SOA APPROACHES ANALYSIS AND INTEGRATION WITH WCF SERVICES

Komal Shringare[1]

Abstract—The technologies to implement SOA will certainly go forward to address emerging needs, but its concepts will remain the same. To address those needs and concerns that SOA is potentially being stretched beyond its limits, a significant and coordinated research program is needed. IT Organizations across many industries have adopted or in the process of adopting towards a service-oriented IT architecture (SOA) with WCF Services. WCF Services is a platform for building, configuring and deploying network-distributed services. WCF service is one the most demanding service oriented technology; it has basic fundamental characteristics which is interoperability to support cross platform. WCF provides a common platform for all .NET communication.

This report outlines the various implementation approaches of SOA and performance analysis of respective approaches like Contract First WCF Services with XML Sterilization and Data Contract Sterilization and RESTful WCF Service.

Keywords— *SOA,Contract-First WCF Service, XML Serialization, Data Contract Serialization, RESTful WCF Service, Performance Analysis, GUI*

I. INTRODUCTION

The Service-Oriented Architecture (SOA) designing and implementation based on a net of software services. Services encompass associated, loosely coupled units of functionality that have no calls to each other embedded in them.

This report presents an overview of the current approaches of the SOA Implementations, performance analysis and focuses specifically on integration of SOA with emerging GUI to make it more portable and device independent as per current market sentiments.

A. WHAT IS ARCHITECTURE?

Software architecture is a description of a software system in terms of its major components, their relationships, and the information that passes among them. A fundamental purpose of software architecture is to help manage the complexity of software systems and the

[1] Asst.Professor, YMT College of Management, Kharghar, Email: komal.loke@gmail.com, Mob: 9969406446

modifications that systems inevitably undergo in response to external changes in the business, organizational and technical environments.

B. WHAT IS SOA?

SOA is concerned with the independent construction of business-aligned services that can be combined into meaningful, higher-level business processes and solutions within the context of the enterprise. Anybody can create a service; that is not the challenge of SOA. The real value of SOA comes when reusable services are combined to create agile, flexible, business processes. Unfortunately, that does not just happen by itself. Achieving it might be easier to manage if a single organization is creating all of the services, but that is not the case at most large organizations. So, part of the architecture of SOA is responsible for creating the environment necessary to create and use compassable services across the enterprise.

The important parts of SOA are:

- Services - Modular units of business functionality
- Integration - Connection to and exposure of existing applications and/or data as services
- Existing systems - Existing legacy systems, commercial off-the-shelf (COTS) applications, and data that the enterprise wants to leverage
- Semantics - The underlying meaning of information that is exchanged in processes
- Transformation - The conversion of information from one format or semantic to another
- Communications - The ability of services to communicate with each other

II. LITERATURE REVIEW

For the purpose of this research, adopted the definition for WCF services put forth by the World Wide Web Consortium (W3C), which states that WCF services are software component to support cross platform so that interaction can easily carried out over the network with defined interface. The message exchange must be in specific format like in the form of Web Services Description Language (WSDL). WCF Service mainly interacts with other systems using SOAP messages as prescribed/defined in WSDL. Typically the message exchange happen using HTTP or HTTPS protocols in XML from in conjunction with serialization as per standards. A service registry based on the Universal Description Discovery & Integration (UDDI) standard can be employed to publish and discover Web services. Web services enable the SOA concept to be applied in a Web based environment.

METHODOLOGY

The project work employs a multiple case research strategy to explore how organizations are approaching the use of a service-oriented architecture across multiple platforms and integrating it with web application so that it can be use by multiple devices as per current technology trends. This approach was chosen because the SOA is a contemporary event that can be observed in a real-life context and for which substantial scientific theory has not yet been established. The project work is to addresses the factors that influence the integration of an SOA as well as how and why these factors play a role.

III.PROBLEM STATEMENT

Generally, the SOA service comprises applications that serve client (or consumer) applications. This means that service demands might not always be met within the required time constraints, if the capacity management is reactive instead of proactive; thus, there is a danger that the infrastructure will act as a bottleneck.

Conversely, infrastructure resources can remain idle should the service demand be below that which is provisioned by capacity planning. This represents a waste of money on unused infrastructure. Furthermore, capacity planning and modeling, when performed properly, can itself be a very costly and time-consuming exercise and would need to be performed regularly in order to track changing SOA-service workloads over time. Therefore, a more efficient and cost-effective solution is needed for designing and implementing SOA services.

A. STATEMENTS OF PROBLEM

- Integrating homogeneous components routine, relatively smooth
- Performance gain

B. OBJECTIVE

The objective is to create WCF service in Contract First approach (DataContarct Serializer and XML Serializer) and in Restful framework based on Service Oriented Architecture (SOA).

C. SCOPE

There are various ways where SOA can be implemented in both technologies and methodologies. However, we are limiting this edition to .Net and Contract First Approach and Restful.

IV. PROPOSED SOLUTION

For performance analysis, different type of approaches has been chosen to design WCF service. It includes Code First Approach, Contract First Approach and Restful Service. For performance check Data Contract Serialization and XML contract Serialization has been chosen. With Data Contract Serialization there is significance performance gain. To verify this it is really required to do performance comparison analysis. As a part of research work WCF Services has to be created to perform the performance analysis in XMLSerilizer and DataContarctSerilizer. Then and there needs to do performance comparison analysis to verify performance gain.

V. CONTRACT FIRST WCF SERVICE

This is especially true when it comes to designing Services using SOA. Careful attention should be paid toward the formal interface used for communication. This interface dictates the usability and interoperability of your system. Consequently, designing this interface, which is also known as the contract, during early phases of the lifecycle is significant. We are now going to design and develop contracts first for Windows Communication Foundation (WCF)-based services. With WS contracts, two major methodologies one is Code First development and other is Contract First development. The focus is mainly on the latter.

A. CONTRACT-FIRST DEVELOPMENT

A contract of a function defines the expectations and commitments of that function. Users of the function only need to know about the contract to use it.Typically,services interact with their clients by exchanging SOAP messages. Modeling these message contracts is the second step of contract-first development. That depends on which SOAP messaging format we prefer to use:RPC/Encoded,RPC/Literal,Document/Literal,Document/Literal/Wrapped

The focus is only on Document/Literal/Wrapped because it's Web Services Interoperability Organization (WS-I)-compliant. Defining a message contract has two aspects. First, we should define the structure of the SOAP body. For that we use XSD to do this and also we can use data contracts that we defined earlier. The other aspect of the message contract is defining the structure of soap headers. Headers for the messages are defined in WSDL.

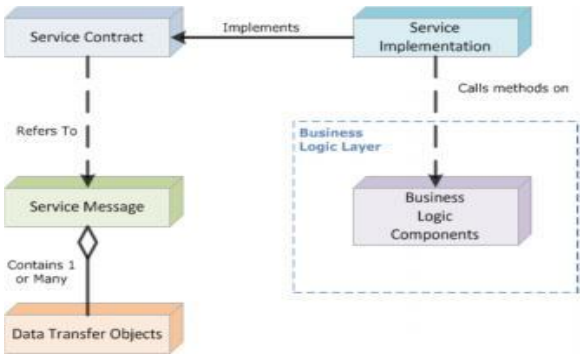

Fig 1: Contract-First Development Flow

B. DATA TRANSFER OBJECTS PATTERN

Data Transfer Objects (DTOs) gained popularity in the Java programming language. The simplest way to understand what they are is to think of them as reusable complex data types that typically represent business entities. DTOs are often confused with Domain Objects, but there are key differences between the two. DTOs do not contain any business logic whatsoever, nor do they contain any logic to perform Create, Retrieve, Update or Delete (CRUD) operations. Their purpose is to be used as convenience containers to carry information to and from consumers. They are specifically optimized for the transfer of data, but must not map directly to objects either in the domain model or database schema. They are typically comprised of primitive data types (e.g. string, double, etc.), but may contain any combination of primitive data types and complex types (i.e. other DTOs).

C. SERVICE MESSAGING PATTERN

The pattern of using Service Messages as a means of communication between service consumer and service provider is a best practice when designing services. There are some very good reasons to use this message-based style of communications:

- Promotes easier extensibility
- Allows greater control over the message itself (e.g. the ability to add custom headers and/or security policies on message headers and/or bodies).

Service Messages are containers that wrap one or more DTOs, and they specify the information that must be passed to or from service operations. Most operations will implement the Request/Response Message Exchange Pattern and will have one inbound Request Message

and one outbound Response Message, as illustrated in figure

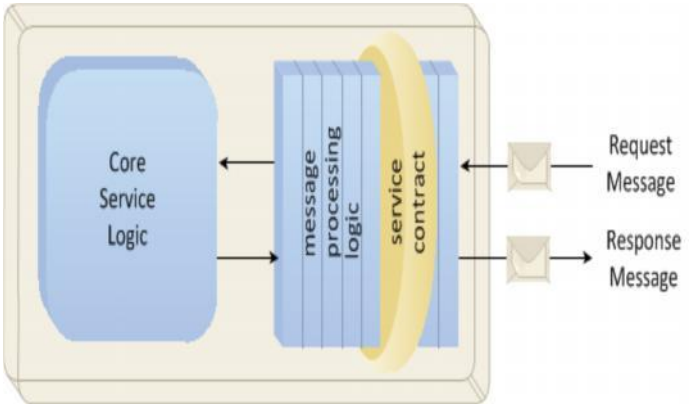

Fig 2: Message Exchange Pattern

D. SERVICE CONTRACT

The Service Contract defines the operations that a service will provide, the Service messages used in each operation, and the Message Exchange Patterns (e.g. Request/Response) used by each operation. This is the entity that clearly communicates to all potential consumers the purpose and rules for usage of the service.

E. SERVICE IMPLEMENTATION

The Service Implementation fulfills the Service Contract. This component should be viewed as nothing more than a gateway into the business logic layer, and does not expose the underlying details of how it achieves the goals of its service operations. There should not be any actual business logic embedded in the Service Implementation; instead, the Service Implementation invokes methods on business components and in many cases other service implementations. Following this pattern provides a clear separation of concerns, and each layer serves a distinct purpose.

The Service Implementation serves as both an Aggregator (i.e. service composition) and a Façade. It decides, directs and composes the business components or services that need to be called to achieve the goals of the service operation, while hiding the complexity of this from all of its consumers. Composition at the services layer is only for short lived processes (i.e. less than 15 seconds) that do not require human intervention. Figure below illustrates a simple service composition at the service layer.

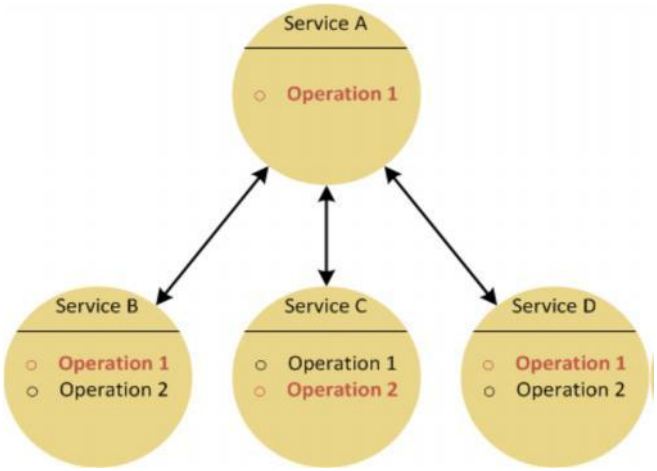

Fig 3: Service Compositions

If a service composition becomes more complex (i.e. longer in duration, requires some level of human intervention, etc.), then it should be encapsulated within a true orchestration layer such as IBM Business Process Manager.

In addition to three primary contract types -- data, message and interface contracts -- a service contract also has a policy, bindings and endpoints. Figure below summarizes which WSDL/schema constructs are used to represent different artifacts in Web service contracts.

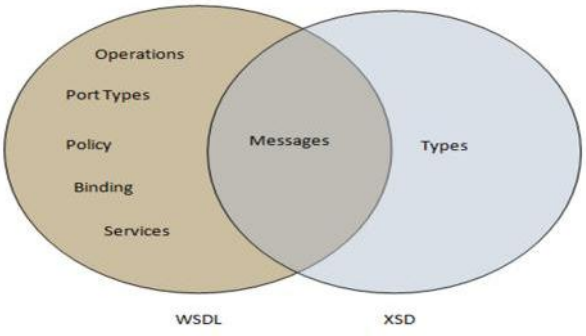

Fig 4: WSDL/Schema Constructs

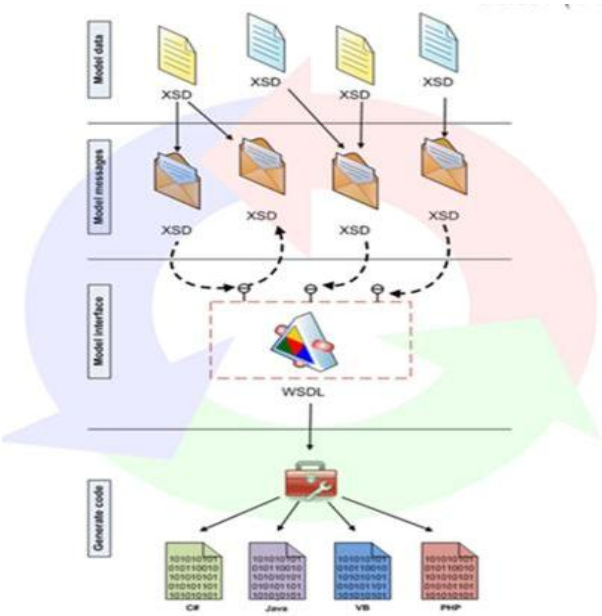

Fig 5: Contract-First Approach

E. SERIALIZATION

Serialization has been a key part of .Net since version 1. It is basically the process of converting an object instance into a portable and transferable format. The objects can be serialized into all sorts of formats. Serializing to Xml is most often used for its interoperability. Serializing to binary is useful when you want to send the object from one .Net application to another. .Net even supports the interfaces and base classes to build your own serializes. There are libraries out there to serialize to comma delimited strings, JSON, etc.

Deserialization is basically the reverse of serialization. It's the process of taking some data (Xml, binary, etc) and converting it back into an object.

XMLSERIALIZER VS. DATACONTRACTSERIALIZER

XMLSERIALIZER	DATACONTRACTSERIALIZER
1.Web service uses XMLSerializer attribute.	1.WCF uses DataContractSerializer attribute.
2. XMLSerializer does not provide better performance when compare with DataContractSerializer becauseXMLSerializer does not indicate which fields or properties of the type are serialized into XML.	2. A practical benefit of the design of the DataContractSerializer is better performance over XmlSerializer. This is because DataContractSerializer explicitly shows the fields or properties are serialized into XML.
3. XMLSerializer cannot translate the Hash Table into XML.	3. The DataContractSerializer can translate the Hash Table into XML.
4. XmlSerializer cannot serialize private members.	4. DataContractSerializer serializes private members.
5. Only the public members are serialized not the private members. Suppose we do not need any of the member to be serialized we can use [XmlIgnore] attribute	5. We can serialize a type that marked with [Serializable] attribute with DataContractSerializer. It serializes all the members (private, public) even they are marked with [XmlIgnore].

Table 1: XmlSerializer vs. DataContractSerializer

The WCF Service can be designed using both ways of Sterilization techniques but with DataContractSerilization there is significance performance gain. As a part of research work comparative study has been carried out to present the report. The comparison has detailed out in next section.

VI. RESTFUL WCF SERVICE

Windows Communication Foundation (WCF) is an SDK for developing and deploying services on Windows. WCF provides a runtime environment for your services, enabling you to expose CLR types as services, and to consume other services as CLR types.

A. WHAT IS REST?

Based on the Roy Fielding theory "Representational State Transfer (REST), attempts to codify the architectural style and design constraints that make the Web what it is. REST emphasizes things like separation of concerns and layers, statelessness, and caching, which are common in many distributed architectures because of the benefits they provide.

Actually only the difference is how clients access our service. Normally, a WCF service will

use SOAP, but if you build a REST service, clients will be accessing your service with a different architectural style (calls, serialization like JSON, etc.). REST uses some common HTTP methods to insert/delete/update/retrieve (CRUD operations) information which is below:

- GET - Requests a specific representation of a resource
- PUT - Creates or updates a resource with the supplied representation
- DELETE - Deletes the specified resource
- POST - Submits data to be processed by the identified resource

B. WHY AND WHERE TO USE REST?

During research, the attempt made to build a sample service which can be accessed by heterogeneous language/platform/system. It can be used by iPhone, Android, Windows Mobile, .NET web application, JAVA or PHP. Using web service, it was bit complex for me to expose it to everyone using uniform system. Then I decided to use REST, which was easily espoused over cloud. Below are some points which will help you to understand why to use the RESTful services.

- It does not contain SOAP header so less overhead and better performance.
- HTTP operations DELETE, PUT and GET support less duplication, easy implementation, and testing and increase readability.
- XML is not really required to implement.

C. WCF SOAP & WCF RESTFUL SERVICE

WCF is the Microsoft framework for building applications that communicate over a network, regardless of the style or protocol. The concept behind WCF was to create a framework that was extensible and pluggable so that developers could learn one programming and configuration model and be able to apply those skills to many different kinds of distributed systems. While it is true that much of WCF is geared toward RPC (using SOAP), it actually has had the ability to expose and consume REST services since it was first released as part of the .NET Framework 3.0. What it lacked was a programming model needed to make using REST with WCF easy. There also were some pieces of infrastructure that you had to build to make REST work with the .NET Framework 3.0. Both a programming model and those pieces of infrastructure were added to WCF in the .NET Framework 3.5 in the System.ServiceModel.Web assembly.

And the .NET Framework 3.5 SP1 added a few small improvements as well.

The main difference between SOAP and REST is the fact that while REST is implemented directly on top of the HTTP protocol, SOAP introduces an abstraction layer (SOAP messaging), that can be implemented on top of any transport. Standardized SOAP bindings currently exist for HTTP, SMTP and JMS, but non-standard bindings have been implemented for other transport solutions. This additional abstraction layer that provides decoupling between existing transports and SOAP-based implementations is the root cause of major differences between SOAP and REST Web Services.

VII.SIMULATION RESULTS

The WCF Service can be designed using both ways of Sterilization techniques (XML Serialization and Data Contract Serialization) but with Data Contract Serialization there is significance performance gain as mentioned in the previous section. As a part of research work a simple EmployeeService has been created to perform performance using XMLSerilizer and DataContarctSerilizer.

Also the comparative study has been carried out for performance analysis of RESTful WCF Service and WCF WSDL Service carried out during research work and it was found that RESTful WCF Service returns the smaller size response object than WCF WSDL Service.

PERFORMANCE ANALYSIS – XMLSERIALIZER & DATACONTRACTSERIALIZER

As a part of research work a simple EmployeeService has been created to perform comparative study of performance using

XMLSerilizer and DataContarctSerilizer. This Service just contains one operation RetrieveEmployeeDetails to retrieve employee details based on provided input parameters. It has been noticed there is significance performance gain in designing WCF service in DataContarct Serialization. The performance gain is about 30%. The result can vary based on data volume and complexity of the operation. In research work medium complexity service has been chosen to consolidate the result data.

VII. CONCLUSION AND FUTURE WORK

It has been clearly observed that by designing/implementing SOA using Contract-First Approach with Data Contract Sterilization saves time and money if SOA has to implement using SOAP WCF Service. With the comparative performance analysis it has been proved. Also with this is approach WS* security concern can achieved properly.

REFERENCES

[1] Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., & Holley, K. “SOMA: A Me-thod for Developing Service-Oriented Solutions”. 2008.
[2] Brown, Kyle. & Ellis, Michael. “Best Practices for Web Services Versioning: Keep your Web Services Current with WSDL and UDDI.”. 2004.
[3] Eric Newcomer, “Understanding Web Services: XML, WSDL, SOAP, and UDDI”. Addison-Wesley Professional, 2002
[4] Thomas Erl, “Service-Oriented Architecture: Concepts, Technology, And Design”, 2005
[5] Mark Pilgrim, "HTML5: Up and Running". 2010.
[6] Matthew MacDonald, “ASP.Net: The Complete Reference”. 2002.
[7] http://msdn.microsoft.com/
[8] http://www.w3schools.com/