

Mircea Radu Serban Dobrau

Distributed Comp Sem 1

Assignment Report Web Dev

by Mircea-Radu-Serban Dobrau

FILE	2787_MIRCEA-RADU-SERBAN_DOBRAU_MIRCEA_RADU_SERBAN_DOBRAU_DISTRIBUTED_C OMP_SEM_1_ASSIGNMENT_REPORT_WEB_DEV_94244_600031464.DOCX (48.23K)		
TIME SUBMITTED	16-JAN-2018 02:32PM (UTC+0000)	WORD COUNT	2868
SUBMISSION ID	81872837	CHARACTER COUNT	18806

Mircea-Radu-Şerban Dobrău

P147987

Web Development (BSc)

**Distributed Enterprise Apps/Distributed and
Cluster Comp (M6X01094)**

Dr. Tim Bashford

Abstract

Middleware technologies enable large businesses and enterprises to build distributed software components or systems. As a response to existing middleware technologies Microsoft has developed a new SDK for developing distributed software/systems called Windows Communication Foundation (WCF). This report starts with a brief history and introduction of WCF, along with the rationale behind its development and what previous middleware technologies have been surpassed by it. Following this is a brief description of the ABCs of WCF along with the distributed systems concepts which underpin them, along with an exploration of the principles and process of WCF metadata exchange. Afterwards a brief listing of WCF hosting options is presented alongside with an explanation of the service application's configuration file. Next is a definition and code illustration of the four major WCF contract types, Service, Operation, Data, and Fault, followed by a conceptual exploration of 5 supported exchange patterns: Request-response, One-way, Duplex, Asynchronous and Peer-to-Peer (P2P). Finally, the report concludes with an explanation on how WCF enables and supports interoperability with other hardware/software stacks.

Table of Contents

Windows Communication Foundation	4
History	4
Superseded Technologies	5
Rationale	6
ABCs of WCF	6
WCF Service Hosting	7
Distributed Systems Concepts Underpinned	8
Principles and process of metadata exchange	8
Service Contract	9
Operation Contract	9
Data Contract	10
Fault Contract	11
Request-response Pattern	11
One-way Pattern	11
Duplex Pattern	12
Asynchronous Pattern	12
Peer-to-Peer (P2P) Pattern	12
Interoperability	13
References	14

Windows Communication Foundation

History

A large number of enterprises provide consumers with hardware, software, storage resources and others as platforms or software “as a service”, for example cloud computing, hosting and storage. Additionally, new technologies, data transfer or data migrations from one storage or “cloud” provider to another are possible scenarios for organizations and enterprises.

These tasks however present difficulties to businesses as the implementation of software or of newer technological innovations could be significantly different due to architectural differences and design. Furthermore, large businesses require a large amount of processing, software and hardware power for functional distributed computing. [1, p. 1]

Development of distributed applications on Windows is a difficult task; Namely, these factors come as most important when it comes to distributed computing: Interoperability, security and performance. [2, p. ii] With the scope of making this task easier various technologies under the “middleware” term have emerged. These technologies enable distributed communication between software components.

The most recent development of this type of technology, Windows Communication Foundation (WCF) comes with Microsoft’s 3.0 version of its .NET framework, released in 2006 as a SDK for service development and deployment. [3, p. 1] Working upon previous middleware such as Web Services, COM (Component Object Model), and RPC (Remote Procedure Call), the Windows Communication Foundation’s aim is to provide a unified programming model [2, p. 17] for distributed applications, which in turn provides a better unification of technologies, interoperability and a service-oriented architecture from the start.

Superseded Technologies

- **Remote Procedure Call (RPC)**, a middleware technology for synchronous remote communication in which the network communication details hidden from the code [2, p. 6].
- **Common Object Request Broker Architecture (CORBA)**, which provides interoperability between different programming languages, machines and products through an open standard for the creation of distributed object systems [2, pp. 7-8].
- **Distributed Component Object Model (DCOM)**, the enhanced version of the Component Object Model middleware which enables developers to build DCOM services for building distributed applications that interact with each other through the common COM interface. In comparison to COM which was designed only to work locally on machines using the Windows platform, DCOM runs processes across heterogeneous networks [2, p. 9].
- **Remote Method Invocation (RMI)**, a distributed object model for working and developing distributed Java applications. The processes involved require Java virtual machines (JVM) to invoke methods of remote objects [2, pp. 10-11].
- **.NET Remoting**, part of the .NET platform which provides developers with a large number of structures, classes, namespaces, delegates and interfaces for developers to work with. It also provides support for Remote Procedure Calls (RPC) and Component Object Models (COM+). Communication between remote and local components is done through a proxy layer which uses either the binary or Simple Object Access Protocol (SOAP) formats and the HTTP or TCP Channels [2, p. 12].
- Finally, **Web Services (WS)**, a software system in which its bindings and interfaces are described in XML. These systems communicate with each other by XML based messages conveyed by Internet protocols. Web services have played a major role in the growth of the Service Oriented Architecture (SOA) for heterogeneous systems [2, p. 14].

Rationale

First, WCF attempts to unify some of the existing distributed technologies listed above so that programming complexity is reduced. In other words, the WCF attempts to **standardize** and bring distributed technologies down to one platform. Additionally, WCF provides the developer with standardizations of all the features attributed to previous distributed technologies such as RESTful communication, queued messaging and distributed transactions. [2, pp. 17-18]

Next, as a solution for tightly coupled or platform dependent software, WCF uses a standard messaging protocol which is programming language and platform agnostic. As a result, WCF provides loose coupling and cross-platform/technology interoperability, so that organizations can more easily merge systems according to business needs. [2, p. 18]

Finally, WCF presents a step towards Service-Oriented Development which encompasses encapsulating functionality and code as services, in contrast to the Object Oriented Programming paradigm's approach to using encapsulating reusable functionality inside classes. The idea expands towards services communicating with each other without any regard to their environment or what programming language they are written in. [2, p. 19]

ABCs of WCF

WCF Services rely on three important concepts:

1. **Addresses** [3, pp. 4–7]: Each service's address is composed of its location and its transport protocol or transport scheme. The location can be a local machine, a site, network; Furthermore, it can be a communication port, pipe, etc; It can also be supplied with a extra more specific path which can be a unique string such as a globally unique identifier (GUID). Next, the transporting scheme is specified and several transport schemes are supported such as HTTP, TCP, IPC, Peer network or MSMQ.

Its syntax is as follows:

```
[transport]://[machine or domain][:optional port]/[optional URI]
```

An example would be:

```
net.tcp://localhost:8002/SomeService
```

Which roughly translates to: **"Go to the localhost machine (or SomeService for short) using the TCP protocol where on port 8002 someone is waiting for calls"**.

2. **Bindings** [3, pp. 24–29]: A large number of transport protocols exist which in turn have a wide number of encoding options that are each fit for differing scenarios. Additionally both services' transport protocols and options must be compatible.

For this reason WCF provides the user with a small number of “bindings” which represent an opinionated approach to managing service transport protocols, minimizing the effort required to ensure that all aspects of the service-to-service communication are properly tailored and service-compatible. Common bindings include Basic (ASMX), TCP, IPC, Web Service (WS) and MSMQ.

3. **Contracts** [3, pp. 7–11]: Conceptually similar to interfaces, all services employ a range of platform-agnostic and standard contracts which have the role of enforcing and declaring what a service is going to do in regards to operations, passing data types, raising errors, and interacting by way of messages.

WCF Service Hosting

WCF services provide the user with multiple hosting options [4, Ch. 11]:

- Internet Information Services (IIS)
- Windows Service Hosting
- Windows Activation Service (WAS)
- Self-hosting by using a console application

Furthermore, the configuration of the service application can be done by editing a web.config file, which is normally comprised of 5 elements [5, pp. 44–45]:

- **Configuration**: Root element of the file.
- **system.serviceModel**: Root element/node for WCF-service specific configuration
- **serviceHostingEnvironment**: Element used to specify which hosting environment of the service
- **serviceActivations**: Element used to specify service name as well as relative address.
- **ServiceBehaviors**: Element used for defining specific behavior of a service, for example enabling metadata exchange with `<serviceMetadata httpGetEnabled=" true" >`

Distributed Systems Concepts Underpinned

The major distributed systems concept that underpins WCF is its emphasis on, and goal, of building software which follows the Service-Oriented Architecture (SOA). Service-Oriented Architecture has emerged as a response to the limitations of Object Oriented Programming and Component Oriented Programming, the majority of which is tightly coupled functionality [4, pp. 1-10]. Service-oriented architecture strives to solve these problems by following 4 tenets [2, p. 19]:

- **Explicit service boundaries:** Services can freely access other services without care of environment or programming language.
- **Services as being autonomous:** Services must be versioned and managed independently.
- **Share schema, not class:** Services transfer schema or contracts rather than the classes.
- **Compatibility is based on policy:** Policies between services must be similar so that they communicate. WCF powers this by negotiating communications channels dynamically and also by creating service policies automatically depending on signatures, class attributes and configuration.

Principles and process of metadata exchange

The metadata of a service has the role of describing the service's characteristics so that other external entities or services can interact with it successfully. It includes XML schema documents which define the data contract and WSDL documents that describe the methods or functionality of a service.

Although services always contain this metadata, WCF gives the users the option to hide it from external entities and transfer it through other means for extra security. It can be enabled through the configuration for the metadata behavior [5].

Furthermore, WCF enables easy usability of the metadata exchange concept by providing the user with the option of having the service host automatically provide an implementation interface of the metadata exchange endpoint industry standard, called `IMetadataExchange`. Finally, endpoint configuration can be easily added in the service's configuration file and WCF provides the user with predefined options for HTTP, TCP and IPC protocols. [3, pp. 39–49]

Service Contract

[7, p. 64] Service contracts act as the interface to the service, exposing the service's methods. In other words, service contracts define what is expected of the service. Service contracts represent .NET classes which in turn implement .NET interfaces. Interfaces can be defined as service contracts by appending the attribute "ServiceContract" on top of the interface. [2, p. 29]

```
[ServiceContract]

public interface IWcfServerInterfaceExample
{
    ...
}
```

Operation Contract

The operation contract is defined within the service contract; its scope is to define the parameters and return type of an operation. Additionally, it also defines operation-level settings such as transaction flow and direction of operation, and the fault contract of the operation. [8]

```
[ServiceContract]

public interface IWcfServiceInterfaceExample
{
    ...

    [OperationContract]
    string SomeStringMethod(string StringValue)
} [2, p. 29]
```

Data Contract

[2, p. 20] A data contract defines what kind of data is exchanged between the service and the client. More specifically, data contracts define parameters and return types in regards to data sent/returned and enable type marshaling and conversions [3, p. 103]

```
[DataContract]

public class SpecialDataType
{
    [DataMember(Name=" SpecialType1" )]
    public string SpecialType1;

    [DataMember(Name=" SpecialType2" )]
    public int SpecialType2;

    [DataMember(Name=" SpecialType3" )]
    public decimal SpecialType3;
}
```

The DataMember attribute present a few parameters which control how the members will be serialized/de-serialized [4, pp. 67–68]:

- **Name:** Defines name of member for data contract, which appears in XML schema
- **IsRequired:** Gets/sets a value which tells serialization engine that member must be present
- **Order:** Gets/sets the order of serialization and deserialization of a member

Fault Contract

Fault Contracts define what errors are raised by the service and how the service propagates errors to its clients [3, p. 7]. Fault contracts are defined with the `FaultContractAttribute` [3, p. 263] on a per-method basis:

```
[FaultContract(typeof(SomeException))]  
  
string SomeMethod(string first, string second);
```

Request-response Pattern

[3, p. 225] The most common communication pattern, the request-response pattern is comprised of a client issuing a request towards the service and it stays locked until receiving a message. If the service does not respond back within a predefined time (by default 1 minute) then the client throws a `TimeoutException`. It is the default operation mode of contracts.

One-way Pattern

In some cases an operation may not return a value; In this case, the client can't react if the invocation succeeded or failed. For this case scenario, WCF has support for one-way operations. With this operation the client makes a call WCF generates a request message but otherwise does not return any reply back to the client. Although this operation does not return values or exceptions from the service itself the client can still expect exceptions and deduce that the call failed on the service.

Additionally, whether or not one-way calls are dispatched at once depends on how the service is configured to manage concurrency. If the service's message capacity is exceeded then the client is blocked from issuing further calls. The client however will be unblocked once the call is deposited into the buffer. [3, pp. 226–229]

Duplex Pattern

Commonly known as callbacks, WCF also supports duplex operations; In duplex operations, the client-service role is reversed: The service becomes the client issuing calls and the client responds back to these calls. Duplex operations are useful for situations in which the client needs to be notified that some action has occurred on the side of the service.

The only commonly used bindings that have support for duplex operations out of the box are the TCP and IPC protocols due to their bidirectional nature; Although the HTTP protocol cannot be used for duplex operations by default due to its connectionless nature, WCF provides a workaround for this situation with the `WSDualHttpBinding` binding, which sets up two Web Service channels: one for client-service calls and one for service-client calls. [3, pp. 230–232]

Asynchronous Pattern

Whenever the client issues a call to the service, the client is temporarily blocked until the service successfully executes the client call. Calling operations asynchronously involves the client having its control handed back immediately after having issued a call whether or not the service executed the command successfully. This way, multiple calls can be made in rapid succession, increasing communication speed and flexibility. During this process the client should ideally be notified whenever any of the calls have been successfully executed as well as be provided with the results of the invocation.

WCF intrinsically offers asynchronous call support for any type of service. Additionally, as the asynchronous invocation support takes place on the client side any binding is compatible with asynchronous invocation. [3, pp. 439–440]

Peer-to-Peer (P2P) Pattern

Peer-to-Peer networks involve collaboration between individuals and groups of users towards sharing computing power and bandwidth, in comparison to the traditional approach of relying on a small number of more powerful servers on the network. In a peer-to-peer (P2P) network each computer connected to the network is called a peer or a node. Nodes alone are responsible for uploading and downloading data among each other with no need of a central server.

There exist two types of P2P networks: “Pure” P2P networks have no concept of a client or a server and its nodes act both as client and server at the same time. These types of networks however are rare. On the other hand, the much more common “Hybrid” P2P networks act very similarly, with the addition of a central server which keeps track of the network’s various peers. The server does not store any data; it is however responsible of responding to requests from peers for peer information such as what files are stored by what peer. [7, pp. 370–372]

WCF supports P2P networking by way of implementation of the P2P networking stack, called a peer channel. [7, p. 380]

Interoperability

WCF provides cross-technology and cross-platform interoperability through its use of message-based standards for communication [2, p. 18] following a suite of specifications developed by the Web Services Interoperability (WS-I) organization, comprised of Microsoft, IBM, BEA, and Sun. This organization aims to allow software to communicate with other software regardless of platform.

WCF’s native messaging protocol is SOAP (Simple Object Access Protocol), an open standard that provides WCF services interoperability with cross-platform software and hardware. SOAP is a protocol used for web services which is based on XML (Extended Markup Language) which allows services to invoke methods on remote machines without having any knowledge of the specific details of the platform or software that these machines run. XML represents the data being sent and is structured according to the common SOAP schema that both parties agree to. [7, pp. 10–13]

References

- [1] P. Kaur and M. Mahajan, "Integration of Heterogeneous Cloud Storages through an Intermediate WCF Service," *Int. J. Inf. Eng. Electron. Bus. Hong Kong*, vol. 7, no. 3, pp. 45–51, May 2015.
- [2] M. Hamayun and N. Ahmed, *Performance Evaluation of Windows Communication Foundation's Interoperability*. 2010.
- [3] J. Lowy and M. Montgomery, *Programming WCF Services: Design and Build Maintainable Service-Oriented Systems*. O'Reilly Media, Inc., 2015.
- [4] P. Chirag, *Developing Service-Oriented Applications Using the Windows Communication Foundation (WCF) Framework*. IGI Global, 2017.
- [5] M. Liu, *WCF Multi-layer Services Development with Entity Framework - Fourth Edition*. Packt Publishing Ltd, 2014.
- [6] M. L. Bustamante, *Learning WCF: A Hands-on Guide*. O'Reilly Media, Inc., 2007.
- [7] N. Pathak, *Pro WCF 4: Practical Microsoft SOA Implementation*. Apress, 2011.
- [8] M. Liu, *Wcf Multi-Tier Services Development with Linq*. Packt Publishing Ltd, 2008.

Mircea Radu Serban Dobrau Distributed Comp Sem 1

Assignment Report Web Dev

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

78/100

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

"organisation" in UK English.



Comment 2

No need for a capital after a semicolon

PAGE 5

PAGE 6



Comment 3

no comment



Comment 4

Sort of - I'd say that SomeService is more a specific location on localhost (as we can host more than one service on the same host).

PAGE 7

PAGE 8

PAGE 9



Comment 5

This is a bit too close to the source material - be careful to paraphrase without rewording the original material, or (if absolutely necessary), use quotes.

PAGE 10

PAGE 11

PAGE 12



Comment 6

This is also a bit too close to the source.

PAGE 13

PAGE 14

GRADING FORM: SMU GENERIC_GJ

MIRCEA-RADU-SERBAN DOBRAU

78

RESEARCH-INFORMED LITERATURE



This is an excellent piece of work overall - well done, Serban. A fair number of academically credible sources used well. Good reference to sources generally, though it'd be a good idea to consider using more than one source to make the same point

7

and citing them all [2][3][4][5] etc. We can have a chat about how best to structure an essay.

KNOWLEDGE AND UNDERSTANDING

 Extremely good evidence of knowledge and understanding - robust descriptions of theoretical processes provided throughout. **23**

ANALYSIS

 Good evidence of analysis. It would, again, be good to see multiple sources used to back up a single point. Some conclusions throughout your work would also help the narrative to flow a little better. **18**

PRACTICAL APPLICATION AND DEP.

 Extremely good reference to practical considerations, you obviously have a complete grasp of the technical considerations pertaining to WCF. **23**

SKILLS FOR PROFESSIONAL PRAC.

 You write very well. Be careful to use UK rather than US English (generally "s"es in words rather than "z"s - organise rather than organize etc). A nicely-presented and formatted document. Overall, I'm impressed with this piece of work - you obviously worked hard, extremely well done. We can have a chat about a few minor points which can help you to **7**

achieve even higher marks in the second assignment.