

Distributed Enterprise Apps

by Mustafa Nassir

FILE	6367_MUSTAFA_NASSIR_DISTRIBUTED_ENTERPRISE_APPS_94244_163262967.DOCX (128.76K)		
TIME SUBMITTED	19-JAN-2018 06:26PM (UTC+0000)	WORD COUNT	3724
SUBMISSION ID	82114985	CHARACTER COUNT	21714

Distributed Enterprise Apps

WCF Concepts

Mustafa Nassir

Student No: 1708461



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

RESULT



School of Applied Computing ASSIGNMENT SPECIFICATION

Student name:	Mustafa Nassir	Student P number:	1708461
Programme:	<u>Web Development</u>		
Lecturer:	Dr Tim Bashford	Internal Verifier:	
Assignment Title:	WCF Concepts		
Hand Out Date:	28/11/2017	Submission deadline:	23/01/2018

Referencing:	In the main body of your submission you must give credit to authors on whose research your work is based. Append to your submission a reference list that indicates the books, articles, etc. that you have read or quoted in order to complete this assignment (e.g. for books: surname of author and initials, year of publication, <u>title of book</u> , edition, publisher: place of publication).		
Disclosure:	I declare that this assignment is all my own work and that I have acknowledged all materials used from the published or unpublished works of other people. All references have been duly cited.		
Student's Signature:	<i>Hardcopy assignments will not be accepted without a signature here</i> <i>Mustafa Nassir</i>	Date:	19/01/18

Late submission will result in a late penalty mark, as follows:

Up to one week late, maximum mark of 40% for first attempts. No late submission for resit work.

(Academic Quality Handbook 2016/17 7.5 (5))

Only the Extenuating Circumstances Committee may grant an extension.

Learning Outcomes tested (from module syllabus)

Critically appraise the major concepts, principles and practice to design distributed software components for use in enterprise services.

Demonstrate a comprehensive knowledge and understanding of complex client/server concepts, principles and tools to build, test, debug and critically evaluate software components in scalable, distributed applications.

Distributed Enterprise Application

Distributed Applications refer to programs that can run on multiple computers within a network in unison as well as store data on physical servers or cloud computing services. The Distributed application has some techniques for software building such as .Net frameworks, Web Services (WS), COM+ Enterprise and Message Queuing. The Windows Communication Foundation (WCF), merges technologies into a sole framework responsible for consuming and building services. It was previously called Indigo which involves a runtime and several sets of API in the .Net framework responsible for making linked applications of service delivery (Löwy and Montgomery, 2016, p. 41). A distributed application built with WS refers to as a service-oriented app which allows clients to take up multiple services loosely grouped.

Microsoft initially presented WCF as a share of the .NET 3.0 Framework continuously improving it on the .Net 3.5 and Visual Studio 2008. Through the use of WCF, data sent as asynchronous information from one service point to the other endpoint. It is a service hosted in an application or part of continuously available service. The functions are loosely grouped and characteristically contain Web Services Description Language (WSDL) interface such that any WCF client can use to consume the service, regardless of the platform hosting the service (Peiris, 2007, p. 234). The unified programming model used in constructing service-oriented apps (SOA), employs design techniques where service-oriented architecture principles, support distributed computing for remote clients (Rajjoub and Garibay, 2012, p. 177).

WCF technology has a preference for other types of tech in the same field such as ASP.NET Web interface API which is designed and built from HTTP. It contrasts to WCF in the initial functions and trigger points such as HTTP for the Site interfaces API, SOAP and

web services like TCP, NetTCP, and MSMQ. WCF also has an advantage over ASP.NET since it's retrofitted with a Representational State Transfer (REST) Starter kit (Peiris, 2007, p. 79). The programming models differ in that while developing software in ASP.NET, The model view control (MVC) is simple since it defines interfaces, creates implementation classes and incorporation of attributes (Weiss, 2013, p. 94). It is essential to understand the synchronicity of WCF and the ASP.NET API in that the latter is considered lightweight and can't be as flexible as WCF. Services are running using HTTP as the transport and require to channel other transport such as TCP and MSMQ, prefer to use WCF. Its flexibility also allows the services to combine transport mechanisms and offer responses in JavaScript Object Notation (JSON) and Extensible Markup Language (XML) formats allowing secure configurations (Löwy and Montgomery, 2016, p. 168).

Reasons for WCF

Windows Communication Foundation assists with web services since it offers backing for sending tickets using HTTP, TCP as well as other net protocols like MSMQ. It performs transport switching tasks with minor effort and allowing services to be hosted on its server and not on the web. WCF also has Built-in support for modern Web services standards such as Web Service and SOAP 1.2 plus the capacity to quickly support new versions (Passant, 2011, p. 96). It is more robust with security, reliability and fast transactions. In other setups, the sending of messages using other formats different from SOAP, for example, Representational State Transfer (REST) is also supported (Rajjoub and Garibay, 2012, p. 56).

Hosting a WCF Service

A service class in WCF is characteristically compiled to a library which requires a host windows process to run in. It is responsible for two critical options for hosting libraries that execute services. The First one uses a host process created by either Internet Information Services (IIS) or a correlated technology termed as Windows Activation Service (WAS)

(Weiss, 2013). All the other types allow selected service to be hosted in an arbitrary procedure. Most of the distributed applications use similar settings and can be organized in one environment. The importance of using WCF is the flexibility it offers. Nevertheless, it's also crucial to identify the service that requires hosting for the optimum type of hosting (Liu, 2014, p. 139). The ideal hosting is determined by the nature of the system since WCF (a unified programming model) is used during building service-oriented applications.

Contraction

WCF has five various methods that allow hosting the service. It includes Internet information service (IIS 5/6) hosting, windows server AppFabric, custom hosting in IIS/WAS, self-hosting as well as WAS hosting. The next segment describes all options, starting with the IIS to the WAS approach.

IIS 5/6 Hosting works using the same techniques as a web service where the client can access WCF functions through changing the formats' files from .asmx service to .svc extension (Löwy and Montgomery, 2016, p. 156). For example, when hosting the WCF service through IIS, it hosts the service like a Web Application because of Internet-related technologies such as a web server, not as a hosting engine, making it faster to access information. Self-hosting, on the other hand, has the developer in control of the WCF host Process. H/She uses during processes for clients and WCF therefore, it is considered the simplest and the least significant type of hosting of all (Weiss, 2013, p. 178).

Windows Activation Service (WAS) hosts support several protocols like TCP, IPC, and MSMQ and offer various advantages over Self Hosting like isolation of the host process, idle time management, recycling and application pooling. The main shortcoming is that it's considered a typical hosting engine and is unable to identify whether it's hosting a website or a WCF service. The combined fourth type of hosting is the custom hosting which fuses IIS and WAS to aim at accessing the host instance. WCF offers a hook known as "host factory" which uses the tag in a .svc extension file to access the host instance (Liu, 2014, p. 46).

Lastly, the Windows Server AppFabric is a technique that unlike WAS which is not optimised for hosting the services, it is installed to allow it to run on windows 7 or Windows server 2008 R2. The extension can track both WCF and workflow services as well as hosting and calling of the WCF service.

Implementation of WCF

WCF is implemented with the help of classes set up in visual studio software. Moreover, the re-using of an empty class file created by Visual Studio will require modifications to make it the implementation class of the service. The techniques involve written code that sets up the configuration table of the system (Passant, 2011). For example, a simple calculator program is generated and set up in code snippets that perform specialised tasks. In visual studio, developers rename the empty class file to ICalcService.cs within the file which defines the iCalc interface which is implemented. Running each technique shown in the iCalc in the ICalcService.cs class, for instance, the code sample below represents the service.

Code Example 1.

```
//ICalcService.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
namespace GettingStartedLib
{
    public class CalcService : iCalc
    {
        public double Add(double n1, double n2)
        {
            double result = n1 + n2;
            Console.WriteLine("Received Add({0},{1})", n1, n2);

            // Code output to the console window.
            Console.WriteLine("Return: {0}", result);
            return result;
        }
        public double Subtract(double n1, double n2)
```

```

    {
        double result = n1 - n2;
        Console.WriteLine("Received Subtract({0},{1})", n1, n2);
        Console.WriteLine("Return: {0}", result);
        return result;
    }
    public double Multiply(double n1, double n2)
    {
        double result = n1 * n2;
        Console.WriteLine("Received Multiply({0},{1})", n1, n2);
        Console.WriteLine("Return: {0}", result);
        return result;
    }
    public double Divide(double n1, double n2)
    {
        double result = n1 / n2;
        Console.WriteLine("Received Divide({0},{1})", n1, n2);
        Console.WriteLine("Return: {0}", result);
        return result;
    }
}
}

```

The interface outlines the service's contract, and the file `ICalcService.cs` comprises of code that implements `iCalc`. The `ICalcService.cs` attribute then classifies the `iCalc` as a service contract. In exposing the process to users, using the attribute. For the solution to build an edit of the `App.config` file is required. For example: the `<service name="...">` tag to `<service name = "GettingStartedLib.CalcService">`, Secondly `<add baseAddress="...">` tag to `<add baseAddress = "http://localhost:8733/Design_Time_Addresses/GettingStartedLib/ICalcService/">`, finally the 'contract' attribute of the `<endpoint ...>` tag changes to `<endpoint ... contract="GettingStartedLib.iCalc">`

Principles and Process of Metadata Exchange

Metadata exchange (MEX) refers to a process that describes how interactions with service endpoints to generate client code for access to the service. WCF provides a sufficient infrastructure for Importing, exporting, publishing and retrieving metadata through the use of

service SvcUtil.exe. It assists the client in the generation of proxy class and provides a platform for service endpoint. Exporting service metadata refers to the process of relating the endpoint so that users understand how the service works (Passant, 2011, p. 121).

Retrieving Service Metadata is the method of reclaiming the data through the use of WS- (MEX) or HTTP protocol for retrieving the info. According to Liu, importing metadata refers to the procedure of generating an intellectual representation of the service using data. Publishing service's metadata involves significant effort, in the conversion of Common Language Runtime (CLR) types and securing info into Web Services Description Language (WSDL) or other low-level representation (Liu, 2014, p. 143). The effort used doesn't increase the business value significantly. However, the host recognises the details of the service and the endpoints in order to print the metadata required. In MEX, the focus primarily is on publishing since it holds the majority of the essential functionalities in distributed applications. There are two routes for publishing metadata: first, provision of the data over HTTP-GET which is a commonly used text-based protocol supported by many frameworks which can otherwise be used as a dedicated endpoint (Rajjoub and Garibay, 2012, p. 169).

The prime purposes of service metadata such as generation of client consumer services automatically, implementing it and updating the linkage for a user. What the policies assist in explicit instructions is the decision to manipulate information in the system. Through the addition of specific service behaviour, WCF can provide the required information.

Metadata over HTTP-GET

Behaviour in the application development refers to a local aspect of the service case in point shows if the host metadata can be published over HTTP-GET which can be added administratively or programmatically. Since WCF has the ability offer the metadata for services over HTTP-GET spontaneously, it is enabled through the addition of specific service behaviours. Enabling metadata in an administrative method requires the addresses for use by

the clients in HTTP-GET and is registered as the HTTP base address of a service.

However, configurations in the host are dependent on the base address, where starting a service shows an exception message. Different addresses have the ability to be specified in the generation of metadata over HTTP-GET or URL added to the base address, and the data publishing will be done through setting the *httpGetUrl* property of the service metadata tag (Rajjoub and Garibay, 2012, p. 168).

The example code below shows how metadata exchange is enabled.

```
<system.serviceModel>
  <services>
    <service name = "MyService" behaviorConfiguration =
      "MEXGET">
      <host>
        <baseAddresses>
          <add baseAddress = "http://localhost:8080/" />
        </baseAddresses>
      </host>
    </service>
    <service name = "MyOtherService" behaviorConfiguration =
      "MEXGET">
      <host>
        <baseAddresses>
          <add baseAddress = "http://localhost:8081/" />
        </baseAddresses>
      </host>
    </service>
  </services>
  <behaviors>
    <serviceBehaviors>
      <behavior name = "MEXGET">
        <serviceMetadata httpGetEnabled = "true" httpGetUrl =
          "MyMEXAddress" />
      </behavior>
    </serviceBehaviors>
  </behaviors>
</system.serviceModel>
```

The above code sample demonstrates a host app configuration file where both hosted services, reference an un-conventional behaviour unit that facilitates metadata running over HTTP-GET (Weiss, 2013, p. 178). It enables the exchange over HTTP-GET for easy navigation to the addresses considered. For example, the base address which is set by default, or else a specific address which is readily accessible using a browser.

Contract Types

The actions of service or client are defined through a WCF contract. It's linked to the elements of a WCF endpoint that contains info about the service and aids in serialise service information. In WCF there are several kinds of contracts such as Message Contract, Data Contracts, Service Contracts and Fault Contract.

Data Contracts

The contract outlines the data type to be channelled to or from the client and advocates its leading role in the WCF service to be serialisation and deserialization. Serialization is the procedure in which translation of data structures is carried out into a file or memory buffer. Deserialization is the vice versa process where data structured from some format, is rebuild it into an object (Löwy and Montgomery, 2016, p. 198). There are two types of Data Contracts which allow the passage of data from the client.

- Data Contract

Contract states and outlines a specific class to be serialised for accessibility of the client. The Data Contract tag is important as it is used in any class to be serialised or de-serialised. Case in point:

```
[DataContract]
public class Student
{
}
```

- Data Member

It states and expresses properties inside a class, the feature that doesn't use a Data Member tag will not be serialised or de-serialised (Peiris, 2007, p. 265). For example,

```
[DataContract]
public class Student
{
    [DataMember]
```

```

        public int ID { get; set; }
        [DataMember]
        public string Name { get; set; }
        public string ContactNo { get; set; }
    }

```

In the previous instance, the “ContactNo” property doesn’t have access to the client for the reason that it will not be serialised or de-serialised even though it’s not used as a Data Member attribute.

Service Contracts

The Service Contracts labels the action a client during performing in a service. The attribute is in the System.ServiceModel namespace. Service Contract and Operation Contract are two core types of service contracts (Weiss, 2013, p. 231). The contract affirms an interface in the WCF service for the user to gain access. For example,

```

[ServiceContract]

interface IStudent

```

The Operation Contract defines and declares a function in the interface. Through the contract, a client will be able to call the function. The lack of using the Operation contract in the previous function makes the client unable to call the function.

```

[OperationContract]

Response AddNew(string student name);

```

Fault Contracts

A fault contract defines errors set to be raised by a service as well as how it handles and distributes the error to clients in its network. In a WCF service, the error can be passed to the consumer and declared using the following syntax (Liu, 2014, p. 189).

```
[ServiceContract]
public interface ICalcService1
{
    [OperationContract]
    [FaultContract(typeof(ServiceData))]
    ServiceData TestConnection(string strConnectionString);
}
```

Example of fault contracts with clients code

```
namespace FaultContractSampleWCF
{
    public class Service1 : ICalcService1
    {
        public ServiceData TestConnection(string StrConnectionString)
        {
            ServiceData myServiceData = new ServiceData();
            try
            {
                SqlConnection con = new SqlConnection(StrConnectionString);
                con.Open();
                myServiceData.Result = true;
                con.Close();
                return myServiceData;
            }
            catch (SqlException sqlEx)
            {
                myServiceData.Result = true;
                myServiceData.ErrorMessage = "Connection cannot open this " +
                    "time either connection string is wrong or Server is down. Try later";
                myServiceData.ErrorDetails = sqlEx.ToString();
                throw new FaultException<ServiceData>(myServiceData,
                    sqlEx.ToString());
            }
            catch (Exception ex)
            {
                myServiceData.Result = false;
            }
        }
    }
}
```

```

        myServiceData.ErrorMessage = "Error occurred. Please try
later.";
        myServiceData.ErrorDetails = ex.ToString();
        throw new FaultException<ServiceData>(myServiceData, ex.ToString());
    }
}
}
}

```

Operation Contracts

The contract attribute represents the operations that will be exposed. WCF makes use a request model for defining and matching to a member of its contracts. Methods in the contract interface use the “*OperationContract*” to expose data to the client. It means the interface *ICalcService.cs* can call addition and subtraction operations (Etzkorn, 2017, p. 195).

Example

```

[ServiceContract]
public interface ICalc
{
    [OperationContract]
    int Add(int x, int y);

    [OperationContract]
    int Subtract(int x, int y);

    // Not decorated
    int Multiply(int x, int y);
}

```

Message Contract

A *MessageContract* states and defines the structure of a message body and its serialisation development. The contract is used to send as well as access info located in SOAP headers. Although by default, WCF creates SOAP messages depending on the service's *DataContracts* in addition to *OperationContracts*.

A *MessageContract* can be categorised or uncategorised and is beneficial in interoperability cases when an existing message format is observed. Mainly the developer focusses more on developing the *DataContract*, serialising the data. In such a case, WCF provides the *MessageContract* means to adapt the message dependent on the requirements

(Passant, 2011, p. 212).

Example of a declaration of MessageContract in a class of employee details.

```
[MessageContract]
public class EmployeeDetails
{
    [MessageHeader(ProtectionLevel=ProtectionLevel.None)]
    public string EmpID;
    [MessageBodyMember(ProtectionLevel = ProtectionLevel.S
ign )]
    public string Name;
    [MessageBodyMember(ProtectionLevel = ProtectionLevel.S
ign )]
    public string Designation;
    [MessageBodyMember(ProtectionLevel=ProtectionLevel.En
cryptAndSign)]
    public int Salary;
}
```

Conceptual Message Exchange Patterns

WCF stores its data in XML and JSON format assisting in communication between client and the service. Message exchange in WCF relies on the manipulations of the data stored and in the JSON or XML files (Etzkorn, 2017, p. 145). Libraries allow efficient transfer during request processing. In conceptual patterns, several types of message exchange help in the proper communication of service and client. WCF has the following types of message exchange patterns (MEP).

Request-Response MEP

It is a native pattern for communication which uses service operation calls. When a service is called the operation generates a reaction message to the user. The client will receive a response regardless of the operation's void return type and will wait for the response from service for a certain time known as *RecieveTimeout*. The simple value for *ReceiveTimeout* is approximately a minute, after that, an exception of type *TimeoutException* is displayed

(Löwy and Montgomery, 2016, p. 238). After receiving the response, the following code is executed.

```
[ServiceContract]
public interface RequestReplyMEPSERVICE
{
    [OperationContract(IsOneWay = false)]
    void SaveData(string value);
}

One-Way MEP
```

The pattern uses one direction of message exchange. It works when we are not focused on receiving a response message back but instead aims at executing a certain functionality through sending a response to WCF service. The method is advantageous since no acknowledgement is required to pass the message. However, it's risky especially when an error occurs while sending the message. For that reason, One-Way MEP offers no support for followings in service operations.

- No output parameters as well as
- No reference parameters. Lastly,
- The return value is not included in this type of pattern.

All WCF bands maintain a One-Way MEP and in an attempt to enable it in WCF, set the operation clause to true as shown in the code sample below.

```
[ServiceContract]
public interface OneWayMEPSERVICE
{
    [OperationContract(IsOneWay = true)]
    void SaveData(string value);
}

Duplex Message Exchange Pattern
```

The MEP refers to a two-way message channel which works through getting a confirmation notification from WCF service as an operation completes. In less complicated terms, the two services in addition to the client's messages can use the method to send

messages to each other. Therefore, the technique of sending in the duplex pattern can both be request-reply also known as synchronous or could be one-way (asynchronous).

Peer-to-Peer MEP

The pattern comprises of two types: Pure P2P Network and a Hybrid P2P Network. A Pure P2P Network is a system of interconnected computer systems that just works through users, and the concept of Client or Server is separated. As an alternative, each of the nodes is connected and assumes the role of both Client and Server as required. A Hybrid P2P network refers to a setup which involves of the client-server concept through a server which is just responsible for answering the requests being sent by other nodes to get info (Etzkorn, 2017, p. 231). The network works using a server that performs some tasks such as searching for files. Moreover, a server with WCF needs to be started to facilitate operations for the other clients for connection as well as get their preferred information.

WCF Enables and Supports Interoperability

Interoperability refers to the characteristics of a system whose interface is understood to be compatible with other products or methods at any stage. WCF implements several web service protocols which include a varied number of options and extensibility. WCF offers assistance for web services structure in conventions through channels and application protocols through the message contracts feature. Interoperability, in this case, shows how distributed enterprise application protocols are achieved via XML Schema Description language 1.0 (XSD) and Web Services Description Language 1.1(WSDL) (Löwy and Montgomery, 2016, p. 190). The Infrastructure procedures' interoperability is provided by the WS-* terms.

References

- Etzkorn, L. H., 2017. *Introduction to Middleware: Web services, Object components, and Cloud Computing*. 1st ed. Boca Raton, Florida: Chapman & Hall/CRC, 2017.
- Liu, M., 2014. *WCF multi-layer services development with Entity Framework: create and deploy complete solutions with WCF and Entity Framework*. 4th ed. Birmingham, UK: Packt Publishing.
- Löwy, J. & Montgomery, M., 2016. *Programming WCF Services*. 2nd ed. Sebastopol, CA: O'Reilly Media.
- Passant, A., 2011. *Semantic Web technologies for Enterprise 2.0*. 1st ed. Amsterdam, Netherlands: IOS Press: Akademische Verlagsgesellschaft.
- Peiris, C., 2007. *Pro WCF: Practical Microsoft SOA Implementation*. 3rd ed. Berkeley, California: Apress: Distributed to the Book trade worldwide by Springer-Verlag, 2007.
- Rajjoub, H. & Garibay, R. G., 2012. *Microsoft Windows Server AppFabric Cookbook: 60 Recipes for Getting the Most Out of WCF and WF Services, Including the Latest Capabilities in AppFabric 1.1 for Windows Server*. 3rd ed. Birmingham: Packt Pub.
- Weiss, G., 2013. *Multiagent Systems*. 2nd ed. Cambridge, Massachusetts: The MIT Press.

Distributed Enterprise Apps

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

54/100

PAGE 1

PAGE 2

PAGE 3



Comment 1

Careful with wording - it uses http, but this sentence isn't really accurate as it is.

PAGE 4

PAGE 5



Contraction

Don't use contractions in academic writing. E.g. you should use "do not" instead of "doesn't".

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

MUSTAFA NASSIR

54

RESEARCH-INFORMED LITERATURE

-  A relatively small number of sources used, but of appropriate quality. Unfortunately, you've used the Harvard referencing style - you must use IEEE. Try to diversify your sources a little more - don't be afraid to use 3-4 sources to make a single point.

3

KNOWLEDGE AND UNDERSTANDING

-  Good knowledge and understanding demonstrated. Some good detail in many places - well done, Mustafa.

18

ANALYSIS

-  Some evidence of analysis, but as mentioned above, a lack of synthesis of materials in places. It'd also be good to see conclusions through your work - this'd help tie your sections together a little better.

12

PRACTICAL APPLICATION AND DEP.

-  Good reference made to practical considerations through the report - a few phrases miss the nuance of the situation, however - be precise.

16

SKILLS FOR PROFESSIONAL PRAC.



A few issues with spelling, punctuation and grammar. Be sure to proof read your work thoroughly. Report is nicely presented, however.

5