International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN : 2278-800X, www.ijerd.com
Volume 4, Issue 11 (November 2012), PP. 07-11

Exception Handling Strategies for WS-BPEL Fault Tolerant Service

Mr. V. Mano Gajapathi[1], Ms. T. Revathy[2]
[1]Assistant Professor, Department of Computer Science & Engineering, Christ College of Engineering and technology
[2]Assistant Professor, Department Of Computer Science & Engineering, Shri Krishnaa College of Engineering and technology

Abstract:-Web Services are one of the most promising technologies for the development of component based systems. There are several faults occur during service invocation, it decrease the quality of service. Fault tolerance schemes can be used to increase the availability reliability and performance of network service systems. Failure of web services is not acceptable in many situations such as online banking, so fault tolerance is a key challenge of web services. In this paper we present a set of high-level exception handling strategies for fault tolerant web service. With this approach, business processes can be made to execute in a fault tolerant manner within standard WS-BPEL execution environment.

Keywords:-Fault tolerant, web service, WS-BPEL, quality of service

I. INTRODUCTION

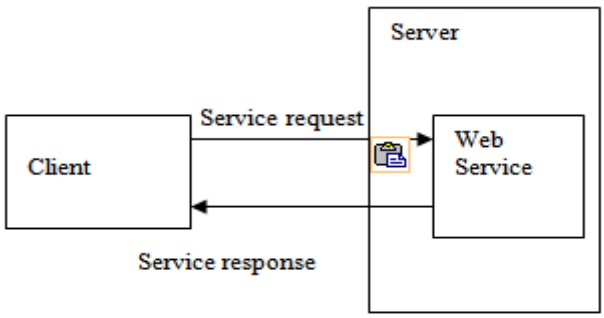
1.1 WEB SERVICE

A Web service is a method of communication between two electronic devices over the web. A "Web service is a software system designed to support interoperable machine-to-machine interaction over a network". It has an interface described in a machine-processable format (specifically Web Services Description Language, known by the acronym WSDL). Web Services can convert your applications into Web-applications. Web Services are published, found, and used through the Web.

Web Services are the technology of choice for Internet-based applications with loosely coupled clients and servers. That makes them the natural choice for building the next generation of grid-based applications.

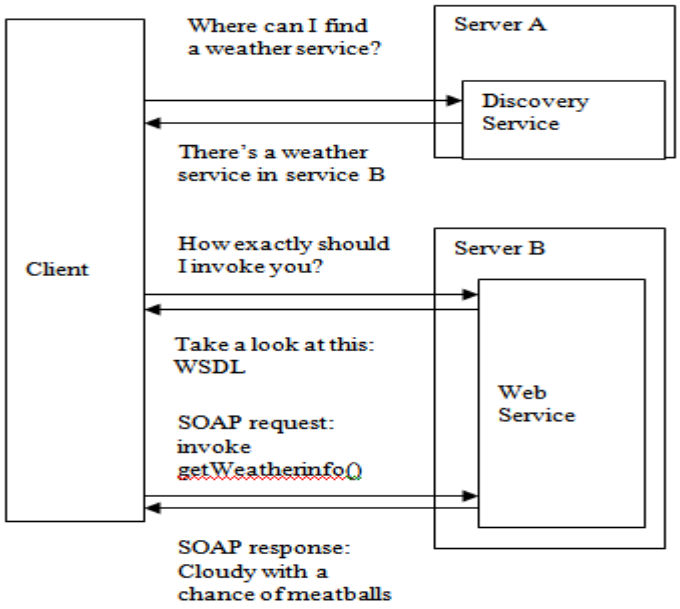
Example:

The clients (programs that want to access the weather information) would then contact the Web Service (in the server), and send a service request asking for the weather information. The server would return the forecast through a service response. This figure shows how the web service work.



1.2 WEB SERVICE INVOCATION:

Invoking a Web Service refers to the actions that a client application performs to use the Web Service. Client applications that invoke Web Services can be written using any technology: Java, Microsoft .NET, and so on



1. As we said before, a client may have no knowledge of what Web Service it is going to invoke. So, our first step will be to discover a Web Service that meets our requirements. For example, we might be interested in locating a public Web Service which can give me the weather forecast in US cities. We'll do this by contacting a discovery service (which is itself a Web service).
2. The discovery service will reply, telling us what servers can provide us the service we require.
3. We now know the location of a Web Service, but we have no idea of how to actually invoke it. Sure, we know it can give me the forecast for a US city, but how do we perform the actual service invocation? The method I have to invoke might be called "string getCityForecast(int CityPostalCode)", but it could also be called "string getUSCityWeather(string cityName, bool isFarenheit)". We have to ask the Web Service to **describe** itself (i.e. tell us how exactly we should invoke it)
4. The Web Service replies in a language called WSDL.
5. We finally know where the Web Service is located and how to invoke it. The invocation itself is done in a language called SOAP. Therefore, we will first send a **SOAP request** asking for the weather forecast of a certain city.
6. The Web Service will kindly reply with a **SOAP response** which includes the forecast we asked for, or maybe an error message if our SOAP request was incorrect.

II. RELATED WORK

The Reliable Hybrid pattern, that can be used to design fault tolerant software applications. The pattern supports development of applications based on classical fault tolerant strategies such as N-Version Programming and Recovery Block, as well as those based on advanced hybrid techniques such as Consensus Recovery Block, Acceptance Voting, and N-Self Checking Programming. An exception handling capability-aware web service selection method focus on how to extend the QoS Criteria, that the exception handling capability could be introduced into the composed Web Services. Based on the work of Atomic sphere, they extend the QoS Criteria, so that the composed services can meet the end users requirements and ensure the security of the execution at the same time. a novel model-driven approach that is based on the use of colored Petri Nets. It focuses on the issue of faults and failures in composite web services (CWS). Single version software (SVS) technique detects errors by executing two copies of an application program with similar inputs and then comparing the results. The approach is able to tolerate a fault or to mask errors through executing the three or more copies of an application program with similar inputs and then comparing or voting upon all the computing results for getting a result in majority. A distributed replication strategy evaluation and selection framework for fault tolerant Web services. Based on this framework, they compare various replication strategies by using theoretical formula and experimental results, and propose a strategy selection algorithm based on both objective performance information as well as subjective requirements of users. A container-based approach is used to improving service availability and reliability. One of the advantages of this technique is that it is likely to be already applied by providers to increase the availability of their websites.

III. OVER VIEW OF THE PROJECT

Web services are self-contained, self-describing, and loosely-coupled computational components designed to support machine-to-machine interaction by programmatic Web method calls, which allow structured data to be exchanged with remote resource. By tolerating component faults, software fault tolerance is an important approach for building reliable systems and reducing the expensive roll-back operations in the long-running business processes.

Fault tolerance becomes considerably more difficult in distributed applications, made up of several processes that communicate by passing messages between themselves. One process may fail without the other processes being aware of the

failure. Today fault tolerant web service application is of great importance. Fault tolerance techniques are often used to increase the reliability and availability of Internet services.

Fault tolerance, which aims at delivering a correct service even in the presence of faults, therefore, becomes a preferred choice for reliable Web services composition. Web services are one of the most important distributed computing technologies that are increasingly used in ecommerce, integrating of organization systems, management of business processes, middleware's and many web based applications. In many situations such as online banking, stock trading, reservation processing, and shopping erroneous processing or outages are not acceptable, so fault tolerance is a key issue of web services. Fault tolerance makes to achieve system dependability. It may be defined as the act of avoiding service failures in the presence of faults. Dependability is related to some QOS aspects provided by the system, it includes the attributes like availability and reliability.

Availability means that a system is immediately ready for use. In general, it refers to the probability that the system is operating correctly at any given moment and is available to perform its functions on behalf of its users.

Reliability indicates that a system can run continuously without failure. During service invocation, errors may occur from latent faults. Partner services may not be available to respond or not respond within time. If the process does not provide fault handling mechanisms, faults will cause the process to stop working. This situation could affect the business process and lead to service unavailability.

This paper aims to propose an approach for applying patterns for fault tolerant software to WS-BPEL. Although faults have occurred, fault tolerant mechanisms can support business process execution and help to maintain its availability, reliability and performance.

IV. FAULT TOLERANT WEB SERVICE

As Web services are inherently unreliable, how to deliver reliable Web services over unreliable Web services is a significant and challenging problem. . In this project we propose a set of exception handling strategies and transaction techniques to improve the reliability and performance of web services. This approach focus on
Physical faults, Logical faults, Content faults ANDService level agreement(SLA) fault.

4.1 TYPES OF FAULTS

Physical Fault

Physical faults involve failures on the server side infrastructures or failures in the network level. The infrastructure of the service provider such as local database or application server may be malfunctioned. Network connection between the execution engine and service provider may be crashed. These faults can lead to Web service unavailability.

Logical fault

Logical faults are deliberately thrown by external Web services as they cannot complete successfully due to various reasons. For example, a flight service is invoked to book four tickets, but only two are available at that time, so the service generates a fault to notify service requesters.

System Faults

System faults are raised by the supporting execution environment. For example, a bike service is undeployed due to business requirements. When a service requester invokes this service, the hosting Web server will inform it that the service does not exist on the specific end point.

Service level agreement(SLA) fault:

SLA faults are raised when Web services complete the functional requirements but violate the predefined SLA. For example, an SLA specifies that the expected execution duration of a service is less than 5 seconds, but the actual execution duration is 8 seconds.

2.2 EXCEPTION HANDLING STRATEGIES

The set of exception handling strategies are used to handle the fault during service invocation.

Reject:

Just as its name says, this strategy does not take any special action to handle a fault, but simply reject it and allows the composite service to continue (e.g., invoke other external Web services). Its failure should not result in the failure of the whole process because the primary business goals are achieved (for example, the flight tickets and hotel rooms have been reserved successfully).

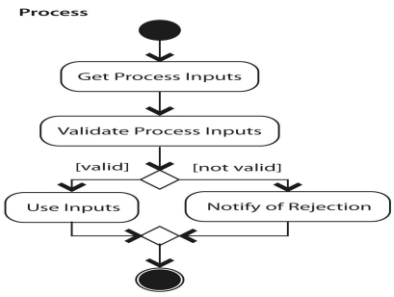


Fig. 4.2.1 The structure of the Ignore pattern in UML

Notify:

When a fault occurs, the notify strategy first writes some information into log files, and then allows the composite service to keep going. The recorded fault information can assist IT or business experts to analyze and refine the technical or business aspects of the composite service.

Skip:

This strategy requires the composite service not to invoke one or more Web services which are the successors of the faulted service in the process. It is helpful to handle SLA faults. Back to the travel arrangement example, a fault occurs when the actual execution duration of the computation service is 8 seconds, which exceeds the promised execution duration, say, 5 seconds.

Failover:

This strategy determines alternative service invocation when first service invocation fails.

Retry:

This strategy repeats the execution of a service until it completes successfully or the associated condition evaluates true. To decrease communication overheads, an interval can be set to specify the amount of time to wait between retries.

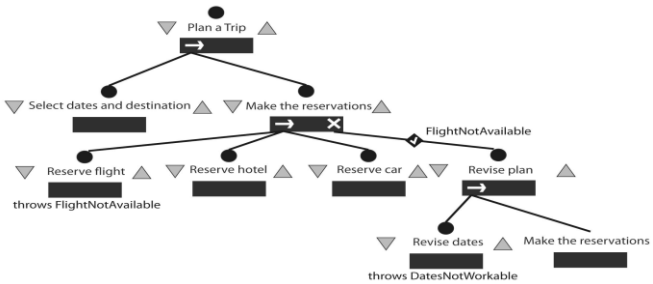


Fig.4.2.2 Using the Retry to replan a trip

Retryuntil:

It is similar to the retry strategy, but the associated condition is a deadline for retrying. That is, a service is reinvoked until it completes successfully or the specified deadline expires.

Substitute

When a service fails, this strategy selects another function-equivalent service to perform the same task. The foundation of this strategy is that a number of Web services have the same or similar functions.

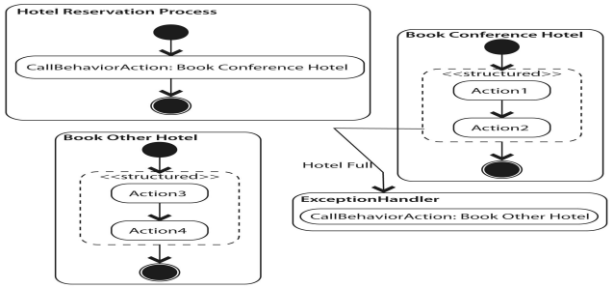


Fig.4.2.3 Substitute strategy.

This strategies can be used to increase the performance and availability of web service.

V. CONCLUSION

We have presented a fault tolerant exception handling and transactional Web services. We identified a set of high-level exception handling strategies, gave a new taxonomy of transactional Web services, and introduced key features including service transfer and vitality degree. This strategy separates the development of normal business logic from fault-handling logic, and summarizes a set of common exception handling strategies to specify fault-handling logic. This improves the reusability of fault-handling logic as well as normal business logic, which is a significant advantage in Web services engineering.

REFERENCES

[1]. Tianyang Zheng, Bing Jia, Wei Pan, Weichang Chen, An Exception Handling Capability-Aware Web Service Selection Method, *2010 Second International Conference on Computer Modeling and Simulation.*

[2]. Karolina Zurowska and Ralph Deters Overcoming Failures in Composite Web Services by Analysing Colored Petri Nets, *March/april 2008*.

[3]. Y.S. Hong and J. H. No and In Han Evaluation of Fault-tolerant Distributed Web System, *10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems 2005*.

[4]. Zibin Zheng and Michael R. Lyu,A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services, *2008 IEEE International Conference on Web Services.*

[5]. Jigang Wang, Guochang Gu, Shi-Bo Xie, and Li-Feng Xu, An Fast Transparent Failover Scheme for Service Availability *First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06) IEEE 2006.*

[6]. Sareh Aghaei, Mohammad Reza Khayyambashi and Mohammad Ali Nematbakhsh. A Fault Tolerant Architecture for Web Services,*2011 international conference on innovations in information technology.*

[7]. Barbara Staudt Lerner, Stefan Christov, Leon J. Osterweil, Reda Bendraou, Udo Kannengiesser, and Alexander Wise Exception Handling Patterns for Process Modeling *IEEE transactions 9.on software engineering, VOL. 36, NO. 2, March/April 2010.*

[8]. Jim Lau, Lau Cheuk Lung, Joni da S. Fraga and Giuliana Santos Veronese, Designing Fault Tolerant Web Services Using BPEL, *Seventh IEEE/ACIS International Conference on Computer and Information Science 2008*.

[9]. Luigi Coppolino, Luigi Romano and Nicola Mazzocca, Sergio Salvi Web Services Workflow Reliability Estimation Through Reliability Patterns.

[10]. Narajess Arari and Denis Barbaron, Fault tolerance for highly available internet services: *concept, Approaches, and issues. 2nd Quarter 2008, volume 10, NO. 2.*

[11]. MingXue Wang, Kosala Yapa Bandara and Claus Pahl, Constraint Integration and Violation Handling for BPEL Processes, *2009 Fourth International Conference on Internet and Web Applications and Services.*

[12]. An Liu, Qing Li, Liusheng Huang, and Mingjun Xiao, FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services *IEEE transactions on services computing, vol. 3, NO. 1, January-March 2010.*

[13]. Lingxia Liu, ZhaoXue Wu, Zhiqiang Ma, Wei Wei A Fault-Tolerant Framework for Web Services, *World Congress on Software Engineering IEEE 2009.*

[14]. Glen Dobson , Stephen Hall and Ian Sommerville, A Container-Based Approach to FaultTolerance in Service-Oriented Architectures.

[15]. Algirdas Avizienis, The N-Version Approach to Fault-Tolerant Software, *IEEE transactions on software engineering, VOL. SE-I 1, NO. 12 December 2001.*

[16]. Hemangi Gawand, R.S.Mundada and P.Swaminathan, Design Patterns to Implement Safety and Fault Tolerance, *International Journal of Computer Applications (0975 – 8887) Volume 18– No.2, March 2011*.

[17]. An Liu, Qing Li, Senior Member, IEEE, Liusheng Huang, and Mingjun Xiao FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services, *IEEE Transactions on services computing, VOL. 3, NO. 1, January-March 2010.*

[18]. G. Dobson, "Using WS-BPEL to Implement Software Fault Tolerance for Web Services, *in Proceedings of Software Engineering and Advanced Applications, 2006. SEAA '06. 32nd EUROMICRO Conference on, 2006, pp. 126-133.*

[19]. Fonda Daniels, Kalhee Kim and Mladen A. Vouk The Reliable Hybrid Pattern A Generalized Software Fault Tolerant Design Pattern.

[20]. R. Hamadi, B. Benatallah, and B. Medjahed, Self-Adapting Recovery Nets for Policy-Driven Exception Handling in BusinessProcesses, *Distributed and Parallel Databases, vol. 23, no. 1, pp. 1-44, 2008*.