Survey on Handling Error using Service Oriented Architecture

Mohd Awais Farooque

Department of Information Technology
Priyadarshini Institue Of Engineering & Technology
RTMNU,Nagpur,India.
Mohd.awais1@gmail.com

ABSTRACT: **Services are performing an increasingly important role in modern application development and composite application The Service-Oriented Architecture (SOA) refers to a style of distributed application. The business applications and data are decomposed into discrete components, known as services, that have particular relevance for various business processes.. The objective of study to create new composite applications that model new or more complex business processes and also to examine the key issues of the user's negative attitude towards introduction of SOA design. Most of the composite applications needed to be reliable and available, however it may appear more difficult to achieved, due to the multi-layered architecture of SOA. To reduce the risk, to reduce the fear of complexity, usable by all types of users when introducing SOA architecture, it is necessary to use error handling methods in order to increase system fault tolerance.**

Keywords: Error Handling WSDL, SOA .BPEL, ESB, Choreography, Services, SOAP,

I. INTRODUCTION

The term Service Oriented Architecture (SOA) refers to a style of distributed application architecture in which key business applications and data are decomposed into discrete components, or "services," that have particular relevance for various business processes. The services can then be used as building blocks to create new "composite" applications that model new or more complex business processes. An SOA can also be compatible with the existing portfolio of business applications. Service-oriented applications have specific characteristics that distinguish them from traditional distributed applications. The services are typically loosely coupled and interact with each other as peers via message-based communications over the network. The basic requirement is that the service requester and the service provider can exchange and interpret messages in a common format. Each service specifies its functionality and quality of service (QoS) characteristics using a common, machine-readable format. This enables dynamic discovery of available services and provides the basis for interoperability in a heterogeneous computing environment. For widest applicability, services must be supported by access and security policies to allow operation across organizational boundaries and administrative domains. This paper identifies common error handling considerations such that architects and designers can address the issues while designing SOA Solutions.

II. BENEFITS

There are many benefits that an enterprise can derive by adopting an SOA style of application infrastructure. Backward compatibility with existing applications and data: With SOA, a application can be offered as a single service, possibly incorporating multiple steps in a business process.

*Extended interoperability***:** SOA can make diverse applications and data appear to the user as an integrated service. In SOA scenario, the user could access all the application resources on the network through a single portal.

Business agility and alignment of IT with business processes**:** Creating application services that mirror steps in various business processes helps to bring application resources into better alignment with business practices. As an enterprise's implementation of SOA matures, it becomes possible to support rapid changes in business processes with new applications created by rearranging existing service components.

*Increased productivity and cost effectiveness***:** Application programmers gain the benefit of code reuse whenever they create a new service based on legacy applications or include an existing service within a new composite application.

III. ELEMENTS OF AN SOA

Using SOA, the service is an abstracted, logical view of how a computer program performs the operations that mirror a business process. As a result, there is a strong relationship between the business process flow chart and the processing flow chart of a service-oriented application. Below figure provides a very simple example of a business process model and the corresponding services.

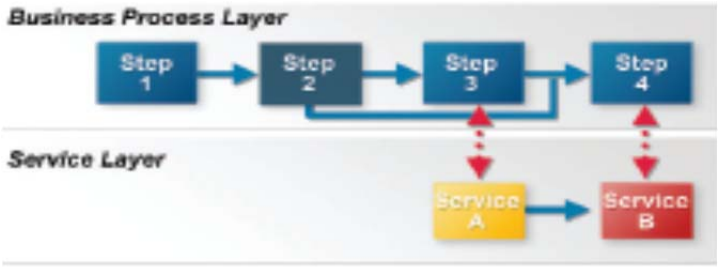
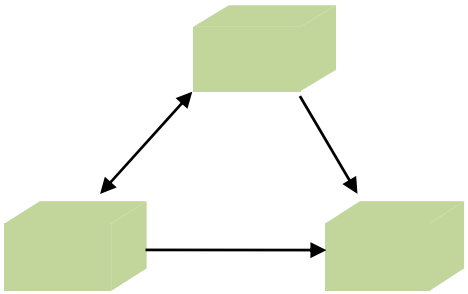


Figure 1: Relationship between business processes and services

IV. 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 [9], [10].
The structure of SOA is shown in Figure. 2.



The service provider register their services at the broker. Services interact with each other and with the service broker using message-oriented communications. A service is described by meta data that provides all the information required to invoke the service. The internal structure of an agent providing the service is essentially hidden behind the service interface. Therefore, any software component or application can become a service if it is given a formal SOA service description and supports the message handling middleware used by the SOA. Messages are sent in a standardized format and delivered through the service interfaces. The services environment, including the service interfaces, brokering, and messaging system, is sometimes described as an enterprise services bus (ESB). The ESB performs all the operations necessary to virtualize services. Services are virtualized in the sense that the service requester is unaware of the service provider's physical location, programming language, runtime environment, hardware platform, or network address.

The service requester discovers the service by exchanging messages with the service broker. In addition to providing a registry of services, the service broker may provide additional functions such as workload management, load balancing among several providers of the same service.

When the service-oriented program is run, the service requester communicates directly with the service provider via the messaging system to invoke the service and receive results.

SOA-specific Errors and Failures

In terms of fundamental concepts of dependability [11], threats for computer systems include errors, faults and failures. An errors are that parts of the system state that may cause a subsequent failure: a failure occurs when an error reaches the service interface and alters the service. A fault is the supposed or hypothesized cause of an error. All faults are gathered into three major fault classes for which defenses need to be devised: design faults,

physical faults, interaction faults. We proceed from the assumption that most of the errors and failures occur during service binding and invocation, messages transferring and requests processing by web service.
In this paper we specified different types of SOA specific errors and failures (see Table 1).

Table 1: SOA errors and failures

Sr.No	Type of error or failure	Error /failure domain
1	Suspension of web service during transaction (getting into a loop)	Service errors and failures
2	System error during processing (like "Divide by Zero")	
3	Calculation error during processing(like, "Operand Type Mismatch	
4	Application error raising user exception(defined by developer)	
5	Error in Target Name Space	Client-side binding errors
6	Error in web service name	
7	Error in service port name	
8	Error in service operation's name	
9	Output parameter type mismatch	
10	Input parameter type mismatch	
11	Error in name of input parameter	
12	Mismatching of number of input service parameters	
13	Web service style mismatching	

V. ERROR HANDLING

Unlike in monolithic applications, error handling becomes a significant step in the design of SOA applications as SOA applications integrate heterogeneous IT systems across the organizational boundaries, vendor and partner IT assets. Focusing on error handling analysis early in the analysis and design phases ensures that appropriate error handling standards/guidelines are put in place for modules in different platforms. This topic identifies common error handling considerations that architects and designers need to address while going through the SOA solution design. SOA analysis and design tasks are broadly classified into three major phases;

i) Service identification,
ii) Service specification and
iii) Service realization

5.1. Error Handling during Service Identification

The goal of service identification is to come up with a candidate service portfolio that leads to identifying re-usable service portfolio [4]. This phase involves analysis of business artifacts package that includes key requirements, business goals, capability models, Business Process Analysis Model (BPAM), use cases, etc.

Types of errors

Recoverable Errors - Recoverable errors are the errors that client programs can recover using appropriate alternate execution paths. Such errors are the result of failure to meet a particular business rule.
Non-Recoverable errors - These are the errors that client programs cannot recover from. This kind of errors are result of some unexpected errors during runtime such as programming errors such null pointers and resources not available.

5.2 Error Handling during Service Specification

Service Specification phase consists of tasks defining inputs and output messages, service and operation names, schemas, service composition, non-functional requirements and other service characteristics such as sync/async, invocation style, etc. for the services that are marked as to be exposed [4].
Characteristics related to Error Handling
Common service characteristics that are related to error handling are:
Assured Delivery - Determine if a service requires assured delivery type of QOS. Such a requirement helps designers put in appropriate asynchronous messaging design patterns or use reliable messaging if implemented as web services.
Monitoring requirements - Determine if the service business critical errors require being setup with proactive monitoring.
Error mapping/transformation rules - Establish transformation rules for errors codes/info returned by the service provider and how it needs to be provided to service consumer. Having standard business error codes helps applications consume these services easily in terms of handling the service errors.
Updated process flows - Existing process flows are to be updated with the newer operations or alternate execution paths as discovered in the identification step to handle business errors.

Transaction attributes and boundaries - Nature of errors such as system Vs application errors influences how different runtime platforms handle automatic roll backs. Transaction attributes and boundaries in a process are to analyzed in the light of errors that can be expected from individual service invocations/transactions.

5.3 Error Handling during Service Realization

Service realization phase is where the service model is mapped to service component and runtime /deployment model [4]. This step typically involves designing service components, allocating the components to SOA stack layers choosing component interaction styles, runtime platforms
and making architectural design decisions (ADD). Subsequent discussion of the subject will be focused around some best practices to implement error handling considerations in the three layers of typical enterprise SOA stack: business processes or choreography, mediation/BUS and component layers.

Error handling in the business process/orchestration layer

Components deployed to this layer implementing business process flows or choreographies. The following error handling considerations apply here [4]:

Fault Handlers - Use of fault handlers is the most popular way of handling service errors returned from the service invocations initiated from within the orchestrations. Fault handlers are attached to specific tasks in a process flow or as a global fault handler for the entire process. When the process results in errors, fault handlers are invoked to implement the corrective tasks. Compensation transactions and manual rollbacks are configured with the fault handlers so that appropriate corrective actions could be applied to handle the process errors. Care should be taken not to use Fault Handlers for alternate execution paths instead should only be used to recover from the errors thrown in the process.

Error handling in the Services/Mediation/ESB layer

Enterprise Service Bus (ESB) layer is at the core of typical enterprise SOA stack (figure 3). This layer supports the transformation and routing Capabilities required off of the enterprise reusable services. Components in this layer provide a well defined interface to the various provider implementations such as existing underlying assets and partner or vendor based services, by applying appropriate message and protocol transformations. Error handling by the mediation components mostly involves transforming the provider error structures into well defined error structures defined in the context of business domain. These components\ also could handle applying some complex transformation and mapping rules on the errors returned from the back end functional components to provide more simplified error info to the service consumers within the enterprise.

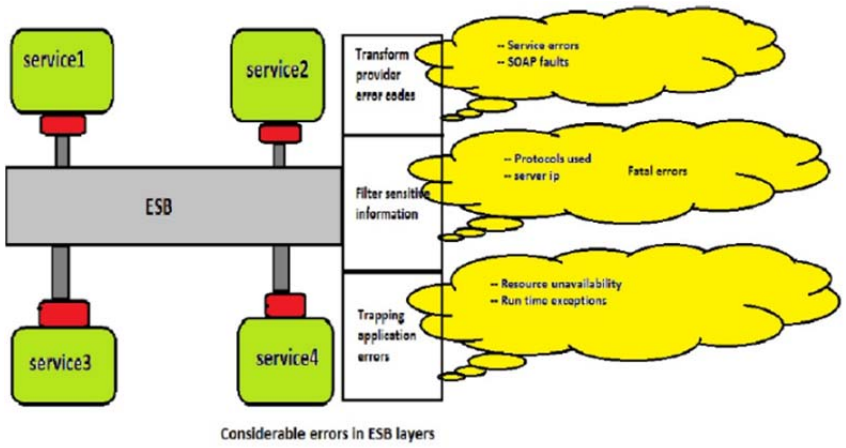


Figure 3: Considerable errors in EBS layer [12]

Transform provider error codes - It is possible that different service providers return service errors using different semantics. The range could involve anywhere from popular SOAP faults to very proprietary structures. Appropriate transformation rules can be applied here so that re-usable enterprise services return errors in a more consistent manner that enterprise applications could easily parse and implement appropriate handlers.

Filter sensitive information- when internal service components throw fatal errors, the stack trace often contains sensitive information such as protocols used, server ips, etc. Appropriate filtering rules are to be established in this layer to filter any sensitive information in the stack trace. This strategy becomes all the more important when service responses are to be given out over the trusted networks.

Trapping application errors - Any kind of technical errors experienced by the service components such as resource unavailability or some runtime exceptions etc. are to be transformed into a simple technical error messages. If native components did not log these errors, then mediation layers could pass all the stack trace info into logging but only return a generic text message back to the service consumer informing about temporary

service unavailability.

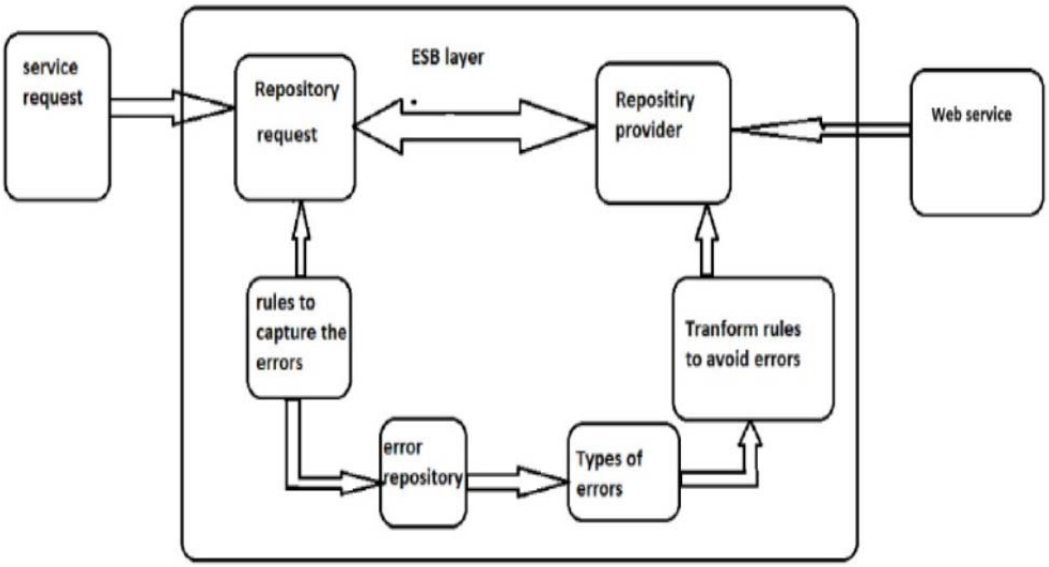


Figure 4: Error handling technique in ESB layer [12]

Error flow steps

The following are the error handling steps in ESB layer (figure3) [12].
Step 1: When a service requesting for another web service the service request reach the request repository in the ESB layer.
Step 2: request repository sends the address of the web service to the Repository provider.
Step 3: Before it reach the repository provider the request repository sends the web address to the Rules to capture the errors.
Step 4: If it finds any error then it sends the errors to error repository.
Step 5: Error repository decides the error is in which type then it sends to the types of error.
Step 6: Next the types of error send it to the transform rules to avoid error, here it applied
some transformation then send it to the Repository provider.
Step 7: Finally the repository provider searches the address of the web service and provide it to the service request.

Error handling in the component layer

Error handling by the components in this layer includes handling abnormal execution conditions such non-availability of a resource or some runtime conditions that the component is not programmed to handle or is considered in violation of logic [4]. Components are required to handle such events to notify client programs and also do appropriate logging to help facilitate troubleshooting and service monitoring. In Java programming language, such events are thrown as exceptions and the API provides two different types of exceptions: checked and unchecked. Checked exceptions inherit from Exception class and are used to handle recoverable errors such as business error scenarios. Unchecked exceptions which are descendents of Runtime Exception class are the ideal candidate exceptions handle non-recoverable errors such as resource no availability. The second part to component level error handling is to do appropriate logging. It is a good practice to perform logging closest to the source where the error occurred. When components throw application errors, they could log the exception at the appropriate interface within the component boundaries and then throw the exceptions. Use of correlation ids to identify the events and passing the same to calling applications would greatly enhance error tracking and monitoring by way of linking logs across different platforms.

VI. CONCLUSIONS

This paper provides SOA architects techniques to discover error handling requirements from the business artifacts package and how to analyze these while going through SOA analysis and design phase. Also provides some best practices to implement error handling in the three layers of SOA i.e. orchestration, mediation and component layers. A thorough upfront analysis of various error handling considerations help architects make the right decisions during design and implementation phases, platform and SOA stack products.

VII. REFERENCES

[1] Chen Liu, Yanting Xu, D., and Xaiyu, L. “A rule-based exception handlings approach in SOA”. International Conference on Computer Application and System Modeling (2010).

[2] Huang T, Wu GQ, W. J.” Runtime monitoring composite web services through stateful aspect extension”. Journal of computer science and technology (2009).

[3] Kareliotis Christos, Dr. Vassilakis Costas, D. G. P. “Enhancing BPEL scenarios with dynamic relevance-based exception handling”. IEEE (2007).

[4] Poolla, H.” Error handling considerations in SOA analysis and design”. Enterprise Architecture (2010).

[5] Shukla, R. K. “Exception handling in service-oriented architecture”. HCL Technologies (2006).

[6] Stefan Bruning, S. W., and Malek, M. “A Fault taxonomy for service-oriented architecture”. IEEE (2007).

[7] Wen Jiajia, Chen Junliang, P. y. and Meng, X. "A multipolicy exception handling system for BPEL process". First International Conference on Communications and Networking in China (06, 2007).

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

[9] W3C, “Web services architecture,” February 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/

[10] 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.

[11] 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.

[12] Prachet Bhuyan, Tapas Kumar Choudhury and Durga Prasad Mahapatra, “A Novel Approach For Exception Handling In SOA”, David C. Wyld, et al. (Eds): CCSEA, SEA, CLOUD, DKMP, CS & IT 05, pp. 425–433, 2012.