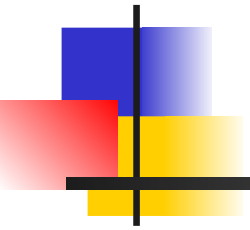
IRA A. FULTON SCHOOLS OF engineering

school of computing, informatics, & decision systems engineering

CSE 446 / 598

Software Integration and Engineering



Unit 2

Software Development by Composition and Integration

Unit 2-3

Workflow Composition and Case Studies

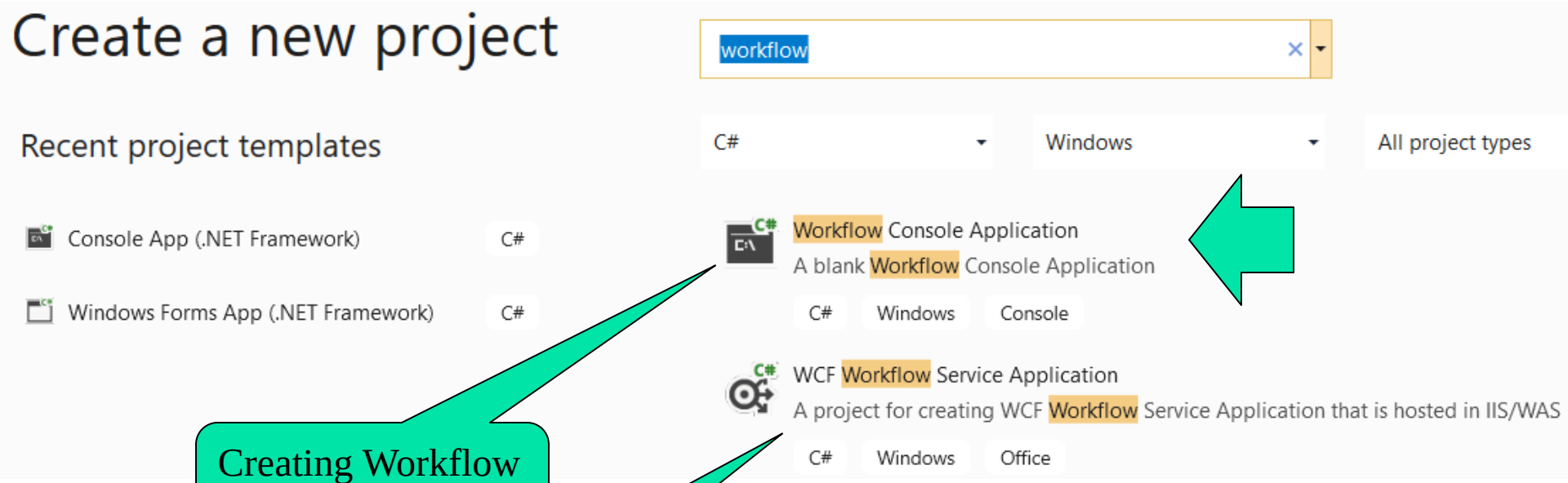
Dr. Yinong Chen

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

Creating a Workflow Console Application

In Visual Studio 2019

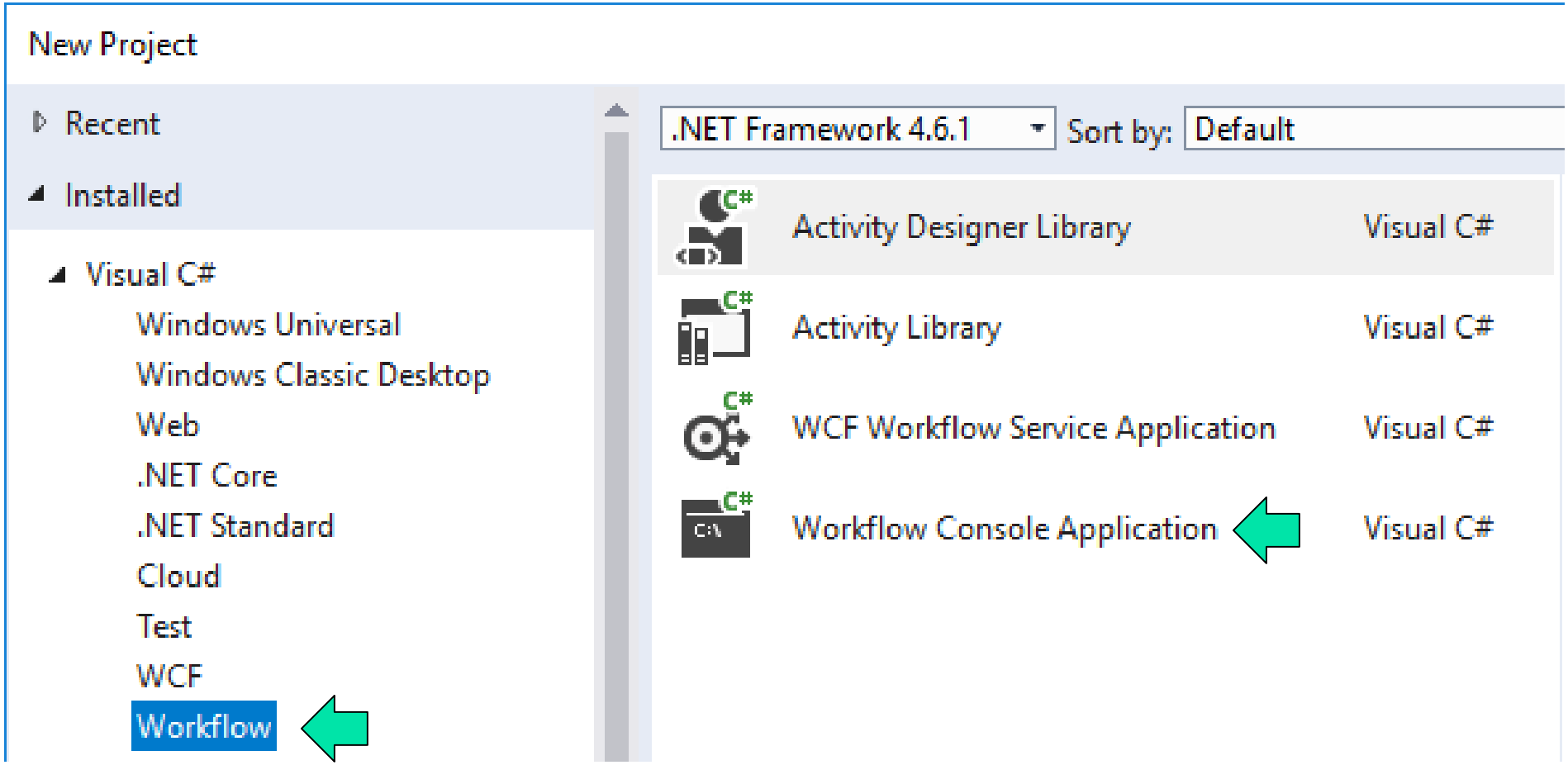


Creating Workflow Application

Creating Workflow Web Service

Creating a Workflow Console Application

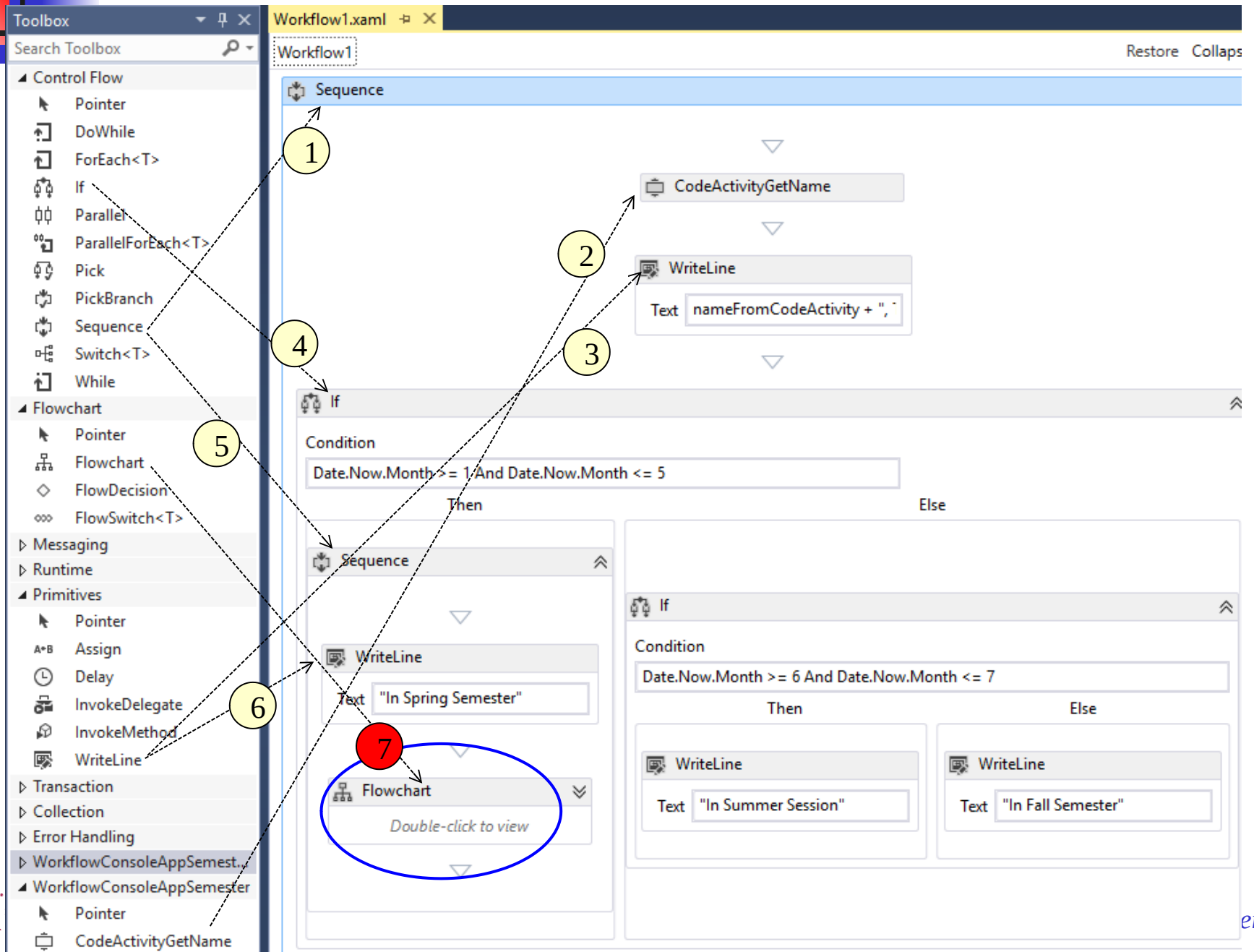
In Visual Studio 2017, Start a New Project



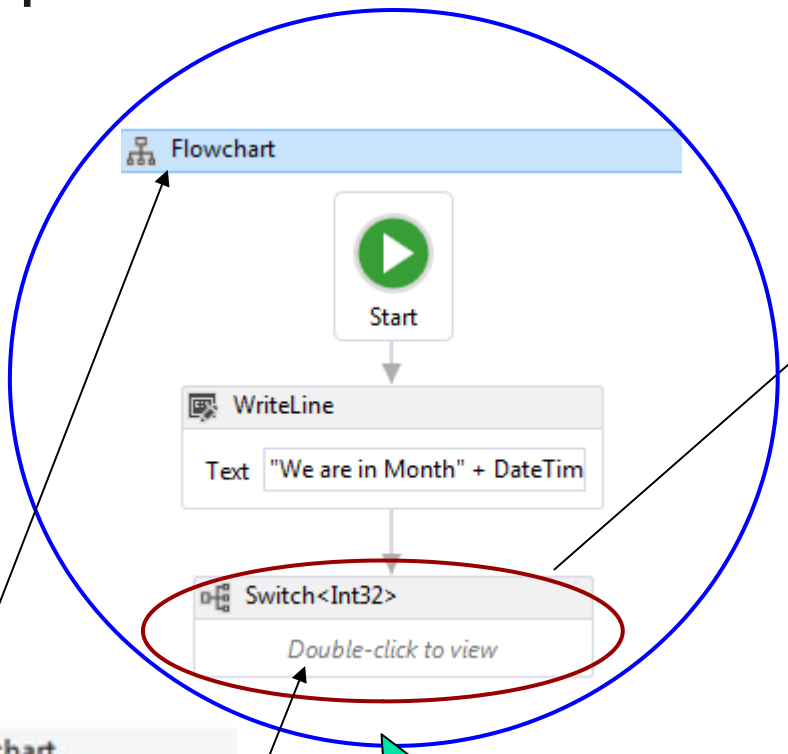
Workflow Console Application

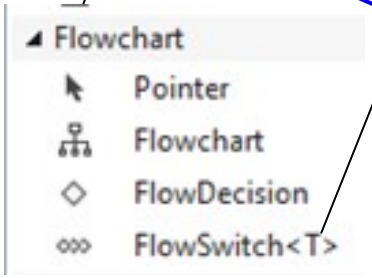
In Workflow Console Application template,
The Main() method in Program.cs is the entry point
The workflowInvoker invokes the Workflow1.xaml file

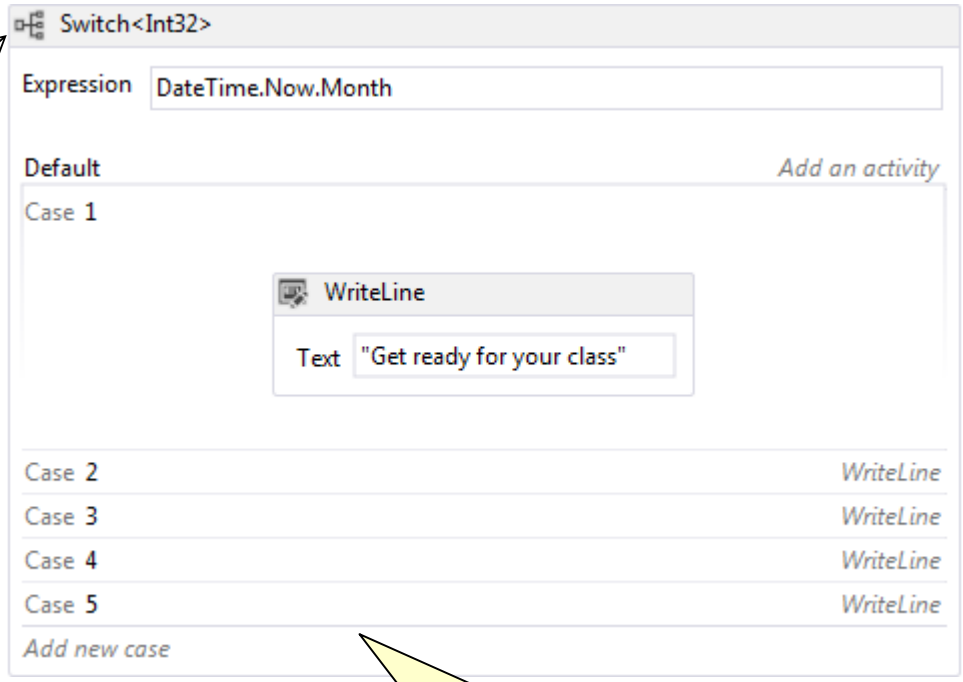
Graphic Development of Workflow.xaml



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

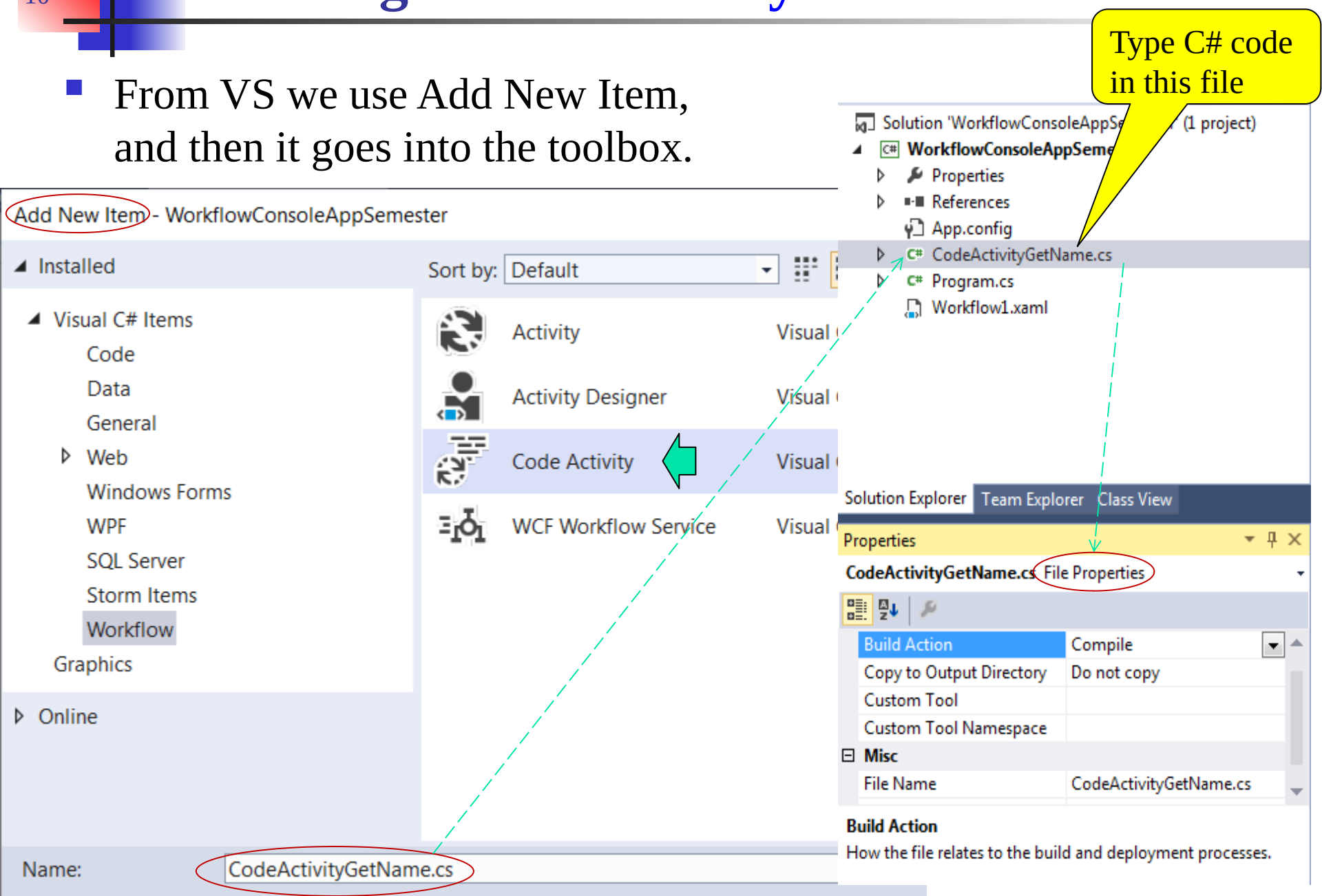
- Creating a Workflow Console Application
  - Adding a Flowchart component into a Workflow
- **Adding Other Components into a Workflow**
  - **Adding a CodeActivity into a Workflow**
  - Adding an External Web Service into a Workflow
- Creating Workflow Services
  - Contract-First Approach, resulting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

Custom Code Activities

- 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.



Template Code in the Code Activity

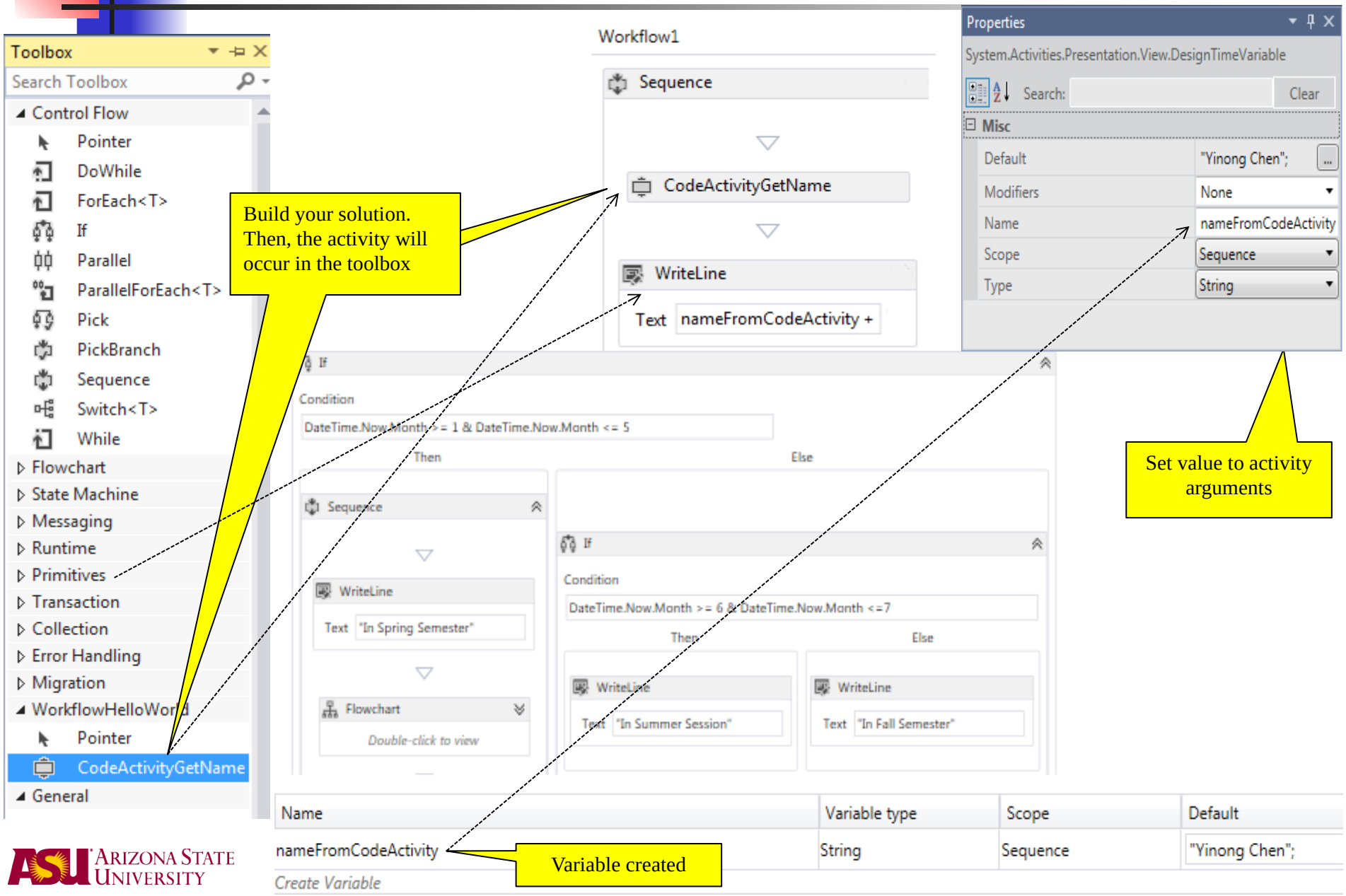
```
namespace WorkflowConsoleAppSemester {
    public sealed class CodeActivityGetName : CodeActivity {
        // Define an activity input argument of type string
        public InArgument<string> defaultName { get; set; }
        // Define an activity output argument of type string
        public OutArgument<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



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.

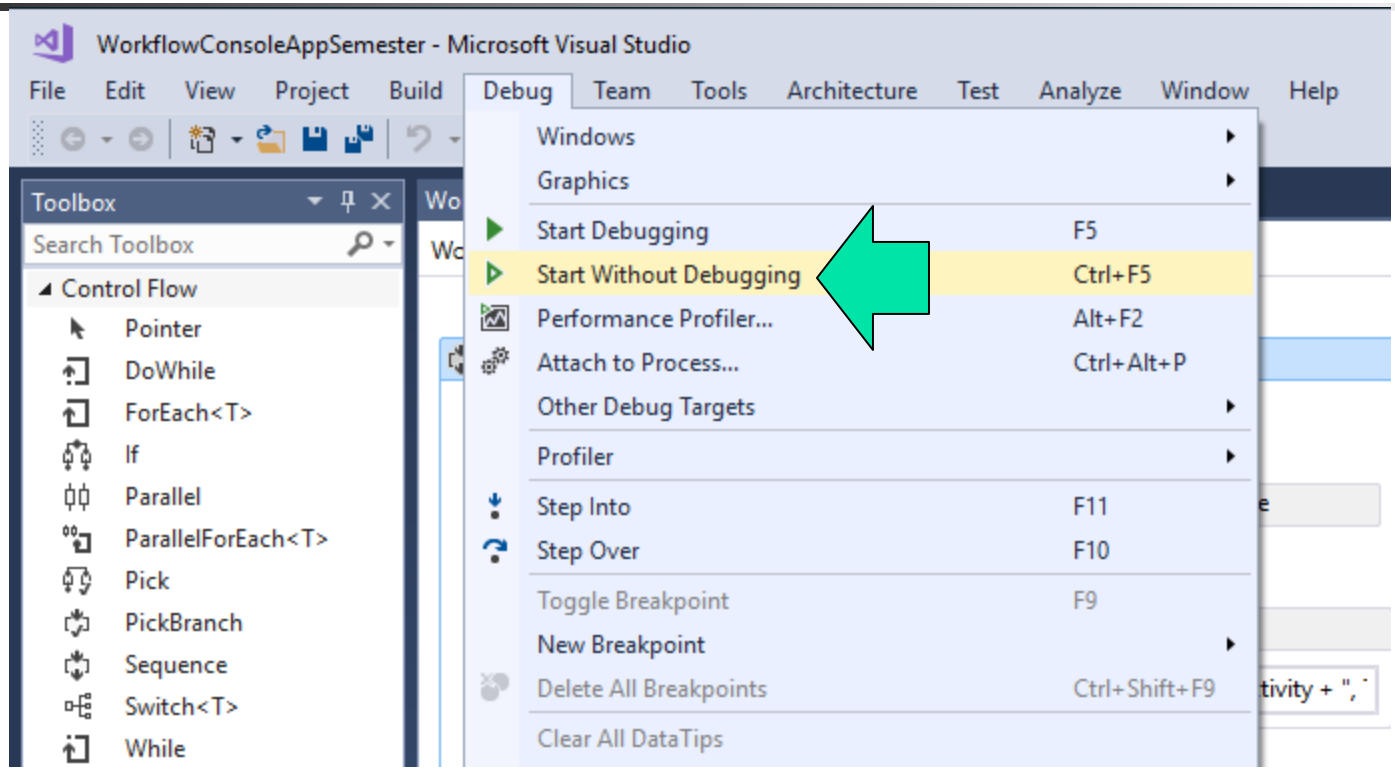
output

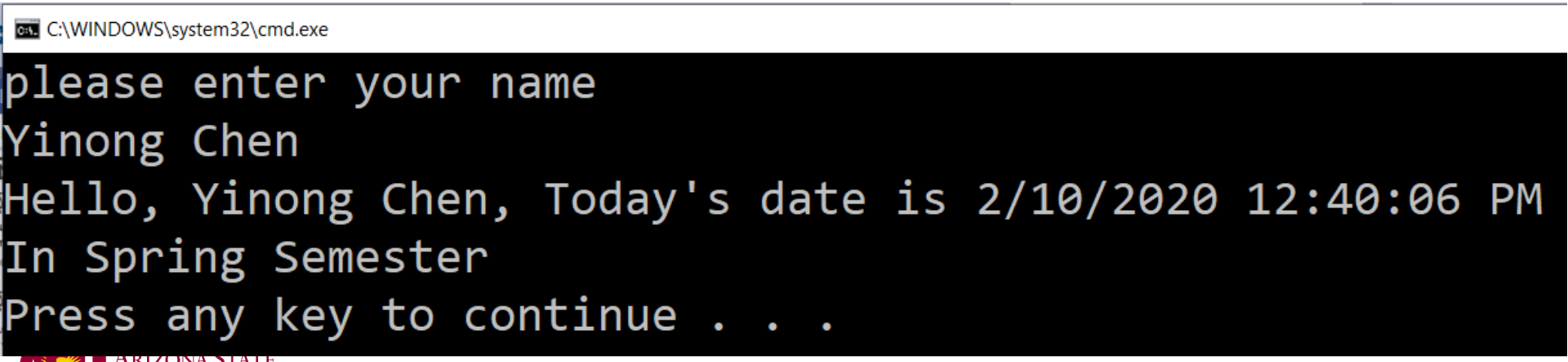
- public OutArgument<string> enteredName
- { get; set; }
  // which returns a value to the calling workflow.

Custom code

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

Test the Code



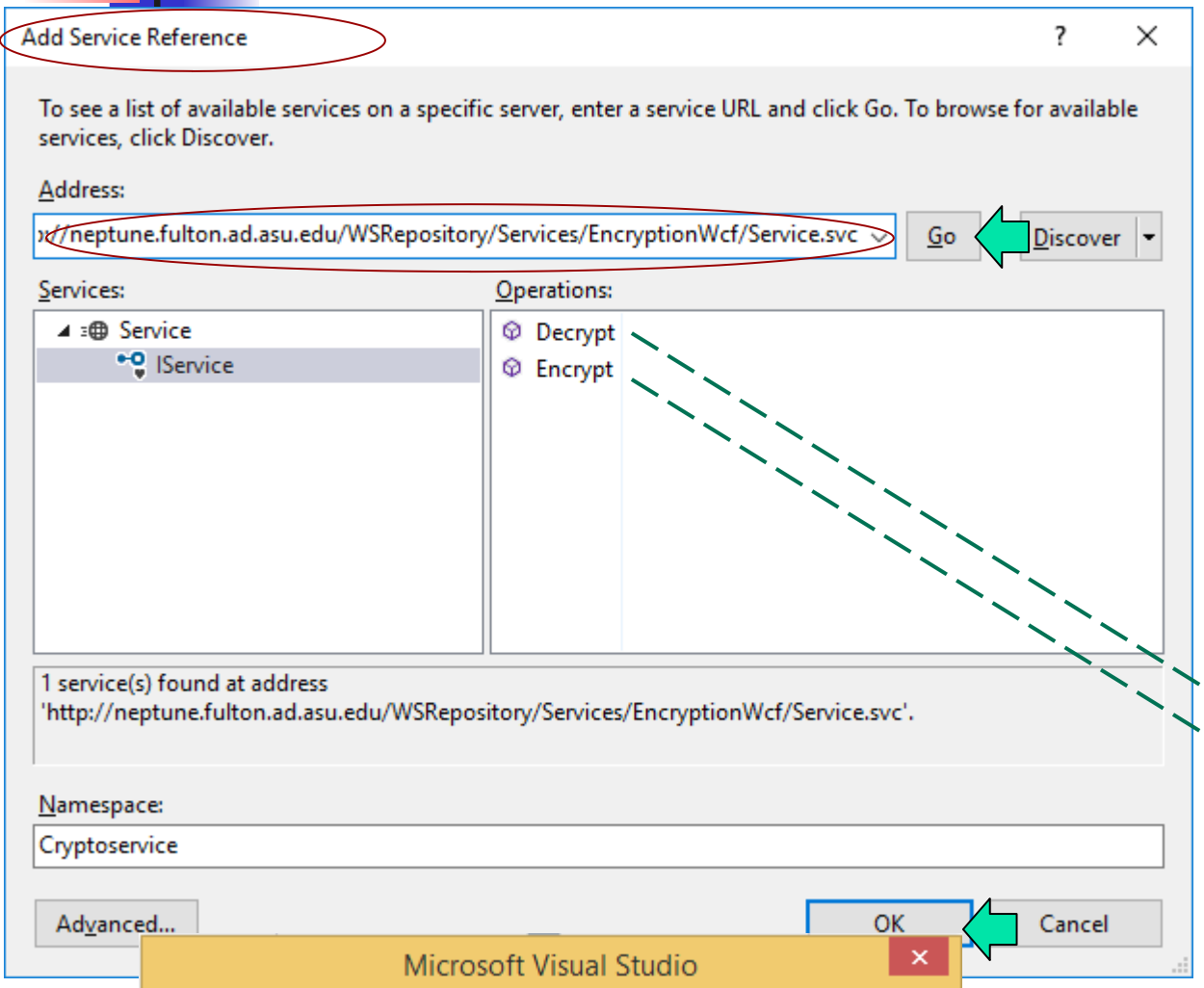


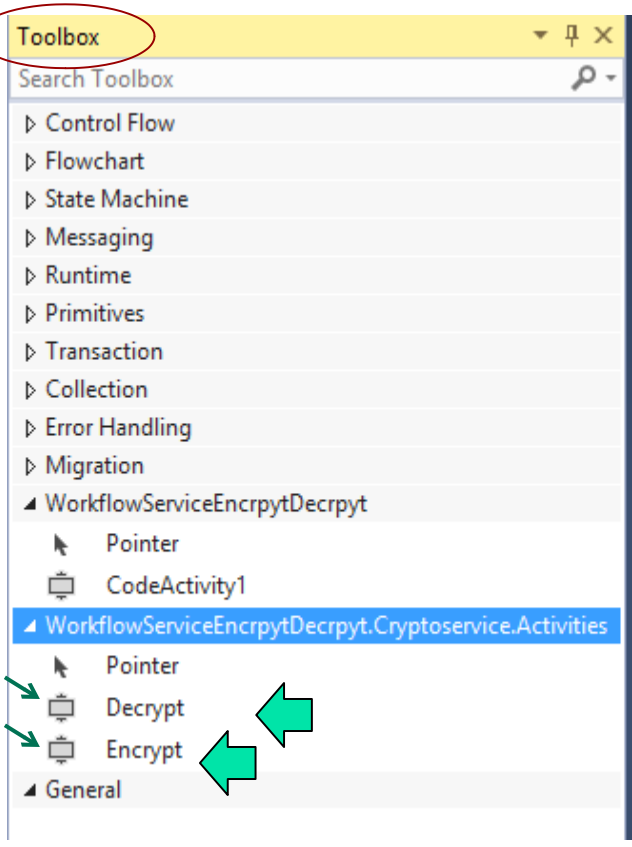
Outline of the Lecture

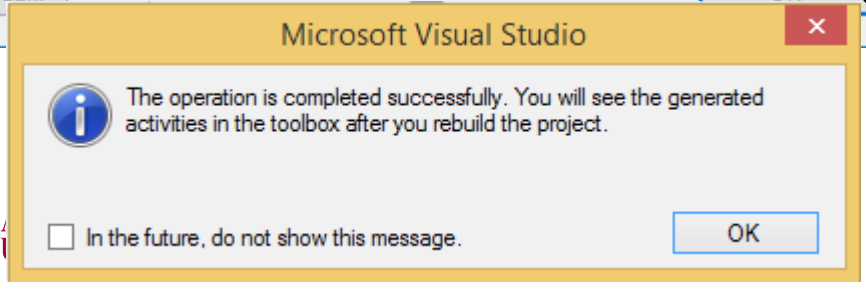
- Creating a Workflow Console Application
  - Adding a Flowchart component into a Workflow
- **Adding Other Components into a Workflow**
  - Adding a CodeActivity into a Workflow
  - **Adding an External Web Service into a Workflow**
- Creating Workflow Services
  - Contract-First Approach, resulting 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

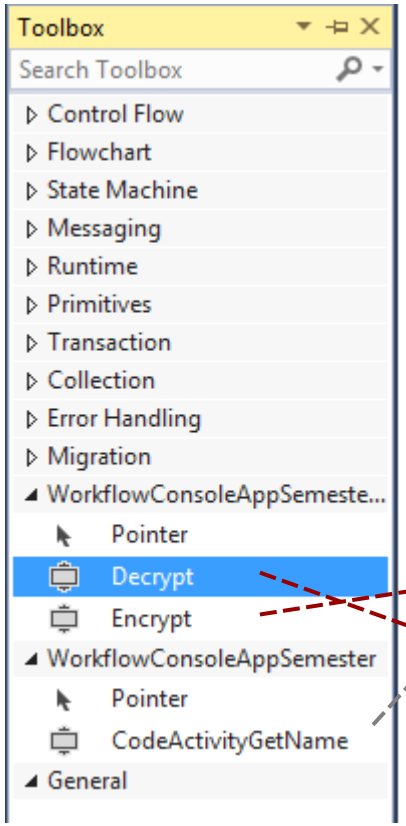
http://neptune.fulton.ad.asu.edu/WSRepository/Services/EncryptionWcf/Service.svc

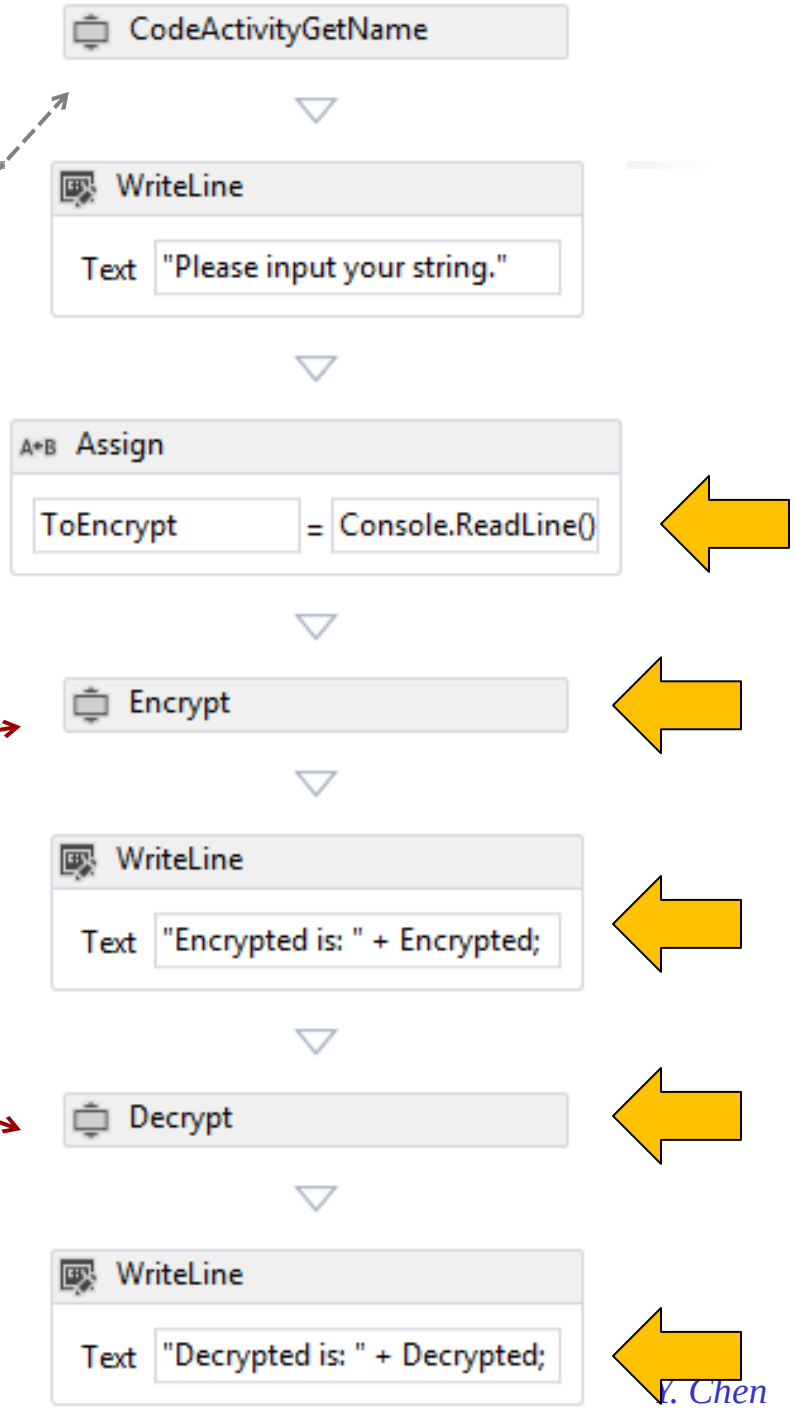






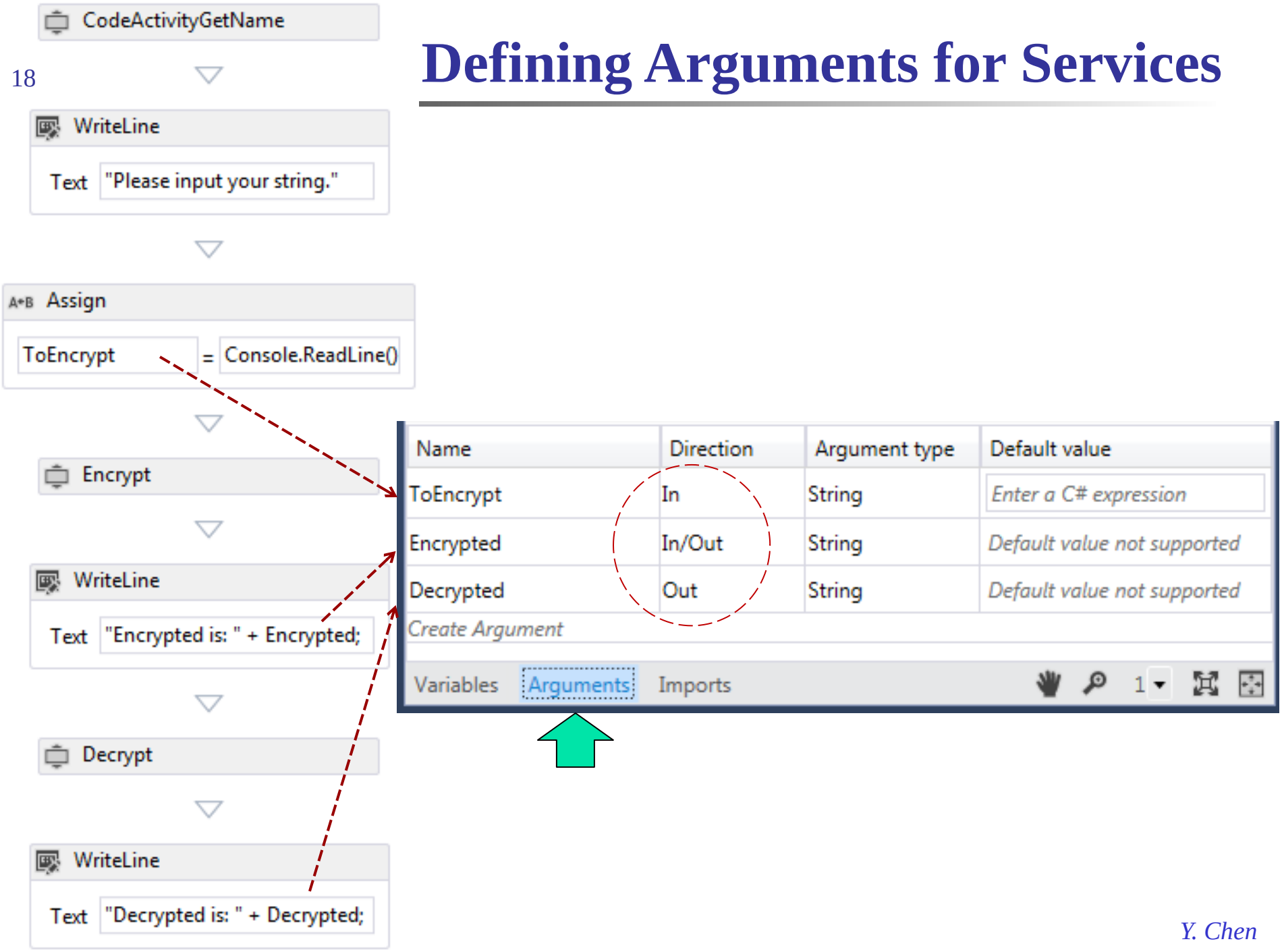
Add Services



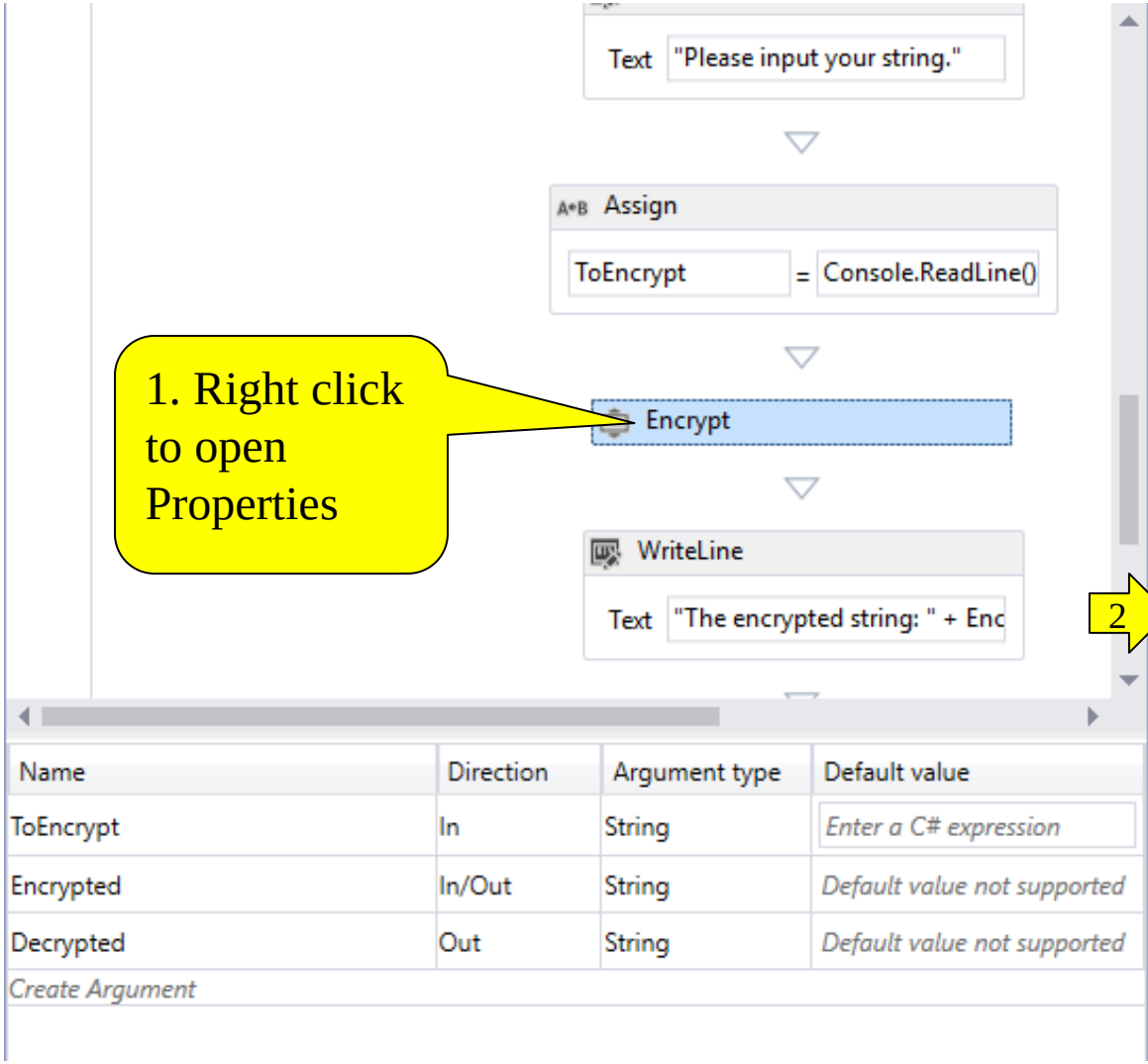


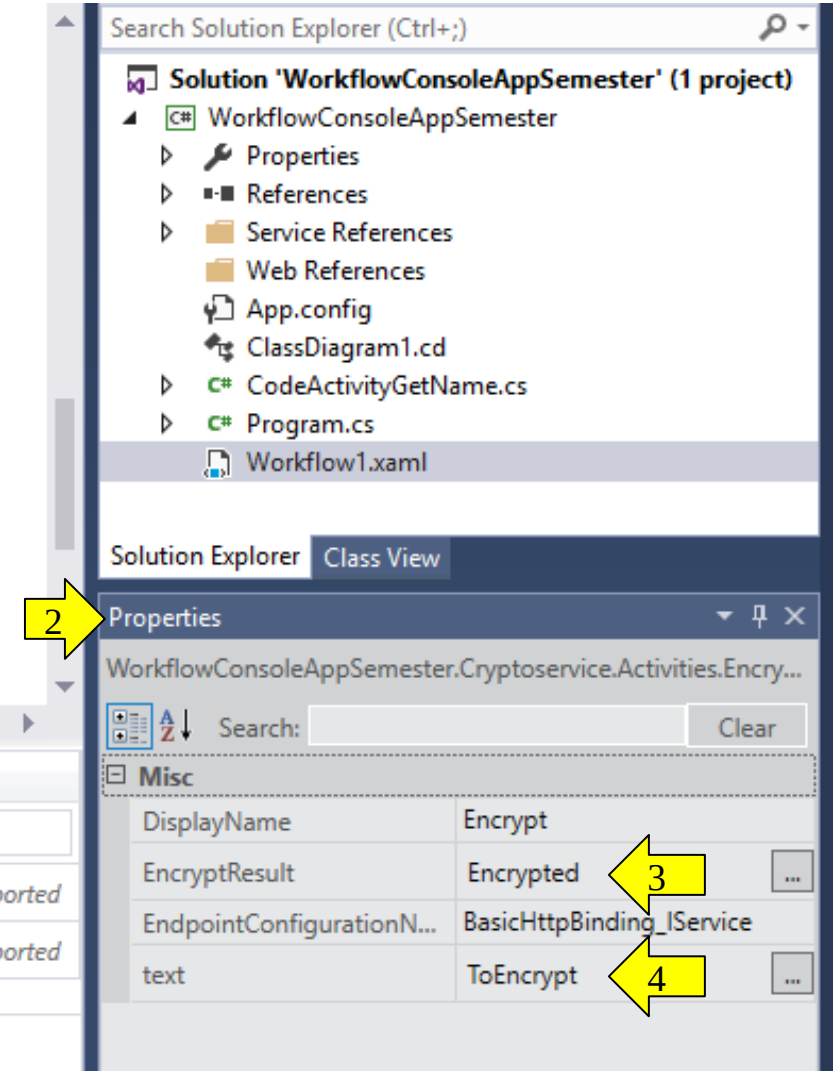
You can use a remote Web service just like a local code activity, by dragging and dropping!

Defining Arguments for Services



Associate the Arguments to the Service

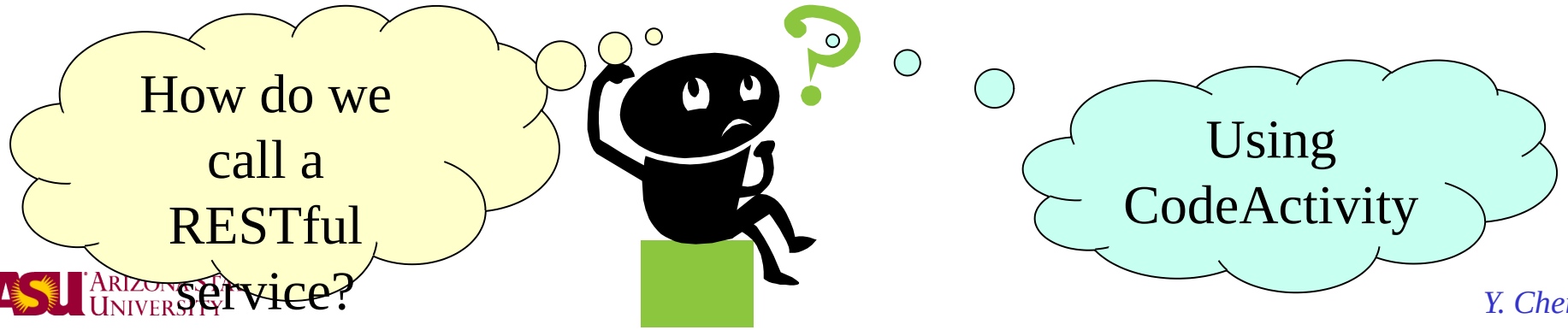




Test the Code

The input number is encrypted and the decrypted

```
Select C:\WINDOWS\system32\cmd.exe
Please input your string.
Hello ASU
The encrypted string: o5HM6k8fIBgjHtE1knGScQ==
Decrypted is: Hello ASU
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-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 Applications and Services

- 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.

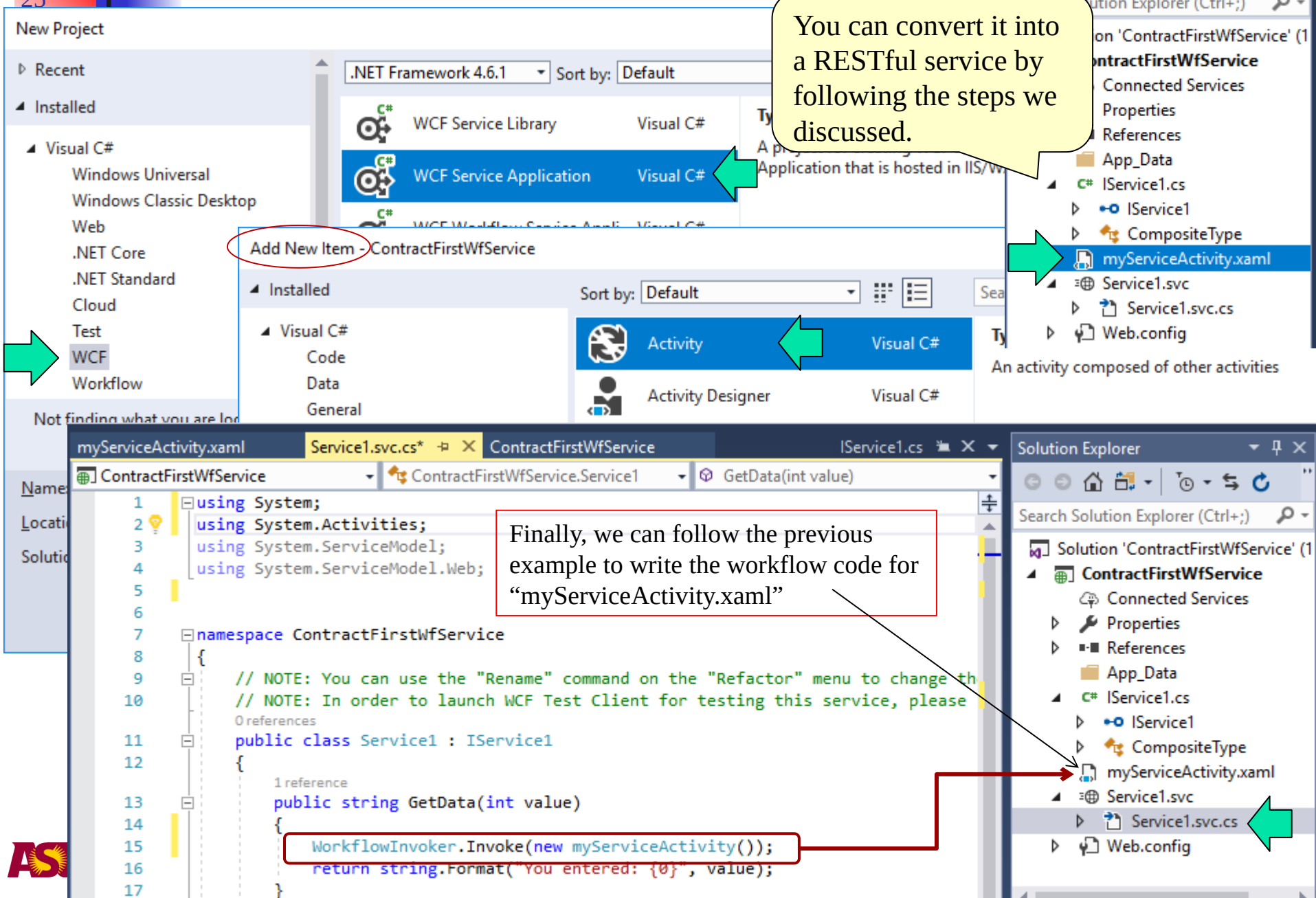
Two Approaches of Workflow Services

- 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

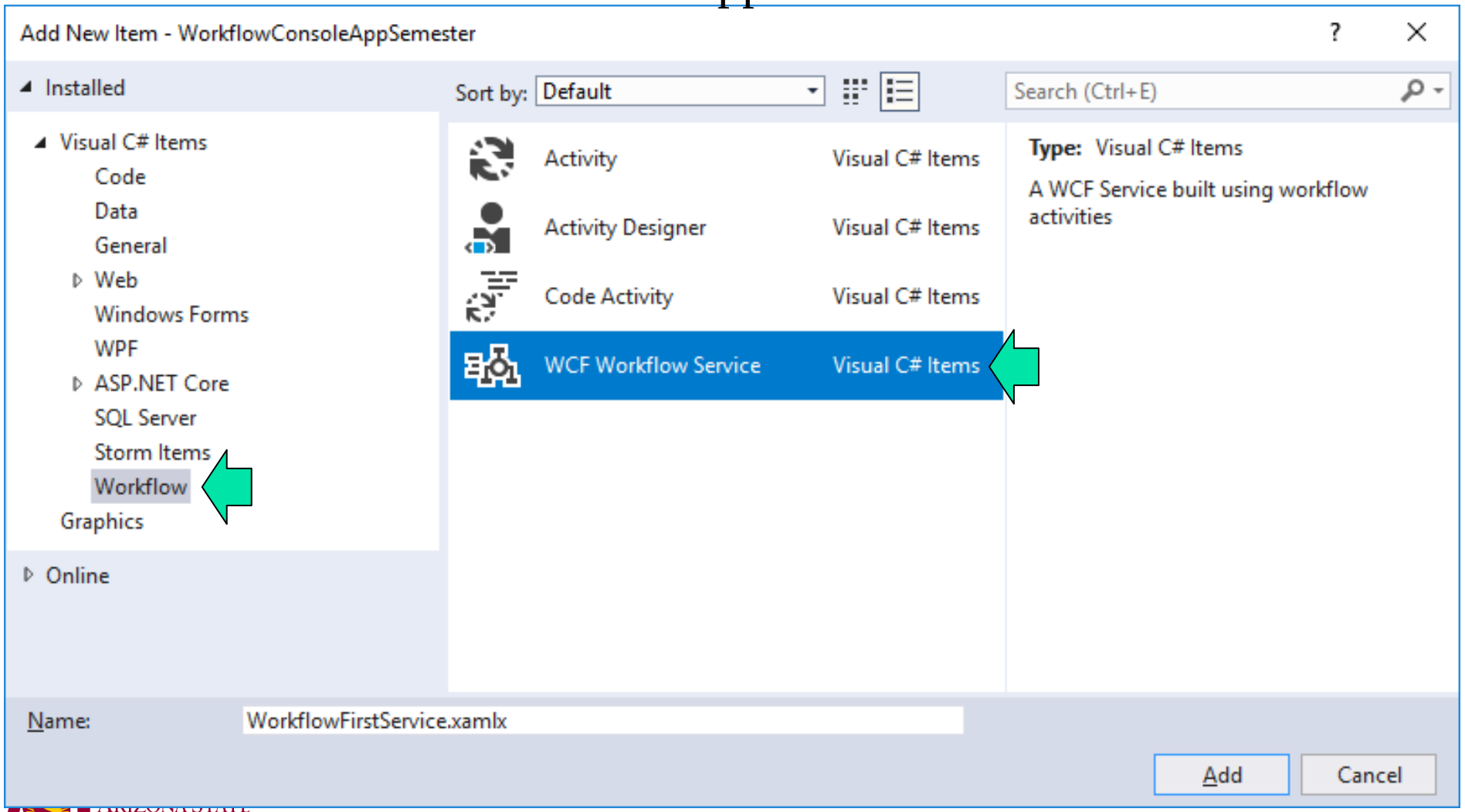


Two Approaches of Workflow Services

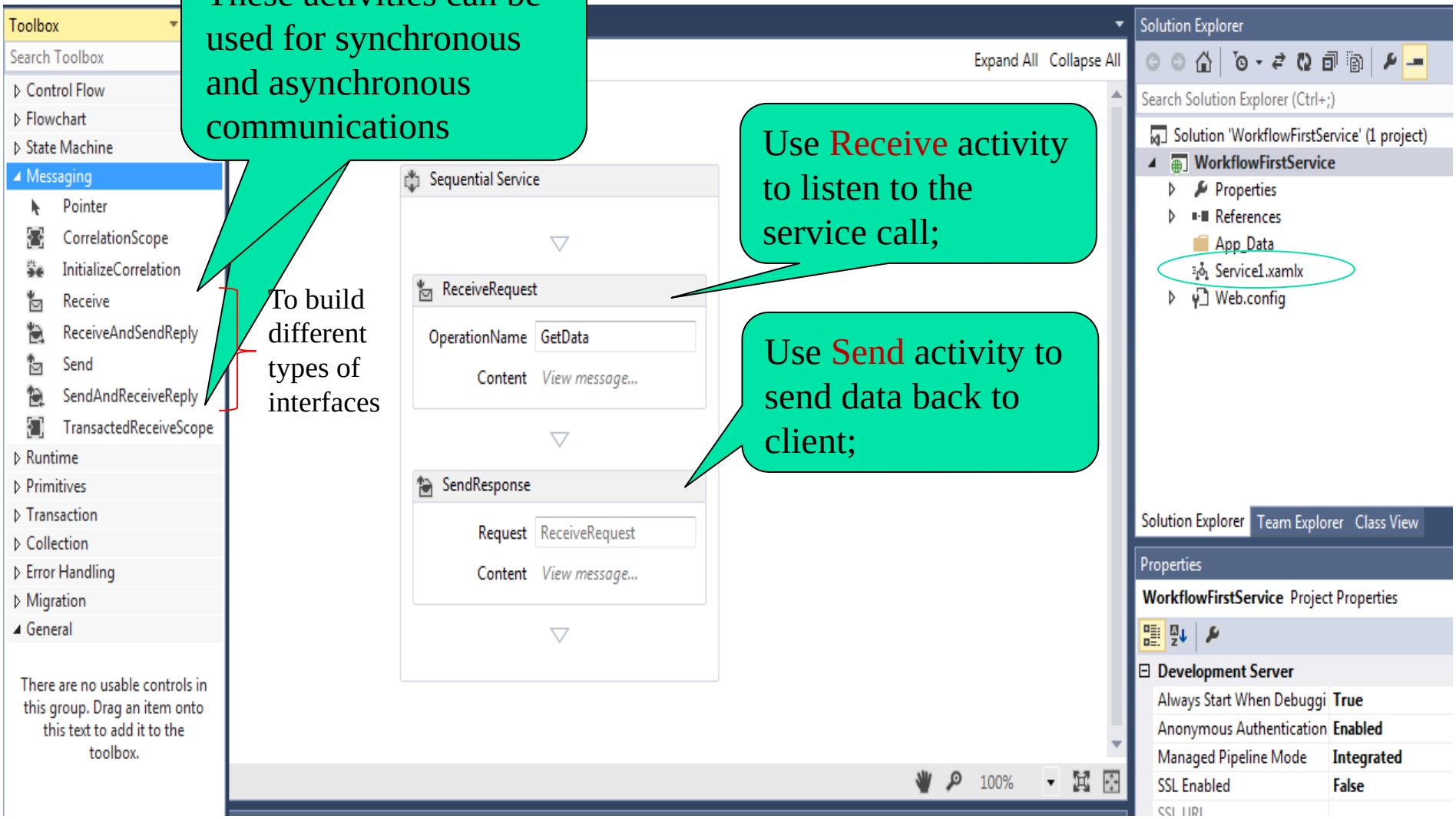
- Contract-First Service Development:
  - Use the "WCF Service Application" template to start an ordinary WCF service project.
  - The service will be exposed as a .svc service
  - Add a workflow activity in the project to perform the high-level code organization work.
  - Create a synchronous service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



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?

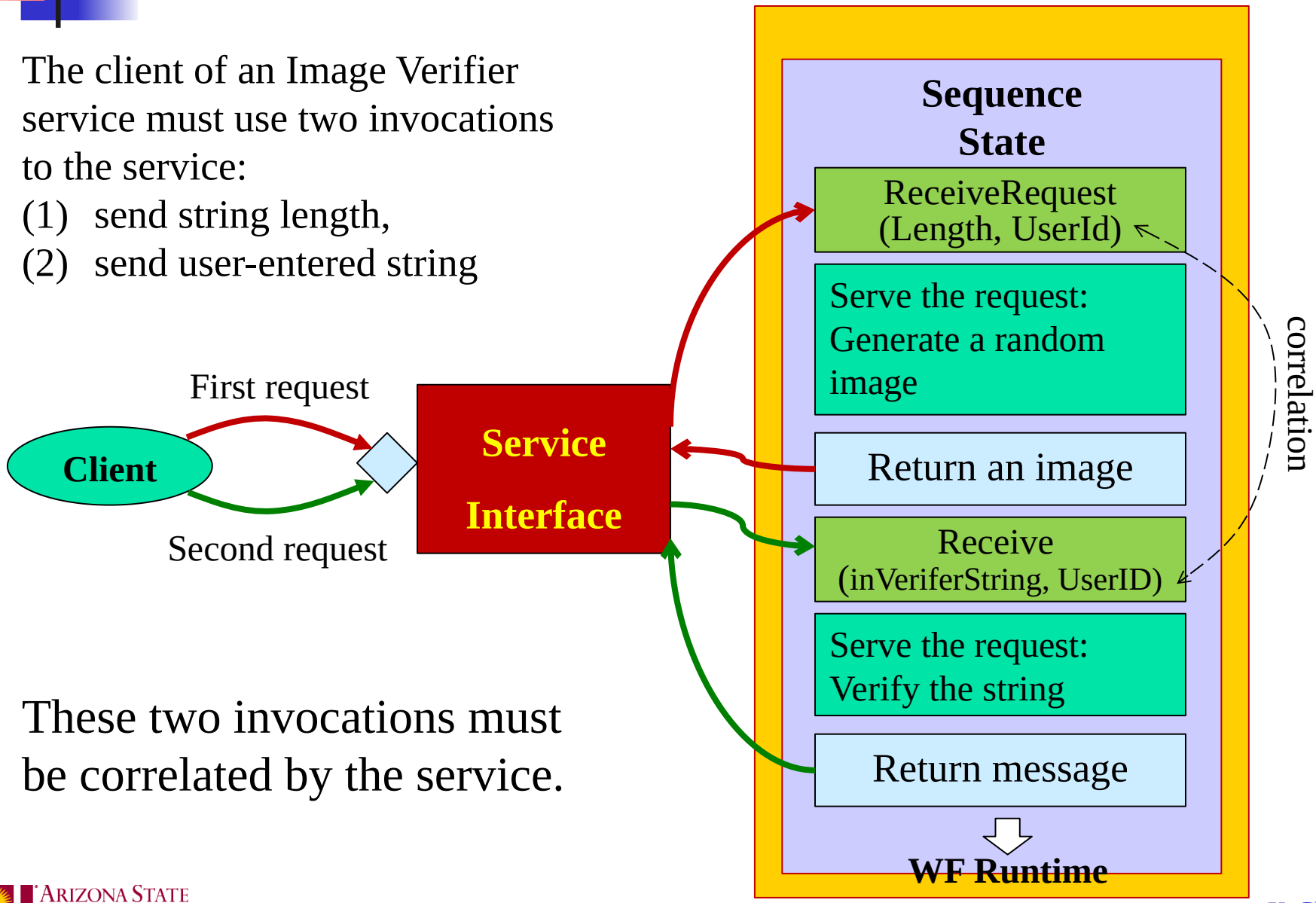
- A persistence service requires more than one access.
- It is similar to asynchronous service, where two calls are required, but it is different in the ways that a persistence service
  - may allow the second call to come after a long time, and thus it must save the partially finished service in a data 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:

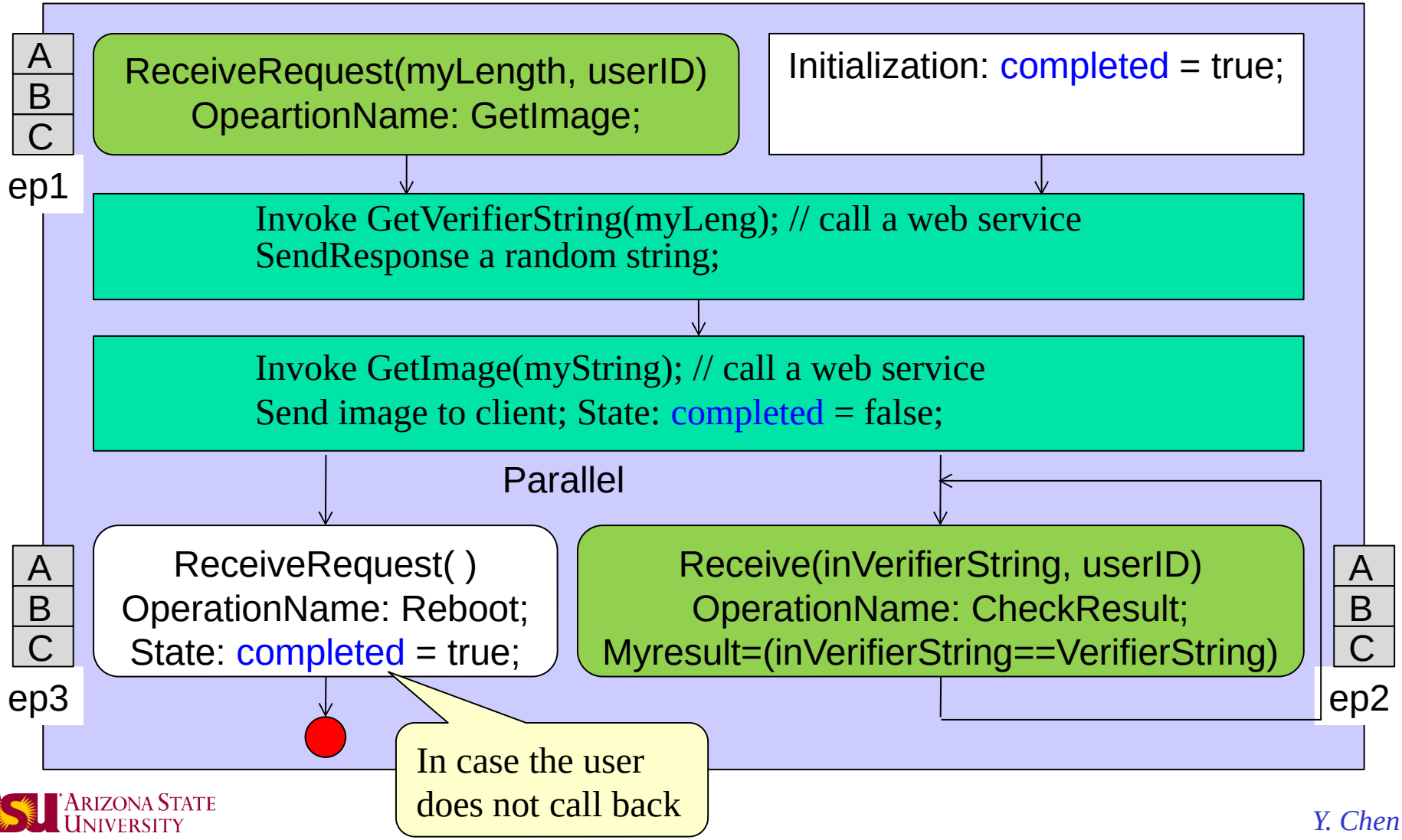
(1) send string length,

(2) send user-entered string

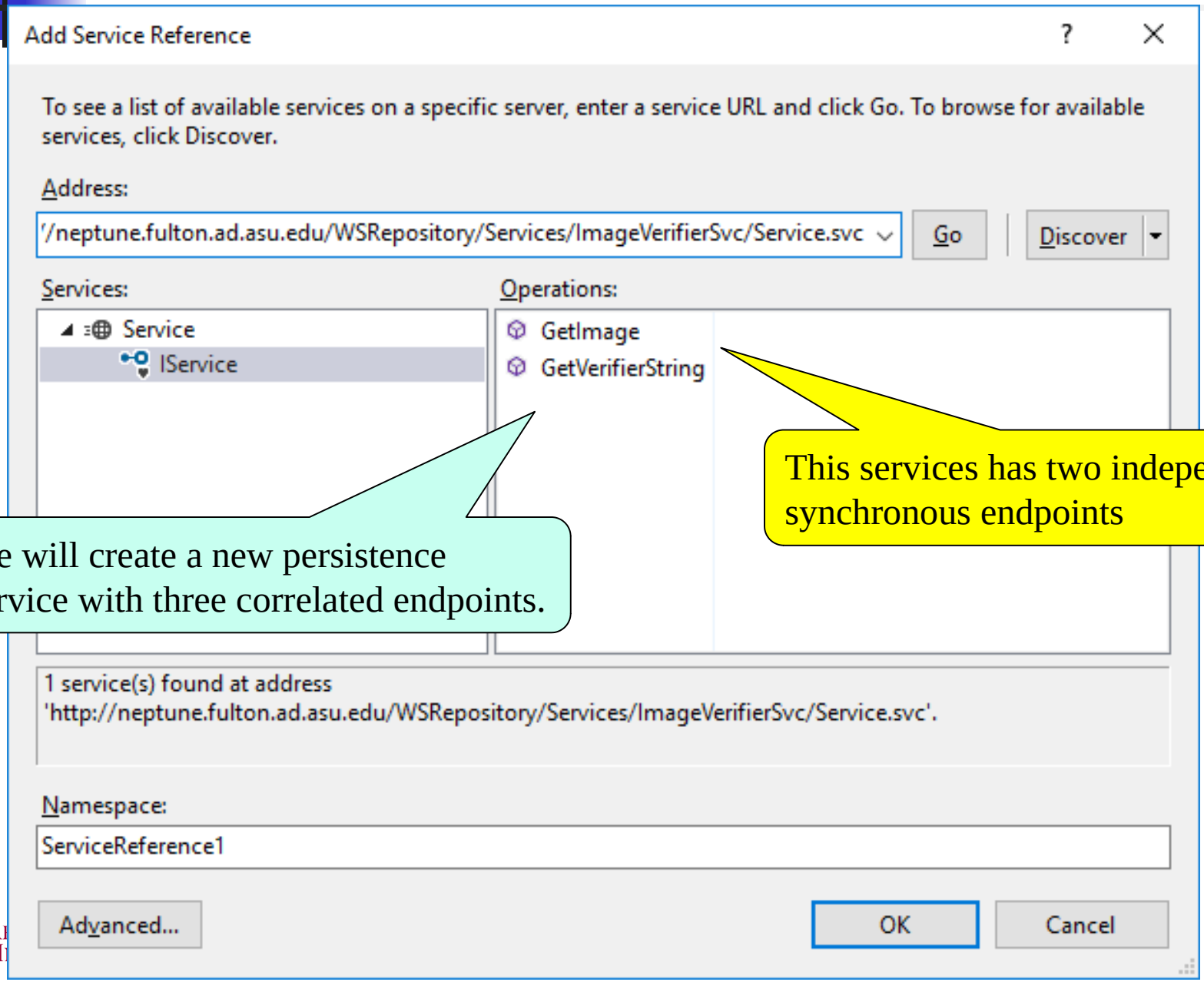


These two invocations must be correlated by the service.

Workflow of Image Verifier Service as a Finite State Machine, with a variable "completed"



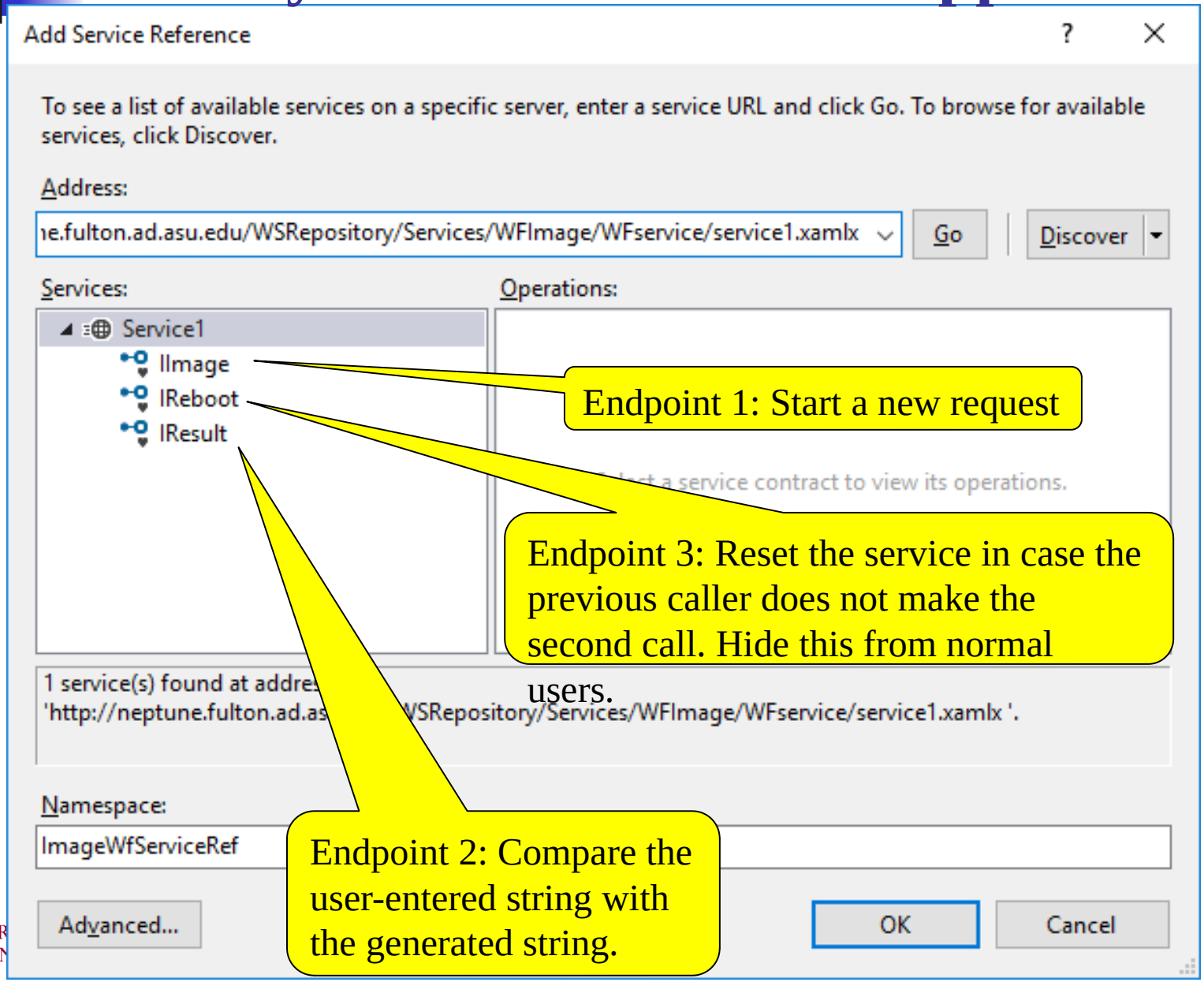
Converting SVC Image Service to Persistence Service



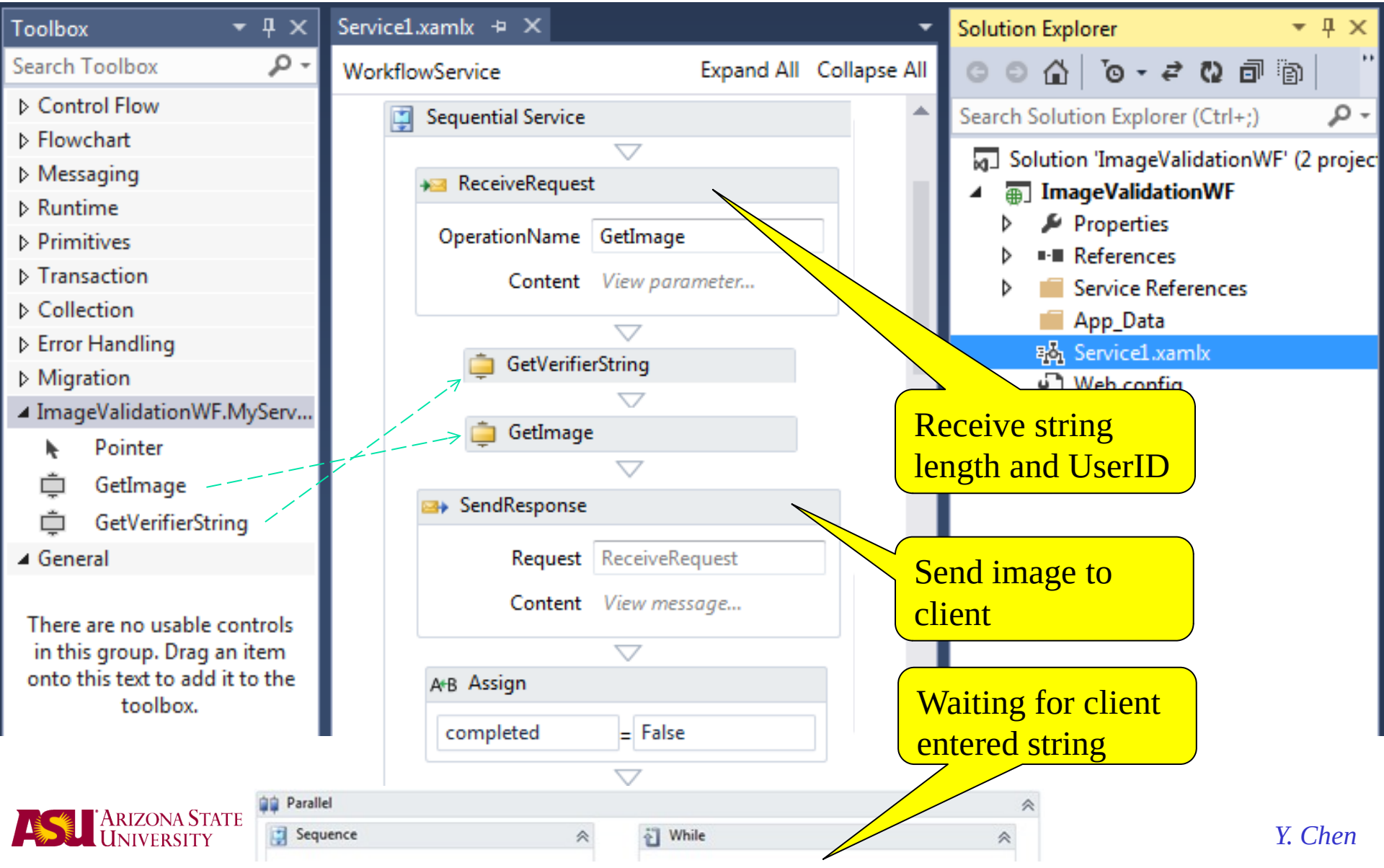
We will create a new persistence service with three correlated endpoints.

This services has two independent synchronous endpoints

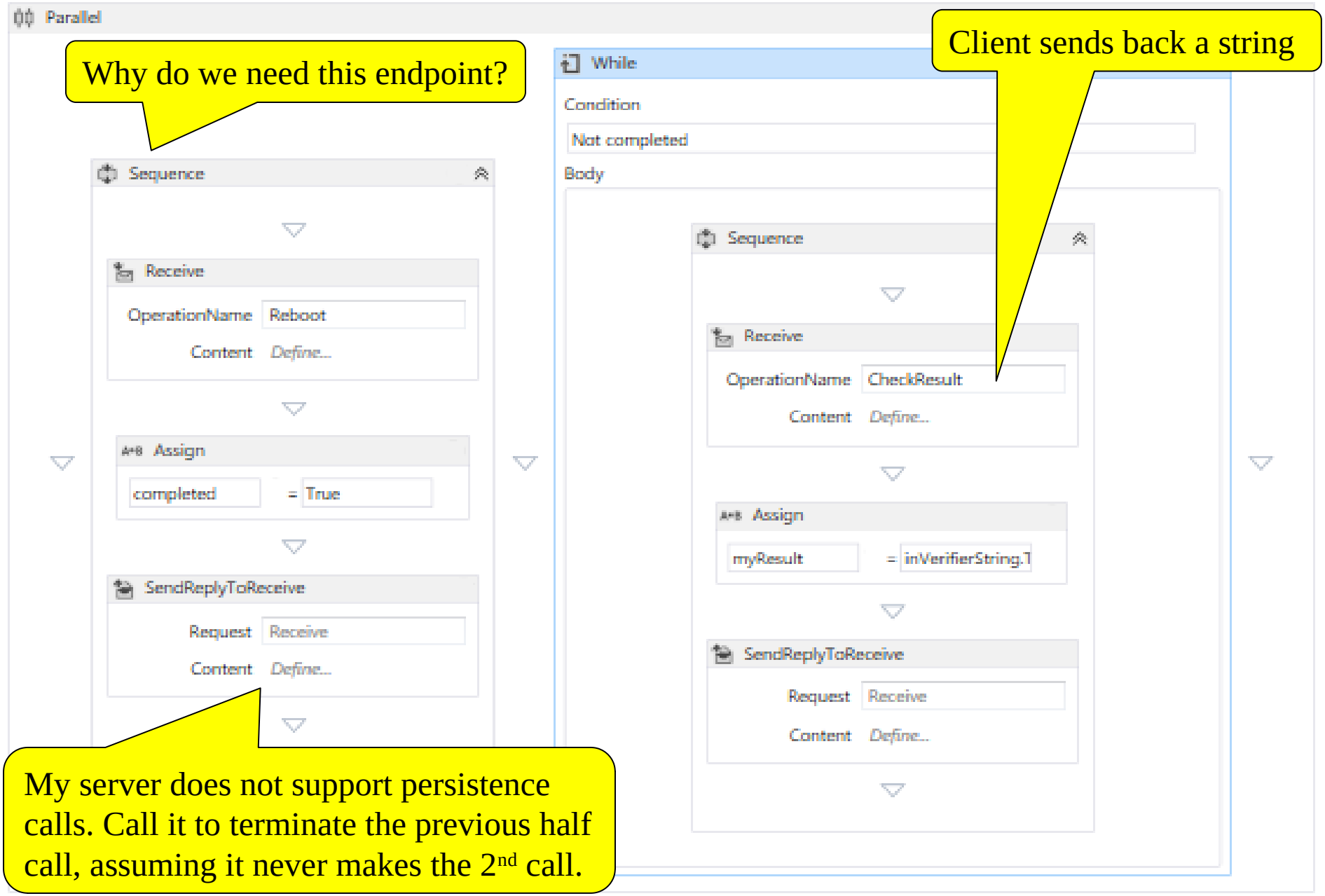
A New Two-Call Service that you can Add to other applications



Workflow of Image Verifier Service
After .svc image service is added

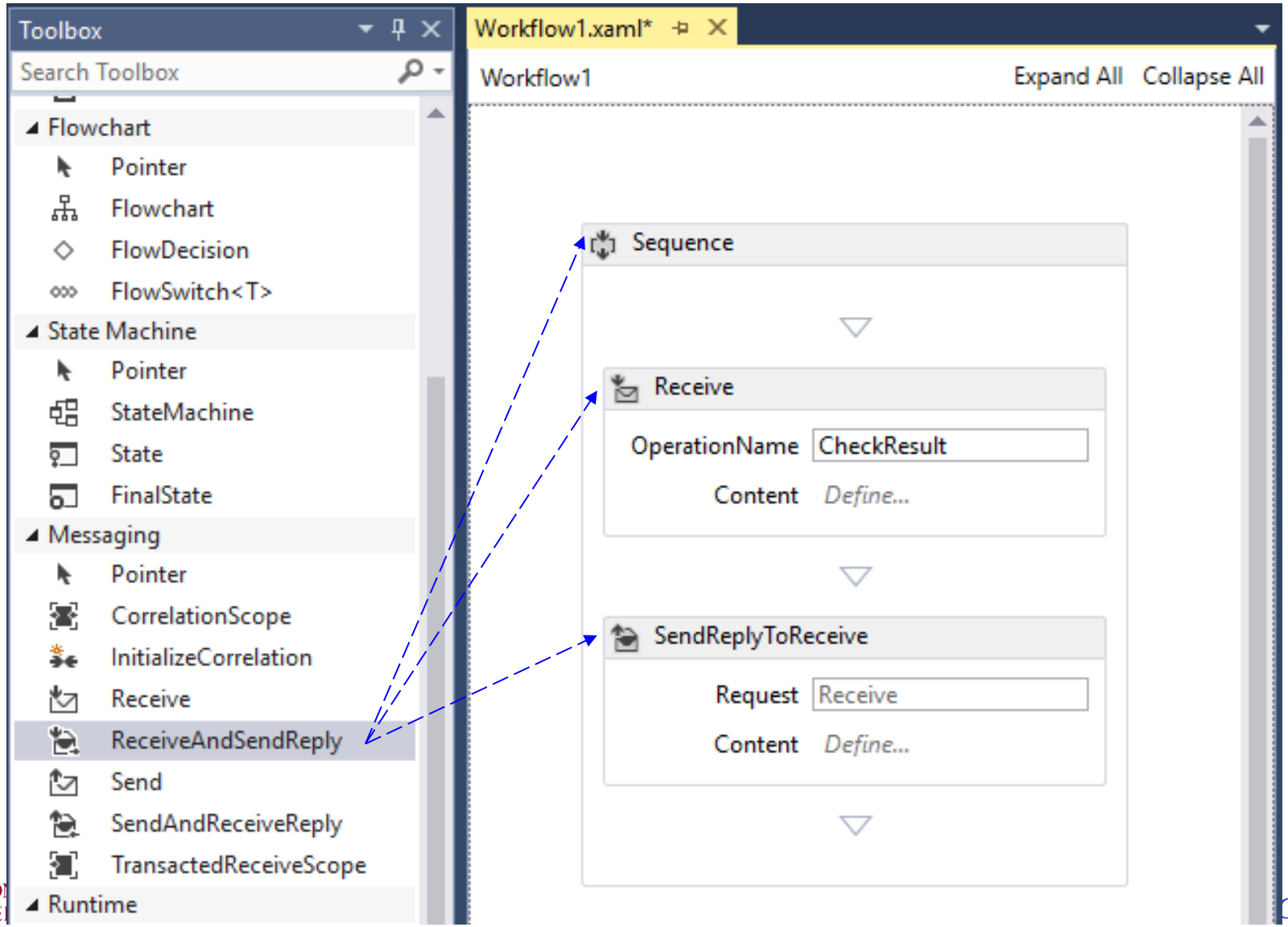


The Main Part of the Workflow

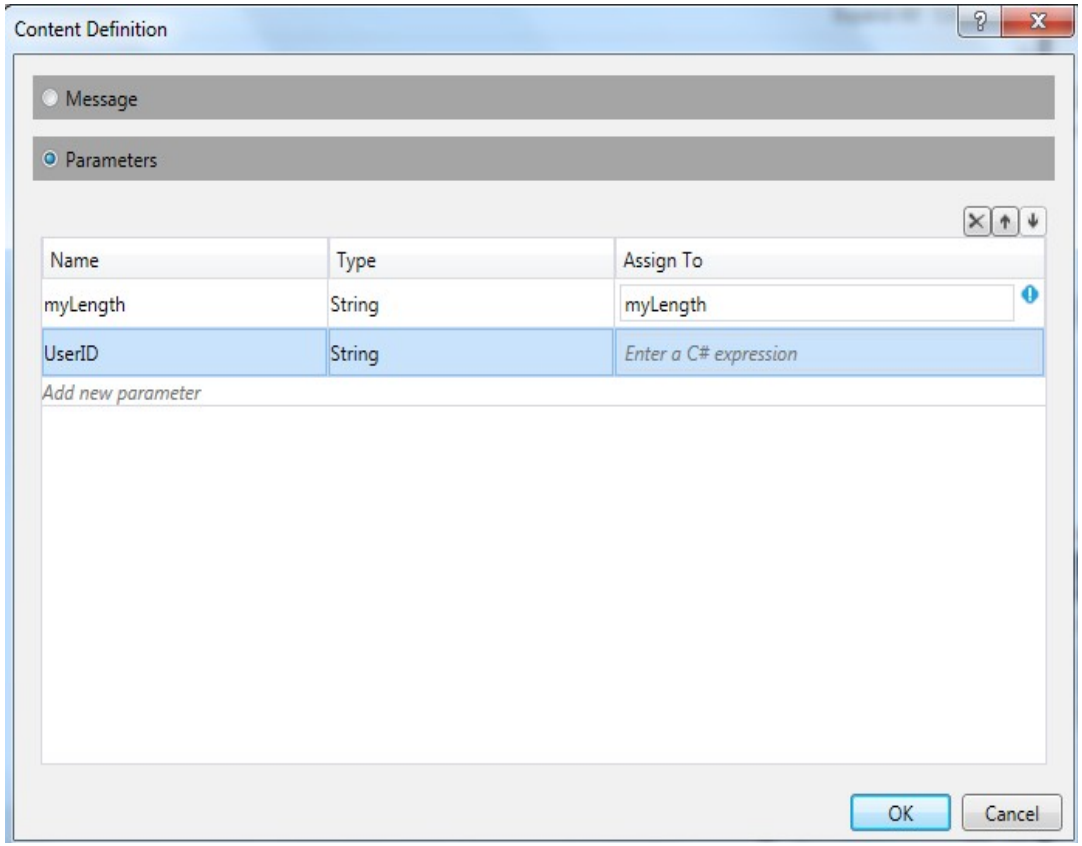


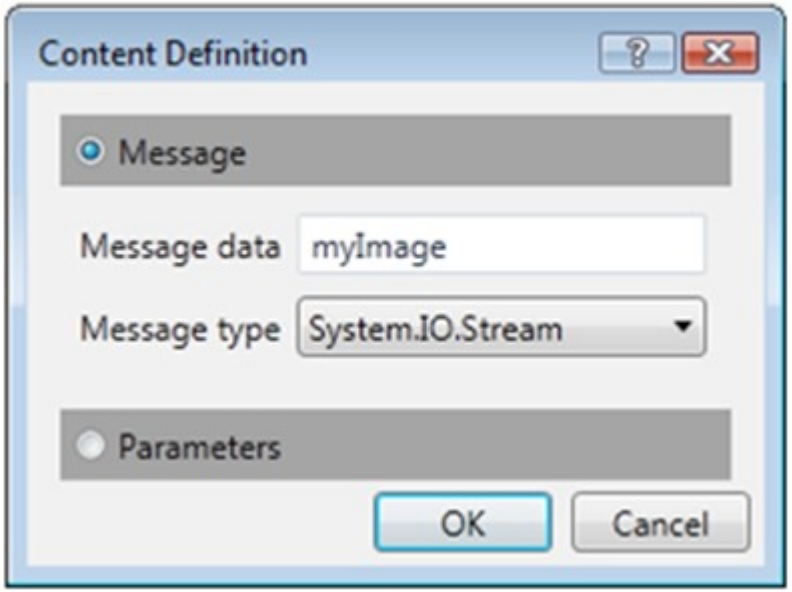
ReceiveAndSendReply Composite Activity

- It consists of Sequence, Receive and SendReplyToReceive



"View parameter…" and "View message…"





Variables Defined for the Workflow

Two calls must be associated through a handle

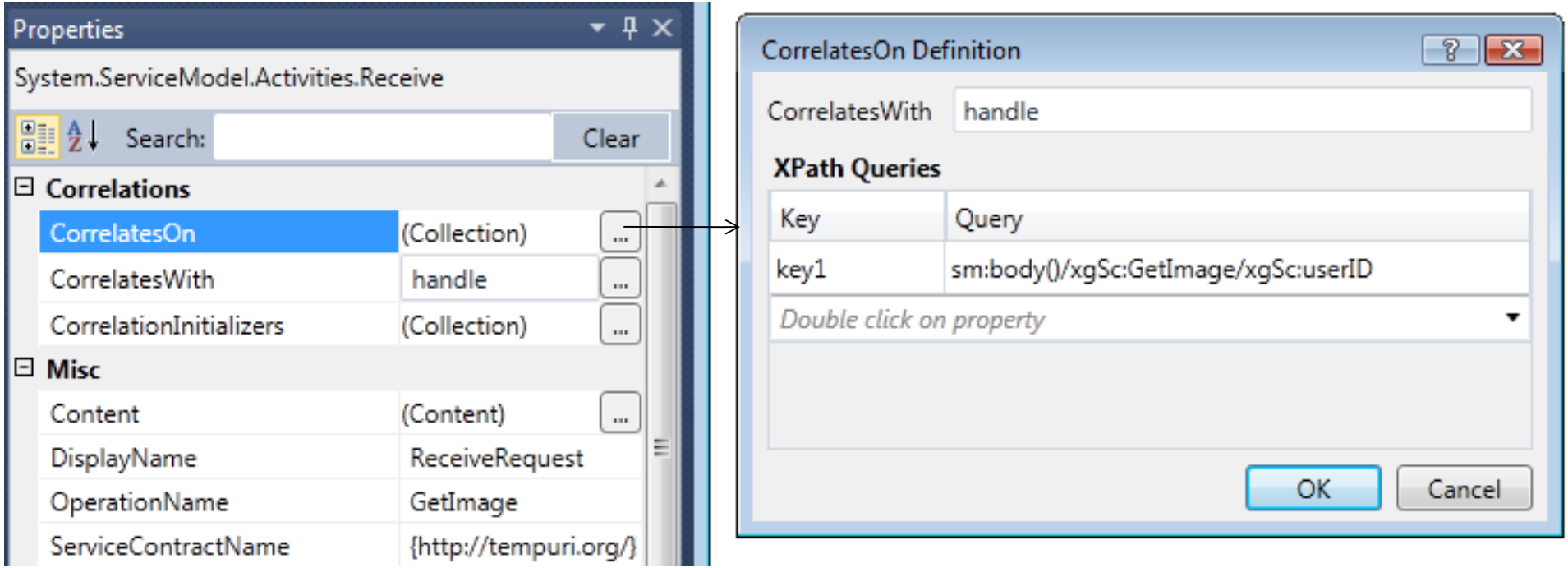
Name	Variable type	Scope	Default
completed	Boolean	Sequence	true
handle	CorrelationHandle	Sequence	*Handle cannot be initialized*
myImage	Stream	Sequence	*Enter a C# expression*
myLength	String	Sequence	"4"
myResult	Boolean	Sequence	*Enter a C# expression*
userID	String	Sequence	*Enter a C# expression*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

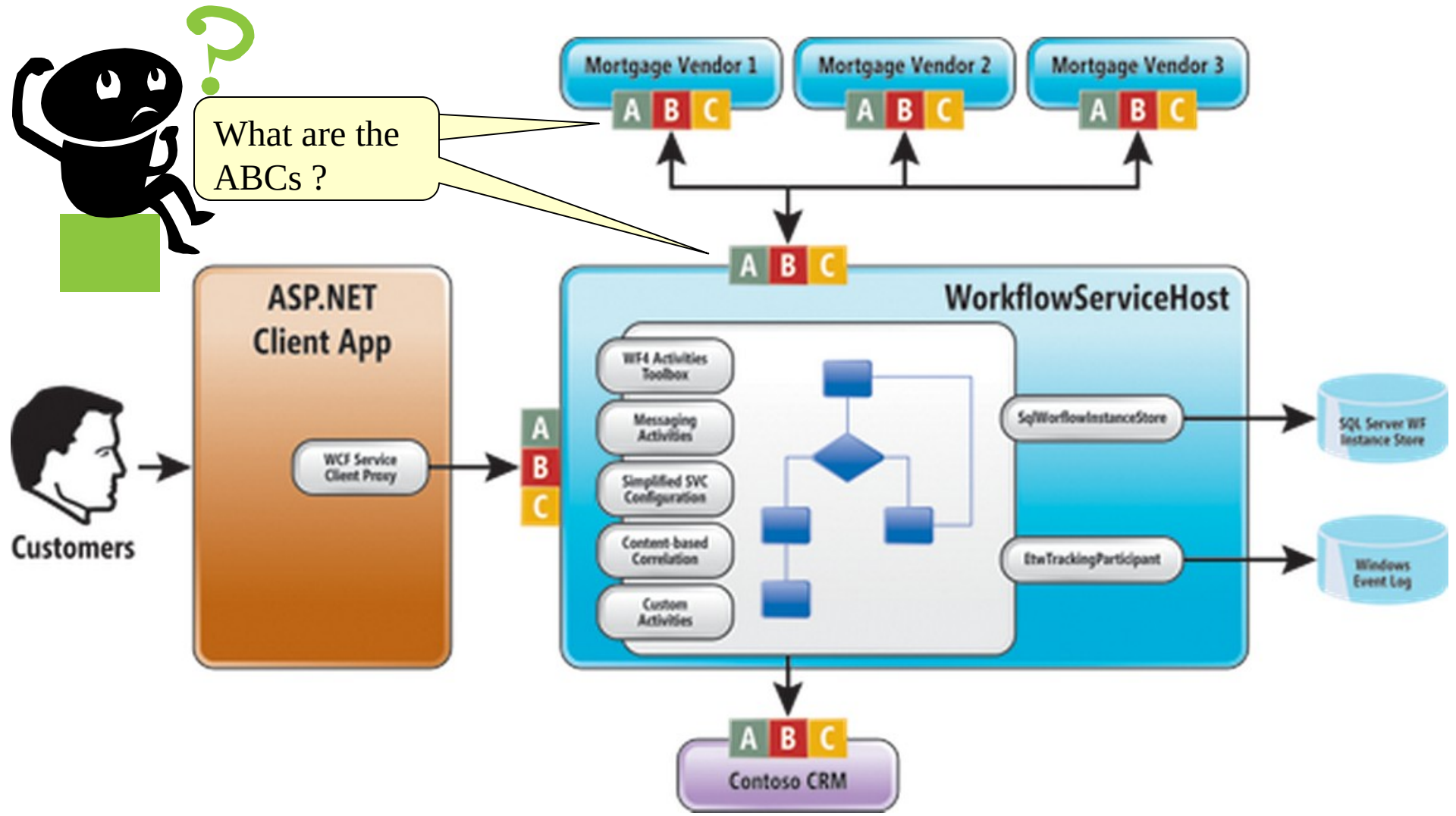
- Creating a Workflow Console Application
- Adding Other Components into a Workflow
- Creating Workflow Services
- Case Studies of Persistence Service:
  - Image Verifier in Workflow
    - Text Section 7.4 with full detail
    - Service in ASU Service Repository
  - **Mortgage Application Integration**
    - **Using Workflow-First approach**
    - **Code downloadable at:**
      http://msdn.microsoft.com/en-us/magazine/ff646977.aspx

Application Requirement (Scenarios)

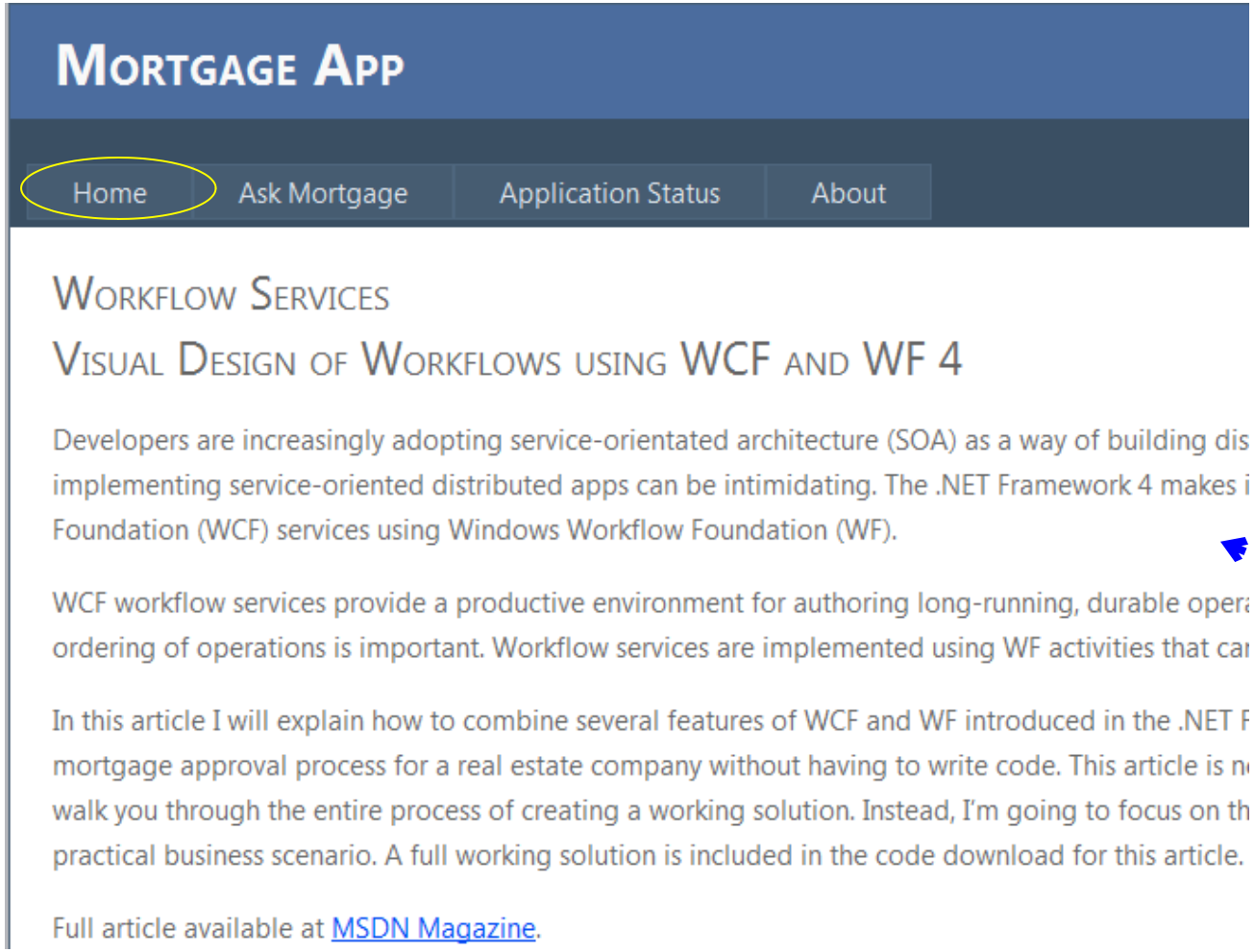
- Contoso Housing is a real estate company that sells houses.
- To provide better customer service and an end-to-end buying experience, Contoso partners with three mortgage companies that assist potential customers with their mortgage needs.
- Each mortgage company offers different interest rates.
- 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)

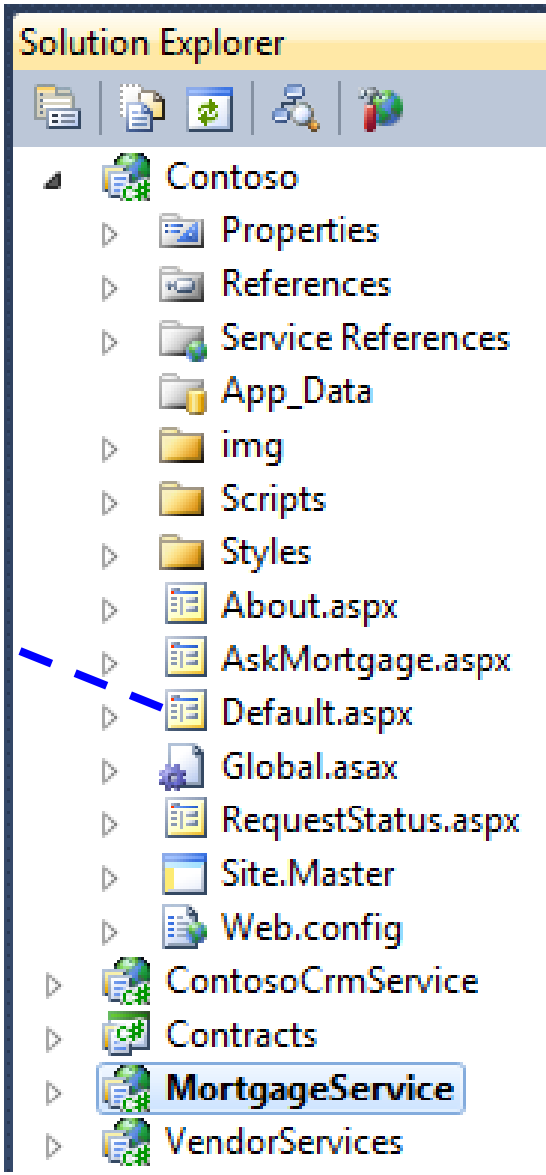
Overview of the Application and Service

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



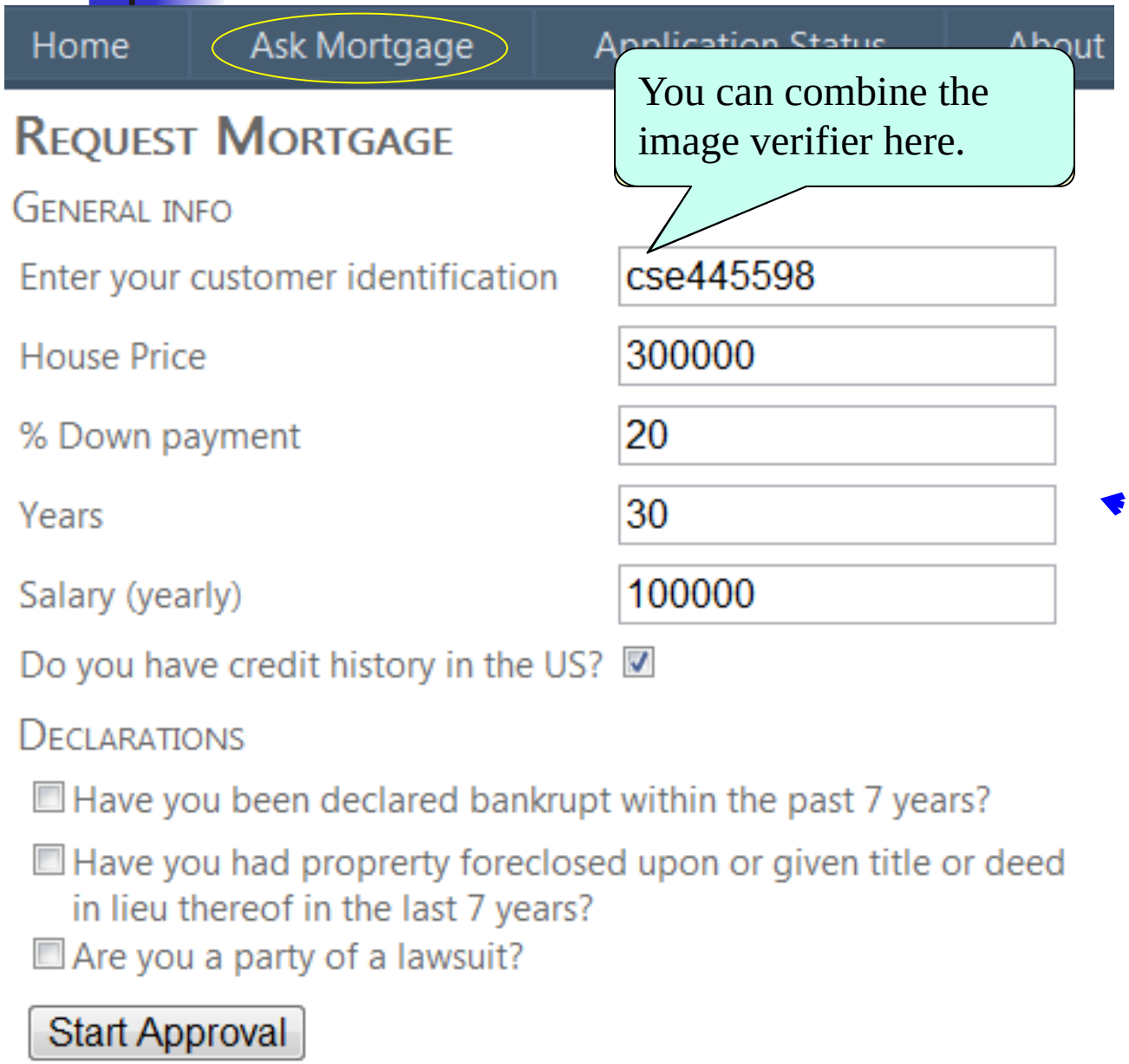
Case Study: Mortgage Application (Client)

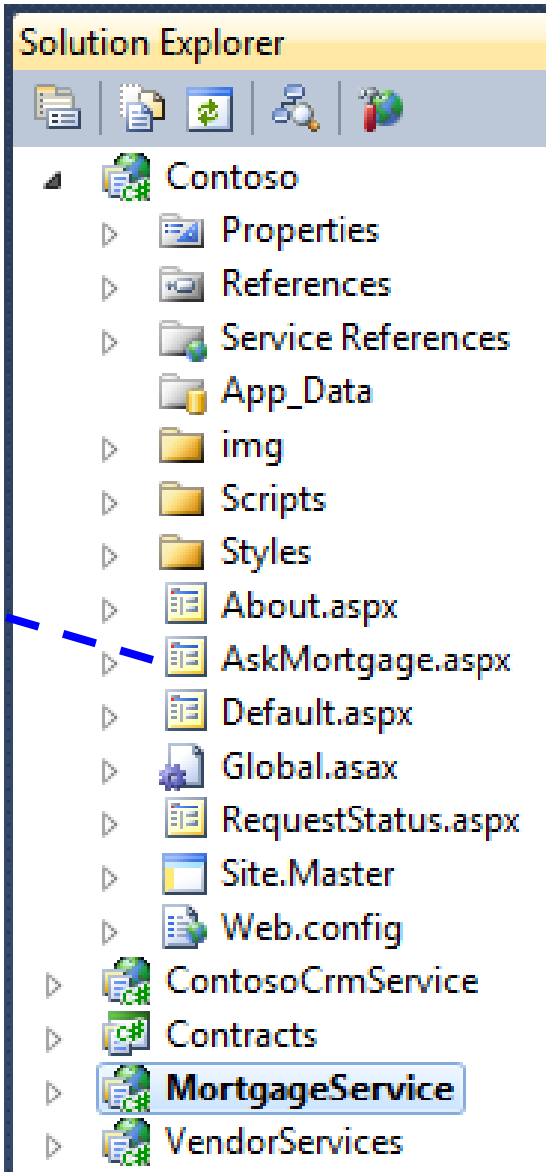




Source: http://msdn.microsoft.com/en-us/magazine/ff646977.aspx

Case Study: Mortgage Application (Client)





Case Study: Mortgage Application (Client)

MORTGAGE APP

Home | Ask Mortgage | Application Status | About

CHECK THE STATUS OF YOUR REQUEST

Enter your customer identification cse445598 Update Status

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

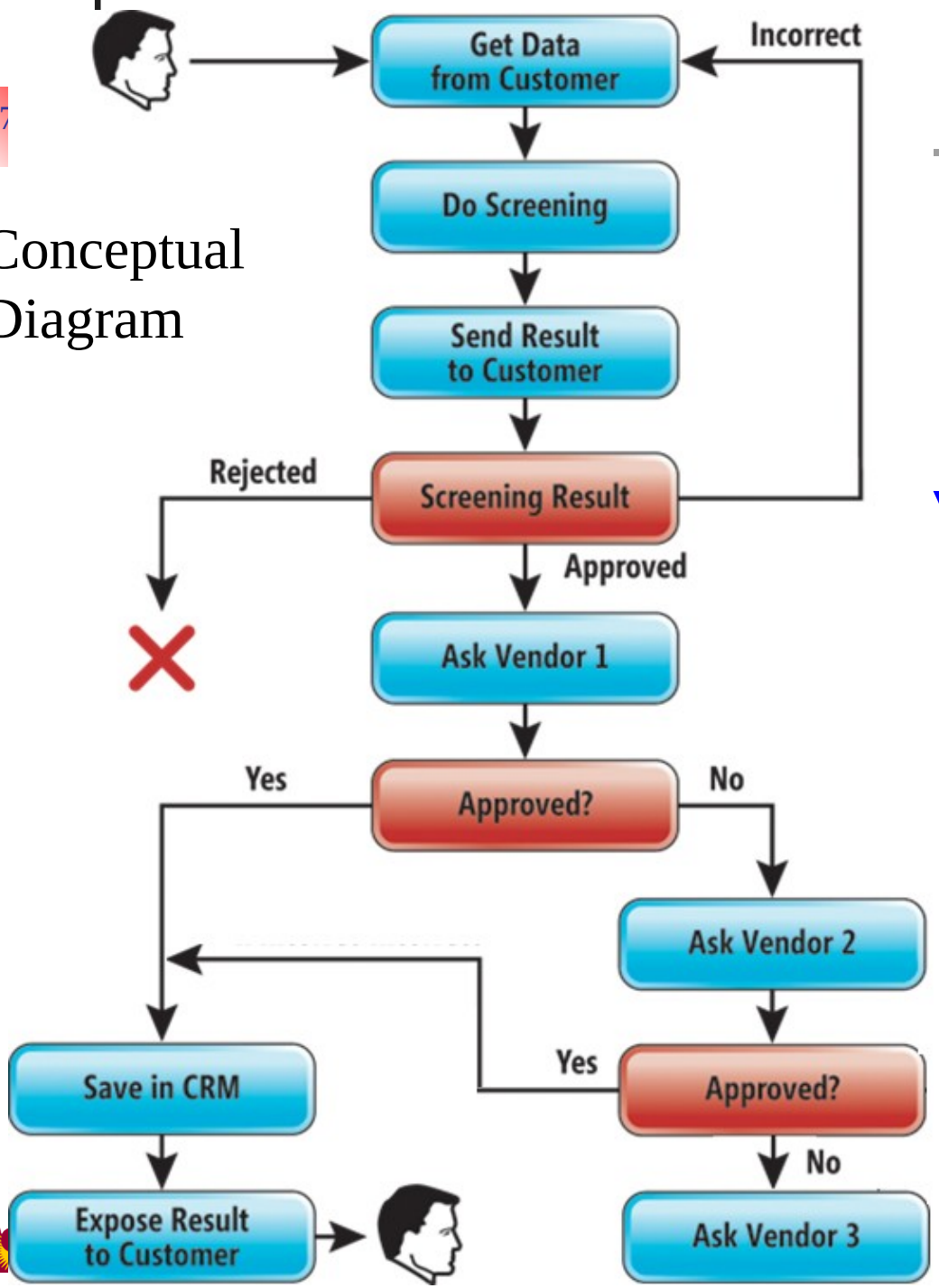
What is a persistence service?

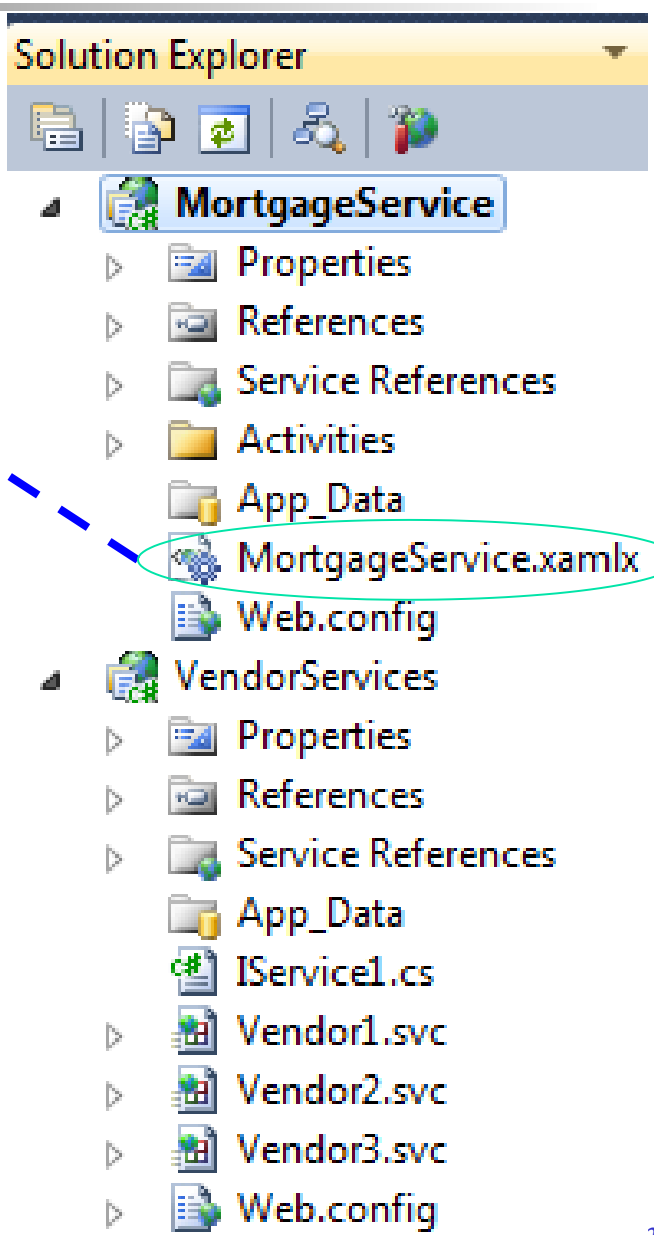
Solution Explorer

- 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

Workflow Service

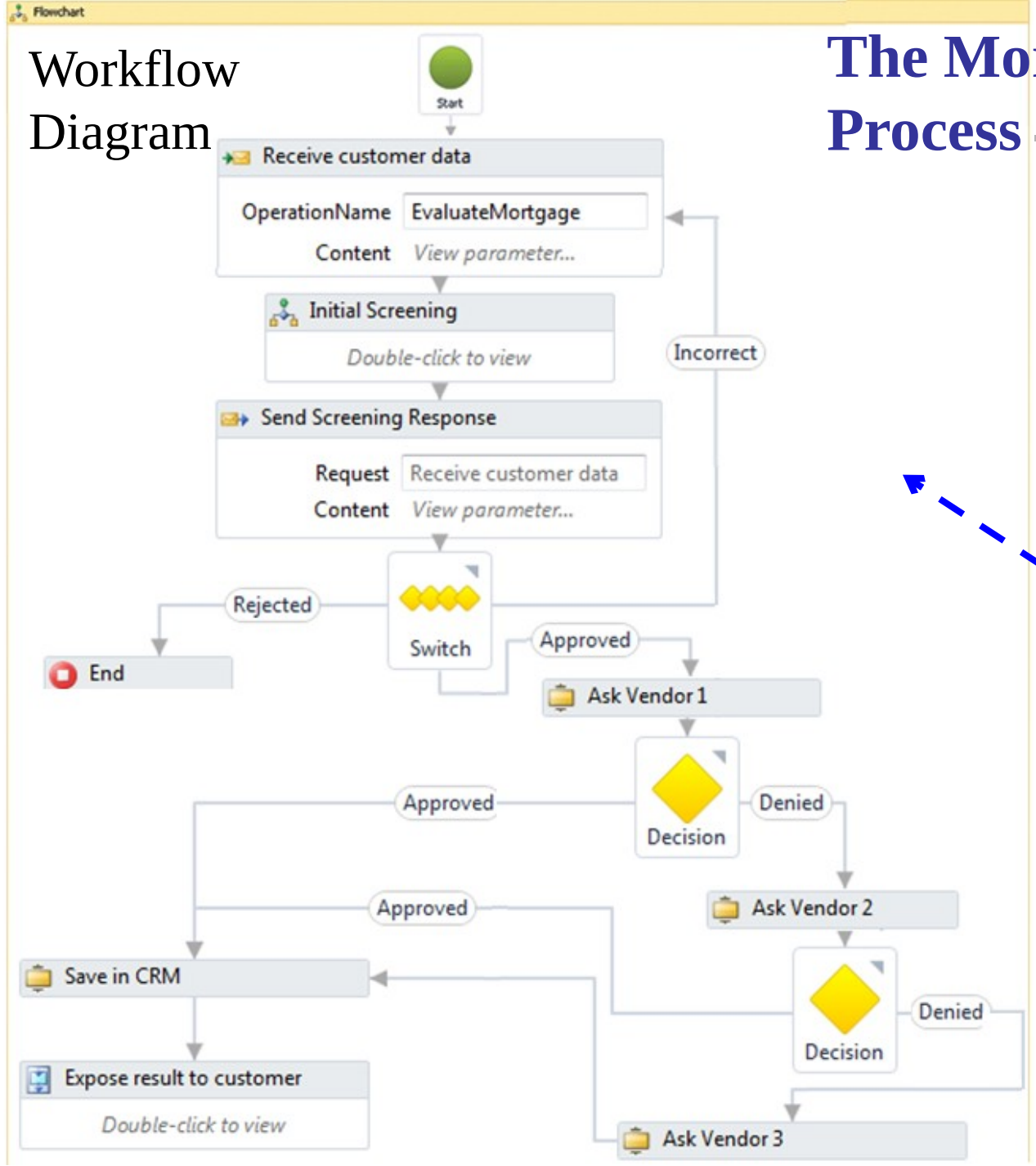
Conceptual Diagram

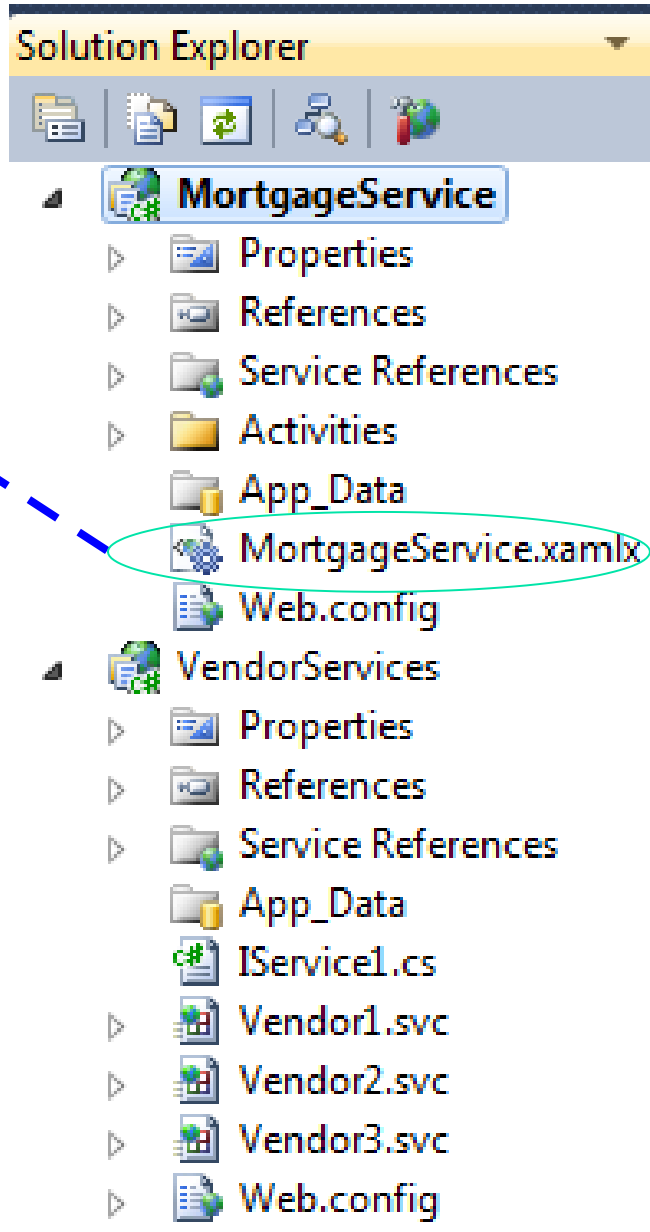




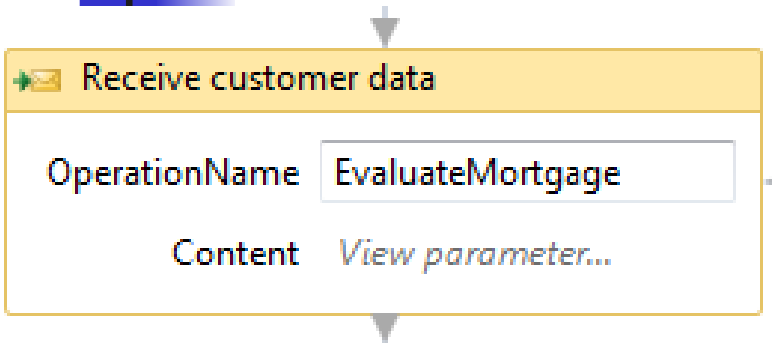
The Mortgage Approval Process as a Flowchart

Workflow Diagram



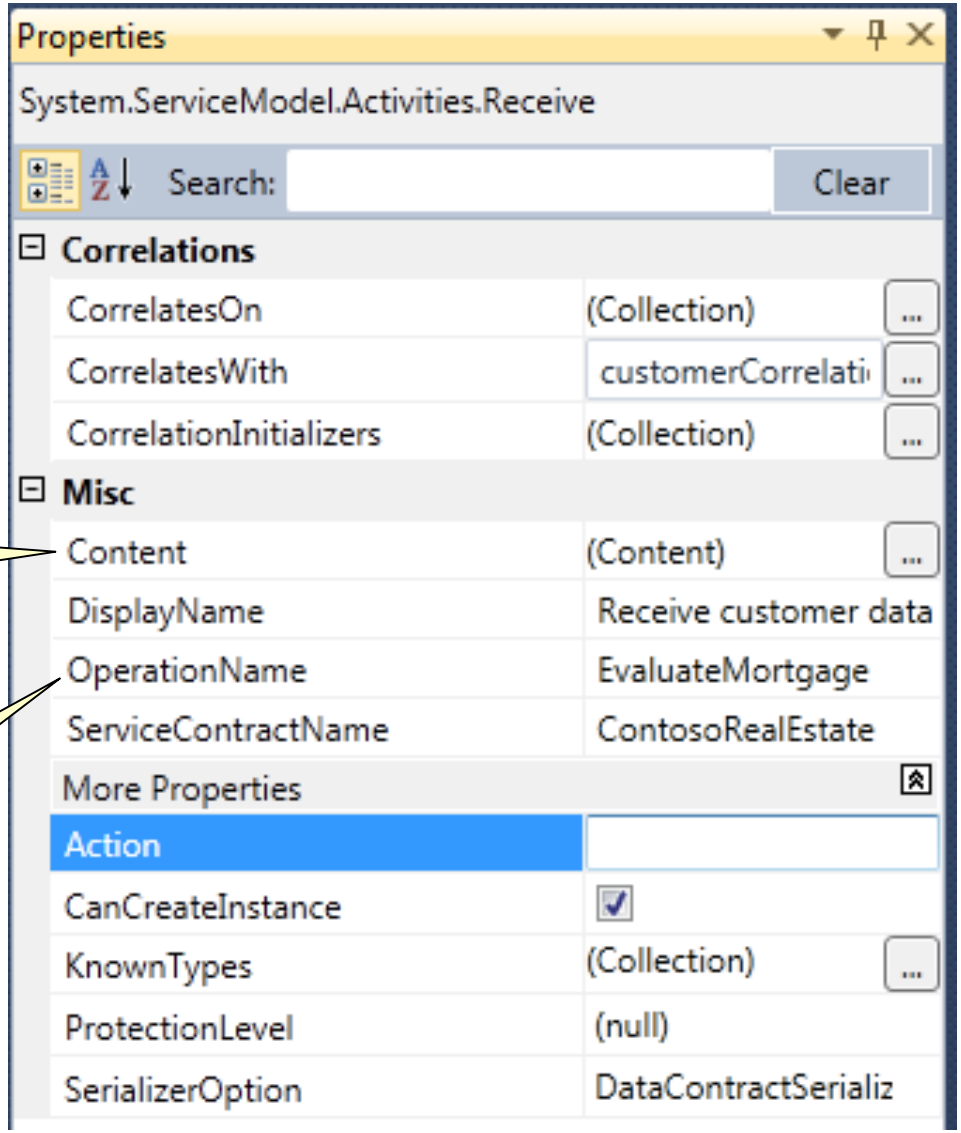


Define the Properties of Receive Activity

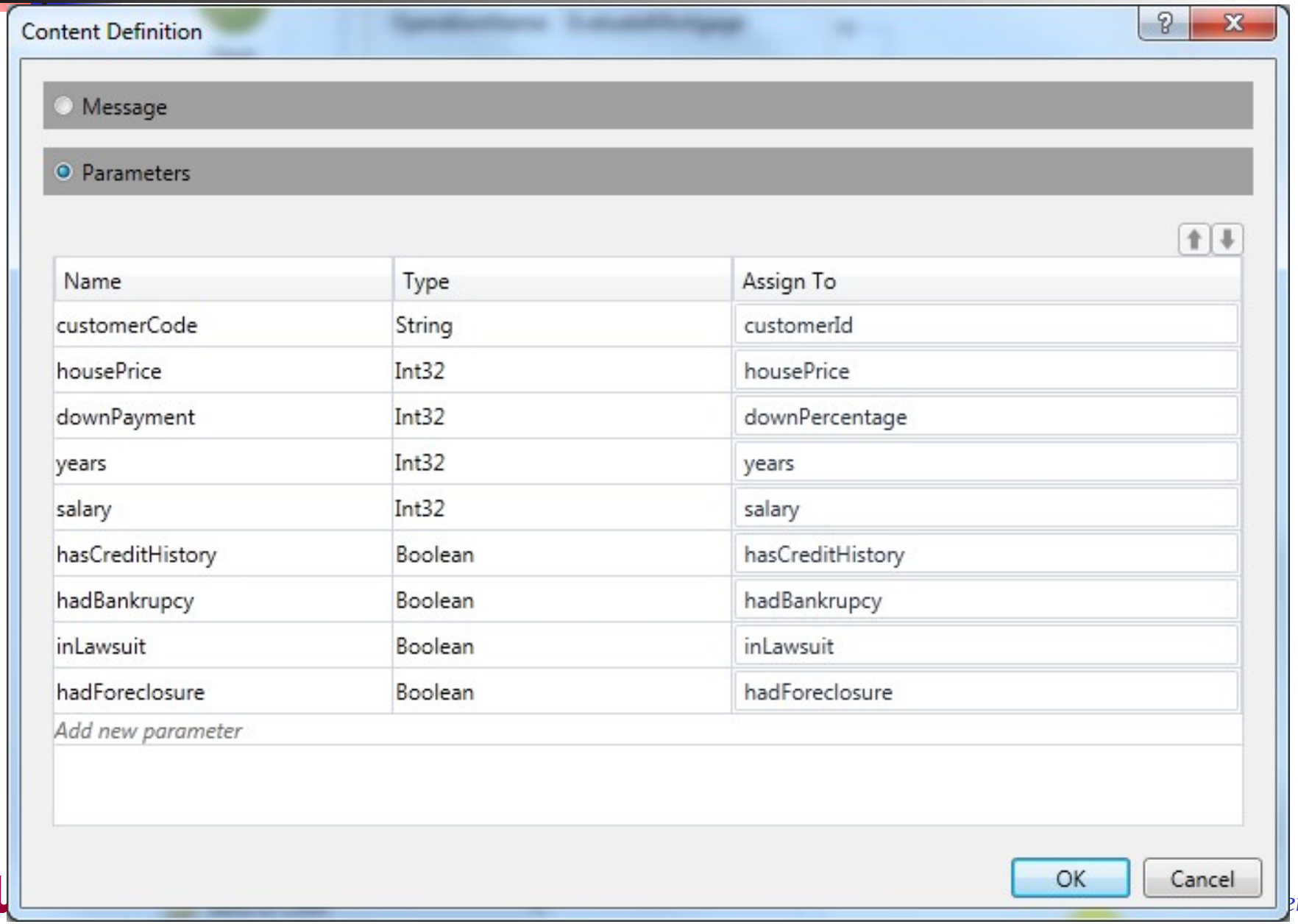


Define the WCF Endpoint:
Operation Contract:
Parameters in & out

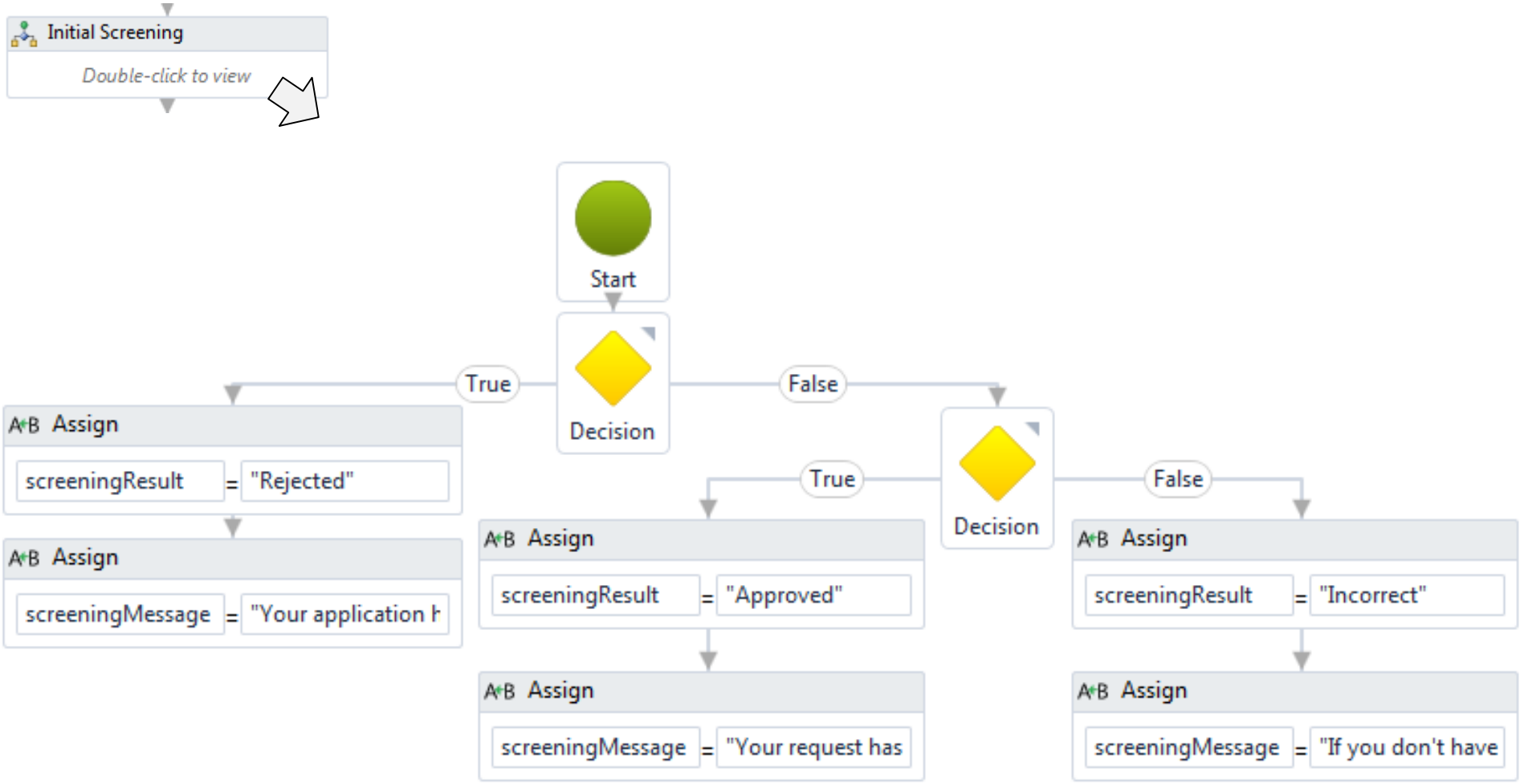
Define the WCF Endpoint:
Operation Contract:
Operation Name



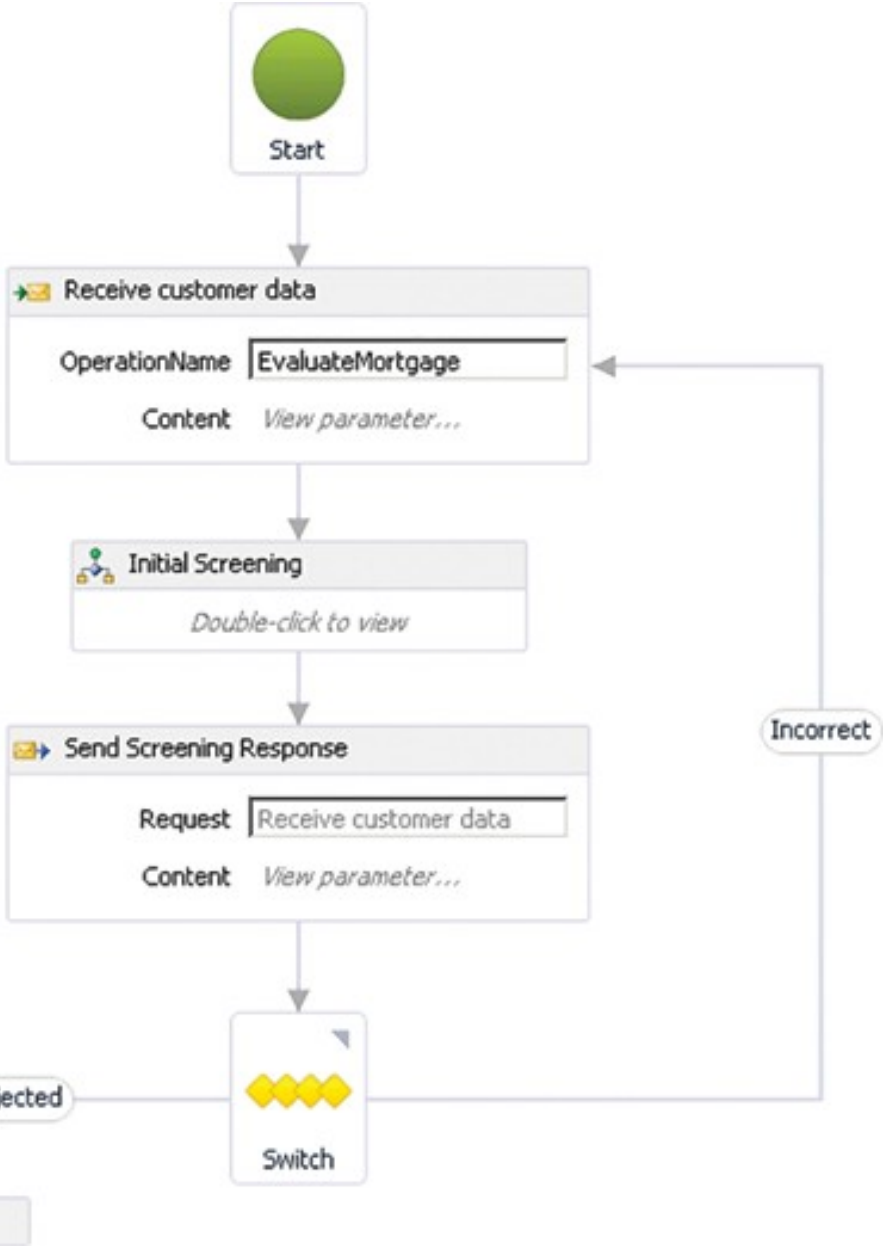
Configuring Input Parameters



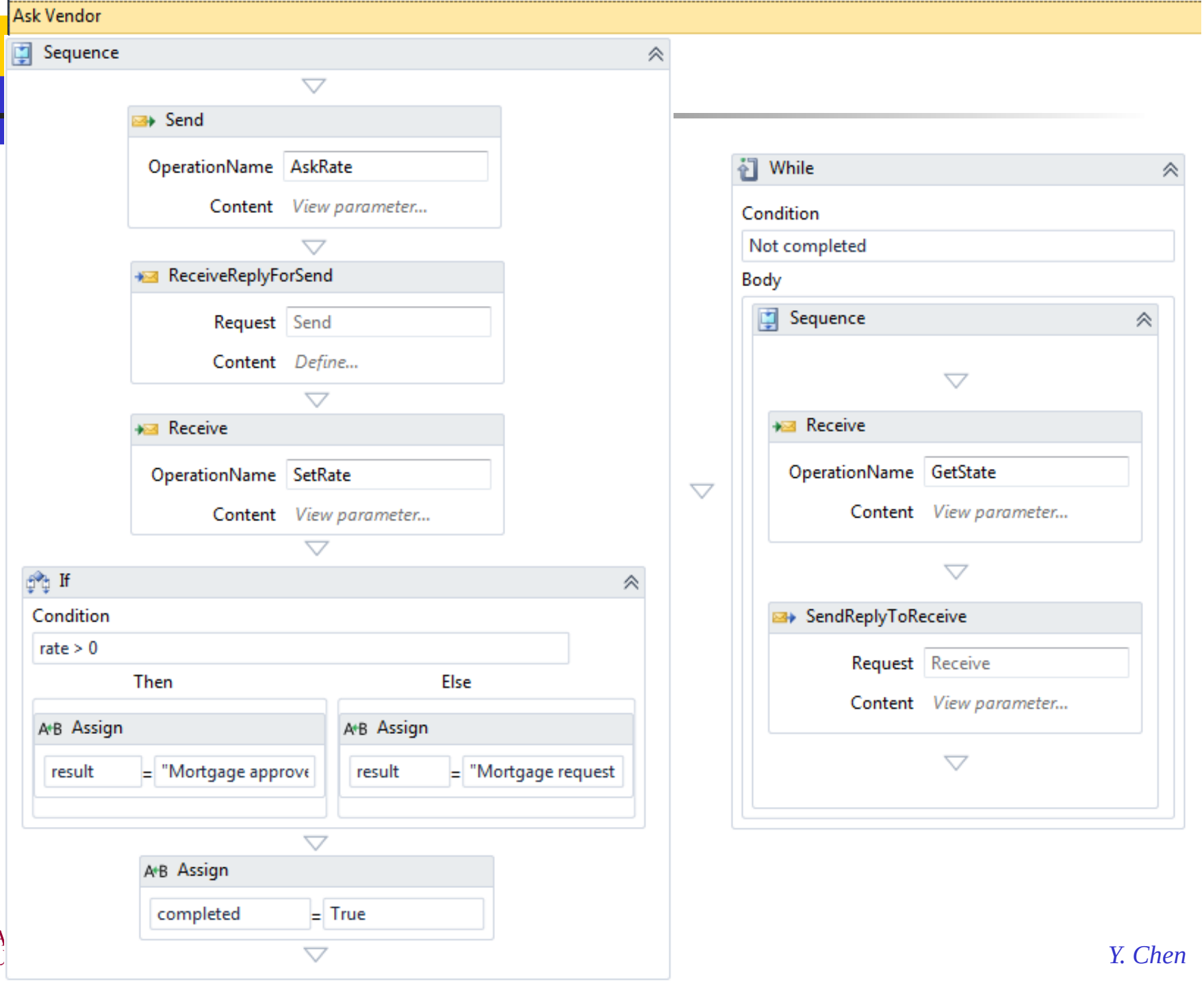
Adding the "Initial Screening" Flowchart



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



Ask Vendor CodeActivity



Service Deployed at

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

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: neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/MortgageService/Service1.xamlx
  neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor1.svc
  neptune.fulton.ad.asu.edu/WSRepository/Services/WFService/VendorService/Vendor2.svc
  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