ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 11 November, 2014 Page No. 9223-9226

A Service-oriented Architecture Based on Windows Communication foundation for Health Monitoring and Tracking”

Maheep Sharma, Mrs. Pooja Sapra
WCTM, Farukhnagar,
Gurgaon,India

ABSTRACT

WCF (Windows Communication Foundation) developed by Microsoft, is an ideal SOA (Service-Oriented Architecture) implementation platform. WCF is designed using service oriented architecture principles to support distributed computing where services have remote consumers. SOA generally provides a way for consumers of services, such as web-based applications.SOA is applied to build the web-based management system for Lifeline due to its flexibility and reusability. All services are programmed on .NET. The results of test system which is constructed on .NET show that Lifeline is valuable for practice. The services are deployed, discovered and consumed as a collection of endpoints. A WCF client connects to a WCF service via an Endpoint. Web Services are applications that can be published, located, and invoked across the Internet. Web Services are implemented by SOA .Web Services are applications that can be published, located, and invoked across the Internet. WCF services provide better reliability and interoperability.

INTRODUCTION

INTRODUCTION OF SERVICE ORIENTED ARCHITECTURE (SOA)

In Distributed Software Engineering, a Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of inter operable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes. SOA design principles are used during the phases of systems development and integration.

SOA generally provides a way for consumers of services, such as web-based applications, to be aware of available SOA-based services. For example, several disparate departments within a company may develop and deploy SOA services in different implementation languages; their respective clients will benefit from a well-defined interface to access them. XML is often used for interfacing with SOA services, though this is not required. SOA defines how to integrate widely disparate applications for a Web-based environment and uses multiple implementation platforms. Rather than defining an API, SOA defines the interface in terms of protocols and functionality. An endpoint is the entry point for such a SOA implementation.

Service-orientation requires loose coupling of services with operating systems and other technologies that underlie applications. SOA separates functions into distinct units, or services, which developers make accessible over a network in order to allow users to combine and reuse them in the production of applications. These services and their corresponding consumers communicate with each other by passing data in a well-defined, shared format, or by coordinating an activity between two or more services.

A service-oriented architecture is a style of design that guides all aspects of creating and using business services throughout their life cycle. An SOA is also a way to define and provision an infrastructure to allow different applications to exchange data and participate in business processes, regardless of the operating systems or programming languages underlying those applications.

An SOA can be thought of as an approach to building systems in which business services (i.e., the services that an organization provides to clients, customers, citizens, partners, employees, and other organizations) are the key organizing principle used to align IT systems with the needs of the business. Services are unassociated, loosely coupled units of functionality that have no calls to each other embedded in them. Each service implements one action, such as filling out an online application for an account, or viewing an online bank statement, or placing an online booking or airline ticket order. Rather than services embedding calls to each other in their source code, they use defined protocols that describe how services pass and parse messages using description metadata. Underlying and enabling all of this requires metadata in sufficient detail to describe not only the characteristics of these services, but also the data that drives them. Programmers have made extensive use of XML in SOA to structure data that they wrap in a nearly exhaustive description-container. Analogously, the Web Services Description Language (WSDL) typically describes the services themselves, while the SOAP protocol describes the communications protocols. Whether these description languages are the best possible for the job, and whether they will become/remain the favorites in the future, remain open questions. As of 2008 SOA depends on data and services that are described by metadata that should meet the following two criteria:

1. The metadata should come in a form that software systems can use to configure dynamically by discovery and incorporation of defined services, and also to maintain coherence and integrity. For example, metadata could be used by other applications, like a catalogue, to perform auto

discovery of services without modifying the functional contract of a service.

2. The metadata should come in a form that system designers can understand and manage with a reasonable expenditure of cost and effort.

SOA aims to allow users to string together fairly large chunks of functionality to form ad hoc applications that are built almost entirely from existing software services. The larger the chunks, the fewer the interface points required to implement any given set of functionality; however, very large chunks of functionality may not prove sufficiently granular for easy reuse. Each interface brings with it some amount of processing overhead, so there is a performance consideration in choosing the granularity of services. The great promise of SOA suggests that the marginal cost of creating the nth application is low, as all of the software required already exists to satisfy the requirements of other applications.

Each SOA building block can play one or both of two roles:

1. **Service provider** - The service provider creates a web service and possibly publishes its interface and access information to the service registry. Each provider must decide which services to expose, how to make trade-offs between security and easy availability, how to price the services, or (if no charges apply) how/whether to exploit them for other value. The provider also has to decide what category the service should be listed in for a given broker service and what sort of trading partner agreements are required to use the service. It registers what services are available within it, and lists all the potential service recipients. The implementer of the broker then decides the scope of the broker. Public brokers are available through the Internet, while private brokers are only accessible to a limited audience, for example, users of a company intranet. Furthermore, the amount of the offered information has to be decided. Some brokers specialize in many listings. Others offer high levels of trust in the listed services. Some cover a broad landscape of services and others focus within an industry. Some brokers catalog other brokers. Depending on the business model, brokers can attempt to maximize look-up requests, number of listings or accuracy of the listings. The Universal Description Discovery and Integration (UDDI) specification defines a way to publish and discover information about Web services. Other service broker technologies include (for example) EBXML(Electronic Business using extensible Markup Language).
2. **Service consumer** - The service consumer or web service client locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its web services. Whichever service the service-consumers need, they have to take it into the brokers, then bind it with respective service and then use it. They can access multiple services if the service provides multiple services.

Characteristics of SOA

Service-oriented architectures have the following key characteristics:

- SOA services have self-describing interfaces in platform-independent XML documents. Web Services Description Language (WSDL) is the standard used to describe the services.
- SOA services communicate with messages formally defined via XML Schema (also called XSD). Communication among consumers and providers or services typically happens in heterogeneous environments, with little or no knowledge about the provider. Messages between services can be viewed as key business documents processed in an enterprise.
- SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Definition, and Integration (UDDI)is the standard used for service registry.
- Each SOA service has a quality of service (QOS) associated with it. Some of the key QOS elements are security requirements, such as authentication and authorization, reliable messaging, and policies regarding who can invoke services.

Benefits of SOA

Some enterprise architects believe that SOA can help businesses respond more quickly and more cost-effectively to changing market conditions. This style of architecture promotes reuse at the macro (service) level rather than micro (classes) level. It can also simplify interconnection to – and usage of – existing IT (legacy) assets.

With SOA, the idea is that an organization can look at a problem holistically. A business has more overall control. Theoretically there would not be a mass of developers using whatever tool sets might please them. But rather there would be a coding to a standard that is set within the business. They can also develop enterprise-wide SOA that encapsulates a business-oriented infrastructure. SOA has also been illustrated as a highway system providing efficiency for car drivers. The point being that if everyone had a car, but there was no highway anywhere, things would be limited and disorganized, in any attempt to get anywhere quickly or efficiently

Short-term benefits of implementation:

- Enhances reliability
- Reduces hardware acquisition costs
- Leverages existing development skills
- Accelerates movement to standards-based server and application consolidation
- Provides a data bridge between incompatible technologies

Long-term benefits of implementation:

- Provides the ability to build composite applications
- Creates a self-healing infrastructure that reduces management costs
- Provides truly real-time decision-making applications
- Enables the compilation of a unified taxonomy of information across an enterprise and its customer and partners

Benefits from the perspective of Business Value

- Ability to more quickly meet customer demands
- Lower costs associated with the acquisition and maintenance of technology

- Management of business functionality closer to the business units
- Leverages existing investments in technology
- Reduces reliance on expensive custom development

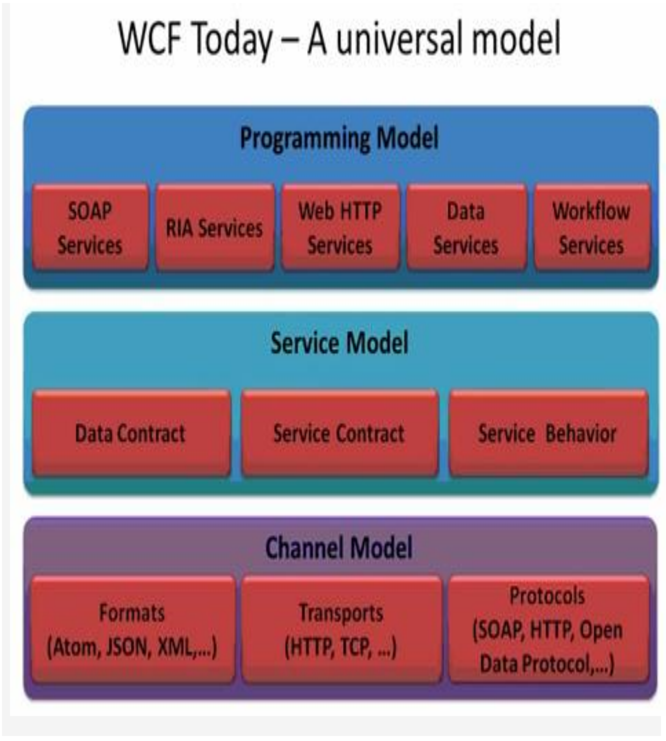
FUTURE SCOPE

Glenn Block, a Windows Communication Foundation (WCF) Program Manager, said during an online webinar entitled "WCF, Evolving for the Web" that Microsoft's framework for building service-oriented applications is going to be refactored radically, the new architecture being centered around HTTP.

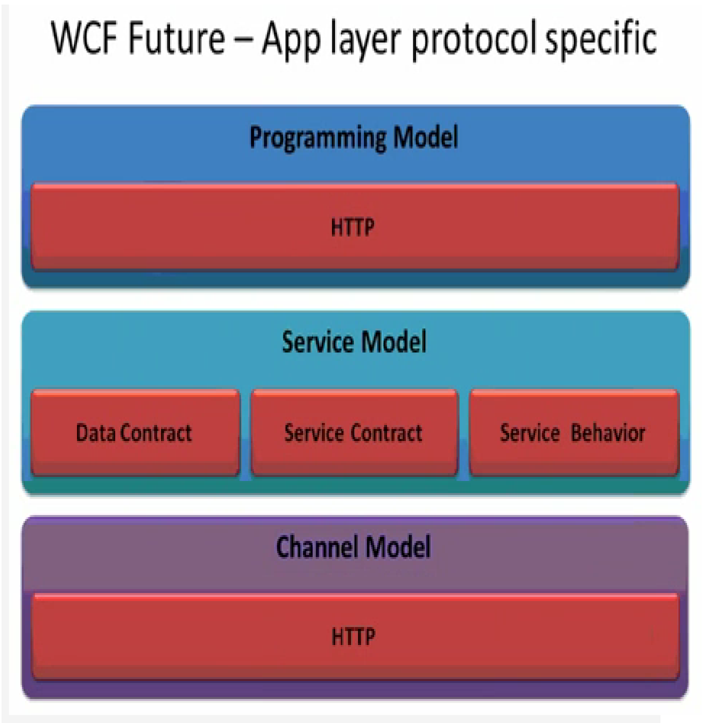
Block started the online session by summarizing the current trends in the industry:

- a move to cloud-based computing
- a migration away from SOAP
- a shift towards browsers running on all sorts of devices
- an increase in the adoption of REST
- emerging standards like OAuth, WebSockets

He mentioned that the current architecture of WCF is largely based on SOAP as shown in this slide:



One of the key features of WCF is support for multiple transports (HTTP, TCP, named-pipes) under the same programming model. Unfortunately when it comes to HTTP, a lot of HTTP goodness (scale, content negotiation) is lost because WCF treats it as a transport. So Block is looking forward to see WCF supporting HTTP as a first class application protocol with simple and flexible programming model as depicted on the following slide:



HTTP was introduced in .NET 3.5, allowing the creation of services accessed via HTTP, but "it does not give access to everything HTTP has to offer, and it is a very flat model, RPC oriented, whereas the Web is not. The Web is a very rich set of resources," according to Block. Instead of retrofitting the current WCF to work over HTTP, Block considers WCF should be re-architected with HTTP in mind using a RESTful approach.

WCF will contain helper APIs for pre-processing HTTP requests or responses, doing all the parsing and manipulation of arguments, encapsulating the HTTP information in objects that can be later transferred for further processing. This will relieve the user from dealing with HTTP internals directly if he wants to. This feature will also present a plug-in capability for media-type formatters of data formats like JSON, Atom, OData, etc. WCF will support some of them out of the box, but the user will be able to add his own formatters.

The new WCF is already being built, Block demoing sample code using it, but he mentioned that the feature set and what WCF is going to look like is not set in stone. They will publish an initial version of the framework on CodePlex in the near future for the community to be able to test and react, shaping the future of WCF. More details are to come during PDC 2010.

Update

CONCLUSION

We asked Glenn Block what it is going to happen to the other protocols, especially SOAP. His answer was that WCF is going to fully support the existing stack, and the current development is meant to evolve WCF to fully support HTTP without renouncing to anything WCF has so far.

A WCF Community website is now set up on CodePlex for all those interested in the evolution of WCF.
Glenn Block presented a more detailed view on the future of WCF and how it relates to current investments in Microsoft's SOA technology at PDC 2010.

REFERENCES

[1] Andrew Troelsen, Pro C# 2008 and the .NET 3.5 Platform, 2006, vol 2, pp.802-803.

[2] Adjuncts,Web Services Description Language,2006, vol 3, pp 8

[3] Adjuncts ,Web Services Description Language (WSDL) ,2006, vol 2, pp.72

[4] Bell, Michael (2008). "Introduction to Service-Oriented Modeling".Service-Oriented Modeling: Service Analysis, Design, and Architecture, 2009, vol 2, pp. 3

[5] Brayan Zimmerli, Business Benefits of SOA, 2009, vol 3, pp.66

[6] Beaglehole R, Dal Poz MR: Public health workforce: Challenges and policy issues.
Human Resources for Health 2003, 1-4

[7] Bell_, Michael , SOA Modeling Patterns for Service-Oriented Discovery and Analysis, 2010,vol 5, pp. 390

[8] Benslimane, Djamal; Schahram Dustdar, and Amit Sheth , Services Mashups: The New Generation of Web Applications, 2008, vol 4, pp.13-15

[9] Bowen crane,SOAP Version 1.2 Part 1: Messaging Framework W3C, 2007, vol 1, pp.67

[9] Benoît Marchal, Uche Ogbuji, Tutorial: XML messaging with SOAP,2008, vol 2,pp 5

[10] Chris Peiris and Dennis Mulder, Pro WCF: Practical Microsoft SOA Implementation, 2010,vol 2, pp.4.

[11] Chris Peiris, Pro WCF:Microsoft SOA Implementation, 2006, vol 1, pp.29.

[12] Channabasavaiah, Holley and Tuggle, Migrating to a service-oriented architecture, 2008, vol 2, pp.125-127

[13] David Chappell, Chappell & Associates, Introducing Windows Communication Foundation,2009. Vol 2, pp,105-109

[14] Health practitioners' and health planners' information needs and seeking behaviour for decision making in Uganda. International Journal of Medical Informatics 2005,714-721

[15] Jiang Jingnan, WCF Technical Analysis,2010, vol 2, pp.482-504.

[16] John Sharp, Microsoft Windows Communication Foundation Step by Step, Microsoft 2011,vol 1, pp.1-6.

[17] Justin Smith, Inside Microsoft Windows Communication Foundation, Microsoft, 2007 , vol 1,pp.17.

[18] Microsoft, Microsoft2008, vol 5, pp 26-31, 72-77, 86, 222, 345.

[19] National Coalition on Health Care accessed in Oct. 2011.

[20] O'Reilly Media, Juval Löwy, Programming WCF Services,2007, vol 2, pp.40-51.

[21] Olson, Mike, Ogbuji, Uche ,The Python Web services developer: Messaging technologies compared ,2003, vol 1, pp.333

[22] Pablo Cibraro, Kurt Claeys, Fabio Cozzolino, Johann Grabner: Professional WCF 4: Windows Communication Foundation with .NET 4, Wrox , 2010, vol 4, pp 56

[23] Radhakrishnan R ,Introduction to WCF Programming,2008,vol 2, pp 45-48

[24] Scott Klein, Windows professional WCF , 2007, vol 1,pp.15.

[25] Steve Resnick, Richard Crane, Chris Bowen, Essential Windows Communication Foundation (WCF): For .NET Framework 3.5, 2008 , vol 2, pp.178