Volume 3, Issue 3, March 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Fault Management of Web Services

Zarana Padiya* *MCA, GTU India*
Yesha Dave *Mca ,GTU India*
Ketan Bhimani *Mca,GTU India*

***Abstract*— The use of service-oriented (SO) distributed systems is increasing now a day. Within service orientation web services (WS) are the de facto standard for implementing service-oriented Systems. The consumers of web services want to get uninterrupted and reliable service from the service providers. But web services providers cannot always provide services in the expected level due to faults and failures in the system. As a result the fault management of these systems is becoming crucial. This work presents a distributed event-driven architecture for fault management of Web Services.**

Keywords— Distributed system, fault management, event-driven architecture, web.

1. INTRODUCTION

In few last years, service-orientation (SO) has emerged as a new system design/integration paradigm for building distributed systems. The Service-Oriented-Architecture (SOA) is a paradigm for designing a software system that offers services to either end-user applications or to other services. SOA was first introduced by Gartner in April 1996. As a system design/integration philosophy it is independent of specific technologies, e.g., Web Services (WS) or J2EE.
This research focuses on the fault management of SOAP based WS due to their wide acceptance in the WS provider community. This work focuses on services where multi-faults can occur simultaneously. The main goal of this research is to detect and manage faults in SOAP based WS.

2. PROBLEM DEFINITION.

The use of WS applications is increasing daily. Their size and complexity is also increasing to meet functional requirements. WS can be found in Online Transaction Processing applications, Banking applications, Database Management applications, Groupware applications, legacy resources and so on. As a result, we become increasingly dependent on them - making it necessary to monitor, track and manage them.
The faults can occur in different places, for example faults can occur in the software application, in the network connection, or in the hardware resources. So the resources that are distributed over the network have to be monitored for availability, performance, and utilization.
IN THIS RESEARCH I WANT TO INVESTIGATE THE FOLLOWING QUESTIONS:

1. How can we monitor SOAP based WS for faults?
2. How can we detect faults in SOAP based WS?
3. Which faults can be detected? Can the system detect faults if multiple faults occur simultaneously?
4. What is the latency between occurrence and detection of faults? How can we improve it?
5. How can WS recover when any fault occurs?

3. WEB SERVICES (WS)

WS are self-describing, self-contained software modules available via a network, such as the internet, which completes tasks, solves problems, or conducts transactions on behalf of a user or application. WS constitute a distributed computer infrastructure made up of many different interacting application modules trying to communicate over a private or public network to virtually form a single logical system.
WS are loosely coupled software modules. The service interface is defined in a neutral manner that is independent of the underlying platform, the operating system, and the programming language the service is implemented in. This allows services, built on a variety of such systems, to interact with each other in a uniform and universal manner. WS are distributed over the internet and make use of existing, ubiquitous transport internet protocols like HTTP. This helps to comply with current corporate firewall policies.

3.1.1 TYPES OF WS

There are two types of web services – Simple or Informational services and Complex services or Business processes. There are two types of WS based on the protocol they use – SOAP based WS and REST (Representational State Transfer) WS. In REST WS the web services are viewed as resources that can be identified by their URLs.

3.1.1.1 SIMPLE OR INFORMATIONAL SERVICES

Informational services are of a simple nature. They provide access to content interacting with an end user by means of simple request/response sequences, or alternatively may expose back-end business applications to other applications.

Informational services can be divided into three subcategories according to the business problem they solve. These are pure content services, simple trading services, and information syndication services.

3.1.1.2 Complex services or business process.

Complex or composite services typically involve the assembly and invocation of many pre-existing services found in diverse enterprises to complete a multi-step business interaction. Complex services can be divided into two groups – complex services that compose programmatic WS and complex services that compose interactive WS.

3.1.2 Web Services Technology Stack

WS allow communication between applications regardless of their platforms, languages, and operating systems. So the WS technology has to allow applications to work together over standard internet protocols, without direct human intervention.

3.1.2.1 Enabling technology standards

At the transport level, WS use HTTP, or JMS, SMTP. WS use XML as the fundamental building block for nearly every other layer in the WS stack.

3.1.2.2 Core services standards

At the core the baseline standards are SOAP, WSDL, and UDDI.

3.1.2.3 Service composition standards

Service composition describes the execution logic of WS-based applications by defining their control flow (such as conditional, sequential, parallel, and exceptional execution) and prescribing the rules for consistently managing their unobservable data. The Business Process Execution Language (BPEL) can achieve composition for WS.

3.1.2.4 Service collaboration standards

Service collaboration describes cross-enterprise collaborations of web service participants by defining their common observable behavior, where synchronized information exchanges occur through their shared contact points, when commonly defined ordering rules are satisfied.

Service collaboration is materialized by the web services choreography description language WS-CDL, which specifies the common observable behavior of all participants engaged in business collaboration.

3.1.2.5 Coordination/transaction standards

Coordination/transaction standards provide mechanisms for defining specific standard protocols for use by transaction processing systems, workflow systems, or other applications that wish to coordinate multiple WS.

3.1.2.6 Value added standards.

Value added standards include mechanisms for security and authentication, authorization, trust, privacy, secure conversations, contract management etc.

3.2 Simple Object Access Protocol (SOAP)

SOAP is an XML-based communication protocol for exchanging messages. WS rely on SOAP for exchanging messages between computers regardless of their operating systems, programming environment, or object model framework.

A SOAP XML document instance is called a SOAP message or SOAP envelope. It is carried as the payload of some other network protocol like HTTP. The SOAP message consists of an <Envelope> element containing an optional <Header> and a mandatory <Body> element.

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">

<env:Header>
   .......
   .......
</env:Header>

<env:Body>
   .......
   .......
 </env:Body>

</env:Envelope>
```

Figure-1 Construct of a SOAP message

SOAP has a clear purpose: exchanging data over networks. It concerns itself with encapsulating and encoding XML data and defining the rules for transmitting and receiving that data.

3.3 Web Services Description Language (WSDL)

WSDL is an XML-based specification schema for describing the public interface of a web service. This public interface can include operational information relating to a web service such as all publicly available operations, the XML message protocols supported by web service, data type information for messages, binding information about the specific transport protocol to be used, and address information for locating the web service.

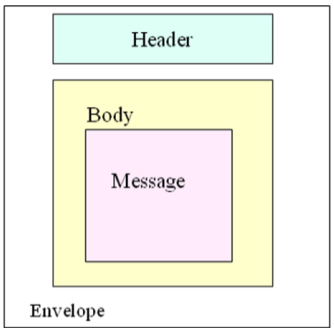

Figure 2. Structure of WSDL.

WSDL defines six major elements. They are:

1. Types: types provide data type definitions used to describe the messages exchanged.
2. Message: message represents an abstract definition of the data being transmitted. A message consists of logical parts, each of which is associated with a definition within some type system.
3. Port Type: port Type is a set of abstract operations. Each operation refers to an input message and output messages.
4. Binding: binding specifies the concrete protocol and data format specifications for the operations and messages defined by a particular port Type.
5. Port: port specifies an address for a binding, thus defining a single communication endpoint.
6. Service: service is used to aggregate a set of related ports.

3.4 Business Process Execution Language (BPEL)

BPEL now is an OASIS standard for defining workflows. Using BPEL web services can be orchestrated. BPEL mainly focuses on modern business processes. BPEL distinguishes five main sections: the message flow, the control flow, the data flow, the process orchestration, and the fault and exception handling sections. Workflows are related to business processes. A workflow system automates a business, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules.

3.5 Enterprise Service Bus (ESB)

The Enterprise Service Bus is an open standards-based backbone designed to enable the implementation, deployment, and management of SOA-based solutions with a focus on assembling, deploying, and managing distributed service oriented architectures. An ESB is a set of infrastructure capabilities implemented by middleware technology that support the SOA and alleviate disparity problems between applications running on heterogeneous platforms and using diverse data formats. The key capabilities of an ESB are listed below.

1. Dynamic connectivity capability.
2. Reliable messaging capability.
3. Topic and content based routing capability.
4. Transformation capability.
5. Service enablement capability.
6. Endpoint discovery with multiple QoS capability.
7. Long running process and transaction capability.
8. Security capability
9. Integration capability.
10. Management and monitoring capability.
11. Scalability capability.

3.6 Event

"An event is an object that is a record of an activity in a system. The event signifies the activity". An event has three aspects - form, significance and relativity. The event's form is an object which can be a single value, or a set of data components. The following figure shows an event class in Java.

3.7 Web Services Eventing (WS-Eventing).

When events occur in different services and applications some WS want to receive messages about the event (subscribe to the event). For receiving events the WS has to subscribe to the event source. WS-Eventing specification defines protocol for the subscriber WS to subscribe with event source.

3.8 Case Based Reasoning

Case-based reasoning (CBR) is a major paradigm in automated reasoning and machine learning. In CBR, a new problem is solved by finding its similarity to one or several problems solved previously and by adapting their known solutions instead of working out a solution from scratch. So in CBR, reasoning is done by remembering the previous solutions and the case based reasoner learns from its experience.

In AI technology CBR can be helpful in five fields. They are

1. Knowledge acquisition
2. knowledge maintenance
3. increasing problem solving efficiency
4. increasing quality of solutions
5. user acceptance

There are two types of CBR tasks.

1. Interpretive CBR
2. Problem solving CBR.

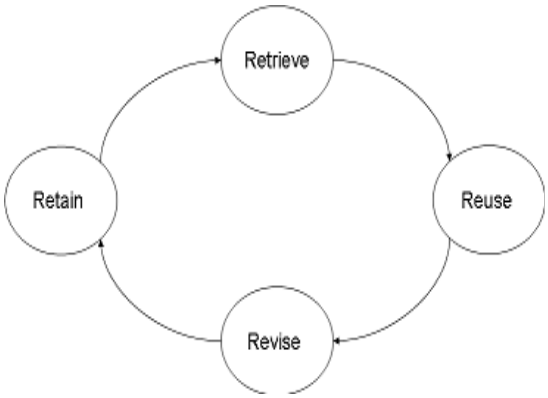

Figure-3 4'R in CBR.
Reference as in [2].

3.9 Fault and Failure

IEEE Standard Glossary of Software Engineering Terminology defined fault and failure. According to IEEE a fault is

(1) A defect in a hardware device or component; for example, a short circuit or broken wire.
(2) An incorrect step, process, or data definition in a computer program.

According to IEEE, "failure" is the inability of a system or component to perform its required functions within specified performance requirements. The fault tolerance discipline distinguishes between a human action (a mistake), its manifestation (a hardware or software fault), the result of the fault (a failure), and the amount by which the result is incorrect (the error).

4. Conclusion:

The fault management system is reliable. As the event processors are distributed over the internet, single component failure will not stop running the management system. This improves the reliability of the fault management system.

There are some limitations in the implemented experimental setup. This work assumed that there will be no noise in reporting the events. But it is not always true in a real world scenario. Network connections are not always reliable. So due to network failure events can be lost. In our work we assumed that the databases are reliable. If database is not reliable in the system then stored events can be lost. Storing the events in multiple databases can prevent event loss in case of database failure.

References:

[1] www.hp.com/fault_management
[2] my.safaribooksonline.com/...fault...web-service.../2_advanced_fault_...

