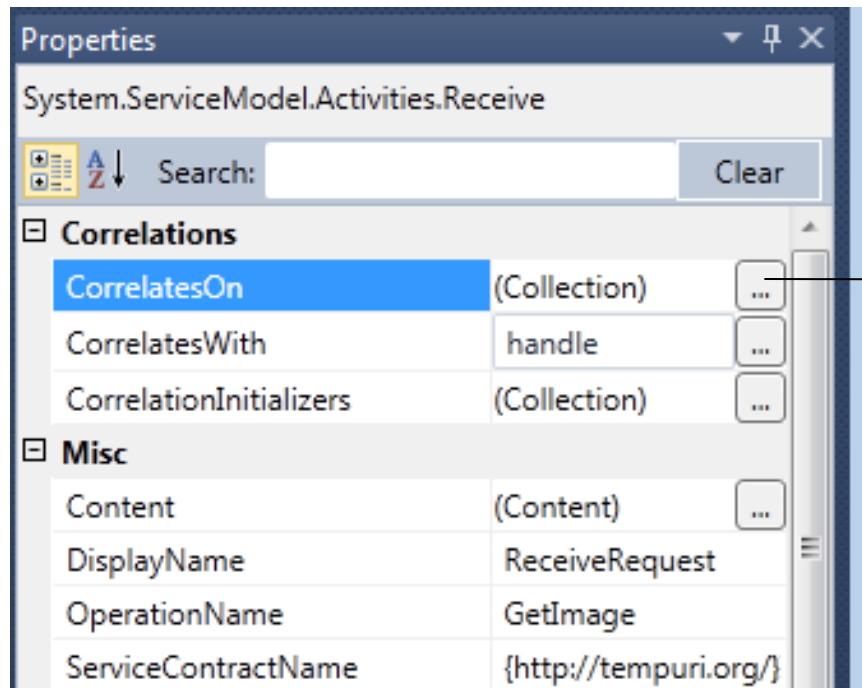
# Variables Defined for the Workflow

Two calls must  
be associated  
through a handle

Name	Variable type	Scope	Default
completed	Boolean	Sequence	true
handle	CorrelationHandle	Sequence	Handle cannot be initialized
myImage	Stream	Sequence	Enter a C# expression
myLength	String	Sequence	"4"
myResult	Boolean	Sequence	Enter a C# expression
userID	String	Sequence	Enter a C# expression
verifierString	String	Sequence	Enter a C# expression
Create Variable			
Variables Arguments			

used by all three receive activities as a  
parameter for identifying the correlation  
among all visits from the same user.

# Definition of Correlation Variable Handle



<http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFImage/WFservice/service1.xamlx>

<http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFImage/WFService/Service1.xamlx?wsdl>

More detail is in text  
section 7.4.5



# Roadmap of the Lecture

---

- Creating a Workflow Console Application
- Adding Other Components into a Workflow
- Creating Workflow Services
- Case Studies of Persistence Service:
  - Image Verifier in Workflow
    - Text Section 7.4 with full detail
    - Service in ASU Service Repository
  - **Mortgage Application Integration**
    - **Using Workflow-First approach**
    - **Code downloadable at:**

<http://msdn.microsoft.com/en-us/magazine/ff646977.aspx>

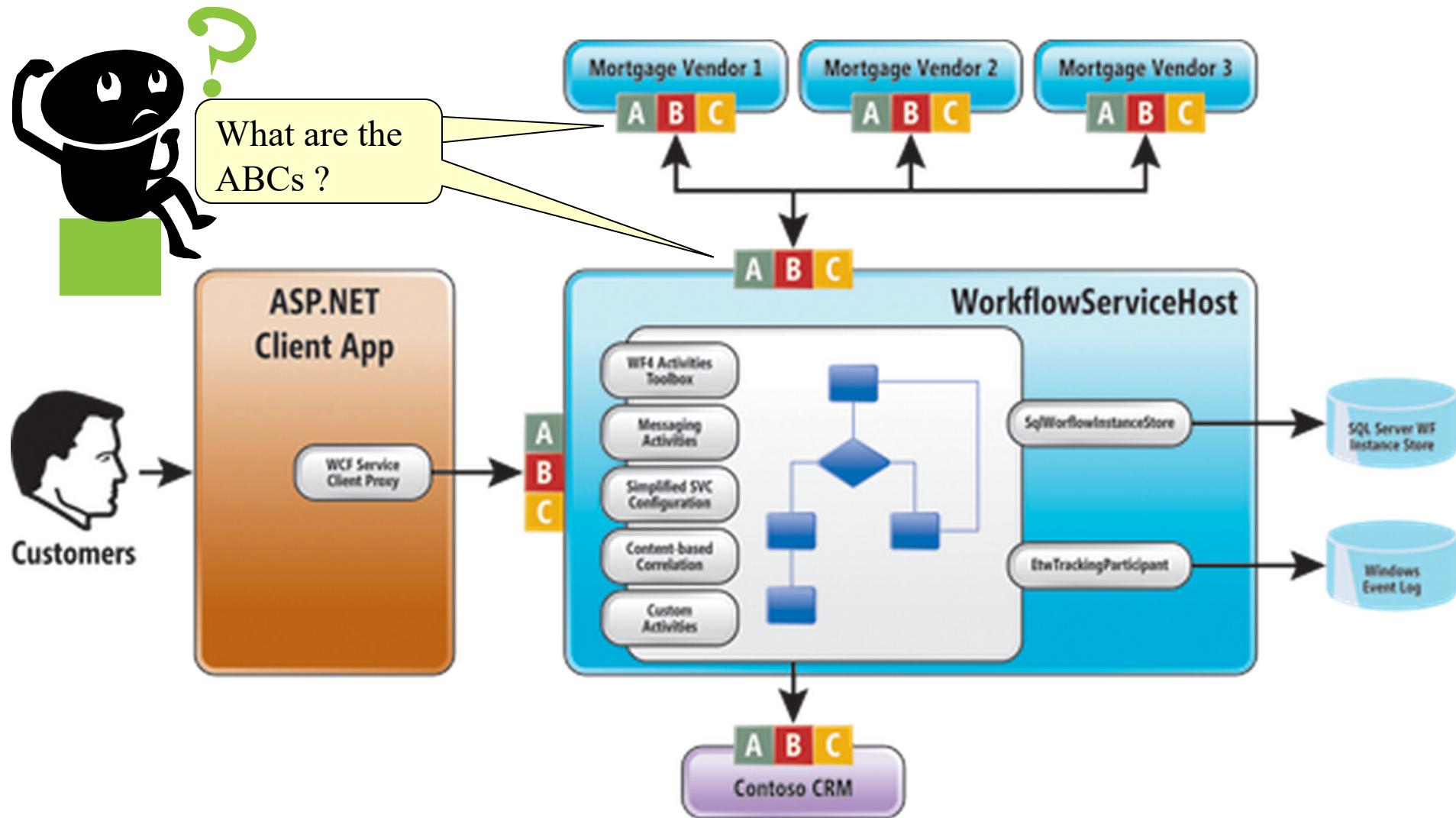


# Application Requirement (Scenarios)

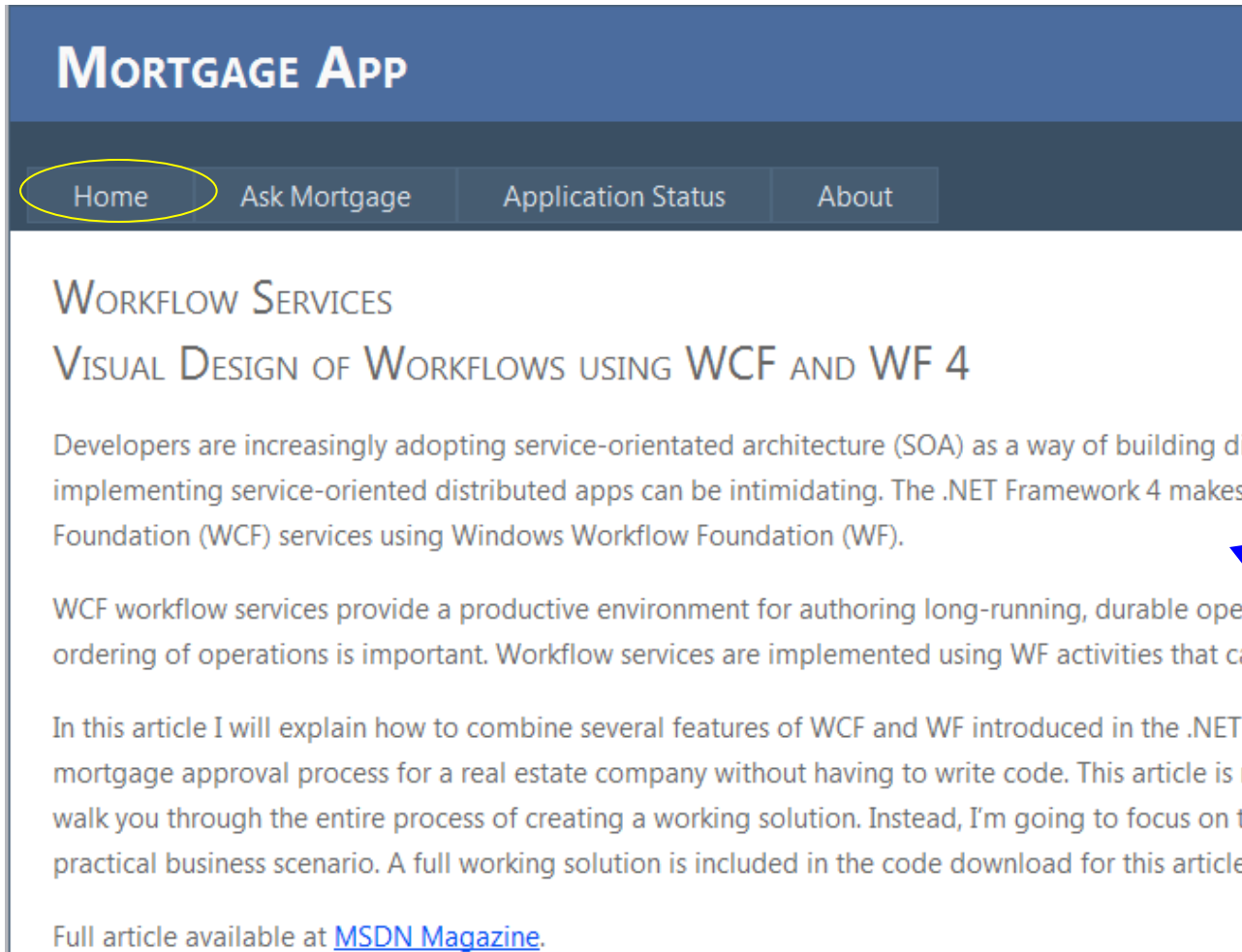
- Contoso Housing is a real estate company that sells houses.
- To provide better customer service and an end-to-end buying experience, Contoso partners with three mortgage companies that assist potential customers with their mortgage needs.
- Each mortgage company offers different interest rates.
- Contoso prioritizes mortgage vendors by their interest rates to ensure that customers get the best deal (using the assumption that a better rate makes the house more likely to sell)

# Overview of the Application and Service

Source: <http://msdn.microsoft.com/en-us/magazine/ff646977.aspx>



# Case Study: Mortgage Application (Client)



The screenshot shows the 'MORTGAGE APP' web application. The header is dark blue with the title 'MORTGAGE APP' in white. Below the header is a navigation bar with four buttons: 'Home' (highlighted with a yellow circle), 'Ask Mortgage', 'Application Status', and 'About'. The main content area has a light blue background and contains the following text:

## WORKFLOW SERVICES

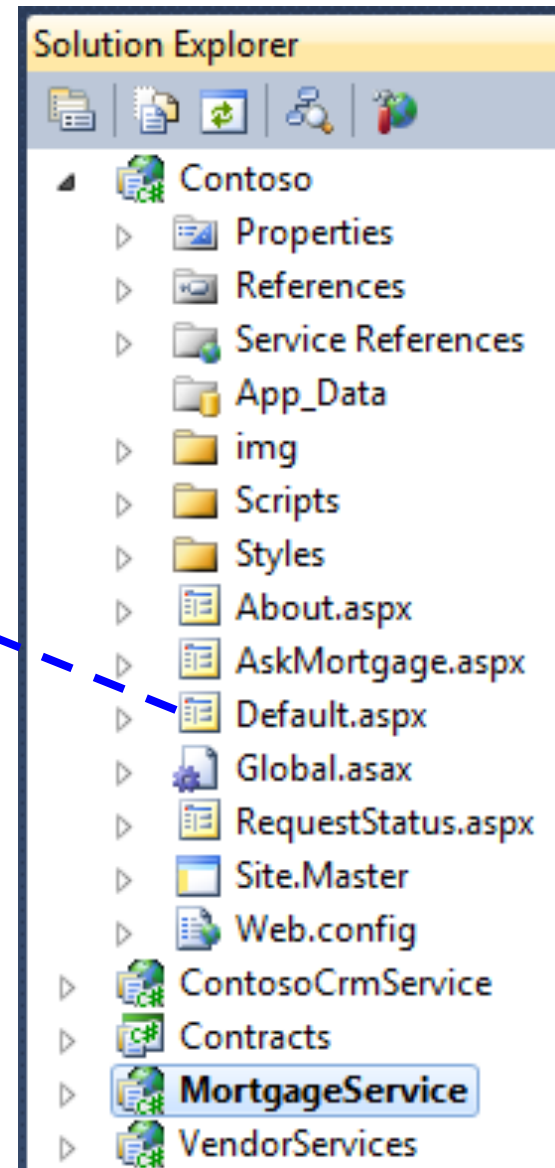
### VISUAL DESIGN OF WORKFLOWS USING WCF AND WF 4

Developers are increasingly adopting service-orientated architecture (SOA) as a way of building distributed apps. Implementing service-oriented distributed apps can be intimidating. The .NET Framework 4 makes it easier to implement service-oriented distributed apps using Windows Workflow Foundation (WCF) services using Windows Workflow Foundation (WF).

WCF workflow services provide a productive environment for authoring long-running, durable operations. The ordering of operations is important. Workflow services are implemented using WF activities that can be visualized and debugged.

In this article I will explain how to combine several features of WCF and WF introduced in the .NET Framework 4 to implement a mortgage approval process for a real estate company without having to write code. This article is not a tutorial; it will walk you through the entire process of creating a working solution. Instead, I'm going to focus on the practical business scenario. A full working solution is included in the code download for this article.

Full article available at [MSDN Magazine](http://msdn.microsoft.com/en-us/magazine/ff646977.aspx).



Source: <http://msdn.microsoft.com/en-us/magazine/ff646977.aspx>

# Case Study: Mortgage Application (Client)

Home **Ask Mortgage** Application Status About

## REQUEST MORTGAGE

GENERAL INFO

Enter your customer identification

House Price

% Down payment

Years

Salary (yearly)

Do you have credit history in the US? ☒

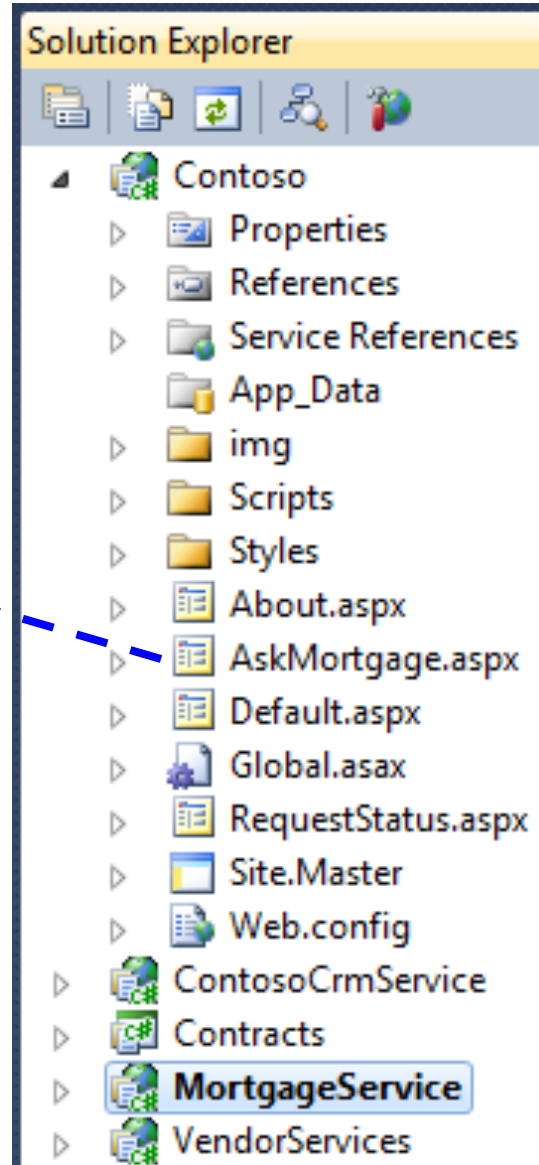
DECLARATIONS

☐ Have you been declared bankrupt within the past 7 years?

☐ Have you had property foreclosed upon or given title or deed in lieu thereof in the last 7 years?

☐ Are you a party of a lawsuit?

You can combine the image verifier here.



# Case Study: Mortgage Application (Client)

## MORTGAGE APP

[Home](#)[Ask Mortgage](#)[Application Status](#)[About](#)

### CHECK THE STATUS OF YOUR REQUEST

Enter your customer identification

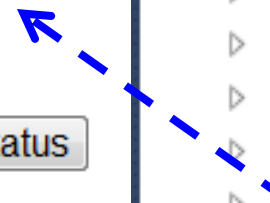
Click 'Update status' to retrieve the status of your application

What is a persistence service?

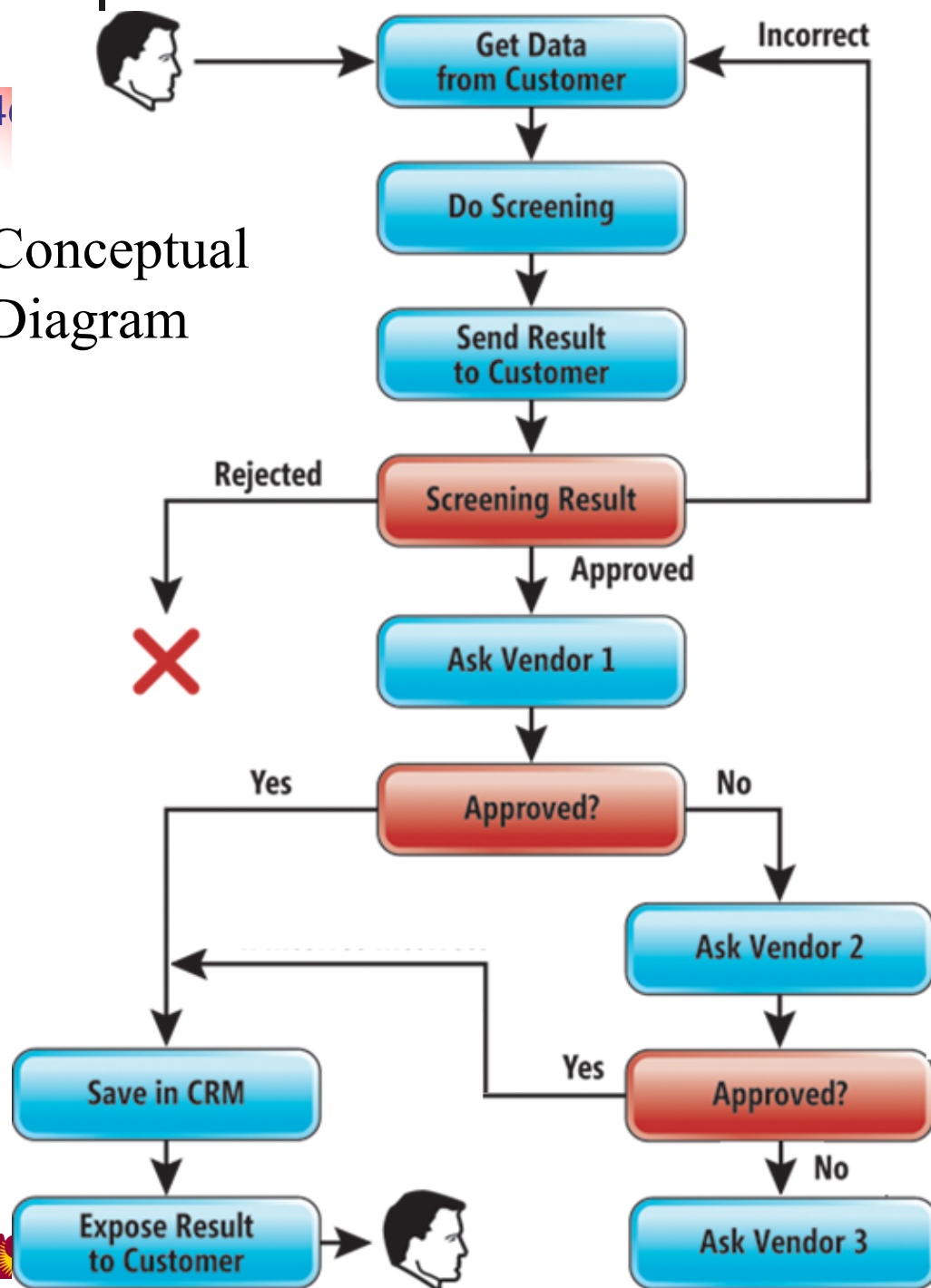


### Solution Explorer

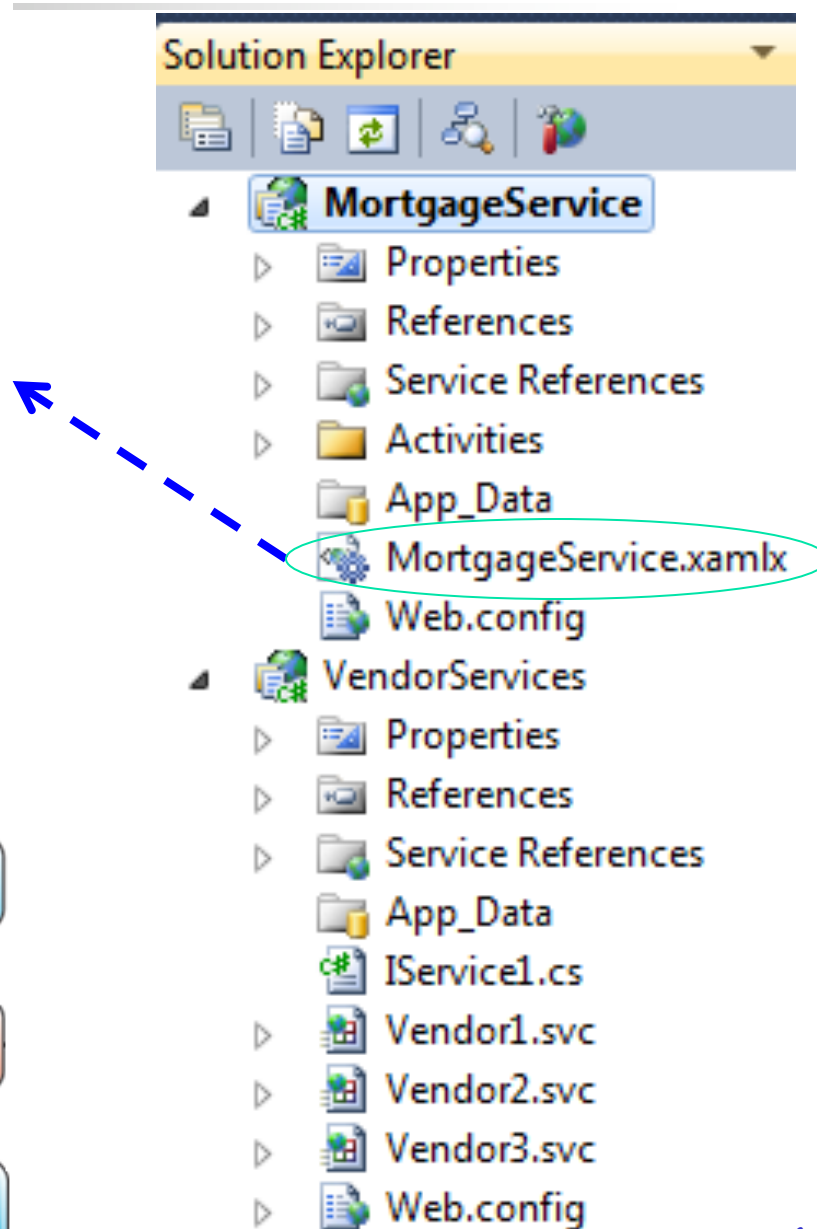
- Contoso
  - Properties
  - References
  - Service References
  - App\_Data
  - img
  - Scripts
  - Styles
  - About.aspx
  - AskMortgage.aspx
  - Default.aspx
  - Global.asax
  - RequestStatus.aspx
  - Site.Master
  - Web.config
- ContosoCrmService
- Contracts
- MortgageService**
- VendorServices



## Conceptual Diagram



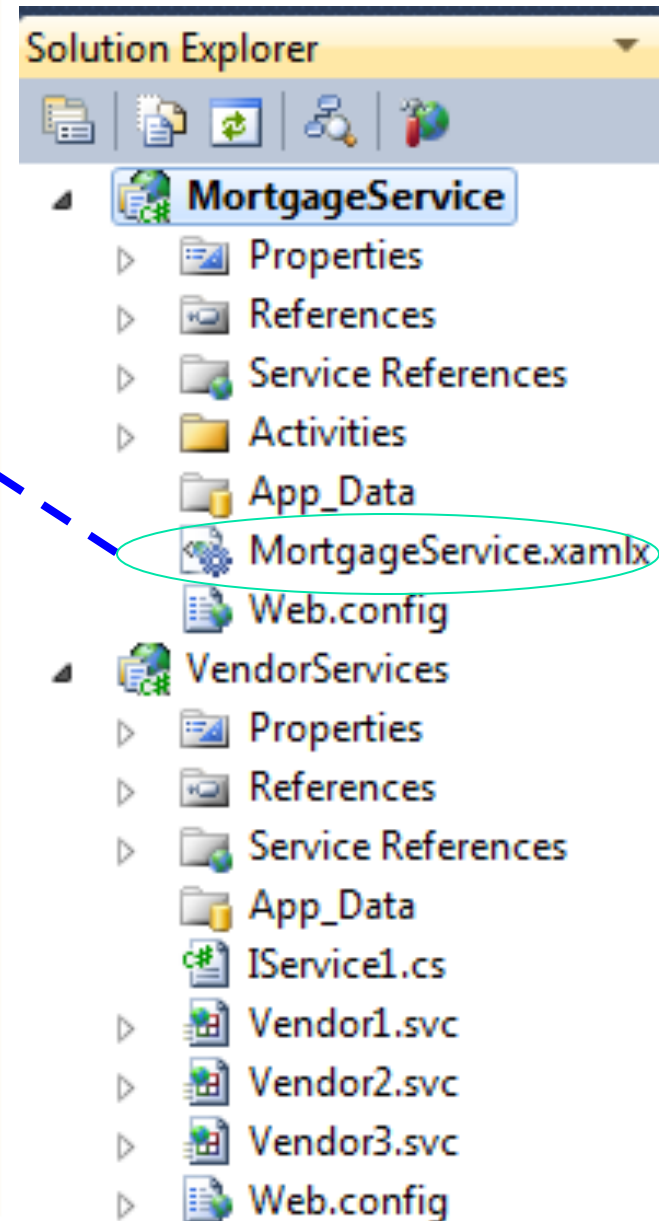
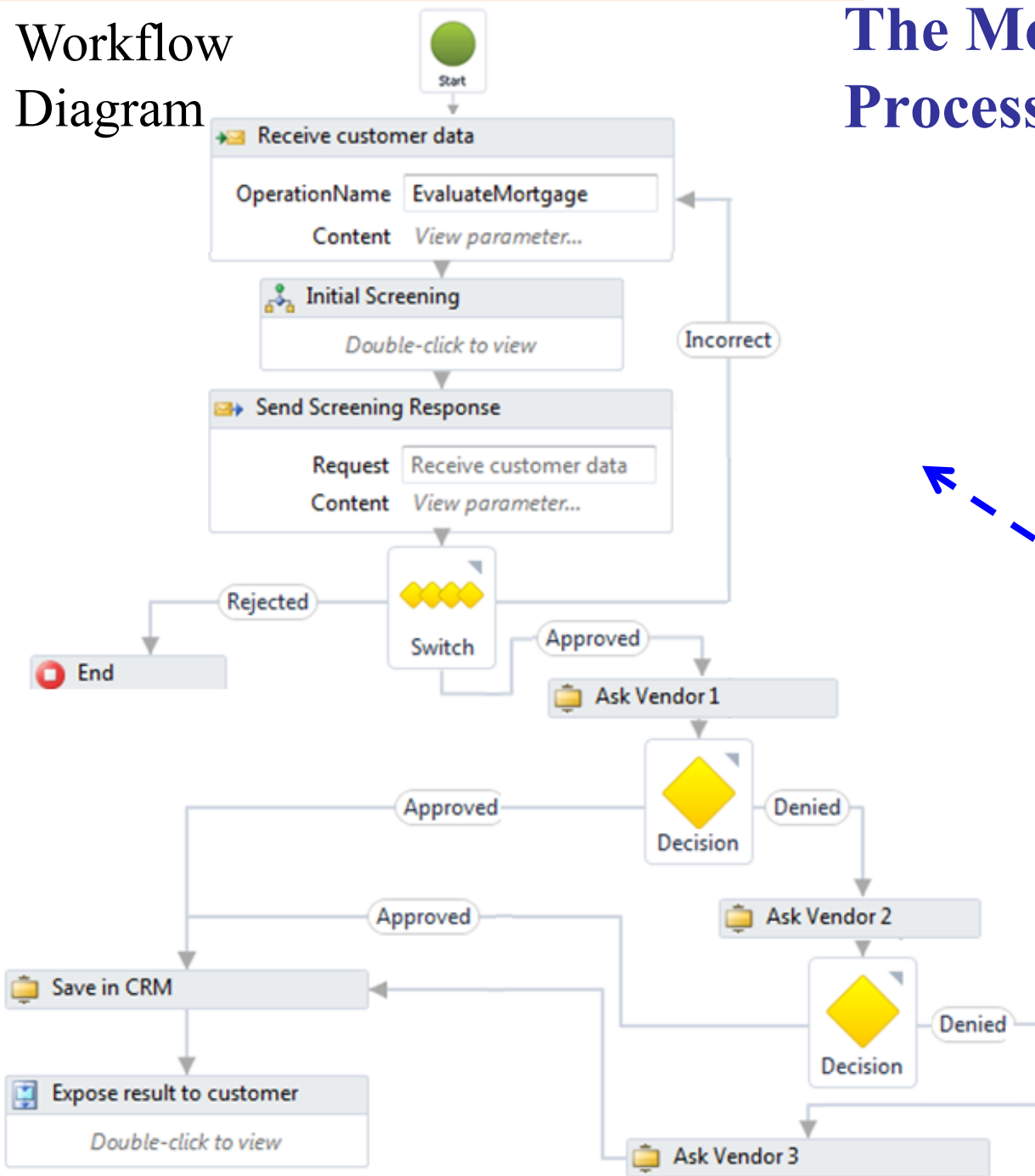
## Workflow Service





# Workflow Diagram

# The Mortgage Approval Process as a Flowchart



# Define the Properties of **Receive** Activity

Receive customer data

OperationName

Content [View parameter...](#)

Define the WCF  
Endpoint:  
Operation Contract:  
**Parameters in & out**

Define the WCF  
Endpoint:  
Operation Contract:  
**Operation Name**

Properties

System.ServiceModel.Activities.Receive

Search:  Clear

**Correlations**

CorrelatesOn	(Collection)	...
CorrelatesWith	customerCorrelati	...
CorrelationInitializers	(Collection)	...

**Misc**

Content	(Content)	...
DisplayName	Receive customer data	
OperationName	EvaluateMortgage	
ServiceContractName	ContosoRealEstate	
More Properties		
Action		
CanCreateInstance	<input checked="" type="checkbox"/>	
KnownTypes	(Collection)	...
ProtectionLevel	(null)	
SerializerOption	DataContractSerializ	



# Configuring Input Parameters

Content Definition

☐ Message

☒ Parameters

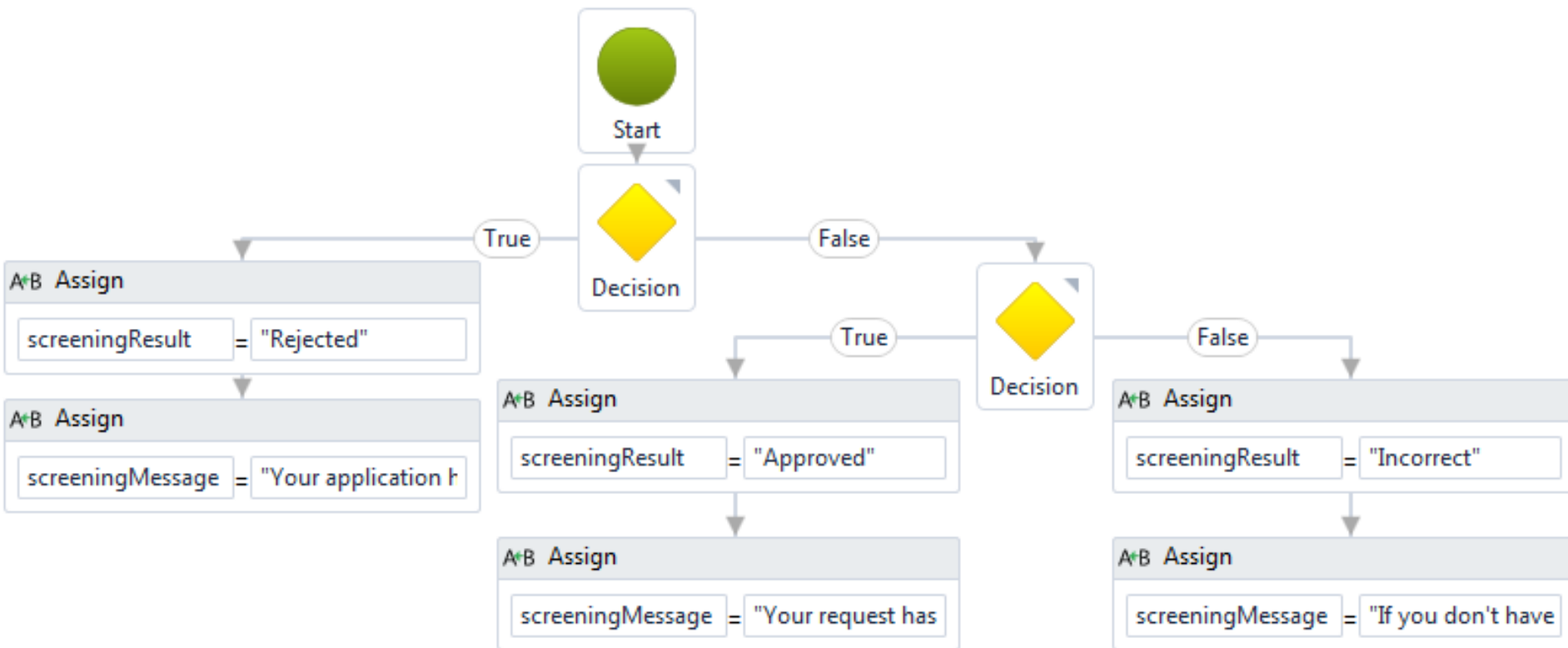
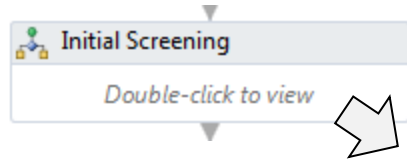
↑ ↓

Name	Type	Assign To
customerCode	String	customerId
housePrice	Int32	housePrice
downPayment	Int32	downPercentage
years	Int32	years
salary	Int32	salary
hasCreditHistory	Boolean	hasCreditHistory
hadBankruptcy	Boolean	hadBankruptcy
inLawsuit	Boolean	inLawsuit
hadForeclosure	Boolean	hadForeclosure

*Add new parameter*

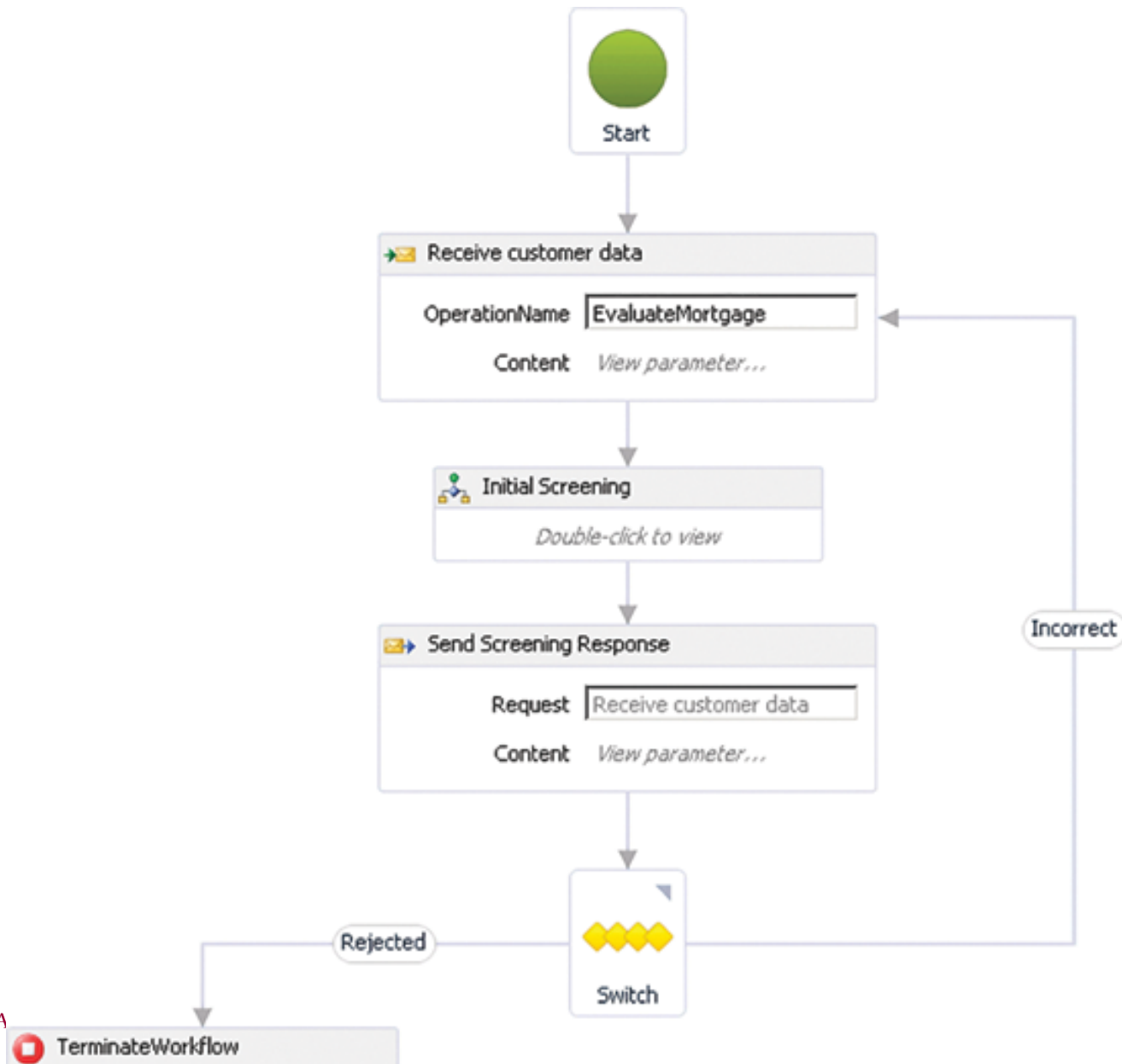
OK Cancel

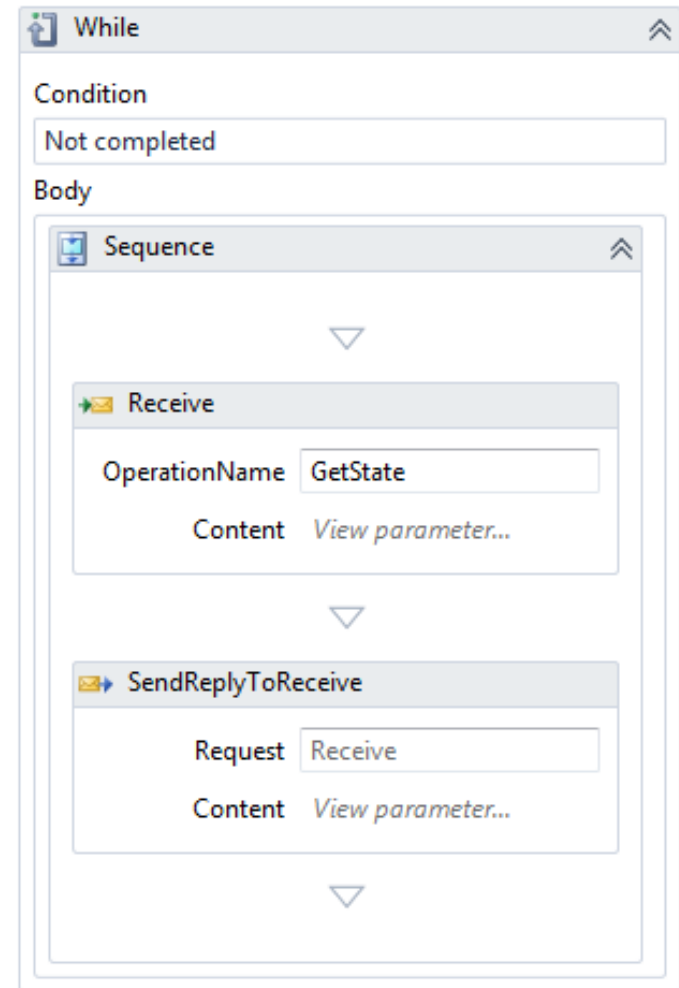
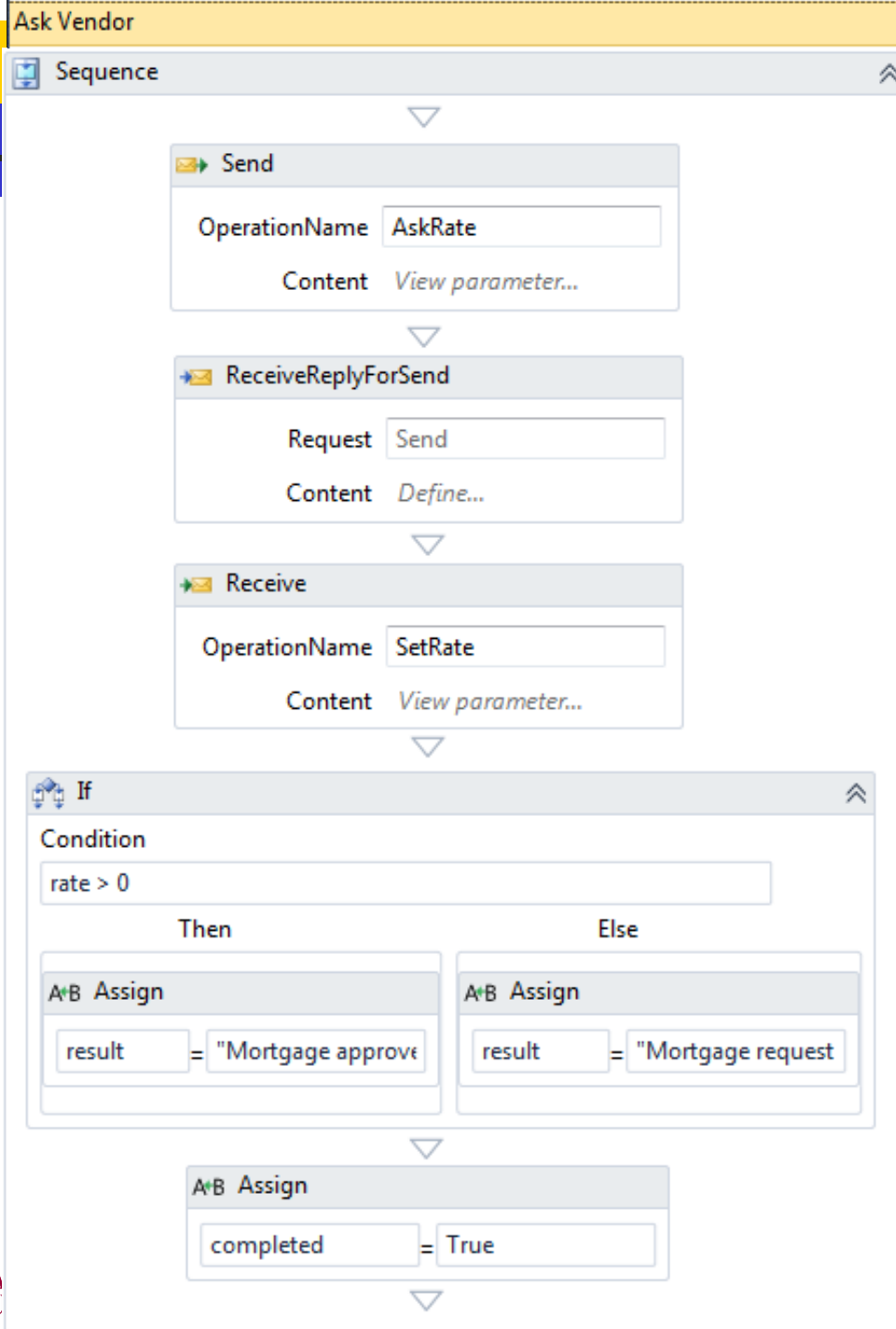
# Adding the “Initial Screening” Flowchart



# Each Outbound Arrow in the Switch Represents a Case in the Switch

51





# Service Deployed at

<http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx>

① [neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx](http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx)

## Service1 Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following

```
svcutil.exe http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx?wsdl
```

You can also access the service description as a single file:

```
http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to ca

C#

```
class Test
{
    static void Main()
    {
        MortgageClient client = new MortgageClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Note that a workflow service generates a **WSDL** interface!

# Testing the Mortgage Example

- The code can be downloaded at  
<http://msdn.microsoft.com/en-us/magazine/ff646977.aspx>
- The code is deployed at:  
The client application (TryIt Page):  
[neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/Services/](http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/Services/)  
[neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx](http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx)  
[neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor1.svc](http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor1.svc)  
[neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor2.svc](http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor2.svc)  
[neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor3.svc](http://neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor3.svc)
- Only the basic functions, including the ASP .Net Client App, the workflow-based mortgage service, and the three vendor services are deployed. The CRM and the database services are deployed.

# Summary of the Lecture

- Workflow applications and services can take different components
  - Flowchart component into a Workflow
  - Custom activities (CodeActivity) in C# and VB
  - External Web Service
- Creating Workflow Services
  - Contract-First Approach, resulting a service.svc file
  - Workflow-First Approach, resulting a service.xamlx file
- Workflow Case Studies with asynchronous service with two correlated invocations
  - Image Verifier in Workflow, persistence service
  - Mortgage Application Integration