The Approach of Web Services Parameter Mutation Operators Based on the Rules Model

Yang Zhao[1]
[1] Jiangsu Automation Research Insititute, Lianyungang, Jiangsu , China
yangzhao@jari.cn
*Yang Zhao

Abstract—Web services have implemented the characteristic such as interoperability and accessibility furthest, making its fault tolerance very important. Use the method of fault injection to test Web services can thoroughly validate its fault tolerance. Existing researches mostly focus on the fault injection at the network layer, or the individual parameters of SOAP (Simple Object Access Protocol) messages or XML document formats, thereby there is lack of considering on the rules mutation among the parameters of Web services. This paper proposes a approach of Web services parameter mutation operators based on the Rules Model. First, Use 12 operators such as "=", "<", ">", "+", "-" to construct atomic rule model which represent the relations among the parameters of Web services or among the parameters and constant. Then combine atomic rule model to multiply rule model using "and", "or". At last, use the SMT solver the multiply rule model to generate mutation data that violates the constraint rules, and inject it into SOAP messages for testing. This approach can effectively assess the validity of business rule implementations in Web services.

Keywords- Web services; Rules Model; SOAP; Fault injection; Mutation operators; Software testing

1. INTRODUCTION

Due to the use of a series of protocols and standards such as SOAP(Simple Object Access Protocol) and WSDL(Web Services Description Language), Web services maximize interoperability, accessibility, and other features. As a result, the Web service provider can be accessed by any service be accessed. Web services may receive and process various types of SOAP messages at any time during the runtime, the stability of application systems largely depends on the reliability of service communication and the fault tolerance of services. In other words, can the Web services handle messages and communication errors during the service invocation process properly? Therefore, compared to traditional software testing, it is particularly important to verify the security and fault tolerance of Web services by injecting different levels of faults to the SOAP messages.

At present, researches at home and abroad mainly integrates on the numerical perturbation and communication perturbation of soap messages, that is, using certain mutation operators to mutate the numerical or communication of soap messages, in order to achieve fault injection and verify the reliability and fault tolerance of SOA systems. Looker[1-2] et al. utilized Web services middleware for error injection to evaluate the reliability of Web service interface method calls. Offutt and Xu[3] proposed modifying the SOAP message by using some mutation operations. Almeida et al. conducted research on the reliability testing of Web services mainly from the perspective of data communication disturbances. Chen[4-5]et al. proposed a mutation testing method for Web security testing, which mutates the security rules of message communication.

In summary, existing researches mostly focuses on the fault injection at the network layer, or the individual parameters of SOAP messages or XML document formats. The data relationships considered are mostly the data types and value ranges of parameters, such as maximum, minimum, boundary values, and length. Constraints often only include all, choice, minOccurs, and any, and there is still a lack of designing mutation operators based on various dependency relationships between parameters, The focus of this paper is to conduct SOAP mutation and testing.

2. BASIC THEORY

2.1. Web services

Web services can be seen as interfaces for certain functions in a Web network, which can be remotely accessed and called by other programs through the Web. Web services are defined as an SOA(Service-Oriented Architecture) technology aimed at cross platform application services that are interconnected through standardized Web protocols and customized services. Web services are actually a software system designed to support the interoperability of applications on different platforms. The Web service architecture framework consists of three parties: Web service interface provider, Web service interface requester, and Web service proxy, and provides three core actions: publish action, which means that the Web service interface provider allows users or other Web service interface providers to perceive the existence and related information of our Web service; The search action is to search and select the optimal and most suitable Web service interface; The binding action is to establish a connection relationship between the Web service interface requester and the Web service interface provider[6].

2835-2823/24/$31.00 ©2024 IEEE
DOI 10.1109/ISSSR61934.2024.00077

2.2. Mutation testing

Mutation testing is a testing technique based on error[7]. Mutation testing was first primarily applied to unit testing, and the concept was first proposed by Demillo[8] et al. and Hamler[9] in 1978. Due to the nature of this type of testing, software testers and researchers have invested a lot of effort and are very interested in it, and mutation testing has provided great help in measuring the effectiveness and generation of subsequent test case suites. The main idea behind mutation testing is to introduce reasonable errors into the original program and run test cases to see if they can recognize these errors. This process allows testers to generate an effective set of test cases, which can help identify real errors in the program.

3. METHOD OF SOAP MESSAGES MUTATION

There are many reasons that can cause Web services failure, including those caused by a single parameter and those caused by the interaction between multiple parameters. In mutation testing for SOAP messages, the core of designing test inputs lies in the design of mutation operators, and the first thing that needs to be clarified is the mutated object.
At present, researches at home and abroad mainly focus on the value variation of single parameters in Web service communication. On the basis of summarizing the single parameter mutation methods, this paper proposes a mutation method for parameter constraint rules. Therefore, the methods for mutating SOAP messages mainly include parameter mutation and parameter rule mutation.

3.1. Mutation of SOAP parameter values

The mutation of SOAP parameter values mainly focuses on constructing abnormal data based on parameter data types。
Based on the research of HECKEL[10]， combined with the characteristics of the Web services, this paper proposes and designs 9 parameter mutation operators for the parameter types in SOAP, as shown in the table below.

Table 1. Mutation operators of the parameter values

ID	Mutation Operator	Description
1	space_Parame	Substitute the value of node N with spaces
2	ill_IntValue	Make integer parameters to outliers
3	illFloatValue	Make float parameters to outliers
4	random_String	Make character string to outliers
5	over_StrLen	Abnormal length of character string
6	ill_TimFomat	Abnormal time format
7	ill_Binary	Make binary to outliers
8	ill_URL	Make URL to outliers
9	ill_Diperse	Make diperse to outliers

3.2. Parameter rule mutation

Due to the different rules between multiple parameters, during the Web testing process, whether it is a normal testing case or a exception testing case, software testing cases generated based on parameter constraint rules are more practical significance. The parameter constrained mutation method proposed in this paper mainly includes three steps.
Step1: construct atomic rule model r:
The atomic rule model represents an independent and complete parameter rule, which is formed by combining parameters, constants, and operators to form a linear inequality or Boolean expression. The commonly used operators are shown in the table below.

Table 2. Atomic rule model operators

ID	Operator	Meaning
1	Less_Than(“<”)	Less than
2	Less_EQ_Than(“<=”)	Less than or equal
3	Equal(“=”)	Equal
4	NeEqual(“!=”)	Not Equal
5	Greater_Than(“>”)	Greater than
6	Greater_EQ_Than(“>=”)	Greater than or equal
7	Add(“+”)	Add
8	Substract(“-”)	Substract
9	Multiply(“*”)	Multiply
10	Divide(“div”)	Divide
11	Mod(“mod”)	Mod
12	Negate(“!”)	Negate

Step2: construct multiple rule model R:
Combine atomic rule model to multiply rule model using "and" or "or" , as shown in the following figure.

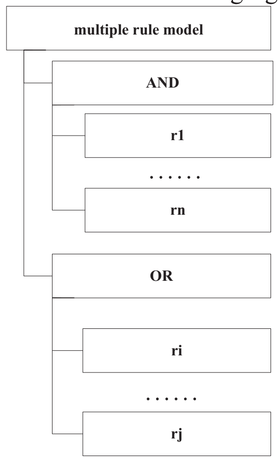

Figure 1 Multiple rule model

Assuming a Web service has three parameters x, y, and z, representing the three degrees of a right triangle. Without loss of generality, there are the following single constraint rules.
r1:x+y+z=180;
r2:x=90
r3:y=90
r4:z=90

Among them, r2, r3, and r4 have conjunction relationships, they and r1 have disjunction relationship，so：

R：（r2 or r3 or r4）and(x+y+z=180)

Step3: Solve !R

First, negate R,then use the SMT solver !R to generate mutation data that violates the constraint rules.

4. EXPERIMENT

In order to generate mutation test data, this article developed an execution program that can read constraint rules and output testing data. Then, select a Web service as the tested software to verify the method proposed in this article.

The Web service receives three positive integer parameters a, b and c. The sum of the three is equal to 10, and any two parameters have the same value. As a result, the constraint rules here can be expressed as follows.

R1:(r1:a>0) and(r2:b>0)and(r3:c>0);

R2:(r4:a+b+c=10);

R3:(r5:a=b) or(r6:a=c) or(r7:b=c);

R: R1 and R2 and R3.

In order to be recognized by the software programs. This paper designs an constraint representation pattern based on XML, as shown in the following figure.

```
<?xml version="1.0" encoding="utf-8"?>
<constraints>
 <and name="R1">
  <constraint name="r1">
   <![CDATA[0<value(a)]]>
  </constraint>
  <constraint name="r2" >
   <![CDATA[0<value(b)]]>
  </constraint>
 ……
 </and>
 <constraint name="R2">
   <![CDATA[value(a)+ value(b)+ value(c)=10]]>
 </constraint>
 <or name="R3">
  <constraint name="r5" >
   <![CDATA[value(a) =value(b)]]>
  </constraint>
    ……
 </or>
</constraints>
```

Figure 2 Constraints rule storage XML

Then, generate the mutation testing data which mutated the combinatorial constraints, as shown in the table below.

Table 3. Mutation testing data

ID	Parameter a	Parameter b	Parameter c
1	1	6	3
2	2	2	8
3	6	6	-2

5. SUMMARY

This paper proposes a approach of Web services parameter mutation operators based on the Rules Model. Design and propose a series of operators to construct multiply rule model, and use the SMT solver the multiply rule model to generate mutation data that violates the constraint rules. The approach in this paper can effectively assess the validity of business rules implementations in Web services.

The testing of Web service is a complex and important issue, it still need further research and improvement in the future. The future research work mainly focus on the enrichment of rule mutation operators and project applications. What’s more, the execution program needs to be further improved.

.

REFERENCES

[1] Nik，Looker，Binka Gwynne，Jie Xu，Malcolm Munro. An Ontology-Based for Determining the Dependability of Service.Oriented Architectures . Proceedings of the 10m IEEE International Workshop on Object. Oriented Real Time Dependable Systems（WORDS’05）.

[2] Nik Looker，Malcolm Munro，Jie Xu. A Comparison of Network Level Fault Injection with Code Insertion. Computer Software and Applications Conference.2005.COMPSAC 2005.29”Annual International Voll，26-28.

[3] W. Xu, J Offutt, Testing Web Services by XML perturbation[J].Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on, 2005.

[4] CHEN JinFu;LI Qing MAO Cheng Ying, A Testing Approach for Web Services Security Based on Security Rules Mutation[J] .Chinese Journal of Computers,2013,10(36):1967-1981.

[5] CHEN Jiamei CHEN Jinfu ZHAN Yongzhao.Design and Implementation of Web Service Vulnerability Testing System Based on SOAP Messages Mutation[J]. Computer Science,2013,7(40):143-186.

[6] XueLing Zhang Vulnerability Testing of Web Services Based on Combination Mutations of Web Service messages[D],2015:6.

[7] Debroy V, Wong W E. Combining mutation and fault localization for automated program debugging[J]. Journal of Systems & Software, 2013, 90(1):45–60.

[8] DeMillo R A,Lipton R J,Sayward F G.Hints on test data selection:Help for the practicing programmer[J].IEEE Computer,1978,11(4):34-41.

[9] Hamlet F G.Testing programs with the aid of a compiler[J].IEEE Trans on Software Engineering,1977,3(4):279-290.

[10] HECKEL, R. AND LOHMANN. Towards contract-based testing of Web Service[C]. In Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS’04).2004.