Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 4, April 2017, pg.97 – 103

DISTRIBUTED APPLICATION USING SOA TO HANDLE ERRORS

Ms. Meenal Raut
meenalraut14@gmail.com
Department of M. Tech(CSE), Nuva College of Engg. & Tech., Nagpur

Prof. Shyam Dubey
shyam.nuva@rediffmail.com

Naziya Pathan
tpcell.nuva@mail.com
Nuva College of Engg. & Tech., Nagpur

Abstract-- Service-Oriented Architecture is a popular design paradigm to simplify the process of creating and maintaining distributed systems. This architecture scheme is becoming more and more popular in web-based applications as well as other distributed systems such as Amazon and Google web services. This paper introduces the results of experimental analysis of the SOA-specific exceptions of Services, implemented by use of different development toolkits and also discusses how SOA is used to simplify the process of servicing the various types of software users. Finally, conclude with the benefits of designing Service Oriented Architecture systems.
Keywords-- SOA, WCF, Services, SOAP, WSDL.

I. INTRODUCTION

Service-Oriented Architecture (SOA) has been introduced for solving the problem of ensuring effective, reliable and secure interaction of complex distributed systems. SOA assumes that such systems are constructed from separate functional application modules (services) that have interfaces, defined by common rules (WSDL - description), and a dedicated invoke mechanism (SOAP messages).

Software trends show that maintainability is paramount in new software design. User's requirements and expectations are continually changing and need to be met. SOA is designed
specifically to satisfy that need[9]. SOA has the intent to have several autonomous services loosely coupled together by a central controller that will coordinate the services for the users. The architecture naturally facilitates design goals such as modifiability, reusability, reliability and much more.

II. BENEFITS

The benefits of using SOA stem mostly from the loose coupling between the individual services. Each service is added to a registry separately, and therefore does not affect how other services operate. The separation of functional blocks promotes an easy system to understand, as developers and testers only need to study code for the individual service, a much smaller scale compared to the entire system. The meaning of processes also become clearer, as services abstracted away the internal details, and only the I/O interfaces need to be grasped. Since services are very easily understood, the system as a whole, very simple to debug and modify. This is an important contributing factor to the systems reliability. Since the services of the system are easily testable, this reduces the chances of the system crashing. Even in the event that a service crashes, SOA is fairly fault tolerant [6]. The system should continue to function properly despite a failed service; it should simply allow the user to query for alternative services that may accomplish the required task.

III. SOA STRUCTURE

The basic assumption of SOA is that there are many consumers that require services. In literature, consumers are also referred to as clients or customers. These terms are used interchangeably here. On the other side, there are many providers that provide services on the network. These two groups have to be linked together in a dynamic and adaptive way. This is usually done by a service broker [7], [5].

Service providers register their services at the broker (registry), service consumers request a service from the service broker, which returns a known provider for the requested service. Consumer and provider agree on the semantics. The consumer then binds himself to the service provider and uses the service. The structure of SOA architecture is shown in Figure. 1.

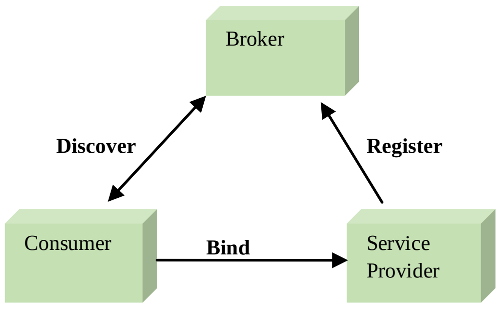

Figure 1: SOA Structure [1]

A. Services

The Service is a unit of software that will perform some functionality for the client. The service can provide meaningful functionality in itself in order to promote reuse. The service must be defined by its ability to communicate with a client, and it must advertise these details in a way that the client can understand. Other important feature of a SOA is low coupling between the client and the service. The IDEs, such as eclipse, have features that allow you to select a service you wish to use in your client, and then the IDE creates client side, ”Proxy or Stub classes”, allowing the client to communicate with service. This practice allows for fast and easy SOA development, but may cause trouble down the line if the client needs to change the service it uses [6].

B. WSDL

Before a service can be discovered and consumed by the client, it must be published with a clearly defined contract outlining its capabilities, as well as input and output interfaces. This is done with the Web Services Description Language (WSDL). WSDL is an XML formatted machine readable language, designed by Microsoft and IBM.[9] It is used for describing how the service can be called, what parameters the service expects, and what kind of data is re-turned. It serves a similar purpose as a method signature in a conventional programming language. [6]

A WSDL document has 4 main sections [10]:

Types:	The data types used by the web service
Message:	The messages used by the web service
Port Type:	The operations performed by the web service.
Binding:	The communication protocols used by the web service.

The Types section defines what data types will be used in the web service. It simply defines variable names and assigns a type such as string or int to them. Variables can be any complex data type that has been defined in XML. Otherwise types must be either a primitive data type, string or an array of primitives.

The Message section defines how the input and output messages are formatted. For each message that will be sent or received by the service, the WSDL document gives the message a name, and declares what parameters will be sent.

The Port Type section declares what operations can be per-formed by the service. Each entry in the Port Type section defines a name for an operation, the message that is needed for input, and the message that will be sent as output. Operations don't always have a request-response structure.

The Binding section defines the message protocol details for a web service. Each Binding entry defines how an operation from Port Type will be accessed using SOAP.

C. SOAP

In order to send and receive the messages defined in WSDL, a standard was needed to allow Internet transfer of these messages over HTTP. SOAP is one standard that can be used. Originally an acronym for Simple Object Access Protocol, It is no longer referred to as an acronym, and is now simply the soap protocol.[9] Soap messages are ordinary XML documents that are sent over HTTP. Each SOAP message contains 4 main sections: [9]
Envelope: A small section identifying the document as a SOAP message
Header: Small section that carries relevant header information
Body: Contains the call and response information
Fault: An element that defines errors and status information

The Envelope element is the root XML element of the message. It has a field to identify the XML as a SOAP message, and another field, containing a URI that details the SOAP encoding used in the message.

The Header section contains some optional application specific information such as authentication and payment. The header section is not mandatory, but it does have a field that can be set to force whatever system receives the message to parses it.

The SOAP Body element contains the actual message that is being sent. The client or service must format this message in a way that is described in the WSDL document. The Fault element is an optional section that indicates error messages. SOAP has some default fault codes that can be sent to indicate whether the problem was on the client side with a malformed message, or on the server side, where the service couldn't respond for any number of reasons.

IV. DISTRIBUTED APPLICATION TOOLKITS

A. Net IDE

Handling Error for SOA application used these steps:

1) Creating a service application.

For creating a service application used following steps, they are:

a) To establish contract between client & the service, use WCF Service Contract & include Service Contract, Operation Contract & Data Contract (i.e. Interface). All these contract can include by adding the reference System.ServiceModel
b) Add WCF Methods/ Operation Contract
c) Create Class Library and add references Service Model & Contract Library.
d) Add one more class which will be the implemented class which is inherited from interface.
e) Add one more class Account.
f) Create Service Provider & add references, Service Model, Contract Library & Implementation.
g) To publish end point, we need App.config which can be created using MS service configuration editor i.e. WCF service configuration editor.
h) Create client application.

App.config :

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.serviceModel>
    <client>
      <endpoint address="http://localhost:9095/BankService"
     binding="wsHttpBinding"
     bindingConfiguration=""   contract="BankContractLib.IBank" name="BankEP" kind=""
          endpointConfiguration="">
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

2) Checking the exception in service

a) The exception in SOA architecture is BasicHttpBinding.

b) To identify, AccID is valid or not, & see the exception on client side. This exception message is very big.

3) Handling the exception

a) The BasicHttpBinding exception is resolve using wsHttpBinding. BasicHttpBinding is a stateless protocol, it doesn't store any previous state. Therefore, change the binding to wsHttpBinding, which is meant for statefull binding.

b) To avoid big message, log it on server side.

B. NetBeans IDE/Sun Java SystemApplication Server bundle

Web services are Web-based enterprise applications that use open, XML-based standards and transport protocols to exchange data with calling clients. NetBeans IDE1, which is a powerful integrated development environment for developing applications on Java platform, supports web services technologies through the Java Platform Enterprise Edition. Sun Java System (SJS) Application Server2 is the Java EE implementation at Sun Microsystems. NetBeans IDE with SJS Application Server support JSR-109, which is a development paradigm that is suited for J2EE development, based on JAX-RPC. NetBeans IDE provides wizards to create web services and web service clients[4].

C. IBMWebSphere Software Developer Kit for Web Services

IBMWebSphere Software Developer Kit for Web Services (WSDK) is an integrated kit for creating, discovering, invoking, and testing Web services. WSDK can be used with the Eclipse IDE. Eclipse provides a graphical interactive development environment, which provides tools for building and testing Java applications. WSDK adds to the standard Eclipse package the tools relating to Web services, making it suitable for building Web services. Supporting the latest specifications for Web services including WS-Security, SOAP, WSDL, and UDDI, WSDK enables to build, test, and deploy Web services on industry-leading IBM WebSphere Application Server. Functionality of the WSDK has been incorporated into the IBMWebSphere Studio family of products [4].

V. ERRORS CORRESPONDENCE ANALYSIS

We were experimenting with simple web services providing arithmetic operations which were deployed on two application services: SJS AppServer and IBM WebSphere. To analyze features of exception raising mechanisms and performance implications in SOA architecture depending on Web Services development toolkit used, we injected errors in the testbed services and client applications, and also simulated network failures. The results of our practice with web services exceptions are shown in Table 2. This table describes a relationship between errors/failures and raised exceptions on different application platforms.

As it was discovered, some errors and failures raise the same exception so we can't recognize exact exception source. There are several groups of unrecognized errors and failures (see Table 2): 1 and 2 (Sun); 3 and 6 (Sun); 4 and 5 (Sun); 1, 2 and 5 (IBM); 3 and 6 (IBM). Some client-side binding errors (11 – "Error in web service name", 12 – "Error in service port name") don't raise exceptions and don't affect service output. It is possible because web service is actually invoked by address location, whereas service name and port name are supplementary information.

Moreover, web service, developed by using IBM WSDK and deployed on IBM WebSphere application server, tolerates such binding errors inside: 10 - "Error in Target Name Space", 14 - "Output parameter type mismatch", and 16 - "Error in name of input parameter". These abilities are provided by features of WSDL-description and built-in function of automatic type conversion. Errors in name of input parameter are tolerated because true order of parameters has a priority over coincidence of parameters' names for IBM implementation of web services.

VI. REAL-WORLD EXAMPLES OF SERVICES

A. Google

Any developer can make use of Google's API of services, some are free while others come at a cost. The Google Maps AIP for example, is free for all developers for the first 25000 visitors but after that point the developer is charged $4USD and again for every 1000 visitors after that point [7].

Although when the Google Code project was first launched in 2005 it mainly contained SOAP services, between 2006 and 2009, Google made the switch from providing SOAP services to rest. Why they made the switch was not made clear, however some speculation has been made that the decision was made to reduce load costs.

B. Cloud

Cloud services have become somewhat of a buzzword over recent years, in reality; it is simply another way of looking at services. The"cloud" is simply a synonym for the inter-net; cloud computing and other services are simply dedicates services instead of services which are add-ons to another application (such as the ones we saw previously in this section). Amazon web services for example[2], provides the ability to store data by simply making a request which includes a file to store the file, then making a request to retrieve the file.

Cloud computing normally encompasses three distinct types of services Software as a Service, Platform as a Service, and Infrastructure as a Service [3].

Software as a Service:	The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited userspecific application configuration settings.
Platform as a Service:	The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.
Infrastructure as a Service:	The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

VII. CONCLUSIONS

Error handling is widely used as the basis of forward error recovery procedure in service-oriented architecture. Effectiveness of exception handling depends on the features of raising exceptions and propagation mechanisms.

REFERENCES

[1] Ms. Meenal Raut, "Survey on Error Handling using Service Oriented Architecture ", IJSTEVSI8022 No. 4271, volume 3, Isuue 8, ISSN 2349-784x, Feb - 2017 .

[2] Amazon web services. http://aws.amazon.com/, 2012.

[3] W. Voorsluys, J. Broberg, and R. Buyya. Introduction to cloud computing. Cloud Computing, pages 1–41, 2011.

[4] Anatoliy Gorbenko, Elyasi Komari Iraj, Vyacheslav S. Kharchenko, Alexey Mikhaylichenko , "Exception Analysis in Service-Oriented Architecture". Department of Computer Systems and Networks, National Aerospace University "KhAI", Ukraine Oct-2010.

[5] G. Denaro, M. Pezz´e, D. Tosi, and D. Schilling, "Towards self-adaptive service-oriented architectures," in TAV-WEB '06: Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications. New York, NY, USA: ACM Press, 2006, pp. 10–16.

[6] L. Srinivasan and J. Treadwell. An overview of service-oriented architecture, web services and grid computing. HP Software Global Business Unit, 2, 2005.

[7] W3C, "Web services architecture," Febuary 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/

[8] Avizienis A., Laprie J.-C., Randell B., Landwehr C. "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Trans. on Dependable and Secure Computing. Vol.1, № 1. – P. 11-33, 2002.

[9] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. Internet Computing, IEEE, 6(2):86–93, 2002.

[10] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1, march 2001. URL: http://www.w3.org/TR/wsdl, 2001.

TABLE 2
EXCEPTIONS RAISED BY DIFFERENT TYPES OF ERRORS AND FAILURES[4]

Sr. No.	**Type of error/failure**	**Exception**	
		Sun Microsystems WS Toolkit	*IBM WS Toolkit (WSDK)*
1	Network connection break-off	HTTP transport error: java.net.UnknownHostException: loony.xai12.ai	faultCode: {http://websphere.ibm.com/webservices/ } Server.generalException
2	Domain Name System (DNS) is down	HTTP transport error: java.net.UnknownHostException: loony.xai12.ai	faultCode: {http://websphere.ibm.com/webservices/ } Server.generalException
3	Lost of packet with client request or service response	Waiting for response during too much time (more than 2 hours) without exception	faultCode: {http://websphere.ibm.com/ webservices/}Server.generalException. faultString: java.io.InterruptedIOException: Read timed out
4	Host unavailable (off-line)	HTTP Status-Code 404: Not Found - /WS/WSCalc	faultCode: {http://websphere.ibm.com/ webservices/}HTTP (404)Not Found
5	Application Server is down	HTTP Status-Code 404: Not Found - /WS/WSCalc	faultCode: {http://websphere.ibm.com/webservices/ } Server.generalException
7	System error during Processing (“Division by Zero”)	java.rmi.ServerException: JAXRPC. TIE.04: Internal Server Error (JAXRPCTIE01: caught exception while handling request: java.lang. ArithmeticException: / by zero)	faultCode: {http://websphere.ibm.com/webservices/ } Server.generalException faultString: java.lang.Arithmetic Exception: / by zero
8	Calculation error during Processing (“Operand Type Mismatch”)	java.rmi.ServerException: JAXRPC.TIE.04: Internal Server Error (JAXRPCTIE01: java.lang.NumberFormatException: For input string: "578ER")	faultCode: {http://websphere.ibm.com/webservices/ } Server.generalException faultString: java.lang.Number FormatException: 5ER
9	Application error raising user exception	java.rmi.RemoteException: ai.xai12.loony.exception. UserException	faultCode: {http://websphere.ibm.com/ webservices/}Server.generalException faultString: (13)UserException
10	Error in Target Name Space	java.rmi.RemoteException: JAXRPCTIE01: unrecognized operation: {urn:WSStREring/wsdl} gluingstring	OK - Correct output without exception
11	Error in web service name	OK - Correct output without exception	OK - Correct output without exception
12	Error in service port name	OK - Correct output without exception	OK - Correct output without exception
13	Error in service operation’s name	java.rmi.RemoteException: JAXRPCTIE01: unrecognized operation: {urn:WSNumeric/wsdl}getMRTult	faultCode: {http://websphere.ibm.com/ webservices/}Server.generalException faultString: WSWS3277E: Error: No such