IOSR Journal of Computer Science (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727
PP 78-83
www.iosrjournals.org

Comparative Study of Languages & SOA Specific Exception Handling

Vinod S. Patil, Prof. Sulabha V. Patil
Department of M. Tech(CSE), Abha Gaikwad-Patil College of Engg., Nagpur, India.
Department of M. Tech(CSE), Abha Gaikwad-Patil College of Engg., Nagpur, India.

ABSTRACT *:Service-Oriented Architecture (SOA) is a popular design paradigm to simplify the process of creating and maintaining distributed software systems. Exception handling is one of the powerful means of achieving high dependability and fault-handling in service-oriented architecture. This paper introduces the comparative study of exception handling related to languages & SOA specific results. This result is due to experimental analysis of the SOA-specific exceptions and fault-tolerance of Web Services. It is 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, we conclude with the benefits of our fault taxonomy for designing and testing SOA-based systems.*

Keywords - *Error Handling, Services, SOA, SOAP, WSDL.*

I. INTRODUCTION

An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. Writing a good computer program includes proper error handling and recovery. Exception mechanism provides simple, streamlined, organized approach to deal with the exception.

a. Service-Oriented Architecture

Computer-based infrastructures are a necessity for many companies to handle their daily work today. The IT system infrastructure of most companies is based on distributed systems, which consist of multiple interdependent computers connected by a network. A common design paradigm for distributed systems is Service-Oriented Architecture (SOA) [1]. The concept of 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).

1.2 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. 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 [2], [3]. 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.

1.3 Objectives

The main objective of this topic is to[4]:

- To propose a reliable exception handing technique for SOA application.
- Adding more exceptions, handlings and rules into our system to complete the strategy.
- Include the creation of a test environment and the development of appropriate fault handling mechanism in SOA.

II. WEB SERVICES DEVELOPMENT TOOLKITS

2.1 Microsoft Visual Studio IDE (.Net)

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms or WPF applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

2.1.1 WCF Programming

Windows Communication Foundation (WCF) is a framework for building service-oriented applications (SOA). As its name suggests, WCF provides the .NET Framework with a basis for writing code to communicate across components, applications, and systems. The below steps presents the fundamentals for creating Windows Communication Foundation (WCF) applications [5].

Basic Programming Lifecycle

Windows Communication Foundation (WCF) enables applications to communicate whether they are on the same computer, across the Internet, or on different application platforms. The basic tasks to perform are, in order[4]:

- Define the service contract. A service contract specifies the signature of a service, the data it exchanges, and other contractually required data.
- Implement the contract. To implement a service contract, create a class that implements the contract and specify custom behaviors that the runtime should have.
- Configure the service by specifying endpoints and other behavior information.
- Host the service.
- Build a client application.

Designing and Implementing Services

This topic provides a high level conceptual orientation to designing and implementing WCF services. Before designing and implementing your WCF application, it is recommended that you:

- Understand what a service contract is, how it works, and how to create one.
- Understand that contracts state minimum requirements that runtime configuration or the hosting environment may not support.

Service Contracts

A service contract specifies the following:

- The operations a contract exposes.
- The signature of the operations in terms of messages exchanged.
- The data types of these messages.
- The location of the operations.
- The specific protocols and serialization formats that are used to support successful communication with the service.

Configuring Services Using Configuration Files

A Windows Communication Foundation (WCF) service is configurable using the .NET Framework configuration technology. Most commonly, XML elements are added to the Web.config file for an Internet Information Services (IIS) site that hosts a WCF service.

Hosting Services

The Internet Information Services (IIS) hosting option is integrated with ASP.NET and uses the features these technologies offer, such as process recycling, idle shutdown, process health monitoring, and message-based activation.

The IIS-hosted services can only use the HTTP transport.

Building Clients

A client application is a managed application that uses a WCF client to communicate with another application. To create a client application for a WCF service requires the following steps:

- Obtain the service contract, bindings, and address information for a service endpoint.
- Create a WCF client using that information.
- Call operations.
- Close the WCF client object.

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 returned. It serves a similar purpose as a method signature in a conventional programming language[6].

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 [7]. Soap messages are ordinary XML documents that are sent over HTTP.

2.2 To create and consume a simple Web Service using JAX WS

One of the most exciting new features of the Java Platform, Standard Edition 6 (Java SE 6) is support for the Java API for XML Web Services (JAX-WS). JAX-WS architecture is an easier-to-understand architecture for web services development [8].

Project Description

- In this example, we create a SOAP based web service for a simple BankService class with operations _GetBalance‘ and _Deposit‘.
- We then create a web service client which then consumes the web service and displays the result of the invoked web service

Creating & Publishing Web service

a) Create a Java Project _BankWebService‘.
b) Create a package _bankPack‘.
c) Create a Java class _BankService‘ and write the code for GetBalance & Deposit method.
d) @WebService annotation at the beginning of the class definition tells the Java interpreter that we intend to publish ALL the methods of this class as a web service. If we want to publish only particular methods then we can use @WebMethod annotation before the method signature.
e) In order to publish our class and its methods as web service we need to crate appropriate stub files or artifacts for web service deployment and invocation. Fortunately Java provides a tool called _wsgen‘ which generates JAX-WS portable artifacts used in JAX-WS web services.
f) Open command prompt and go to the project folder _BankWebService‘.
g) Now issue the following command,
wsgen -cp ./bin -keep -s ./src -d ./bin bankPack.BankServie
h) the –cp option specifies the classpath for our BankService class which is in _bin‘ folder, the –d option specifies where to place generated output files which is also the _bin‘ folder in our case.
i) Now we need to publish our class as a web service endpoint. For that we use the static publish() method of the javax.xml.ws.Endpoint class to publish our _BankService‘ class as a web service in the specified context root.
j) Create a class _RunService.java‘ with main method and type the following code.

```
package bankPack;
import javax.xml.ws.Endpoint;
public class RunService {
public static void main(String[] args) {
System.out.println("Web Service Started");
Endpoint.publish("http://localhost:8080/BankService", new BankService());
} }
```

i. Run this class as Java Application.
ii. You may not get output in the Console. To check whether our class is published as web service, open a browser and type the URL mentioned in the endpoint with a parameter ?wsdl appended.
http://localhost:8080/BankService?wsdl
iii. When you run the application, the Java SE 6 platform has a small web application server that will publish the web service at the address http://localhost:8080/BankService?wsdl while the JVM is running.

Creating and Consuming a Web Service Client (i.e. cross language or language independent)

a) Having published the web service, we now create a client which communicates with the service and displays the result.

b) Create a project ‗BankingConsoleClient' using Microsoft Visual Studio IDE and add Service References of Java Application using the following url; http://localhost:8080/BankService?wsdl & then write the following code for ‗AccountForm.cs' & ‗Program.cs' and run the client.

III. HANDLING EXCEPTION FOR SOA APPLICATION

3.1 Handling Exception For SOA Application Using .Net IDE

Creating a service application.

For creating a service application used following steps[9], 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>
```

Checking the exception in service

- The exception in SOA architecture is BasicHttpBinding.
- To identify, AccID is valid or not, & see the exception on client side. This exception message is very big.

Handling the exception

- 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.
- To avoid big message, log it on server side.

IOSR Journal of Computer Science (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727
PP 78-83
www.iosrjournals.org

IV. BLOCK DIAGRAM OF SOA APPLICATION

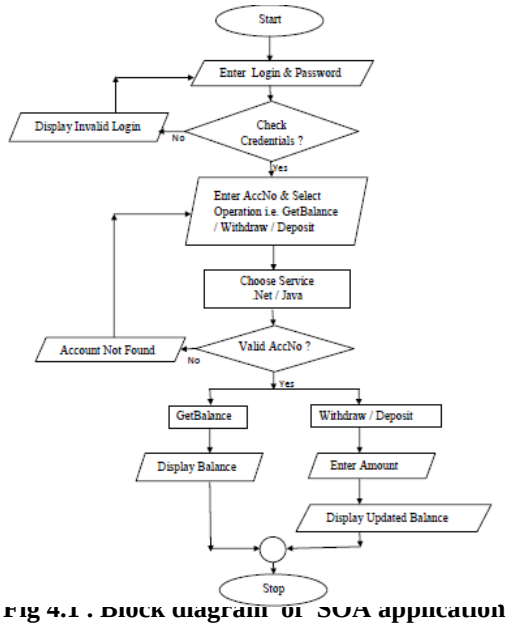

Fig 4.1 : Block diagram of SOA application

RESULT ANALYSIS

5.1 BankDotNetSolution

Account Object Created
Unhandled Exception: System.ApplicationException: Account Not Found
at BankLib.BankService.CheckAccID(Int32 accid) in
D:\SOAProjects\BankDotNetSolution\BankLib\BankService.cs:
at BankLib.BankService.InitializeAccount(Int32 accid) in
D:\SOAProjects\BankDotNetSolution\BankLib\BankService.cs
at BankLib.BankService.Deposit(Int32 accid, Double amt) in
D:\SOAProjects\BankDotNetSolution\BankLib\BankService.cs
at BankConsole.Program.Main(String[] args) in
D:\SOAProjects\BankDotNetSolution\BankConsoleClick\Program.cs

5.2. BankSOASolution

FaultException is based on SOAP Specification, it is for all application related to SOA Architecture. This internally mapped to SOAP Fault.

Result :

Unhandled Exception: System.ServiceModel.FaultException`1[BankContractLib.BankEr ror]: Account Not Found
Server stack trace: at System.ServiceModel.Channels.ServiceChannel.HandleReply(ProxyOperationRuntime operation, ProxyRpc& rpc) at System.ServiceModel.Channels.ServiceChannel.Call(String action, Boolean on eway, ProxyOperationRuntime operation, Object[] ins, Object[] outs, TimeSpan timeout)
at System.ServiceModel.Channels.ServiceChannelProxy.InvokeService(IMethodCall Message methodCall, ProxyOperationRuntime operation)
at System.ServiceModel.Channels.ServiceChannelProxy.Invoke(IMessage message)
Exception rethrown at [0]: at System.Runtime.Remoting.Proxies.RealProxy.HandleReturnMessage(IMessage req Msg, IMessage retMsg)
at System.Runtime.Remoting.Proxies.RealProxy.PrivateInvoke(MessageData& msgDa ta, Int32 type) at BankContractLib.IBank.Deposit(Int32 accid, Double amt)
at BankingConsoleClient.Program.Main(String[] args) in
D:\SOAProjects\BnkingSolution\BankingConsoleClient\Program.cs

Table 1: Compare SOA Application (.Net) and Simple Dot Net Application

SOA Application (.Net)	Simple Dot Net Application
Used SOA Specific Exception	Used Language Specific Exception
Need to create channel	No need to create channel
Add Service references & Contract by adding namespace using System.ServiceModel; using BankContractLib;	Add BankLib reference by adding namespace using BankLib;
Catch Exception of FaultException	Catch Exception of Language Specific Exception

5. CONCLUSION

This topic provides SOA architects techniques to discover error handling requirements from the business artifacts package and to specifies the comparative study of exception handling related to language & SOA specific results. It is implemented by use of different development toolkits. It also specified the execution of SOA application which is used to various types of software users.

REFERENCES

[1] Stefan Bruning, S. W., and Malek, M. "A Fault taxonomy for service-oriented architecture". IEEE (2007).
[2] W3C, "Web services architecture," Febuary 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/
[3] 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.
[4] V. S. Patil, P. Patil & P. Bhanarkar, "Survey On Exception Handling In SOA". International Journal of Engineering Research & Technology –IJERT, Vol. 1 Issue 9, November- 2012ISSN: 2278-0181.
[5] http://msdn.microsoft.com
[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] 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.
[8] http://theopentutorials.com/examples/java-ee/jax-ws/create-and-consume-web-service-using-jax-ws/
[9] V. S. Patil and P. Patil "Handling Exception using Service Oriented Architecture" 2nd National Conference on Emerging Trends In Computing 2013(NCETIC 13), June 2013 ISBN:978-93-82338-64-2.