2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies

Experimental Study of Web Service Availability Based on WSDL Documents

Liyuan Xin, Zhiyong Feng, Shizhan Chen*
Tianjin Key Laboratory of Cognitive Computing and Application
School of Computer Science and Technology, Tianjin University
Tianjin, China
{xinliyuan, zyfeng, shizhan }@tju.edu.cn

***Abstract*—The WSDL document is an XML-based interface that is used for describing the abstract functionality offered by a service and the concrete details of how a service can be called. It thus serves the most important factor of Web service availability over the Internet for that's the only way in which potential clients could search, discover and interact with the described service. In this paper, based on the analytical investigation on current status of Web services that exist on the Web today, we propose a tool which can detect Web service availability and repair some errors in the WSDL document. According to the different types of errors we collected and analyzed, our tool can automatically correct or prompt user interactively to patch the WSDL file. The experiment with more than 17000 Web services in real world indicates that our tool produces a high error detection rate with great repair rate. It would be a useful assistant to service provider to publish their WSDL file, and improve the service retrieval and employ process.**

Keywords-web service; availability; WSDL document; detection

I. INTRODUCTION

Recent year, Service-oriented computing using Web services [1] has emerged as a major research topic and considered as the best practice to support the development of rapid, low-cost, interoperable, evolvable and distributed applications in a dynamic environment. As a contract between a Web service consumer and a service provider, the WSDL [2] provides a machine-readable interface that is used for describing the abstract functionality offered by a service and the concrete details of how a service can be invoked. Consequently, the associated WSDL document serves the most important factor of Web service availability over the Internet for that's the only way in which potential clients could search, discover and interact with the described service. That is, a Web service that can be widely engaged on the Web depends on whether the WSDL document is completely correct and sufficiently accurate. Otherwise, the WSDL document with defects may lead to the clients cannot access the business unit exposed as a Web service even if it is running well.

WSDL document is generally registered into a public registry--for instance, Seekda [1], Xmethods [2], Service-repository[3], etc. by service provider. According to the WSDL document, users can discover the right Web service from a shared registry and can interact with it remotely. However, to our surprise, only 63% Web services are considered to be active on the World Wide Web [3]. In fact, even if a service is active which meanings the service's URL is active, there are many services in an unusable state, because defects and errors in WSDL document will cause huge information loss and the concomitant consequence that users cannot discover or invoke potential Web services. Therefore it is significant to improve the WSDL document quality for Web services availability. In this paper, we conduct a thorough analytical investigation on the current status of Web services that exist on the Web, and then propose a Web service availability detection and WSDL error correction tool (WDC) to detect Web service availability and, farther more, to repair some defects and errors in WSDL documents. The experimental results show that our tool is effective to produce high error detection rate with great repair rate. It would be a useful assistant to service provider to publish their WSDL file, and to improve the service retrieval and composition.

The remaining of the paper is organized as follows: Section II describes some of the related work. Section III discusses the results of analytical investigation. In section IV, we describe the proposal framework. The evaluation measures of the proposed approach and experimental results are presented in Section V. Finally, conclusion and future work are discussed in Section VI.

II. RELATED WORK

With the explosive growth of Web services on the Internet, the quality of Web services, including performance, reliability, availability, security and many other aspects [4], has caused widespread concern in academic and industrial circles. There are various efforts on the quality of Web services and Web service composition based on QoS (quality of service). QoS issues can be evaluated from the perspective of the providers of Web services and from the perspective of

[1] http://www.webservice.seekda.com
[2] http://www.xmethods.net
[3] http://www.wsdll.com

978-0-7695-5044-2/13 $26.00 © 2013 IEEE
DOI 10.1109/EIDWT.2013.68

the users of these services [5]. In [6] proposes a novel QoS-based dynamic service composition technique for web services with Ant Colony Optimization (ACO) in an optimization approach. Based on QoS measurement metrics, [7] proposes multiple criteria decision making and integer programming approaches to select the optimal service. From the user prospective, a Web Service is a black box that needs to be tested before being used, mainly to confirm the claims of services' providers about the quality of their services. In [8] leverages both the information contained in the WSDL/XSD files and the information provided by testers as well as enables testers to customize the fields to be tested, field constraints and data generation rules. In [9] present a collaborative filtering approach for predicting QoS values of Web services and making Web service recommendation by taking advantages of past usage experiences of service users. In another effort, the author in [10] describe a framework for service integrators to collaborate during Web Services testing by making test suites open to the public and share test results. However, these evaluations and detections on Web service quality devote efforts to the available Web service [10, 12]. And there exists a part of Web services that have been neglected, and they would not normally access and use on the Internet.

Considering the WSDL document quality impacting service discovery, the work described in [11] proposes a method for various bad/poor practices and common mistakes in WSDL documents, explain how these impact service discovery. This work focuses on improving Web service description for service discovery, rather than repairing the defects and errors in the WSDL documents. That will cause huge information loss and the concomitant consequence that users cannot discover or invoke potential Web services. Thus, an automated approach to detect and repair errors in the WSDL documents based on Web service availability is on its way.

III. COMMON ERRORS ANALYTICAL INVESTIGATION

To access a Web service, a service consumer must obtain the associated WSDL document. A WSDL is an XML-formatted document that includes information about the Web service, including the following information: operation that you can call on the Web service; input parameters that you pass to each operation; return values from an operation. The users of the Web services have a variety of ways to obtain services, such as public registry, Web service portals or directories, search engines. There are mainly three methods to generate a WSDL document. The first method is to automatically generate code by the program; the second method is generated by half automation tools that can automatically generate WSDL form and the programmers fill the content; the third method is developed by the programmer – that is, programmer write code. The first method would not cause errors in WSDL document if the program is correct. The latter two methods may cause some errors in WSDL document during the development process, such as that the negligence of programmer, programmer's poor competence and imperfect detection. This section introduces and analyzes the investigation results about Web service availability on the Internet. We collected a list of 17224 possible Web service interfaces that gathered by Web crawler Heritrix. Even if they come from the service search engine that human intervention to maintain. Through our investigation result, some of WSDL documents could be found some defects and errors so that the associated Web service unavailability. That gives us a lot of inspiration about improving the investigation approach for error detection and correction that will be discussed in next section.

To define the evaluation standard of analytical investigation, there are mainly three steps as follows: empty document validation, basic validation, parsing validation. If a WSDL document passed the three-step validation process, we think it is correct, and its associated Web service is available to access. Empty document validation is that validated file is an empty file (0 KB); basic validation is to conform if WSDL's syntax rules are correct and URLs state is active; parsing validation refers to validate the elements in WSDL document. That is, WSDL document correctness and associated Web service availability can be ensured by through the three-step validation.

The results of analytical investigation are as shown in fig. 1. It can be observed that empty documents took up a certain proportion, which might be caused by empty document address content or access exceptions. Empty document validation is the first step of our investigation. Only through empty document validation detection, specific content detection can be carried out, which is order to improve the efficiency of the algorithm.

It is possible to throw many runtime exceptions during the parsing validation process. Although a part of WSDL documents conform to WSDL specification and their service URLs are active, they pass the basic validation but not pass the parsing validation. So this part of the documents could contain some problems. Then a detailed error analysis and statistics on those 2682 documents. Detailed error analysis is as shown in Table I. And detailed error statistics is as shown in fig. 2.

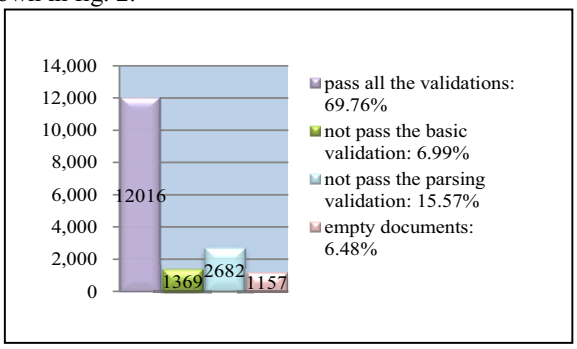


Figure 1. WSDL documents investigation result.

TABLE I. ERROR ANALYSIS

No.	Error analysis	
	Error Name	***Cause Analysis***
e1	Namespace prefix failed to identify	It is important to add namespace prefix before the element to represent its

No.	Error analysis	
	Error Name	*Cause Analysis*
		definition of the scope. The element lack namespace prefix would lead to not recognize its child element namespace prefix.
e2	The complex type element defines error conflict	Iterative method is adopted to deal with the complex type parameter until parse out basic type. If the number of iterations make more than 1,000 times, we think its definition is ambiguous.
e3	Namespace definition	Some namespaces require consistent definition in a WSDL document. If there is a conflict, it will cause error.
e4	Operation input/output parameter has errors	Operation input/output parameters transfer information has error.
e5	Element reference not found	The element reference address is not effective. The corresponding file namespace has error or incorrect.
e6	Contains invalid XML schema	The WSDL document contains invalid XML schema.
e7	Element undefined	A element definition is not found
e8	Target namespace is empty	Target namespaces exist null value
e9	Other unknown error	Unknown reason

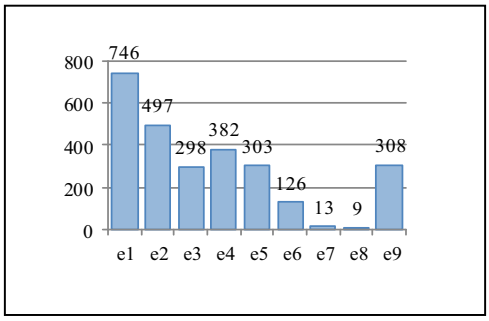


Figure 2. Error statistics

This analytical investigation can help service publishers and consumers understand the current status of the availability of Web services and the associated WSDL documents on the Internet, so as to serve future publishers who can pay attention to these problems, reduce the WSDL document mistakes and guarantee the availability of Web services. Errors in WSDL document will cause the Web service cannot be access. We observe those errors and find that much of those errors are caused by arbitrary definition of namespace, element. There are mainly three kinds of cases as follows: poor programmer competence or carelessness mistakes in constructing WSDL document process; URL of the service was changed, but the associated WSDL document was not changed; and other unknown reasons. An effective WSDL document should edit simply and logic clearly, which let the service consumers understand easily. According to the error analysis, it will give the corresponding modification tips and storage into database – artificially establish an "error_analysis" table, which used for the new tool, including error name, error reason, correction tip, etc.

Most of Web services have significant practical application meaning. Web service provider should detect the WSDL document before publishing, which can validate not only basic format of the WSDL document and service URL, but also document namespaces, elements definition. It also can accurately point out the error reason and error correction tip, so that error can be found in time and the WSDL document quality can be ensured.

IV. WEB SERVICE AVAILABILITY DETECTION AND WSDL ERROR CORRECTION TOOL (WDC)

A WSDL document quality is related to access and use the associated Web service. It is significant to ensure the availability of Web services, namely the WSDL document correctness of the Web service. Based on the investigation result and error analysis in the previous section, we develop a Web service availability detection and WSDL error correction tool, named WDC, which can further detect WSDL document, accurately point out its existing problems, give an error correction tip and automatically correct or prompt user by interaction. Fig. 3 shows the WDC workflow overview. It is composed of three modules: detection manager, error correction manager and user front end. The above box is detection manager and the under box is error correction manager in fig. 3.

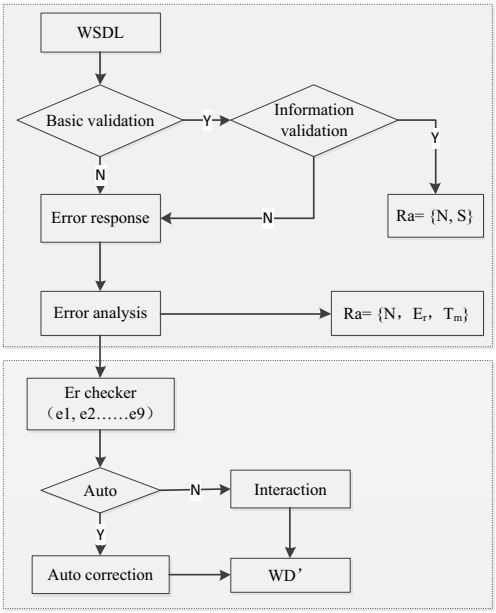


Figure 3. WDC workflow overview

A. *Detection manager*

User inputs a WSDL document path. The WSDL document will be uploaded to the server which executes program and output results. Detecting manager encapsulate many complex functions so that users can very simply detect WSDL document, get error reason and error correction tip by web interface. Input/output of the detecting manager are defined as follows:

In: *DocPath*;

Out：*Ra = {N，Er，Tm，S}*.

DocPath is WSDL document path. *Ra* is result set, including service name *N*, detailed error reason *Er*, error correction tip *Tm*, correct identification *S*.

This module is improved by investigation method. Previous section points out that the evaluation of WSDL document has three steps. In order to explain the detection manager workflow, this section extends to five aspects < *Ul, Vb, Vi, E, R* >. In order to establish detection manager, we adopt Dom, WSDL4J to manipulate WSDL document.

Ul is uploading and reading WSDL document.

Vb is basic validation: 1) validate basic format of the WSDL document; 2) validate whether the Web service is active by accessing Web service URL. We think a Web service URL is effective by the return page state information. Concrete implementation way is to definition a WSDLReader instance, if the program can correctly return an instance definition, then basic validation is successful.

Vi is information validation: To validate information in WSDL document by manipulating the WSDL document. Between father label and son label in the WSDL document have very strong logic relationships. And document format is fixed. Therefore to exact information from the documents in the order given: service → port → binding → portType → operation→message→part.

E is exception handle: This step is extended by investigation method and is the core of this module. Either *Vb* or *Vi* has error response, it would analyze the error. There are various kinds of WSDL document errors. However, not all of errors can be solved in the previous section. Therefore, we listed the errors that detection manager could handle, as follows:

1) WSDL basic format: whether the document conforms to WSDL's syntax rules.

2) Service URL: it will try to connect the service URL. If URL can be connected successfully in 10000 ms, we believe that the service is active.

3) Empty namespace value.

4) Namespace definition conflict: it is important to set namespaces for resolving the problem of naming conflict. However, some relevant position namespace and its content must be same in the WSDL document. If there has a conflict, it will cause an error.

5) Element reference: namely check that referenced address is active.

6) Namespace prefix failed to identify: To avoid naming conflicts, you should use the namespace prefixes for representing its definition of scope. The element lack namespace prefix would lead to not recognize its child element namespace prefix.

7) Lack of effective element definition: Lack of effective element definition will cause that the element cannot found.

8) Ambiguous element definition: It adopts iterative method to deal with the complex type parameter until parse out basic type. If iterations make more than 1,000 times, we think its definition is ambiguous.

Specific method overview is as follows:

Algorithm1: Error handle

```
ErrorHandle (DocPath)
    e =null  // e represents an error
    get WSDL document from DocPath
    try {
        parse WSDL document
        extract information from WSDL document
        N = service name
    } catch ( Exception e) {
        turn e into a numeric code
        switch (e)
        {
            case 1: The document does not conform to WSDL's
                    format
            case 2: Connect service time > 10000 ms
            case 3: There exists an empty namespace
            case 4: There exists a conflict between namespace.
            case 5: There exists a namespace prefix that failed
                    to identify.
            case 6: There exists an element reference that is not
                    active.
            case 7: There exists an element that lack an effective
                    definition
            case 8: The iteration of complex type parameter more
                    than 1000 times.
            default: Any other error response.
        } finally{
            Return N, e
        }
```

R is result analysis: if there is no error response, it will return *Ra = {N, True} (S=True)*, namely the WSDL document has no error; If there is an error response, it will return *Ra = {N, Er, Tm}*, including the service name *N*, detailed error reasons *Er* and error correction tip *Tm*.

Specific method overview is as follows:

Algorithm2: Error Analysis

```
ErrorAnalysis (WSDLSet, Errors)
Er, Tm = null
ErrorSet = Φ
N = WSDLSet[w]
If (Errors == Φ)
   Ra = (N, True)
Else
   For each e in Errors
      Select error reason, correction tip from error_analysis
      where error name = e;
         If (error reason is not null  && correction tip is not
           null)
           Er = append(Er, error reason)
           Tm = append(Tm, correction tip)
      End If
   Endfor
   Ra = (N, Er, Tm)
End If
```

Though this tool could not guarantee all WSDL documents are effective that are passed detection in this paper. But the error detection rate is improved greatly. WDC can accurately detect the overwhelming majority of errors

existing in WSDL documents, so that guarantee WSDL document correctness and Web service availability.

B. *Error correction manager*

After detecting the type of error in a WSDL document, user would choose whether to repair the error. And there are two methods of error correction: automatic correction or interactive adjustment. For the no automatic method to repair the errors, it adopts user interaction that can help users find errors and guide users to repair errors. Table II presents the methods of error correction.

TABLE II. METHODS OF ERROR CORRECTION

Method	Error type
Auto	e1
Interaction	e2, e3, e4, e5, e6, e7, e8

Input/output of the error correction manager are defined as follows:

In：*Er, WD*;

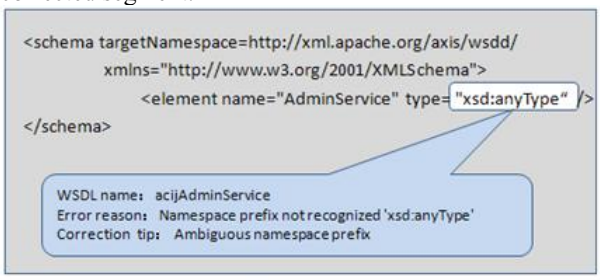
Out：*WD'*.

Detailed error analysis Er is including error type *Ec* and error position *Ew*, *Ec* ∈ (e1 e2... and e9); *WD* is the original WSDL document; *WD'* is the corrected WSDL document. Interaction method will be introduced as an example later. Automatic method overview is described as follows:

Algorithm3: Automatically Modify Error

```
1  ModifyError(WD, Er)
2  If (Er != null)
3      While (Test. WD == false)
4          Extract wp from WD // Exact error segment wp from
           WSDL document
5          If (Er.auto == true)
6              Extract nsprefix,p from wp//Exact unidentified
               namespace prefix nsprefix, nsprefix label p
7              While (p.prefix == null)
8                  Add nsprefix before p
9                  p = p.parent
10             EndWhile
11             Return WD'
12         Else
13             Return wp // response wp to user
14         EndIf
15         If (user.exit == true)
16             Break;
17         EndIf
18     EndWhile
19 EndIf
```

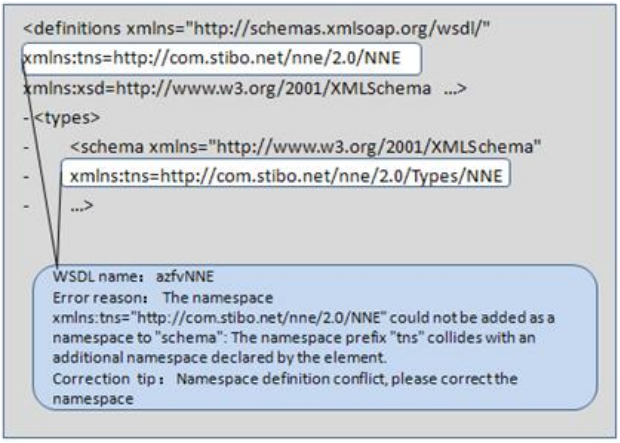
Next, we will use two examples to help you understand how WDC works. Examples are selected real WSDL document segments. Fig. 4 (a) is a WSDL document segment, namespace prefix "xsd" of "xsd: anyType" is not identified in this segment by the detection. And fig. 4 (b) shows error correction result. The parent label lacks a namespace prefix. It is so easy to cause ambiguity that namespace prefix of child label would not be identified. In the analysis error process, the tool can automatically identify the error and repair it. Namely it adds the same namespace "xsd" in front of label <schema> and <element>. Fig. 4 (b) shows the corrected segment.



(a)

(b)

Figure 4. WSDL document correction example 1



(a)

(b)

Figure 5. WSDL document correction example2

Another example is an interactive adjustment example. In fig. 5 (a) two namespace prefix are "tns" in the white box which represent two different namespace. It would cause the namespace conflict. Because that some namespaces require consistent definition in a WSDL document. This error is the type e3.In the analysis error process, the tool can interact with user. So the server would return message to user. Then

user would repair the error and send it to the server. The server would return the corrected WSDL document to the user. Fig. 5 (b) shows a feasible corrected segment.

C. User front end

In the current implementation, the user front end is a set of JSP pages that provide a simple interface for the users to perform several operations and communicate with the server. In the error detection phase, the user inputs a WSDL document path, while the user front end returns the detecting result set Ra to the user. In the error correction phase, it interacts with the user and show the correct WSDL document ultimately.

V. EXPERIMENTAL RESULTS

In order to evaluate the applicability and usefulness of the proposed method, we perform an experiment with 17224 WSDL documents that were used for analytical investigation in section III. Then we define a framework to evaluate and analysis the experiment results. The framework contains measures specific to our detection approach, including detection result DR and evaluation result ER. We use T to represent 17224 WSDL documents. In our work, the detection result can be defined as 7 data sets.

- R is a random set of data extracted from T;
- B is a set of data which are not passed basic validation for R;
- E is a set of data which are not passed empty document validation for R;
- P is a set of data which are passed the basic validation but did not passed the parsing validation for R;
- S is a set of data which can be successfully detected errors for P by using WDC;
- C is a set of data which can be successfully detected errors with an accurate error correction tip for P by using WDC;
- A is a set of data which are passed detection from R by using WDC.

We can use W to represent all WSDL documents which are not passed validation by using WDC. $W = P \cup B \cup E$. Therefore, it can be inferred $R = P \cup B \cup E \cup A$.

The evaluation result includes 4 indicators about WDC, including the following indicators:

- TDr = $(B \cup E)/W$, which is an error detection rate without WDC for W;
- WDr = $(S \cup B \cup E)/W$, which is an error detection rate of successful detection for WSDL documents with errors in W by using WDC;
- Ar = C/S, which is an accuracy rate of prompting correction tip for WSDL documents with errors in S by using WDC;
- Cr = A/R, which is an accuracy rate of WSDL documents in R where no error can be detected.

Among the above evaluation value, WDr and Ar are two important indicators for error detection and error correction of WSDL documents which reflect the validity and accuracy of the WDC. Meanwhile, the contrast of TDr and WDr, which embodies the proposed tool detected diversity of the errors and high detection rate. Cr reflects the proportion of correct WSDL document in the detected sample data.

In order to reflect the effect of the stability of WDC detection, we randomly selected 100, 500, 1000 WSDL documents grouping experiments from the data set T. If using the original validation method, it only validates WSDL basic format and service URL. However, in the WSDL documents, there still exist some errors which cannot be checked out by using general method, so that the associated Web services cannot be really available to users. Therefore, we perform our new method on the sample data. Table III shows the results.

TABLE III. EXPERIMENTAL RESULTS

Group	R	B	E	P	S	C	A
Group 1	100	6	6	17	15	10	71
Group 2	500	41	36	74	59	44	349
Group 3	1000	77	63	161	142	103	699

TABLE IV. EVALUATION RESULTS

Group	TDr	WDr	Ar	Cr
Group 1	41.38%	93.10%	66.67%	71%
Group 2	50.99%	90.07%	74.58%	69.8%
Group 3	46.51%	93.69%	72.54%	69.9%

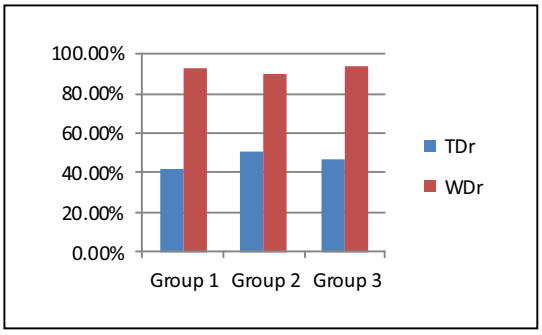


Figure 6. Comparison of error detection rate

Fig. 6 illustrates the comparison of sets of data error detection rate. We can see that TDr is about 46%. If you do not use WDC proposed in this paper, only about 46% error documents could be successfully detected. While using WDC, over 90% (WDr) wrong documents could be successfully detected. It was a substantial increase in error detection rate that was almost 2 fold compared with before. We can infer that error reason e9 (other unknown error) might be failed to detect. To evaluate the error correction tip, we observe each error reason and its error correction tip. Error correction tip accuracy (Ar) can reach about 70% that is a good result. The correct rate (Cr) of the WSDL documents in sample data is consistent with the investigation in Section II, which was further evidence of the experiment reliability.

Then we can discuss the two correction methods, respectively. For interactive adjustment method, we cannot perform an exact experimental result. The reason is that we cannot guess service provider's real intentions, especially empty element definition or reference address failure. But the tool can guide the user by interaction. From the service non-provider perspective, error correction rate may be lower by the interaction method. For the automatic correction method, based on deep analysis of the cause of error e1, we believe that this error could be corrected automatically. A small-scale experiment shows that automatic correction rate can reach 100%. After our correction proposed in this paper, we estimate at present available Web services on the Internet would be increased by 6% than before.

VI. CONCLUSION AND FUTURE WORKS

In our study, we investigated the characteristics and defects in the WSDL documents for the Web service availability on the Internet. Then we proposed an automatic tool to detect Web service availability and associated WSDL document correctness. Correcting error in WSDL document by automatic correction or interactive adjustment would improve the availability of Web services and play a good supporting role to Web service consumers. Meanwhile, we aim to assist developers in finding and repairing some errors as early as possible. If it can be used before registering Web service, some errors in WSDL document would be avoided that is a guarantee of Web service availability.

In the future, we are planning to extend this study for analyzing how to suggest suitable error correction tip and how to interact with users more friendly to improve the correction. Furthermore, we are also going to be concern with how to improve the WSDL document quality for service discovery.

ACKNOWLEDGMENT

Funded by NSFC grant No.61173155 and No.61070202, and the 985 Project of Tianjin University grant 2010XG-0009. Corresponding author: shizhan@tju.edu.cn.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, “Web Services: Concepts, Architectures and Applications,” Springer, 2003

[2] http://www.w3.org/TR/wsdl

[3] E. Al-Masri, and Q. H. Mahmoud, “Investigating WebServices on the World Wide Web,” Proc. 17th Ann, International Conference on World Wide Web(WWW 08), ACM Press, Apr. 2008, pp. 795 - 804.

[4] W3C Working Group. QoS for Web services: requirements and possible approaches. http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

[5] D. A. Menasce, “QoS Issues in Web Services,” IEEE Internet Computing, vol. 6, Nov. 2002, pp. 72 - 75.

[6] W. Zhang, C. K. Chang, T. Feng, and H. Jiang, “QoS-based Dynamic Web Service Composition with Ant Colony Optimization,” Proc. 34th Ann, IEEE Computer Software and Applications Conference, IEEE Press, Jul. 2010, pp. 493 - 502.

[7] A. Huang, C. Lan, and S. Yang, “An Optimal QoS-based Web Service Selection Scheme,” Information Sciences, vol. 179, Sep. 2009, pp. 3309–3322.

[8] Z. Li, J. Zhu, L. Zhang, and N.M. Mitsumori, “Towards a practical and effective method for Web services test case generation,” Automation of Software Test(AST 09), IEEE Press, May. 2009, pp. 106-114.

[9] Z. Zheng, H. Ma, M.R. Lyu, and I. King, “QoS-Aware Web Service Recommendation by Collaborative Filtering,” IEEE Transactions on Services Computing, vol. 4, Apr-Jun. 2011, pp. 140 - 152.

[10] N. E. Ioini, and A. Sillitti “Open Web Services Testing,” IEEE World Congress on Services, IEEE Press, Jul. 2011, pp. 130 - 136.

[11] M. Crasso, J. M. Rodriguez, A. Zunino, and M. Campo, “Revising WSDL documents: Why and how,” IEEE Internet Computing, vol. 14, Sept.-Oct. 2010, pp. 30–38.

[12] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini, “WS-TAXI: a WSDL-based testing tool for Web Services,” International Conference on Software Testing Verification and Validation (ICST 09), IEEE Press, Apl. 2009, pp. 326-335.

[13] W. Hui, F. Zhiyong, C. Shizhan, X. Jinna, and S. Yang, “Constructing service network via classification and annotation,” IEEE International Symposium on Service Oriented System Engineering, Jun. 2010, pp. 69-73.

[14] A Averbakh, D Krause, and D Skoutas, “Exploiting user feedback to improve semantic web service discovery,” The Semantic Web, Vol. 5823, Oct. 2009, pp 33-48.

[15] N. Milanovic, and M. Malek, “Current Solutions for Web Service Composition,” IEEE Internet Computing, vol. 8, Nov.-Dec. 2004, pp. 51- 59.

[16] S.C. Oh, D. Lee, and S. R. Kumara, “Effective Web Service Composition in Diverse and Large-Scale Service Networks,” IEEE Transactions on Services Computing, vol. 1, Jan.-Mar. 2008, pp. 15-32.

[17] Y. Guo, S. Chen, and Z. Feng, “Composition Oriented Web Service Semantic Relations Research,” IEEE International Joint Conference on Service Sciences(IJCSS), IEEE Press, May, 2011, pp. 69- 73.