International Journal of u- and e- Service, Science and Technology
Vol.9, No. 9 (2016), pp.221-242
http://dx.doi.org/10.14257/ijunesst.2016.9.9.22

Matching Algorithm Based on Semantic Similarity of Service Requirement Ontology for CAE Simulation in Cloud Platform

Ziyun Deng[1, 2], Jing Zhang[1*],Lijun Cai[2, 3],Lei Chen[1, 2] and Tingqin He[2, 3]

[1]*Electrical and Information Engineering,Hunan University,Changsha,Hunan, 410082,China*
[2]*National Supercomputing Centers in Changsha, Changsha,Hunan, 410082, China*
[3]*Information Science and Engineering,Hunan University,Changsha,Hunan, 410082,China*
**zhangj@hnu.edu.cn*

Abstract

When Cloud Platform for Computer Aided Engineering Simulation (CPCAES) are developed to use Service-Oriented Architecture (SOA), on the one hand, it's necessary to accurately express the service requirements of users, on the other hand it's need to semantic of the existing services in the platform, and then to match the ontologies. The existing Ontology Web Language for Services / Universal Description Discovery and Integration (OWL-S / UDDI) algorithm has weak matching precision, and not meet the Computer Aided Engineering (CAE) simulation applications. The other semantic similarity algorithms also have weakness in judgment factor, or do not meet the requirements of CAE simulation applications for matching to service requirement ontology in cloud platform. Such that, the authors present a kind of ontology of service requirement for CAE simulation, model the content of the ontology, including resource requirements, computing requirements, computing job requirements, input, output, etc. The authors give the ontology mapping relationship between Ontology Web Language for Services (OWL-S) and the ontology, give the matching decision rules for the ontologies, propose a matching algorithm for matching the ontologies, and compare with the classic OWL-S / UDDI matching algorithm and a similarity matching algorithm proposed in other paper in the ability for measure the similarity quantity of services, the ability to suit the applications, as well as algorithms recall rate and precision rate. The results show the matching algorithm proposed in this paper is more suitable for service requirement ontology of CPCAES, can use the quantify value for the similarity analysis to the ontologies, and has higher precision rate and higher recall rate, which can reach about 90%. The research work in this paper is used in the second prototype of CPCAES, and the research team is developing the third prototype based on semantic Web Services and SOA framework.

Keywords: *Cloud Platform for CAE Simulation, service requirement ontology, matching decision rule, semantic similarity, service matching algorithm*

1. Introduction

After supercomputer had been developed, related software applications become a hot topic of engineering technology [1-2]. Because of the traditional Linux operation command way requires the user to know the supercomputer hardware structure, operating systems, job scheduling software, it's difficult to use and have unfriendly interface, so the

[*]Corresponding Author

application users want to use the computing resources of supercomputer for parallel computing in a simple Web-friendly way, avoiding directly to the command processing operations.

Cloud service is an ideal choice for CAE users to access the supercomputing services. Many supercomputing centers and computing center has a High-Performance Computing (HPC) service capabilities are seeking ways to build user-oriented cloud services platform, and cloud platform based on SOA architecture is a viable way [3]. The Web Services in SOA architecture includes WSDL description, and composed using Business Process Execution Language (BPEL), Unified Modeling Language (UML) and other process definition technologies, such that the architecture can be statically defined services and business processes, but cannot dynamically support services and service flow generation, is not conducive for service to be automatic discovery, matching, composition, process generation, performance monitoring, the semantic technology for Web services can solve the problem [4-5].

In order to provide a good user experience service for CAE users, the National Center of Supercomputing center in Changsha designs and implements CPCAES based on SOA architecture. In this paper, the authors discuss the following ideas.

(1)First, the authors propose a service requirement ontology from the perspective of user requirement described, and give comparative analysis and mapping between the ontology and OWL-S ontology.

(2)Next, the authors give the matching decision rules for matching service requirement and existing service in CPCAES.

(3)And then, the authors design a measure approach used to quantify the similarity matching degree, give the similarity calculation algorithm, and then analysis algorithm through compare the qualitative and quantitatively with other algorithms.

(4)Finally, the authors introduce the application about the algorithm in CPCAES, and then the authors propose their work in route location and research focus on the technical method.

1. Related Research

Many researchers and engineers are developing CPCAES based on SOA architecture and workflow technology [3] and [8-15]. Literature [3] proposed a series of architecture for cloud simulation platform, pointed out the seven key technologies of the platform, which had resource management techniques based on Web Services / grid and semantic resource discovery technologies for simulation model and developed COSIM-CSP prototype of cloud simulation platform. Literatures [8-15] also proposed the model of the cloud manufacturing platforms and designed the prototype systems. Literature [12] used a unified ontology to enhance semantic interoperability in the whole process of cloud services, used cloud management engine to manage all user-defined cloud. However, CAE simulation process is complex, and require integrated many software interfaces, it is difficult to deal with all cases with a unified ontology, and the functions in all above cloud platform prototypes are also very limited.

The matching technology between requirements of cloud simulation users and existing services is the key issue need to be solving so as to build semantic Web Services in CPCAES, there are some scholars and engineers have carried out research work in this area [3] and [8-15]. Literature [3] described the requirements for simulation problems on the portal, and build dynamic simulation system through the cloud simulation platform, and the functions of the platform including discovery, scheduling, various compositions for environmental simulation models and software service resources. The authors proposed a matching algorithm to model based on ontology reasoning. Literature [8] studied the automatic composition of semantic Web services technology, but didn’t discuss service matching, semantic ontology description for Web services in the platform.

Literature [15] proposed the cloud model of the manufacturing platform. In semantic layer, [15] used extensible Markup Language / Resource Description Framework / Ontology Web Language for Services (XML / RDF / OWL-S) to describe the ontology concerned with manufacturing operations, but did not implements the platform and did not discuss the semantic Web services ontology.

Matching algorithm and engineering technology and automatic composition of semantic Web services, mainly includes OWL-S [16], Web Services Modeling Ontology (WSMO) [17], Adding Semantics to WSDL (WSDL-S) [18] , all the three kinds are distinctive [19].the semantic description of OWL-S to services is bidirectional to bidirectional matching. The semantic description of required services is not exist in WSDL-S, so that matching results is relatively low degree of versatility. WSMO introduced Meditor to solve the problem of heterogeneous elements, but it also causes WSMO very complex, inconvenient project realization. Therefore, it is more based on OWL-S to improve its relevant theories and engineering research [19-25]. Literature [19] proposed an adaptive semantic matching approach for Web services to improve the precision and recall for service discovery, in this method, the requirements document and OWL-S service ontology are converted to outline the ontology tree by similarity classification of similar properties and grading methods to structural calculation of the corresponding tree node, the matching algorithm is proposed semantic Web services and achieve OWLS-CPS prototype system. Literature [21] developed a semantic-based Web services composition system OntoPipeliner to meet the needs of users using the way of the wizard, interactive semi-automatic composition of Web services. Literature [22] described the first one to use a hybrid semantic Web technologies matcher OWLS-MX, which in the original logic-based OWL-S service matching technology, if the match fails to use a token-based syntax similar to the degree of matching methods, by introducing a non-logic-based information retrieval technology to enhance the matching ability of matcher.

Put the context description into service ontology description, and use the semantic similarity to measure matching degree, these may help to enhance the ability of the matching service. Literature [23] proposed a design method based on semantic context representation model that extends OWL-S joined the context of the conditions and adapt to the rules by constantly updated context information, which can automatically discover and compose services. Literature [24] put the context describe that influence HPC into the ontology description of service requirements, and the context of HPC service discovery is divided into five categories including user context, work context, data context, service quality context and host context. Literature [25] systematically studied the calculation method for semantic similarity, and proposed a matching algorithm WSMatch for semantic Web services, the algorithm calculate the matching degree between two Web services based on Input / Output / Precondition / Effect (IOPE) , the basic idea is to calculate the input parameters, output parameters, the effects of pre-conditions and post-effects between two services through a linear composition of them, but did not carry out research work bonding applications.

In summary, there are have certain research about the manufacture cloud platform based on semantic services, but has yet to meet the HPC users about CAE simulation applications related research and literature. In addition, the service matching algorithms already have some basic research. In view of this, the authors propose the ontology using to describe the user requirement for CAE simulation applications. As one of the key technologies of CPCAES, the matching algorithm is improved based on some basic matching algorithm what has been providing by other literature [20-28].

2. Ontology Modeling for Service Requirement

At present, the development work of ontology modeling about the CAE simulation requirements has not been found yet. The authors put forward the ontology model, give the corresponding relation between OWL-S and the ontology in CPCAES, and perform the example analysis.

2.1. Service Requirement Ontology

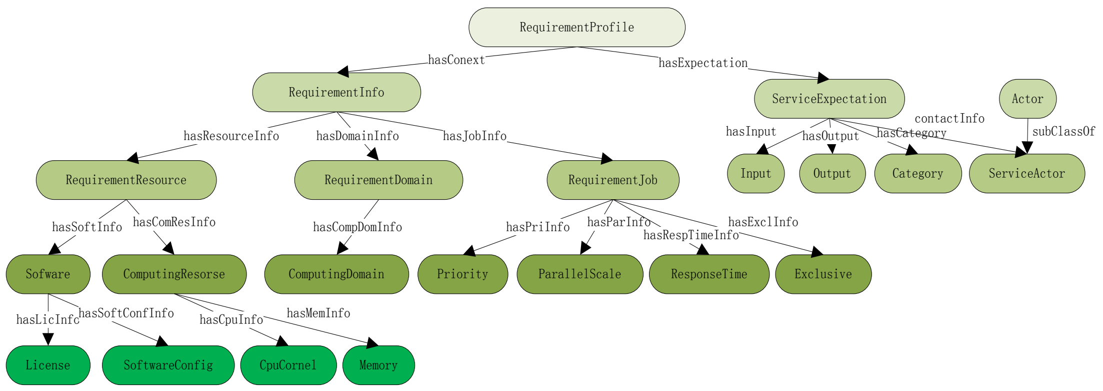


Figure 1. Requirement Ontology for CAE Simulation in CPCAES

To design the requirement ontology of the CAE simulation in CPCAES, the following four factors should be taken into consideration: (1) learn from the existing research results; (2) try to be close to the OWL-S as far as possible for mapping when matching with the ontology; (3) be helpful for the implementation of engineering design; (4) take the input data of the user interaction into consideration and excavate the implicit requirements of user by programming.

According to the above factors, in this paper, the authors design a kind of service requirement ontology. In the ontology, it is extended from ServiceProfile in the OWL-S to RequirementProfile, as shown in Figure 1.

RequirementProfile includes two categories, namely ServiceExpectation and RequirementInfo. In the OWL-S, the ServiceExpectation is the other content of the ServiceProfile which remove Precondtions, Effects and Service Quality. RequirementInfo is the relevant information about calculation request of user. The specific content for RequirementProfile is shown as follows:

(1)ServiceExpectation

ServiceExpectation is an expectation for the requirement service of user, including basic information, functional properties and nonfunctional information. In order to keep consistent with OWL-S for matching, expectations have four subclasses, including Input, Output, Category and ServiceActor. ServiceActor is a subclass of Actor. Compared with OWL-S, the ServiceExpectation has no Precondtions and Effects, because Precondtions describe the conditions satified before the service is executed, Precondtions is more intuitive keep it in user requirement information. Effects describe the impact of the service execution, and it is no need to logic in requirment service expectations of users.

There is no description about the Quality of Service (QoS) in ServiceExpectation because these contents are built in the RequirementInfo class.

(2) RequirementInfo

The requirement informations of users include requirement informations for computing resource (RequirementResource), informations for computing fields (RequirementDomain) and requirement informations for computing job (RequirementJob).

In term of the parallel computing of the HPC CAE simulation, each calculation will be transformed into a parallel computing job in supercomputer or computer cluster. The requirements for computing resources for job should be put forward by users in advance. The computing resources include software, Central Process Unit (CPU), memory, etc. Software class shows which CAE software and its version user should use, and how much license resources user should need. Of course, it is a recessive requirements how many License may user need. Users may only put forward how many CPU cores they will use. Most commercial CAE software is based on parallelism (the number of CPU cores of parallel computing jobs at the same time) to control the scale of calculation and sell their software in the business. Therefore, in this case, it is need to program to automatically fill the requirement number of the license.

The requirements information for computing field refers to which fields for parallel computing tasks, inluding stress and strain analysis, impact analysis, harmonic analysis, large deformation analysis and electromechanical liquid joint simulation etc.

The requirements information for computing job includes Priority, ParallelScale, ResponseTime, and Exclusive. Priority refers to the job priority in the job scheduling system (such as SLURM, LSF) in supercomputer or computer cluster. The Priority is higher, and then the greater likelihood is to get computing resources. However, when users perform requirement submission, it may be a fuzzy representation, such as very high, higher, general and low. Through programming, it is converted to the clear numerical in job scheduling system. ParallelScale usually stands for how many computing nodes user need work together. When user submit a job, generally, the system only come up with how many CPU cores user need to do parallel computing. This is because the application domain users are not typically clear for specific hardware configuration situation of super computers or computer cluster, which can be obtained by program automatic conversion. For example, “Tianhe No.1” uses Intel CPU and the CPU number of each node is two-way 6 cores, namely a total of 12 CPU kernels. Then, it is converted into computing nodes needing to participate in according to the number of CPU used cores. ResponseTime refers to the maximum time range that user can accept for completing the computing task. Exclusive refers to the user computing jobs whether to allow other user job to preempt its computing resources.

2.2. Mapping Relationship Between Service Requirement Ontology and Existing OWL-S Service Ontology

The ServiceProfile of OWL-S is bidirectional matching. The service provider and consumer describe Web Services in the same described way. But requirements description for Web services of CAE simulation cannot do it, and it is necessary to appoint mapping relationship between service requirement ontology and the ServiceProfile of OWL–S. Finally, the system matches it using a certain algorithm.

As shown in Figure 2, from the design of service requirement ontology once upon a time, the system can determine the mapping relationship of RequirementProfile and ServiceProfile from the description information. ServiceActor in ServiceExpectation matches with Actor in ServiceProfile and they are both the description for contact information of service provider. The input and output of Service are matched by the corresponding item between Function Description in ServiceExpectation and ServiceProfile. RequirementResource and RequireDomain can be mapped to the Preconditions of Function Description in ServiceProfile because the corresponding portion of service requirement ontology actually describes the requirement conditions invoking a service. RequirementJob in the service requirement ontology point out job response requirements that Service need and its essence is the quality requirements. So it can be corresponding to the Service Quality in ServiceProfile.

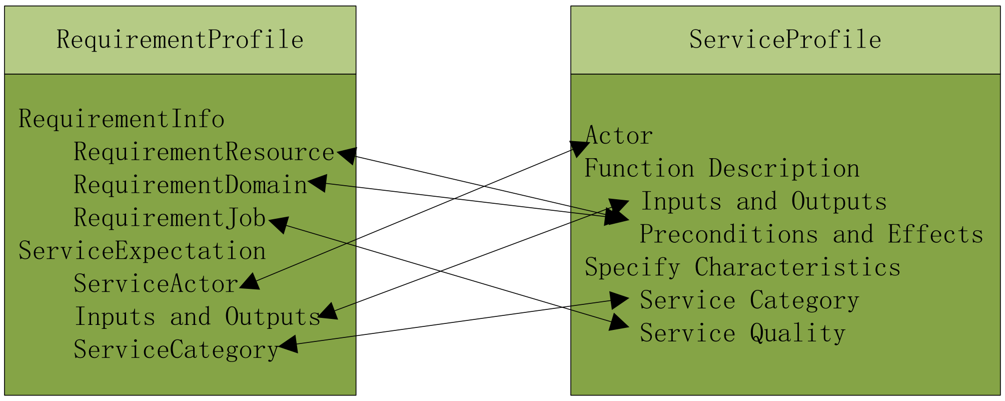


Figure 2. Mapping Relation between Service Requirement Ontology and Existing OWL–S Service Ontology

2.3. Matching Analysis for Service Requirements

The functions of service requirement information for service matching are represented in the following three aspects:

(1) Matching the premise. Service requirement information reflects the condition which is necessary for the computing requirements in terms of computing resources and computing fields, which match with the premise in the published service.

(2)Judge whether QoS meet requirements. Computing job requirements in the service requirement information actually reflects the requirements on the quality of calculation service, which match with the service quality information in the published service.

(3)Filtrate services. According to the two above requirements, service matcher needs to design is then composed with functional requirements matching results, which can filtrate services in accordance with user requirements.

Some researchers put forward the method of phase matching approach for implementing matcher [26-27] to reduce the service screening quantity. In order to improve the accuracy of service matching, and considering that the CAE simulation of oriented application field is relatively single, therefore in this paper, the authors do not use phases matching approach. Without loss of generality, the authors below illustrate the functions of service requirement information in the service matching. The authors assume have provided some computing services for CAE simulation as shown in table 1.

(1)Service A. It can provide the ANSYS calculation file set on super computer “Tianhe No.1” to carry out calculation. The user can get ANSYS calculation result file set and the mark of computing success or computing failure. ANSYS license provided by services can implement parallelism computation range from >=32 to <=64. The software version of ANSYS is 15.1, which can carry out mechanical statics parallel solving calculation asking for computational memory requirements within >=1G and <=30G. According to the historical conditions, response time is calculated as average service time, which is about 10 minutes. When the job is executing, it is exclusively using the node resources. The type of service is the 0 layer service to initial matching.

(2) Service B. It can provide the LS-DYNA calculation file set on super computer “Tianhe No.1” to carry out calculation. The user can get LS-DYNA calculation result file set and the mark of computing success or computing failure. LS-DYNA license provided by services can implement parallelism computation range from >=16 to <=32. The software version of LSDYNA is 9.7, which can carry out collision analysis parallel solving calculation asking for computational memory requirements within >=500M and <=10G. According to the historical conditions, response time is calculated as average service time, which is about 15 minutes. When the job is executing, it is exclusively using the node resources. The type of service is the 0 layer service to initial matching.

(3)Service C. It can provide the ANSYS calculation file set on super computer “Tianhe No.1” to carry out calculation. The user can get ANSYS calculation result file set and the mark of computing success or computing failure. ANSYS license provided by services can implement parallelism computation range from >=32 to <=64. The software version of ANSYS is 13.2, which can carry out heat conduction analysis parallel solving calculation asking for computational memory requirements within >=1G and <=20G. According to the historical conditions, response time is calculated as average service time, which is about 20 minutes. When the job is executing, it is exclusively using the node resources. The type of service is the 0 layer service to initial matching.

Table 1. Semantic Description for Provided Services

Service Name	Input	Output	Preconditions	QoS	Service type
Service A	Calculation file set: ANSYS calculation file set. Target cluster: Tianhe No.1	Result file set: ANSYS calculation result file set. Calculation success or fail: isSuccess	License: >=32 parallelism, <=64 parallelism. Software: ANSYS 15.1. Memory :> =1G, <=30G. Calculation field: machinery statics	Priority: high. Parallelism: > =32 cores, <=64 cores. Response time: about 10 minutes. Exclusive: monopolization	0 layer service to initial matching
Service B	Calculation file set: LS-DYNA Calculation file set. Target cluster:Tianhe No.1	Result file set: LS-DYNA calculation result file set. Calculation success or fail: isSuccess	License: >=16 parallelism, <=32 parallelism. Software: LS-DYNA 9.7. Memory: >=500M, <=10G. Calculation field: collision analysis	Priority: middle. Parallelism: >=16 cores, <=32 cores. Response time: about 10 minutes. exclusive: monopolization	0 layer service to initial matching
Service C	Calculation file set: ANSYS Calculation file set. Target cluster: Tianhe No.1	Result file set: ANSYS calculation result file set. Calculation success or fai:isSuccess	License: >=32 parallelism, <=64 parallelism. Software: ANSYS 13.2. Memory :> =1G, <=20G. Calculation field: heat transfer analysis	Priority: high. Parallelism :> =32 cores, <=64 cores. Response time: about 20 miniutes. Exclusive: monopolization	0 layer service to initial matching

Service requirements requestA and requestB are both shown in the Table 2. RequestA requests for parallel computing of ANSYS and Goal cluster is supercomputer "Tianhe No.1". From the perspective of the input and output matching of requestA, ServieA and ServiceC can meet the requirements, and it also needs to carry out the information matching for user requirement. According to the information matching for user requirement of the requestA, it need 48 parallelism Licenses and calculate by using ANSYS 15.1, consuming 12 G memory to carry out mechanical statics calculation. So ServiceA conforms only to the requirements. From the perspective of the job requirement description in the information for user requirement, ServiceA and ServiceC meet to requirements. In conclusion, service requirements of ServiceA and requestA match with each other. We can see that if the input and output match, the matching results for requirement information might be different when the same service aims at different service requirement.

When we match analysis from the input, output and requirement information in term of service requirement reqestB, only ServiceB meets to the requirements.

Table 2. The Semantic Description and Matching Results of Service Requirements

Service request	Input	Output	User requirement information	Matching result
requestA	Calculation file set:ANSYS Calculation file set, Target cluster:Tianhe 1	Result file set:ANSYS calculation result file set, Calculation success or fail:isSuccess	License:48 parallelism software:ANSYS 15.1 memory:12G calculation field:mechanical statics priority:high parallelism:48 cores response time:<=1 hours exclusive:monopolization	ServiceA
requestB	Calculation file set:LS-DYNA Calculation file set, Target cluster:Tianhe 1	Result file set:LS-DYNA calculation result file set, Calculation success or fail:isSuccess	License:24 parallelism software:LS-DYNA 9.7 memory:2G calculation field:collision analysis priority:middle parallelism:24cores response time:<=2 hours exclusive:monopolization	ServiceB

3. Service Matching Algorithm Based on Similarity

Firstly, it is necessary clear the decision rules to matching algorithm based on service requirements, and then give some definitions of terms to calculate the similarity. Finally, a specific algorithm is given in this paper.

3.1. Matching Decision Rules for Service Requirements

According to the analysis mentioned above, the description of service requirement ontology does not include the description of service results. It is no need to match with the service results. Accordingly, in the match between the service requirement ontology and the existing OWL-S service ontology, the input, the output, the premise condition and QoS are the four basic arguments to match.

The main relationship in the ontologies is the subclass relationship between concepts (subClassOf). The relationship is also known as inheritance relationship. If the concept A is the parent class of the concept B, the former includes the latter, which is denoted by $A \supset B$. The inclusive relationship between concepts can be passed. The ontology can be expressed as a classification tree based on the hierarchy relationships. As shown in Figure 2, it is the classification tree of the requirement ontology. A classification tree can be used to judge whether services are matching with each other.

Assume that Inputs, Outputs, RequirementResource+RequirementDomain and RequirementJob of the service requirement ontology are represented as $\mathrm{WS_r} = \{\mathrm{I_r}, \mathrm{O_r}, \mathrm{P_r}, \mathrm{Q_r}\}$, and Inputs, Output, Preconditions, QoS of the existing service ontology are expressed as $\mathrm{WS_a} = \{\mathrm{I_a}, \mathrm{O_a}, \mathrm{P_a}, \mathrm{Q_a}\}$. The decision matching rule is as follow:

(1) The inputs of the service requirement ontology include all the inputs in the existing service ontology, which represents the input matching, denoted by $I_r \supseteq I_a$。

(2) The output of the existing service ontology include all the outputs in the service requirement ontology, which represents the output matching, denoted by $O_r \subseteq O_a$.

(3) The RequirementResource+RequirementDomain of the service requirement ontology include the Preconditions in the existing service ontology, which represents precondition matching, denoted by $P_r \supseteq P_a$.

(4) The QoS of the existing service ontology include all the job requirements in the service requirement ontology, which represents QoS matching, denoted by $Q_r \subseteq Q_a$.

3.2. Definition of Semantic Distance and Definition of Semantic Similarity

Here $\mathrm{distance}(A, B)$ expresses the semantic distance metric between concept A and concept B , and $\mathrm{sim}(A, B)$ expresses the matching degree metric by the semantic similarity matching degree between concept A and concept B.

Definition 1[20]: semantic distance and semantic similarity between two nodes without inclusion relation. We assume v_i and v_j what are two nodes in ontology classification tree, if v_i does not include v_j, denoted as $v_i \not\subset v_j$, then $\mathrm{distance}(v_i, v_j) = \infty$, $\mathrm{sim}(v_i, v_j) = 0$.

This definition suggests the similarity of two nodes without inclusion relation is 0, and the two nodes are not matched.

Definition 2[20]: semantic distance and semantic similarity between two same nodes. We assume v_i and v_j what are two nodes in ontology classification tree, if v_i and v_j are same concept, denoted as $v_i = v_j$, then $\mathrm{distance}(v_i, v_j) = 0$, $\mathrm{sim}(v_i, v_j) = 1$.

This definition suggests that the similarity of two nodes with same concept is 1, and the two nodes are perfect matched.

Definition 3[20]: semantic distance and semantic similarity between two nodes in adjacent layer with inclusion relation. We assume v_l and v_{l-1} are located respectively at l layer and $l - 1$ layer in ontology classification tree, then,

$$\mathrm{distance}(v_l, v_{l-1}) = \frac{1}{m} + \frac{m^2-1}{m^2}\left(\frac{m+1}{m}\right)^{-l} \qquad l \geq 1, m \geq 2$$

Here m is layer total number of ontology classification tree, and root node is the zeroslayer of the tree, When the tree only has a root node, $l = 0, m = 1$.

Thus, we can arbitrarily the distance of any two nodes (v_i,v_j) with inclusion relation in ontology classification tree is shown as follows,

$$\mathrm{distance}(v_i, v_j) = \mathrm{distance}(v_{l+1}, v_l) + \mathrm{distance}(v_{l+2}, v_{l+1}) + \cdots + \mathrm{distance}(v_k, v_{k-1})$$

Among them, l and k are respectively the layer number, and $k \geq l \geq 1, m \geq 2$.

Definition 4[20]: semantic similarity between any two nodes. We assume v_i and v_j what are two nodes in ontology classification tree, then,

$$\mathrm{sim}(v_i, v_j) = \begin{cases} 1 & v_j = v_i \\ \frac{1}{2} + \frac{1}{2} \times \frac{1-\mathrm{distance}(v_i,v_j)}{m-1} & v_j \subset v_i, v_j \neq v_i, j > i \geq 1, m \geq 2 \\ \frac{1}{2} \times \frac{1-\mathrm{distance}(v_i,v_j)}{m-1} & v_j \subset v_i, v_j \neq v_i, j > i \geq 1, m \geq 2 \\ 0 \quad v_j \not\subset v_i, v_i \not\subset v_j & \end{cases}$$

3.3. Matching Degree between Service Requirement Ontology and Existing Service Ontology

Definition 5: we assume service requirement ontology is $WS_r = \{I_r, O_r, P_r, Q_r\}$, and existing service ontology is $WS_a = \{I_a, O_a, P_a, Q_a\}$, then we can get semantic matching degree between service requirement ontology and existing service ontology,

$$\mathrm{degreeOfInputMatch}(I_a, I_r) = \mathrm{sim}(I_a, I_r)$$
$$\mathrm{degreeOfOutputMatch}(O_r, O_a) = \mathrm{sim}(O_r, O_a)$$
$$\mathrm{degreeOfPreconditionMatch}(P_a, P_r) = \mathrm{sim}(P_r, P_a)$$
$$\mathrm{degreeOfQoSMatch}(Q_r, Q_a) = \mathrm{sim}(Q_a, Q_r)$$

Then,

$$\mathrm{degreeOfWebServiceMatch}(WS_r, WS_a) = k_1\mathrm{sim}(O_r, O_a) + k_2\mathrm{sim}(I_a, I_r) + k_3\mathrm{sim}(P_r, P_a) + k_4\mathrm{sim}(Q_a, Q_r)$$

Among them, $\sum_{i=1}^{4} k_i = 1$.

3.4. Matching Algorithm Between Service Requirement Ontology and Existing Service Ontology

The input, output parameters of a service may have more than one, and we need to apply definition 4 and definition 5 to calculate the total similarity on multiple parameters.

We assume the inputs of service requirement ontology are $R(ri_1, ri_2, \ldots, ri_n)$, the outputs of service requirement ontology are $R(ro_1, ro_2, \ldots, ro_m)$, the inputs of existing service ontology are $A(ai_1, ai_2, \ldots, ai_u)$, the outputs of existing service ontology are $A(ao_1, ao_2, \ldots, ao_v)$. The similarity calculation algorithm to outputs is shown as algorithm 1.

Algorithm 1: $\mathrm{degreeOfOutputMatch}(R(ro_1, ro_2, \ldots, ro_m), A(ao_1, ao_2, \ldots, ao_v))$

```
1 define $\mathrm{simOut}[][] = \mathrm{float}[m][v]$ // Define the similarity matrix
  // Fill in the values in the similarity matrix
2 $\mathrm{for}(\mathrm{int}\ i = 1; i \le m; i++)$
3     $\mathrm{for}(\mathrm{int}\ j = 1; j \le v; i++)$
4         $\mathrm{simOut}[i][j] = \mathrm{sim}(ro_i, ao_j)$
5     end for
6 end for
7 define $\mathrm{simSelected}[] = \mathrm{float}[m]$ // Define the output similarity set
  //Define the output weight coefficient set, and set the initial values
8 define $\mathrm{simRatio}[] = \mathrm{float}[m]\{t_1, t_2, \ldots, t_m\}, \sum_{i=1}^{m} t_i = 1$
9 $\mathrm{for}(\mathrm{int}\ i = 1; i \le m; i++)$
  // Find the maximum similarity of the current output
10    $\mathrm{simSelected}[i] = \mathrm{simRatio}[i] \times$
                  $\max\{\mathrm{sim}(ro_i, ao_1), \mathrm{sim}(ro_i, ao_2), \ldots, \mathrm{sim}(ro_i, ao_v)\}$
11 end for
12 $\mathrm{sim} = \sum_{i=1}^{m} \mathrm{simSelected}[i]$
13 return sim
```

In algorithm 1, first, we use a double circulation filling similarity values $\mathrm{sim}(ro_i, ao_j)$ in each position in similarity matrix, and the similarity value is calculated according to definition 4. The next, as a benchmark what the number of output parameters of service requirement ontology, the maximum value of all output similarities is calculated between each output of requirement service ontology and all outputs of existing service ontology, and the sum of all weighted maximum is calculated. The weights of the output parameters reflect the user attention to these parameters. By default, the weight value is average value $1/\mathrm{m}$.

Similarly, we can get the similarity calculation algorithm to inputs. The algorithm is shown as algorithm 2.

```
**Algorithm** 2:$\mathrm{degreeOfInputMatch}(\mathrm{A}(ai_1, ai_2, \ldots, ai_u), \mathrm{R}(ri_1, ri_2, \ldots, ri_n))$
1 define $\mathrm{simIn}[][] = \mathrm{new\ float}[u][n]$ // Define the similarity matrix
  // Fill in the values in the similarity matrix
2 $\mathrm{for}(\mathrm{int}\ i = 1; i \leq u; i++)$
3    $\mathrm{for}(\mathrm{int}\ j = 1; j \leq n; i++)$
4       $\mathrm{simIn}[i][j] = \mathrm{sim}(ai_i, ri_j)$
5    end for
6 end for
7 define $\mathrm{simSelected}[] = \mathrm{float}[u]$ // Define the output similarity set
  //Define the input weight coefficient set, and set the initial values
8 define $\mathrm{simRatio}[] = \mathrm{float}[u]\{t_1, t_2, \ldots, t_u\}, \sum_{i=1}^{u} t_i = 1$
9 $\mathrm{for}(\mathrm{int}\ i = 1; i \leq u; i++)$
  // Find the maximum similarity of the current input
10   $\mathrm{simSelected}[i] = \mathrm{simRatio}[i] \times$
            $\max\{\mathrm{sim}(ro_i, ao_1), \mathrm{sim}(ro_i, ao_2), \ldots, \mathrm{sim}(ro_i, ao_n)\}$
11 end for
12 $\mathrm{sim} = \sum_{i=1}^{u} \mathrm{simSelected}[i]$
13 return sim
```

In algorithm 2, first, we use a double circulation filling similarity values $\mathrm{sim}(ai_i, ri_j)$ in each position in similarity matrix, and the similarity value is calculated according to definition 4. The next, as a benchmark what the number of input parameters of existing service ontology, the maximum value of all input similarities is calculated between each input of existing service ontology and all inputs of requirement service ontology, and the sum of all weighted maximum is calculated. The weights of the input parameters reflect the user attention to these parameters. By default, the weight value is average value $1/\mathrm{m}$.

The similarity of Preconditions and the similarity of QoS can be determined by the logical expression. Here we first discuss how to determine the similarity of Precondition. According to the rules mentioned above, one of the rules is $P_r \supseteq P_a$. In service requirements ontology, We assume RequirementResource and RequirementDomain together are $\mathrm{R}(rp_1, rp_2, \ldots, rp_x)$, RequirementJob is $\mathrm{R}(rq_1, rq_2, \ldots, rq_y)$. In existing service ontology, Precondtions is $\mathrm{A}(ap_1, ap_2, \ldots, ap_z)$, QoS is $\mathrm{A}(aq_1, aq_2, \ldots, aq_w)$. Then we can get the similarity algorithm to Preconditions. The algorithm is shown as algorithm 3.

Algorithm 3:$\mathrm{degreeOfPreconditionMatch}(R(rp_1, rp_2, \ldots, rp_x), A(ap_1, ap_2, \ldots, ap_z))$

```
1 define $simPrecondtion[] = int[x]\{0, \ldots, 0\}$ // Define the similarity matrix
  // Fill in the values in the similarity matrix
2 $for(int\ i = 1; i \le x; i++)$
3    $for(int\ j = 1; j \le z; i++)$
4       $if(isMapping(rp_i, ap_j)\ and\ isSatisfy(rp_i))$
5          $simPrecondtion[i] = 1$
6    end for
7 end for
  //Define the Precondition weight coefficient set, and set the initial values
8 define $simRatio[] = float[x]\{t_1, t_2, \ldots, t_x\}, \sum_{i=1}^{x} t_i = 1$
9 $sim = \sum_{i=1}^{x}(simRatio[i] \times simSelected[i])$
10 return sim
```

The description of Preconditions is either logical expression or relation expression. We assume if one of Preconditions is satisfied, then the similarity of the Precondition is 1. If one of Preconditions is not satisfied, then the similarity of Precondition is 0. In Algorithm 3, we first define the initial values in the similarity array of Preconditions are all 0, and then judge in a double circulation. If a RequirementResource or a RequirementDomain is mapped to a Precondition in current existing service ontology, and the RequirementResource or the RequirementDomain is satisfied, then the similarity of current Precondition is 1. The next, as a benchmark what the number of parameters of RequirementResource and RequirementDomain in the service requirement ontology, and the sum of all weighted maximum is calculated. The weights of the Preconditions parameters reflect the user attention to these parameters. By default, the weight value is average value $1/x$.

Similarly, we can get the similarity calculation algorithm to QoS. The algorithm is shown as algorithm 4.

Algorithm 4:$\mathrm{degreeOfQoSMatch}(A(aq_1, aq_2, \ldots, aq_w), R(rq_1, rq_2, \ldots, rq_y))$

```
1 define $simQoS[] = int[w]\{0, \ldots, 0\}$ // Define the similarity matrix
  // Fill in the values in the similarity matrix
2 $for(int\ i = 1; i \le w; i++)$
3    $for(int\ j = 1; j \le y; i++)$
4       $if(isMapping(rq_i, aq_j)\ and\ isSatisfy(rq_i))$
5          $simQoS[i] = 1$
6    end for
7 end for
  //Define the QoS weight coefficient set, and set the initial values
8 define $simRatio[] = float[w]\{t_1, t_2, \ldots, t_w\}, \sum_{i=1}^{w} t_i = 1$
9 $sim = \sum_{i=1}^{w}(simRatio[i] \times simSelected[i])$
10 return sim
```

The description of QoS is either logical expression or relation expression. We assume if one of QoS is satisfied, then the similarity of QoS is 1. If one of QoS is not satisfied, then the similarity of QoS is 0. In Algorithm 4, we first define the initial values in the similarity array of QoS are all 0, and then judge in a double circulation. If a RequirementJob is mapped to a Precondition in current existing service ontology, and the RequirementJob is satisfied, then the similarity of current QoS is 1. The next, as a benchmark what the number of parameters of RequirementJob in the existing service ontology, and the sum of all weighted maximum is calculated. The weights of the QoS parameters reflect the user attention to these parameters. By default, the weight value is average value 1/w.

3.5. Matching Calculation Example

We describe the existing service ontology in Table 1, and requirement service ontology in Table 2 to as a matching calculation example. We assume the computing file set in Inputs and the computing file set in Outputs use the ontology as shown in Figure 3. The target cluster in Inputs uses the ontology as shown in Figure 4. The isSuccess in Outputs paremeters uses data type Boolean in XML Schema.

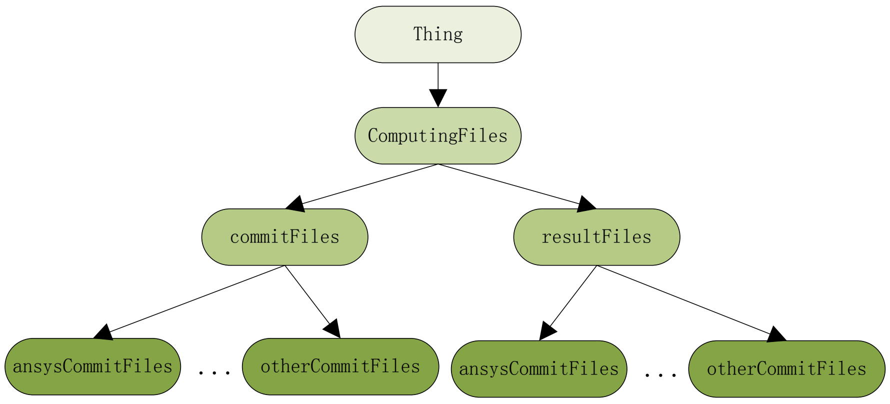


Figure 3. Ontology of Computing File Set

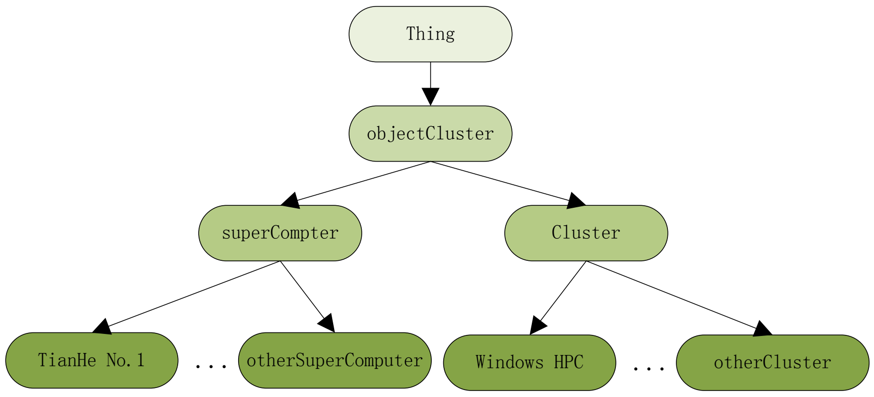


Figure 4. Ontology of Target Cluster

We calculate similarity using requestA in Table 2, and two services what is ServiceA and ServiceB in Table 1.

(1)The matching degree of requestA and ServiceA is calculated as shown below.

$$\text{sim}(\text{serviceA's input reference 1}, \text{requestA's input reference 1}) = \text{sim}(\text{ansysCommitFiles}, \text{ansyscommitFiles}) = 1$$

$$\text{sim}(\text{serviceA's input reference 1}, \text{requestA's input reference 2}) = \text{sim}(\text{ansysCommitFiles}, \text{TianHe No. 1}) = 0$$

$$\text{sim}(\text{serviceA's input reference 2}, \text{requestA's input reference 1}) = \text{sim}(\text{TianHe No. 1}, \text{ansysCommitFiles}) = 0$$

$$\text{sim}(\text{serviceA's input reference 2}, \text{requestA's input reference 2}) = \text{sim}(\text{TianHe No. 1}, \text{TianHe No. 1}) = 1$$

According to the above calculation, we can get the the similarity matrix of inputs,

$$\text{InputMatchMatrix}(\text{serviceA's input}, \text{requestA's input}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Accordingly,

$\text{degreeOfInputMatch}(\text{serviceA}, \text{requestA}) = \frac{1}{2} \times \max\langle 1,0 \rangle + \frac{1}{2} \times \max\langle 0,1 \rangle = 1$

Similarly,

$\text{degreeOfOutputMatch}(\text{requestA}, \text{serviceA}) = 1$

$\text{degreeOfPreConditionMatch}(\text{serviceA}, \text{requestA}) = 1$

$\text{degreeOfQoSMatch}(\text{requestA}, \text{serviceA}) = 1$

$$\begin{aligned}&\text{degreeOfWebServiceMatch}(\text{WS}_{\text{ra}}, \text{WS}_{\text{sa}}) = \\ &\frac{1}{4} \times \text{degreeOfInputMatch}(\text{serviceA}, \text{requestA}) + \\ &\frac{1}{4} \times \text{degreeOfPreConditionMatch}(\text{serviceA}, \text{requestA}) + \\ &\frac{1}{4} \times \text{degreeOfPreConditionMatch}(\text{serviceA}, \text{requestA}) + \\ &\frac{1}{4} \times \text{degreeOfQoSMatch}(\text{requestA}, \text{serviceA}) \\ &= 1\end{aligned}$$

(2) The matching degree of requestA and ServiceB is calculated as shown below.

$$\begin{aligned}&\text{sim}(\text{serviceB's input reference 1}, \text{requestA's input reference 1}) \\ &\quad = \text{sim}(\text{lsdynaCommitFiles}, \text{ansyscommitFiles}) = 0 \\ &\text{sim}(\text{serviceB's input reference 1}, \text{requestA's input reference 2}) \\ &\quad = \text{sim}(\text{lsdynaCommitFiles}, \text{TianHe No. 1}) = 0 \\ &\text{sim}(\text{serviceB's input reference 2}, \text{requestA's input reference 1}) \\ &\quad = \text{sim}(\text{TianHe No. 1}, \text{lsdynaCommitFiles}) = 0 \\ &\text{sim}(\text{serviceB's input reference 2}, \text{requestA's input reference 2}) \\ &\quad = \text{sim}(\text{TianHe No. 1}, \text{TianHe No. 1}) = 1\end{aligned}$$

According to the above calculation, we can get the the similarity matrix of inputs,

$$\text{InputMatchMatrix}(\text{serviceB's input}, \text{requestA's input}) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Accordingly,

$$\text{degreeOfInputMatch}(\text{serviceB}, \text{requestA}) = \frac{1}{2} \times \max\langle 0,0 \rangle + \frac{1}{2} \times \max\langle 0,1 \rangle = 0.5$$

Similarly,

$\text{degreeOfOutputMatch}(\text{requestA}, \text{serviceB}) = 0.5$

$\text{degreeOfPreconditionMatch}(\text{serviceB}, \text{requestA}) = \frac{1}{4} \times (0 + 0 + 0 + 0) = 0$

$\text{degreeOfQoSMatch}(\text{requestA}, \text{serviceB}) = \frac{1}{4} \times (0 + 0 + 1 + 1) = 0.5$

$\text{degreeOfWebServiceMatch}(\text{WS}_{\text{ra}}, \text{WS}_{\text{sb}}) = \frac{1}{4} \times (0.5 + 0.5 + 0 + 0.5) = 0.375$

4. Algorithm Evaluation

Here the similarity matching algorithm proposed in this paper is compared with the classic OWL-S/UDDI matching algorithm [28] and a similarity matching algorithm proposed in literature [20] in the ability for measure the similarity quantity of service, the ability to suit the applications, as well as algorithms recall rate and precision rate.

The Ontology matching algorithm proposed about CAE simulation in this paper uses similarity matching degree value to measure matching degree between requirement service and existing service. The algorithm in this paper has some advantages compare with classic OWL-S/UDDI algorithm, as shown below.

(1)The algorithm in this paper has definite numerical measure of the degree of matching, and the OWL-S/UDDI matching algorithm only has 4 grades.

(2)In the calculation of similarity, the similarity is designed with weights, and can be freely regulated based on the number of input and output parameters, the Preconditions number, QoS number, so the precision of the algorithm is higher than the classic OWL-S/UDDI algorithm.

(3)The algorithm in this paper can be adapted to many parameters, but the classic OWL-S/UDDI algorithm only can be adapted to single parameter.

(4)Because the algorithm takes into QoS matching, the recall of the algorithm is better than the classic OWL-S/UDDI algorithm.

(5)We add RequirementJob information in requirement service ontology corresponds to the QoS of OWL-S ontology, so the algorithm enhances the matching ability.

A service semantic similarity algorithm in literature [20], the algorithm of ontology tree node similarity is same than the algorithm in this paper. The service similarity algorithm is proposed in this paper has some advantages compare with the service similarity algorithm proposed by [20], as shown below.

(1)This paper designs a specifically ontology for CAE simulation, and It is very suitable for application in CPCAES. The ontology can be applied to other needs appropriate transformation. In [20] literature, the authors put forward the context information of users into the requirements of context ontology, not suitable for a variety of professional fields.

(2)This paper describes the requirement ontology to increase RequirementJob information, which corresponds to the QoS of OWL-S ontology, and [20] has no the description of ontology.

(3)In this paper, the similarity matching algorithm considers the QoS similarity. The similarity matching algorithm in [20] does not consider the QoS similarity.

(4)The similarity matching algorithm of Preconditions is not given in [20], and this paper has carried on the supplement.

But the service requirement ontology and application of the algorithm proposed in this paper is biased in favor of CPCAES, application in the other areas need to be revised, in general weak than classic OWL-S/UDDI algorithm and the matching similarity matching algorithm proposed in [20].

Here we select 100 services from CPCAES. We use OWL-S service ontology to descript these services, and use requirement service ontology proposed in this paper to descript the requirement of user. We respectively use these matching algorithms to calculate the similarity, including the matching algorithm in this paper, the classical OWL-S/UDDI algorithm and the matching algorithm in [20]. In a plurality of parameters, the classic OWL-S/UDDI matching algorithm uses the minimum grade in matching grade of all parameters. The three algorithms return respectively these services which similarity is the similarity in more than 0.2, more than 0.4, more than 0.5, more than 0.8, 1. To the classical OWL-S/UDDI algorithm, we take subsumes matching results for more than 0.2 and more than 0.4. We take plugIn matching results for more than 0.5 and more than 0.8. The experimental results are shown in figure 5 and figure 6.

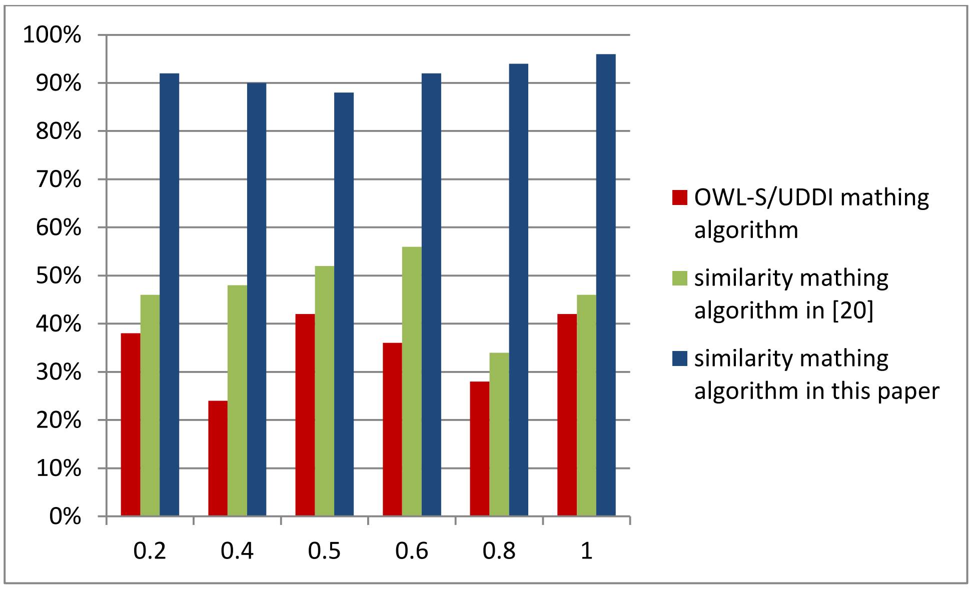


Figure 5. Precision Rate Comparison

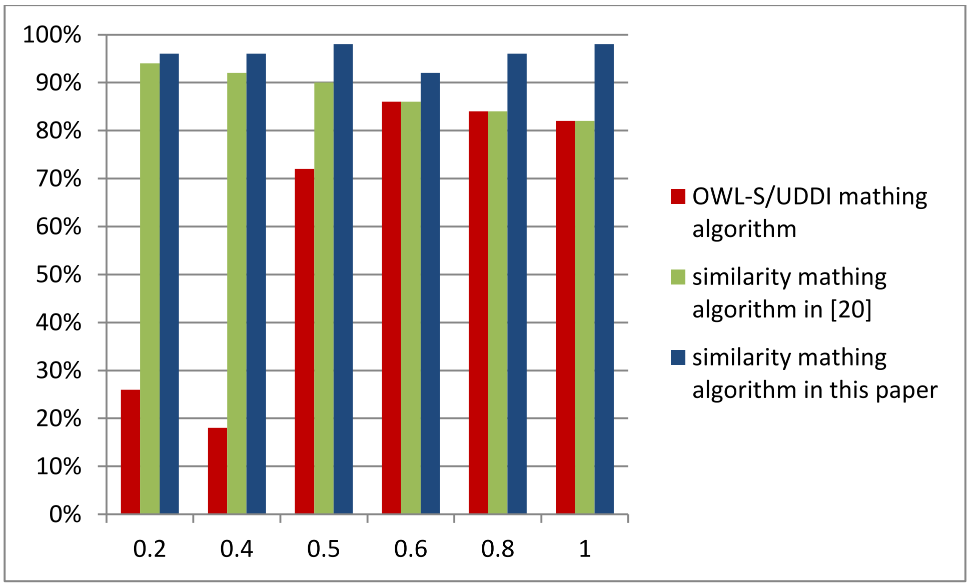


Figure 6. Recall Rate Comparison

Figure 5, and Figure 6, show that the similarity between the precision rate of the similarity matching algorithm in this paper is much better than the classical OWL-S/UDDI algorithm, and is much better than the similarity matching algorithm in [20], and the recall rate is slightly better. The main reason is the matching algorithm in this paper with QoS matching calculation, and it is more excellent expression ability in CAE simulation.

5. Engineering Prototype

The research framework of CPCAES is shown in Figure 7. Starting from the construction demand of the platform, the authros research and establish the SOA architecture. The authors make comparative analysis based on the reference to the similar platform, and adopt the SOA architecture, and put forward the overall architecture of

CPCAES base on SOA through the semantic Web technology research and the SOA theory research. The architecture is suitable for "Tianhe No.1" supercomputer and its software environment. In the process of development to CPCAES, the authors constantly tune architecture and carry out the evolution by using the method of rapid prototyping. After having completed the platform development of the third prototype, the authors finalized the overall system architecture and the system architecture of each layer. The authors design and implement the technical framework of cloud computing platform including the software selection, the composition technique solutions, the data flow. At present, the platform architecture and the technology framework have been completed, the authors has completed the development task of the second prototype of CPCAES, which shows in literature [6-7].

Sencondly, In order to study the feature analysis of the cloud services for CAE simulation in CPCAES. The authors have obtained the HPC service characteristics of the cloud platform for CAE simulation. Through the composition of the characteristics of supercomputer "Tianhe No.1", the authors carry out the study to description language of the semantic Web services and the service composition. In addition, the authors are not only study the automatic composition technology, but also implement the platform technology prototype by using the composition of some open source softwares.

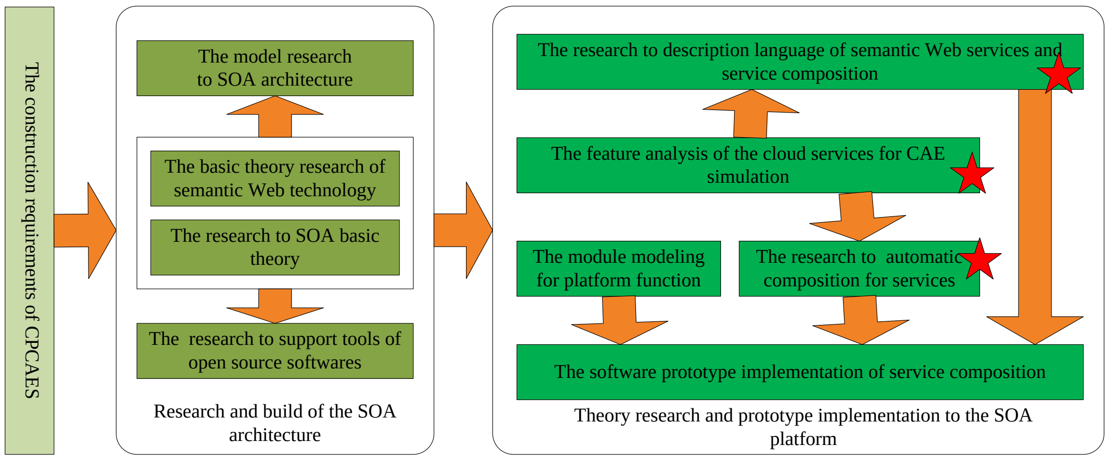

The construction requirements of CPCAES
The model research to SOA architecture
The basic theory research of semantic Web technology
The research to SOA basic theory
The research to support tools of open source softwares
Research and build of the SOA architecture
The research to description language of semantic Web services and service composition
The feature analysis of the cloud services for CAE simulation
The module modeling for platform function
The research to automatic composition for services
The software prototype implementation of service composition
Theory research and prototype implementation to the SOA platform

Figure 7. Overall Research Framework of CPCAES

The authors assume many HPC services have been able to be provided to the CAE simulation users. These services follows the description rules of OWL–S, and are stored in the database of CPCAES. When a service is joined each time in the platform, the service automatic composition technology is used to do composed modeling. The calculation process model will eventually become the initial requirement services for the CAE simulation users. As shown in the Figure 7, like star, The research work of the authors focuses on some fields, including the research to description language of semantic Web services and service composition, the feature analysis of the cloud services for CAE simulation, the research to automatic composition for services.

According to the algorithm in this paper, the authors use Eclipse as the development tool, Java as the development language, OWL-S as the description language for Web services, Tuscany as the container for Web services, WSDL2OWLS as the translation tool, OWL-S API as the interface tools. The overall implementation framework with the technology composition to open source softwares is shown in Figure 8.

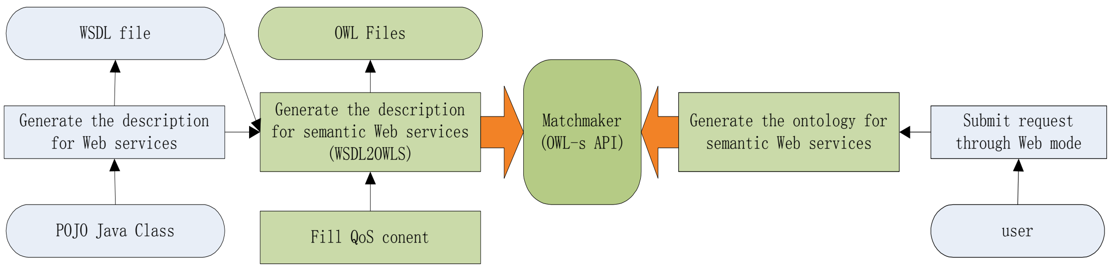
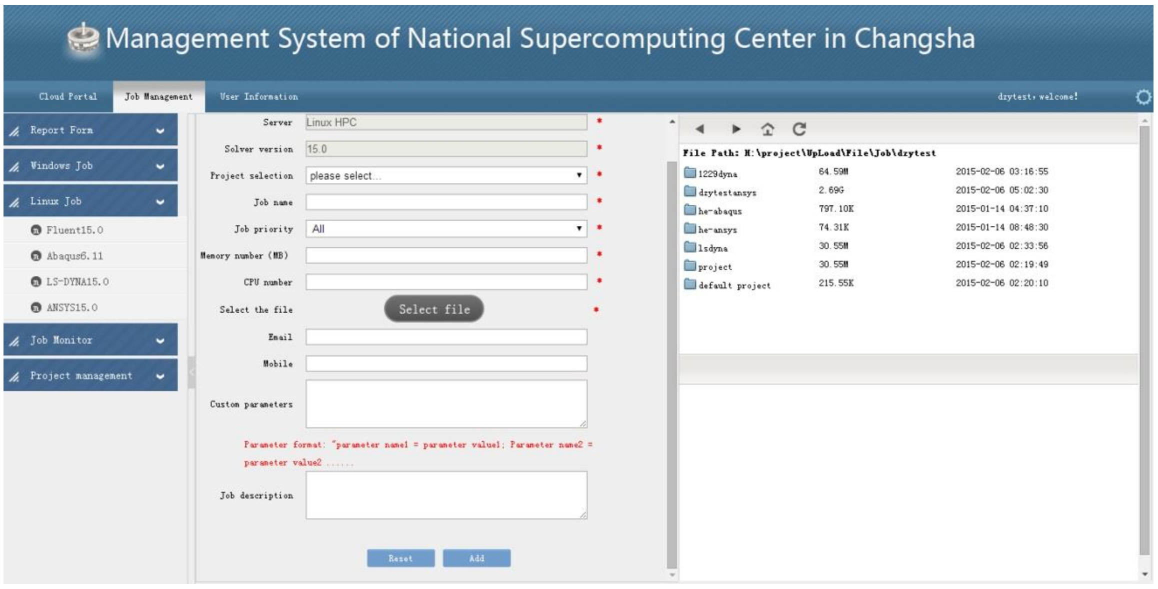

WSDL file
OWL Files
Generate the description for Web services
Generate the description for semantic Web services (WSDL2OWLS)
Matchmaker (OWL-s API)
Generate the ontology for semantic Web services
Submit request through Web mode
POJO Java Class
Fill QoS conent
user

Figure 8. Overall Implementation Framework with the Technology Composition to Open Source Softwares

All users submit their job requirements through the Web interface in CPCAES, as shown in Figure 9. After CPCAES receives the request, the service requirement ontology is generated. The Matchmaker is used to match the existing services in CPCAES through the matching algorithm described in this paper.


Management System of National Supercomputing Center in Changsha

Cloud Portal | Job Management | User Information | drytest, welcome!

Report Form
Windows Job
Linux Job
Fluent15.0
Abaqus6.11
LS-DYNA15.0
ANSYS15.0
Job Monitor
Project management

Server	Linux HPC
Solver version	15.0
Project selection	please select...
Job name	
Job priority	All
Memory number (MB)	
CPU number	
Select the file	Select file
Email	
Mobile	
Custom parameters	
Job description	

Parameter format: "parameter name1 = parameter value1; Parameter name2 = parameter value2 ......

Reset | Add

File Path: H:\project\UpLoad\File\Job\drytest

1229dyna	64.59M	2015-02-06 03:16:55
drytestansys	2.69G	2015-02-06 05:02:30
he-abaqus	797.10K	2015-01-14 04:37:10
he-ansys	74.31K	2015-01-14 08:48:30
lsdyna	30.55M	2015-02-06 02:33:56
project	30.55M	2015-02-06 02:19:49
default project	215.55K	2015-02-06 02:20:10

Figure 9. The Interface for Submitting Jobs in CPCAES

6. Further Research

Based on the research of this dissertation, the authors intend to continue to carry out the following research work.

(1)Make a choice between fully automatic composition for Web services and semi-automatic composition for Web services, and then realize the third prototype of CPCAES.

(2)Continue to research the description technology and the description tools for service composition, and study on the formal description technique of Petri nets, and the realization of open source software tools, such as the relationship between the OWL-S services by Petri net description and the Activiti process, the realized for the services in Activiti.

(3)Research the methods and tools for automatic service composition, especially the process of how to form the composite service. Study on the automatic search algorithm, the calculation approach for similarity, the computing operator for composition, and the rules for computing operator.

7. Conclusion

The service requirement ontology has its own characteristics in CPCAES, such as the descriptions of the computational field, the requirement resources for HPC, and the performance requirements for computing job. The ontology is proposed in this paper, and it composed with above information and the original IOPE description in OWL-S.

According to the matching rules given in this paper, it is necessary to meet the 4 conditions, including $I_r \supseteq I_a$, $O_r \subseteq O_a$, $P_r \supseteq P_a$, $Q_r \subseteq Q_a$. The similarity matching algorithm for Web services is proposed in this paper, and the authors give the similarity weights for the similarity calculation results to these 4 conditions. These weights reflect the emphasis on determination conditions. In calculating the similarity of each decision condition, the authors give the similarity weights for every similarity of parameter between service requirement ontology and existing service ontology. These weights reflect the importance degree of parameters. The authors get the total similarity by similarity weighted sum of the similarity calculation results to these 4 conditions.

Through qualitative and quantitative analysis, the matching algorithm proposed based on similarity has higher precision rate and recall rate than the classic OWL-S/UDDI matching algorithm and the similarity matching algorithm in [20]. The recall rate and precision rate can reach about 90%. The algorithm provides a basis for the research work on automatic service composition in CPCAES. The authors' research team has developed the second prototypes of CPCAES based on semantic, and they are developing the third prototype of CPCAES.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61174140, 61472127, 61272395); China Postdoctoral Science Foundation (2013M540628,2014T70767); Natural Science Foundation of Hunan Province (14JJ3107); Excellent Youth Scholars Project of Hunan Province (15B087); National Key Technology Support Program of China Grant No.2012BAH09B02.

References

[1] J. Bigot, Z.Hou, C. Pérez and V. Pichon, “A low level component model easing performance portability of HPC applications”, Computing, vol. 96, no. 12, **(2014)**, pp. 1115-1130.
[2] L. Li, “Pay attention to coordination to break through the bottlenecks of software development in super computer”, Science & Technology Review, vol. 32, no. 25, **(2014)**, pp. 9.
[3] B. Li, X. Chai, B. Hou, T. Li, Y. Zhang, H. Yu, J. Han, Y. Di, J. Huang, C.Song, Z. Tang, P. Wang, G. Shi and X. Wang, “Networked modeling & simulation platform based on concept of cloud computing—cloud simulation platform”, Journal of System Simulation, vol. 21, no. 17, **(2009)**, pp. 5292-5299.
[4] Y. Donghee, “Hybrid query processing for personalized information retrieval on the Semantic Web”, Knowledge-Based Systems, vol. 27, no. 3, **(2012)**, pp. 211–218 .
[5] F. He, “Research on some key technologies of semantic Web services composition”, Science Press, **(2013)**.
[6] Z. Deng, J. Zhang, S. Bai, Z. Liu, L. Chen and W. Zhang, “Architectural design of CAE integration platform based on super computation”, Journal of Hunan University(Natural Sciences), vol. 40, no. 7, **(2013)**, pp. 80-85.
[7] Z. Deng, J. Zhang, L.Yangbin and J. Xiao, “Research on middleware of cloud platform for industrial design and simulation on “TianheNo.1” super computer”, China Mechanical Engineering, vol. 26, no. 6, **(2015)**, pp. 766-772 to 798.
[8] H. Ji, “Product platform design services based on cloud computing”, Ph.D Thesis, Mechanical Science Research Institute in China, **(2012)**.
[9] J. Yang and G. Guo, “Design a New Manufacturing Model: Cloud Manufacturing”, Proceedings of the 2012 International Conference on Cybernetics and Informatics, **(2012)**.
[10] L. Zhang, Y. Luo, F. Tao, B. Li, L. Ren, X. Zhang, H. Guo, Y. Cheng, A. Hu and Y. Liu, “Cloud manufacturing: a new manufacturing paradigm”, Enterprise Information Systems, vol. 8, no. 2, **(2014)**, pp. 167-187.
[11] L. Ren, L. Zhang, F. Tao, C. Zhao, X. Chai and X. Zhao, “Cloud manufacturing: from concept to practice”, Enterprise Information Systems, vol. 9, no. 2, **(2015)**, pp. 186-209.

[12] Y. Lu, X. Xu and J. Xu, "Development of a Hybrid Manufacturing Cloud", Journal of Manufacturing Systems, vol. 33, no. 4, **(2014)**, pp. 551-566.
[13] S. H. Cho, "Development of simulation service platform based on cloud computing for manufacturing industry", AsiaSim2011, **(2011).**
[14] S. H. Cho, "CAE services on cloud computing platform in South Korea", AsiaSim 2012, **(2012)**.
[15] J. Yu, J. Cha, Y. Lu, W. Xu and M. Sobolewski, "A CAE-integrated distributed collaborative design system for finite element analysis of complex product based on SOOA", Advances in Engineering Software, vol. 41, no. 4, **(2009)**, pp. 590-603
[16] C. Ke and Z. Huang, "Self-adaptive semantic web services matching method", Knowledge-Based Systems, vol. 35, no. 11, **(2012)**, pp. 41-48.
[17] OWL for Services (OWL-S), http://www.ai.sri.com/daml/services/owl-s/.
[18] Web services Modeling Ontology (WSMO), http://www.w3.org/Submission/WSMO/.
[19] Web services Semantics - WSDL-S, http://www.w3.org/Submission/WSDL-S/.
[20] H. Peng, "matching and application for semantic Web services", Beijing University of Posts and Telecommunications press, **(2015).**
[21] S. Lee, S. Jeong, H. G. Kim, H. Jung, M. Lee, S. Song and B. J. You, "OntoPipeliner: An ontology-based automatic semantic service pipeline generator", Expert Systems with Applications, vol. 38, no. 8, **(2011)**, pp. 9472–9482.
[22] M. Klusch, B. Fries and K. Sycara, "OWLS-MX: A hybrid semantic web services matchmaker for OWL-S services, Web Semantics: Science, Services and Agents on the World Wide Web", vol. 7, no. 2, **(2008)**, pp. 121-133.
[23] A. Furno and E. Zimeo, "Context-aware composition of semantic web services", Mobile Networks and Applications, vol. 19, no. 20, **(2014)**, pp. 235-248.
[24] X. Zhao, "Research on context-aware HPC service discovery technology", the Degree of Master of Engineering, PLA Information Engineering University, **(2012)**.
[25] L. Song, "Research on semantic similarity computation and applications", Ph.D Thesis, Shangdong University, **(2009)**.
[26] B. Lantow and K. Sandkuhl, "An Analysis of Applicability using Quality Metrics for Ontologies on Ontology Design Patterns", Intelligent Systems in Accounting,Finance and Management, vol. 22, no. 1, **(2015)**, pp. 81-99
[27] Y. Zhang, L. Zhang, F. Qian and H. Liu, "Study on the cloud service selecting algorithm under cloud manufacturing mode", Journal of Chinese Computer Systems, vol. 35, no. 11, **(2014)**, pp. 2390-2395.
[28] S. Sharma, J. S. Lather and M. Dave, "Semantic Approach for Classification of Web Services Using Unsupervised Normalized Similarity Measure", Journal of Emerging Technologies in Web Intelligence, vol. 6, no. 3, **(2014)**, pp. 364-372.

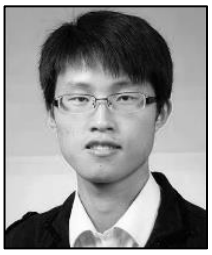
Authors

Ziyun Deng is a doctoral student of Hunan University. He is a professor. His recent research interests are in hign performance computing, logistics information technology.

Jing Zhang is a professor of Hunan University. His recent research interest is in control and optimization of complex industrial.

Lijun Cai is a professor of Hunan University, His recent research interests are in hign performance computing, cloud computing, bioinformatics.

Lei Chen is a doctoral student of Hunan University. His recent research interests are in cloud computing, big data.



Tingqin He is a doctoral student of Hunan University. His recent research interests are in cloud computing, database system.