Zhao, J., Zhou, J. and Yang, H. (2018) 'A matching approach to business services and software services', *International Journal of Computational Science and Engineering*, 16 (2), pp. 123-131.

Official URL: http://dx.doi.org/10.1504/IJCSE.2018.090458

A Matching Approach to Business Services and Software Services

Junfeng Zhao
Assistant Professor, College of Computer Science
Inner Mongolia University
Hohhot, China
cszjf@imu.edu.cn

Jiantao Zhou
Professor, College of Computer Science
Inner Mongolia University
Hohhot, China
cszjtao@imu.edu.cn

Hongji Yang
Professor, Centre for Creative Computing
Bath Spa University
Bath, England
h.yang@bathspa.ac.uk

Abstract—***Recent studies have shown that Service-Oriented Architecture (SOA) has the potential to revive enterprise legacy systems [1-10], making their continued service in the corporate world viable. In the process of reengineering legacy systems to Service Oriented Architecture, some software services extracted in legacy system can be reused to implement business services in target systems. In order to achieve efficient reuse to software services, a matching approach is proposed to extract the software services related to specified business services, where service semantics and structure similarity measures are integrated to evaluate the similarity degree between business service and software services. Experiments indicate that the approach can efficiently map business services to relevant software services, and then legacy systems can be reused as much as possible.***

Keywords-*software service; business service; matching approach; sematics similiarity measure; structure similarity measure.*

I. Introduction

Service Oriented Architecture has been deployed in software reengineering, which can help reuse legacy systems efficiently and protect the existing assets of IT systems. Through reengineering to SOA, legacy systems can revive and operate in innovative and advanced ways, so that they can meet the new need of users better.

In the process of reengineering legacy systems to SOA, software developers implement service-oriented analysis and designs according to the requirements of target systems, and then software acts as a group of loosely coupled business services which describe the work related to some business but does not produce a tangible commodity. The discovered business services are described by WSDL files according to web service specification. To legacy systems, business modules can be extracted to form software services by bottom-up analysis and relevant descriptions can be created by analysing the implementation of business modules. Contrary to business service, software service is the reality existence that realises specific business functions. Based on the description of business services and software services, an efficient matching algorithm need to be designed to discover the most compatible software service with the desired business service, so business logic in legacy system can be reused. In this paper, an iterative matching approach, which combines text semantics and structure similarity measure, is proposed to achieve this goal. Text semantics similarity measure can assess the compatibility between business service and software service on the whole, and structure similarity measure

This work was partially supported by National Natural Science Foundation of China (No. 61462066, No. 61262082, No. 61562064), Key Project of Chinese Ministry of Education (No. 212025).

Jiantao Zhou is the corresponding author (email:cszjtao@imu.edu.cn).

evaluates the compatibility by matching the interfaces of business service and software service specifically.

The rest of the paper is organised as follows. Section Ⅱ introduces the description specification of software service. Section Ⅲ presents the matching approach between business service and software service. Related experiment is presented in Section Ⅳ. Section Ⅴ introduces the related work to service identification and similarity measure. Conclusions and future work are presented afterwards.

II. DESCRIPTION OF SOFTWARE SERVICES

For the effective reuse of software services extracted from legacy systems, matching between business service and software service need to be considered according to their description documents. Therefore, effective description to software service is necessary. For this purpose, a specific description specification is proposed to express the functional and structural information of software service.

For the purpose of automatic matching between software service and business service, the description specification should be compatible with WSDL specification that is used to describe business service. WSDL specification is shown in Fig. 1, where "Types" is the container of data types; "Message" is the abstract type definition for data structure of communication message that is defined by the types described in "Types"; "Operation" describes the operations supported by service, and an operation represents the request/response message pair of an access point; and "service" is the collection of related service access point.

Referencing WSDL specification, the description specification of software service is defined by XML Schema as Fig. 2. The root element "reusablecomponent" represents software service, and attribute "name" is used to identify a specific software service. The root element contains three child elements, respectively "desc", "types" and "operations". Element "desc" explains the function of software service by a section of text document. Element "types" contains child element "type" which is used to define the complex type related to the implementation of software service. Complex type is described by exhibiting its attributes which are represented as child element "attribute" in specification. Element "operation" defines the external interface of software service, which includes child element "note", "input" and "output". Element "note" describes the function of interface, "input" shows the input parameters of interface, and "out" indicates the result type of interface. Through the definition of description specification for software service, the related information and description format are specified, which facilitates the follow-up matching between software service and business service.

III. MATCHING ALGORITHM BETWEEN SOFTWARE SERVICE AND BUSINESS SERVICE

Based on the description specification of business service and software service, a novel algorithm is proposed to realise matching between business service and software service, which integrates text similarity and structure similarity measures. According to the text description of business service and software service, text similarity measuring achieves comparison between business service and software service in general. In addition, taking structure description of business service and software service as input, structure similarity measure carries out comparison between business service and software service in detail, which takes advantage of semantics of identifiers and compatibility of data types. Due to the possible granularity incompatibilities between business service and software service, the combination adjustment to software services and iterative matching are considered in the process of matching. The execution process of the matching algorithm is shown in Fig. 3.

A. *Text similarity measurement*

The process of text similarity measure can be divided into three steps. Firstly, the indexes of text description for software services can be created by Lucene. Then, semantic expansion of text description for business service will be obtained by using WordNet. Finally, taking the expanded description of business service as query text and retrieving it from the indexes of text description of software services by Lucene, the similarity degree of text description between business service and software services can be obtained. The process of text similarity measure is shown as Fig. 4, which corresponds to the activity of text similarity measurement in Fig. 3.

The implementation of text similarity measurement is based on Lucene toolkit which can be embedded into applications to realise full-text information retrieval [17]. When performing keyword retrieval, Lucene uses TFIDF algorithm to compute the relevance between keywords and documentation.

B. *Structure similarity measurement*

Structure similarity measurement usually adopts the method of signature matching. In order to overcome the shortcoming of signature matching, structure similarity measure for business service and software service integrates the measure to semantic similarity of identifier and compatibility of data type, not only fall back on data type. Description information of software service includes interface signatures and custom types. The identifiers of interface and input parameter can reflect relevant functional semantics, and the identifiers of data type, message and operation have functional semantics as well. In reality, the identifiers marking same entity concepts or function concepts may not be based on different words literally, but have similar meaning semantically. For example, the interface identifiers with 'calculate' and 'count' may describe the same business logic, but with different words. So the similarity measurement to identifiers should carry out on semantics. The experimental results in [11] show that the semantic structure measure can obtain better matching effects. Structure similarity measure to business service and software service need to combine data type similarity measures in addition to semantic structure similarity measures, because the identifier of operation result in software service cannot be obtained but relevant data type can. So the measurement on operation results of business service and software service need to be executed according to the compatibility of data types. The process of structure similarity measurement is shown as Fig. 5.

Semantic similarity measure to service names adopts method Lin in Java WordNet Similarity. Java WordNet Similarity is an open source project for semantic similarity calculation based on Java and WordNet, where many semantic similarity algorithms are implemented[18]. Lin considers concept similarity from the viewpoint of information theory, and believes that similarity degree depends on the intercommunity and difference between meaning of concepts.

Operation similarity measurement is composed of similarity measure to identifiers, input parameters and return results. Similarity measure to input parameters can be divided into two kinds of cases. When the data types of input parameters are both simple, the similarity degree will be obtained by executing semantic similarity measure to parameter identifiers. On the other hand, the attributes of complex type will be extracted as new input parameters, then a new nested call to parameter similarity measure will be carried out; For the similarity measure to return results, it is different to input parameters due to the absence of return result identifiers in description of software service. The similarity measurement to return results relies on the compatibility measure to the data types of return results. When the types are both simple, the similarity degree is assigned according to the compatibility between the types; otherwise, the comparison should be executed according to the attributes in complex types.

C. Adjustment to software service based on association degree

In the process of matching between business services and software services, the first matching may not produce the desired result. There may be two kinds of reasons. One is that the desired software service related to business service does not exist in legacy system originally. The other case may be that unreasonable extraction of software service from legacy system leads to unsatisfactory matching result, although the relevant business logic related to business service does exist.

To the second reason, one specific case is that the granularity of software service may be incompatible to the desired business service. Under the premise that software service extraction prefers finer granularity, the adjustment is to enlarge the granularity of software services by combing existing software services, through which the adjusted software services will possess expanded business functions. The adjustment can be carried out according to the association degree between software services.

In fact, multiple matching iteration may be needed before obtaining the desired results. After the first round matching, the software service with the highest similarity degree to business service can be discovered. Then calculating association degrees between this software service and all other software services, the software service with the highest association degree will be chosen to combine with this software service. So a software service with coarser granularity will be created, meanwhile adjusting the relationship between new software service and other software services. When the software service with the highest association degree is not unique, the software service with minimum number of classes should be selected. If it is still not unique, the software service with higher overall similarity degree will be selected. To evaluate the association degree between software services, a calculation expression is constructed as (1):

$$\mathrm{Cou}_{i,j} = \left(\mathrm{I}_{i,j} + \mathrm{I}_{j,i}\right) * \mathrm{W}_{\mathrm{I}} + \left(\mathrm{G}_{i,j} + \mathrm{G}_{j,i}\right) * \mathrm{W}_{\mathrm{G}} + \left(\mathrm{A}_{i,j} + \mathrm{A}_{j,i}\right) * \mathrm{W}_{\mathrm{A}} + \left(\mathrm{D}_{i,j} + \mathrm{D}_{j,i}\right) * \mathrm{W}_{\mathrm{D}} \tag{1}$$

$\mathrm{Cou}_{i,j}$ represents the association degree between software service i and j. $\mathrm{I}_{i,j}$ and $\mathrm{I}_{j,i}$ represent the number of edges with implementation relationship between software service i and j; similarly, $\mathrm{G}_{i,j}$ and $\mathrm{G}_{j,i}$ corresponds to generalisation, $\mathrm{A}_{i,j}$ and $\mathrm{A}_{j,i}$ corresponds to association, and $\mathrm{D}_{i,j}$ and $\mathrm{D}_{j,i}$ corresponds to dependency; W_{I}, W_{G}, W_{A}, W_{D} represent the weights of implementation, generalisation, association and dependency.

After adjustment, the new software service will possess expanded text description and more comprehensive function interfaces, due to the combination of previous software services. Evaluating the similarity degree between the new software service and business service, then the new degree will be compared with the highest similarity degree obtained by previous round. If the new similarity degree excels the previous, combination, matching and comparison will be carried out again in the same way. Until the matching result is inferior to the previous, the matching will stop and the matching result obtained by previous round will be returned as the desired match.

The similarity degree between business service and software service is obtained by adding up text similarity degree and structure similarity degree. In order to realise the comparison between matching results, the text similarity degree and structure similarity degree will be normalised. In practical terms, every text similarity degree will be divided by the sum of all text similarity degrees in the matching round, and every structure similarity degree will be divided by the possible maximum value. So that the scopes of text similarity and structure similarity will fall between 0 to 1. The calculation expression is shown as in Formula (2), where $sim_{c_i,s}$ represents the overall similarity degree between the ith software service and business service s, which is the sum of normalised text similarity degree $nomdocsim_{c_i,s}$ and normalised structure similarity degree $nomstrsim_{c_i,s}$; numope_{c_i} indicates the number of operations in ith software service, and numope_s indicates the number of operations in business service; docsim_{sum} expresses the sum of text similarity degrees before normalisation; and $\mathrm{strsim}_{c_i,s}$ represents the structure similarity degree between business service and the ith software service.

$$\begin{gathered} sim_{c_i,s} = nomdocsim_{c_i,s} + nomstrsim_{c_i,s} \\ \mathrm{docsim}_{sum} = \sum_{i=1}^{n} \mathrm{docsim}_{c_i,s} \\ \mathrm{nomdocsim}_{c_i,s} = \mathrm{docsim}_{c_i,s}/\mathrm{docsim}_{sum} \\ \mathrm{nomstrsim}_{c_i,s} = \mathrm{strsim}_{c_i,s}/(3 * \max(\mathrm{numope}_{c_i}, \mathrm{numope}_s) + 1) \end{gathered} \quad (2)$$

IV. EXPERIMENT

JUnit is a simple framework to write repeatable tests for Java developers, which makes Java unit test more standard and efficient [19]. It is an instance of the xUnit architecture for unit testing frameworks. We take it as a legacy system to verify the validity of our approach.

Based on the clustering analysis to JUnit, 6 software services were discovered in the implementation of JUnit. By analysing the related public interfaces and their corresponding comments, the description file of the software services was obtained. The excerpt of the description file is shown as Fig. 6.

A business service related to core function of JUnit was defined, which includes three operations such as starting test, ending test and test assertion. The excerpt of description file is shown as Fig. 7. Taking the description files in Fig. 6 and Fig. 7 as inputs, the matching algorithm is used to measure the similarity degree between business services and software services. The matching result of the first round is shown in Table 1, which describes text similarity degrees, structure similarity degrees and overall similarity degrees between the business services and the software services. It can be seen from the table that the text similarity degree between the first software service and the business service is maximum and the second software service is the most similar to business service from structure. On the whole, the first software service is the most similar to the business service.

Based on the relationship between the software services, association degrees between software services can be obtained by performing Equation (1). Through analysing the association degrees, we found that the 3rd software service had the highest association degree with the first software service. The first software service and the 3rd software service were combined to form a new software service with coarser granularity. Then the similarity measurement between the new software service and business service was carried out again, and the matching result is shown in Table 2. The data in table indicates that the similarity degree between the new software service and the business service is higher than the first round match. So the new software service is considered to be more compatible with the business service than the first software service before adjustment.

Adjusting software services according to the same way, further similarity measure was executed between the new software service and the business service. Table 3 shows the measure result. The degree of similarity does not increase as before, on the contrary, it is lower than the previous degree of similarity. It is suspected that further adjustments are redundant, the matching algorithm terminates and returns the previous matching result.

By analysing the business function of software services, we can find that the first software service includes the operations of test startup and end, and the second software service realises test assertion. So the matching result conforms to the practical case. In addition to this business service, other three business services related to the core functions of JUnit were defined and matched with software services, the matching results are consistent to the matter of fact on the whole. It can be concluded that the matching approach is effective.

V. RELATED WORK

In service oriented projects, service identification plays an important role as the foundation for the follow-up work. Service identification is used to define a set of services supporting business architecture in enterprise application environment, which is a process of extracting services from business requirement or existing IT systems [1]. Analysis and design about services can be divided into three kinds of methods, namely top-down, bottom-up and convergence in the middle. Top-down methods take system business requirements as a starting point, and abstract business requirements as a set of related business services. Bottom-up methods start from legacy systems, and extract business logic as software services to support business services. Convergence in the middle methods combine bottom-up and top-down methods. To reuse the business logic of legacy systems in SOA, convergence in the middle methods not only make full use of the existing assets of legacy system, at the same time but also achieve the real application requirements, where top-down methods implement the abstraction of business services; bottom-up methods realise the extraction of software services; and finally business services are mapped to software services.

For top-down methods, there exist many implementation approaches, including model-driven approaches [2, 3], pattern matching approaches [4, 5], ontology mapping approaches [6, 7] and so on. The abstracted business services can be described according to WSDL specifications. Likewise, many technologies are adopted in bottom-up methods, including concept analysis [8], information retrieval [9] and clustering analysis [10]. Therein, the clustering analysis is paid more attention in many studies. For example, an improved agglomerative hierarchical clustering algorithm was proposed to restructure legacy code and to facilitate legacy code extraction for web service construction, which supports service identification and service packaging and archives legacy system migration into service-oriented architectures by providing functional legacy code as Web services [11].

To achieve recognition services from legacy code, an approach combining text similarity measurement with structure match was proposed in [12]. The text similarity measure is based on free text query and text description extracted from legacy code, and structure match based on WSDL document and interface characteristics extracted from source by reverse engineering. Two major steps are carried out to implement service recognition. First of all, candidate services will be extracted from source code of legacy system, where method signatures will be expressed as WSDL operations, and the complex types that are used in method signatures will be represented as XML schema. Then, the target service will be compared with all possible candidate services. The comparison is divided into two stages. First, the information retrieval technology is adopted to calculate the similarity between description text of target services and candidate services, then using the matching algorithm in [13] to measure structural similarity, and comparison results are ranked according to the relevance.

In [13], an evaluation approach based on vocabulary and structure similarity measurement was proposed to measure similarities between service interfaces, which is adopted to realise web service discovery, service version comparison, service classification and so on. Vocabulary match is used to calculate the semantic similarity between concept description, and structure match is used to measure the similarity between combined concepts, including service, operation, message, data type and so on.

Similarly, a group of web service discovery approaches are described in [14], which combine information retrieval and structure matching. Therein, a kind of structure matching extends the signature matching of component retrieval, involving the comparison of operation sets of services. Another kind of structure matching is related to semantics, where the identifiers in service description are considered in addition to the types of programming language and syntactic relations. WordNet is deployed to calculate the semantic distance between the related element identifiers in WSDL specification, rather than the measurement based on the compatibility of data types. An approach for web service retrieval based on the evaluation of similarity between web service interfaces is presented in [15], which combines the analysis of structures and the terms used in them. A practical approach is proposed to measure the similarity of web services based on their interfaces in [16], where both semantic and lexical metrics are combined to match web services, operations, messages, parameter identifiers and types. The experiment results show its practicability. In [20], a semantic matchmaking algorithm was proposed to search cloud services that best meet the users' requirements, and experimental results show that the semantic technologies can enhance the performance of supply-demand matchmaking. A context-aware discovery process was proposed to identify the related services in [21], which is based on the similarities of semantics and structure. An approach was proposed to identify the best services in [22], which makes use of functional and non-functional characteristics, moreover focuses on computational optimization of selection.

The application context of our paper is similar to [12], which is aimed at recognising services from legacy code. [13, 14, 15, 16] have identical background, namely measuring the similarity of web services. Our paper and [12] measure structural similarity in the similar way with [13, 14, 15, 16], meanwhile there exist some obvious differences between these studies and our approach. The candidate services all come from individual classes in the existing study, so the granularity of software service is too small to match target service. On the contrary, the candidate services will be extracted from legacy systems by adopting clustering in our study, so the granularity will be more compatible to target services. In addition, our approach have difference in structural similarity measure, which combines semantic and data type similarity measure, meanwhile considers the adjustment to candidate services.

VI. CONCLUSIONS

Reengineering legacy system to SOA can make legacy systems survive and meanwhile take full advantage of the existing assets of legacy system. To achieve the effective reuse of software services extracted from legacy systems, a matching approach between business service and software service was proposed. Based on the description information of business service and software service, text similarity measurement and structure similarity assessment are adopted to discover the software service related to business service. In view of the granularity difference between software service and business service, an iteration strategy is adopted to realise the match between business service and combined software service, which can improve the possibility of discovering more accurate software service related to business service.

Currently, the scale of software systems analysed in the experiments is relatively small, and therefore large-scale and complex legacy systems should be used in follow-up experiments. The efficiency of matching algorithm can be further verified, and certain adjustments and optimisation to the matching algorithm may be carried out to promote its practical role in reengineering.

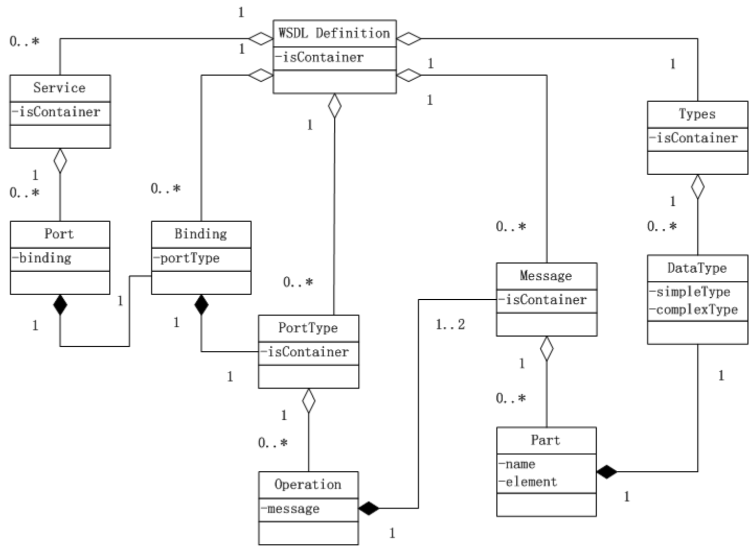

Figure 1. WSDL Specification

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="reusablecomponent">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="desc" type="xs:String">
        <xs:element name="types">
          <xs:complexType>
            <xs:element name="type" minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:element name="attribute" type="xs:String" maxOccurs="unbounded"/>
                <xs:attribute name="name" type="xs:string" use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:complexType>
        </xs:element>
        <xs:element name="operation" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="note" type="xs:String" minOccurs="0"/>
              <xs:element name="input" type="xs:String" minOccurs="0" maxOccurs="unbounded"/>
              <xs:element name="output" type="xs:String"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 2. Software Service Description Specification

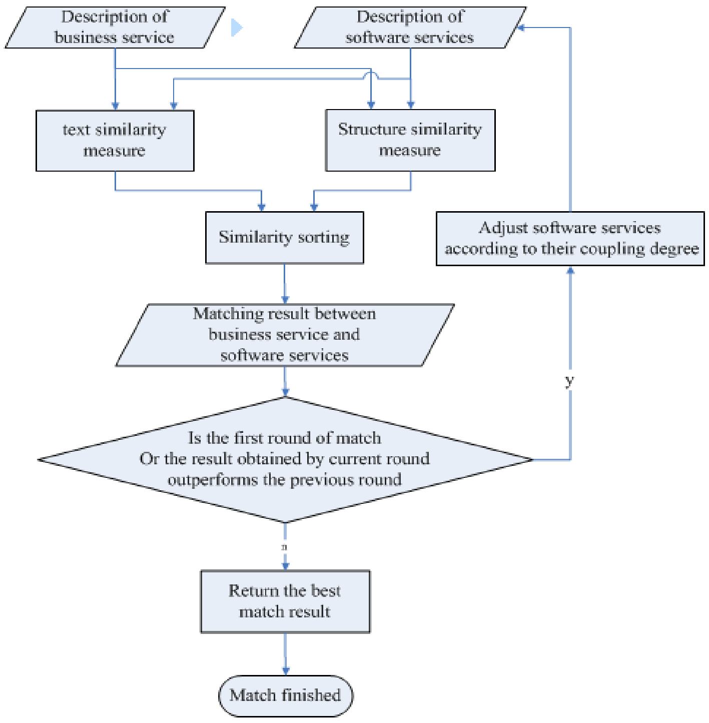
Description of business service
Description of software services
text similarity measure
Structure similarity measure
Similarity sorting
Adjust software services according to their coupling degree
Matching result between business service and software services
y
Is the first round of match Or the result obtained by current round outperforms the previous round
n
Return the best match result
Match finished

Figure 3. Process of Matching

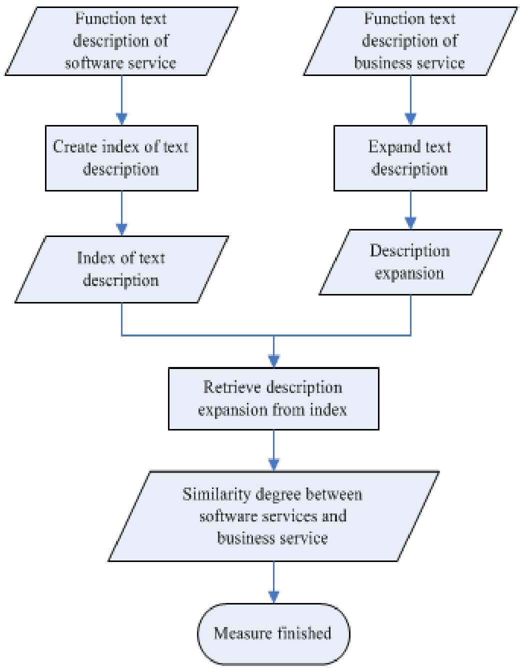
Function text description of software service
Function text description of business service
Create index of text description
Expand text description
Index of text description
Description expansion
Retrieve description expansion from index
Similarity degree between software services and business service
Measure finished

Figure 4. Process of Text Similarity Measuring

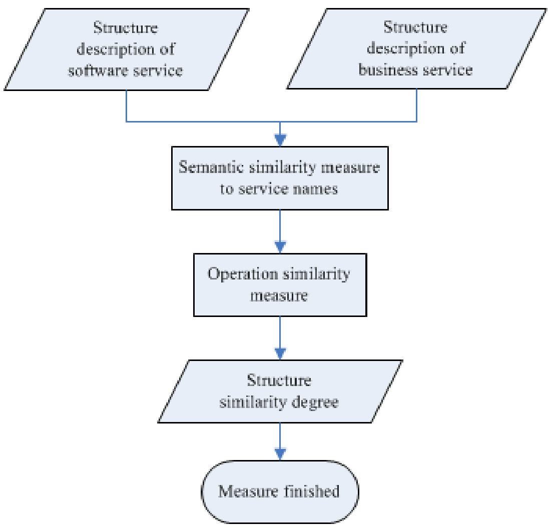
Structure description of software service
Structure description of business service
Semantic similarity measure to service names
Operation similarity measure
Structure similarity degree
Measure finished

Figure 5. Process of Structure Similarity Measurement

```
<reusablecomponent id="1">
     <document>test startup and result collection</document>
     <operation name="run">
       <input>junit.framework.TestResult:result</input>
       <output>void</output>
     </operation>
     <operation name="countTestcases">
       <output>int</output>
     </operation>
     <operation name="runBare">
       <output>void</output>
     </operation>
     <operation name="runTest">
       <input>Test:test</input>
       <input>TestResult:result</input>
       <output>void</output>
     </operation>
     ...
</reusablecomponent>
```

Figure 6. Description File of Software Services in JUnit

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.imu.edu.cn/wstest"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://imu.edu.cn/wstest"
name="Task">
<documentation>
a web service is defined to start unit test and collect the result of test.
</documentation>
<types>
<xsd:schema>
......
<operation name="startTest" parameterOrder="test">
<input message="tns:test"></input>
<output message="tns:startTestResponse"></output>
</operation>
<operation name="endTest" parameterOrder="test">
<input message="tns:test"></input>
<output message="tns:endTestResponse"></output>
</operation>
......
<service name="unittest">
<port name="TaskPort" binding="tns:unittestPortBinding">
<soap:address location="http://localhost:8080/unittest"></soap:address>
</port>
</service>
</definitions>
```

Figure 7. Description File of A Business Service

TABLE I. MATCHING RESULT BETWEEN BUSINESS SERVICE AND SOFTWARE SERVICES

Software services	**Text similarity degree**	**Structure similarity degree**	**Overall similarity degree**
The first software service	0.69	0.41	1.1
The second software service	0.11	0.56	0.67
The third software service	0.05	0.50	0.55
The 4th software service	0.05	0.11	0.16
The 5th software service	0.05	0.18	0.23
The 6th software service	0.05	0.23	0.28

TABLE II. MATCHING RESULT BETWEEN BUSINESS SERVICE AND SOFTWARE SERVICES AFTER THE FIRST ADJUSTMENT

Software services	**Text similarity degree**	**Structure similarity degree**	**Overall similarity degree**
Combination of the first and the 3th software service	0.68	0.52	1.2
The second software service	0.12	0.56	0.68
The 4th software service	0.07	0.11	0.18
The 5th software service	0.07	0.18	0.25
The 6th software service	0.07	0.23	0.30

TABLE III. MATCHING RESULT BETWEEN BUSINESS SERVICE AND SOFTWARE SERVICES AFTER THE SECOND ADJUSTMENT

Software services	**Text similarity degree**	**Structure similarity degree**	**Overall similarity degree**
Combination of the first, the second and the 3th software service	0.64	0.49	1.13
The 4th software service	0.12	0.11	0.24
The 5th software service	0.12	0.18	0.30
The 6th software service	0.12	0.23	0.35

REFERENCES

[1] Cai S, Liu Y, Wang X. A survey of service identification strategies [C]. In: Proceedings of 2011 IEEE Asia-Pacific Services Computing Conference (APSCC), IEEE, 2011, pp.464-470.

[2] Gašević D, Hatala M. Model-driven engineering of Service- Oriented systems [J]. International Journal of Service Science, Management, Engineering, and Technology, 2010, 1(1): 17-32.

[3] De Castro V, Marcos E, Vara J M. Applying CIM-to-PIM model transformations for the service-oriented development of information systems[J]. Information and Software Technology, 2011, 53(1): 87-105.

[4] Chengjun W. Applying pattern oriented software engineering to web service development [C]. In: Proceeding of International Seminar on Future Information Technology and Management Engineering (FITME'08), IEEE, 2008, pp. 214-217.

[5] Elgedawy I. Reusable SOA assets identification using e-business patterns [C]. In: Proceeding of 2009 World Conference on Services-II, IEEE, 2009, pp. 33-40.

[6] Tian, C., Cao, R. Z., Ding, W., Zhang, H., & Lee, J. Business Value Analysis of IT services[C], In: Proceeding of IEEE International Conference on Services Computing (SCC 2007), 2007, pp. 308-315.

[7] Chen, F., Zhang, Z., Li, J., Kang, J., & Yang, H. Service identification via ontology mapping[C]. In: Proceedings of 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09), IEEE, 2009, vol. 1, pp. 486-491.

[8] Zhang Z, Yang H, Chu W C. Extracting reusable object-oriented legacy code segments with combined formal concept analysis and slicing techniques for service integration [C]. In: Proceedings of 6th Int ernational Conference on Quality Software (QSIC 2006), IEEE, 2006, pp. 385-392.

[9] Sindhgatta R, Ponnalagu K. Locating components realizing services in existing systems[C]. In: Proceedings of 2008 IEEE International Conference on Services Computing (SCC'08), IEEE, 2008, vol. 1, pp. 127-134.

[10] Khadka R. Service Identification Strategies in Legacy-to-SOA Migration[C]. In: Proceedings of the 27th IEEE International Conference on Software Maintenance-Doctoral Consortium (ICSM-DC 2011), 2011.

[11] Zhang Z, Yang H. Incubating services in legacy systems for architectural migration[C]. In: Proceedings of 11th Asia-Pacific Software Engineering Conference, IEEE, 2004, pp. 196-203.

[12] Aversano L, Cerulo L, Palumbo C. Mining candidate web services from legacy code[C]. In: Proceedings of 10th International Symposium on Web Site Evolution (WSE 2008), IEEE, 2008, pp. 37-40.

[13] Kokash N. A comparison of web service interface similarity measures[J]. Frontiers in Artificial Intelligence and Applications, 2006, 142: 220-231.

[14] Stroulia E, Wang Y. Structural and semantic matching for assessing web-service similarity[J]. International Journal of Cooperative Information Systems, 2005, 14(04): 407-437.

[15] Plebani, Pierluigi, and Barbara Pernici. URBE: Web service retrieval based on similarity evaluation. Knowledge and Data Engineering, IEEE Transactions on 21.11 (2009): 1629-1642.

[16] Tibermacine, Okba, Chouki Tibermacine, and Foudil Cherif. WSSim: a Tool for the Measurement of Web Service Interface Similarity. Proceedings of the french-speaking Conference on Software Architectures (CAL). 2013.

[17] http://lucene.apache.org/

[18] http://code.google.com/p/ws4j/

[19] http://junit.org/

[20] Modica G D, Tomarchio O. A semantic framework to support resource discovery in future cloud markets[J]. International Journal of Computational Science & Engineering, 2015, 10(1/2):1-14.

[21] Ulizia A, Ferri F, Grifoni P. A similarity assessment method for discovering and adapting business services[J]. International Journal of Computational Science & Engineering, 2010, 5(2):97-109.

[22] Surianarayanan C, Ganapathy G. Towards quicker discovery and selection of web services considering required degree of match through indexing and decomposition of non-functional constraints[J]. International Journal of Computational Science & Engineering, 2015, 10(1/2).