IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 8, 2020, accepted February 24, 2020, date of publication March 31, 2020, date of current version April 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984648

A Framework for Trustworthy Web Service Composition and Optimization

CHUNLING HU[1], XIAONA WU[2], AND BIXIN LI (Member, IEEE)[2]
[1]School of Artificial Intelligence and Big Data, Hefei University, Hefei 201900, China
[2]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China

Corresponding author: Bixin Li (bx.li@seu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61806068 and Grant 61672204, in part by the Key Technologies Research and Development Program of Anhui Province under Grant 1804a09020058, and in part by the Major Program for Scientific and Technological of Anhui Province under Grant 17030901026

ABSTRACT Recently, Web service composition and optimization have received an increasing attention from both academia and industrial community. Most current methods have not paid enough attention to the user specific trust requirement for composite services. However, Trust is an important metric to judge whether a composite service can behave as user expected. In this work, firstly, a multifactor concept of trust of composite service is defined based on the trust of component services, interface compatibility and optimal binding schemes. Secondly, a trustworthy Web service composition and optimization framework called TWSCO is proposed to guarantee the trust of composite service and efficiency of Web service composition process. The interface-matching problem among component services and user preference are considered in TWSCO, which firstly uses component services filter to remove untrusted component services. Secondly, the concrete services, based on interface similarity, are organized as clusters. Thirdly, a composite template among component services is formed at the cluster level to guarantee the trust and efficiency of composite service. finally, the best binding scheme is discovered by an optimization method based on user specific trust metrics. In the end, experiments based on real Web services are presented to illustrate the proposed framework TWSCO can effectively guarantee the user preference trust of the composite service.

INDEX TERMS Service composition, service optimization, trustworthiness, interface matching, optimal binding scheme, composite template.

I. INTRODUCTION

Web services have recently received an increasing attention. Atomic services and composite services are two forms of service existence [1]. An atomic service fulfills the requests from consumers independently. When an atomic service cannot fully satisfy the complex requirements of users, there is a need to create a new value-added service by service composition. A composite service is actually a conglomeration of outsourced Web services working together to offer an intended function.

A wide range of researches on Web service composition have been carried out [2], such as service composition languages, service composition methods [3] (Workflow-based, Model driven, AI-planning), execution monitor [4] and the QoS(Quality of Service) modeling [5]. Service workflows and their applications exemplify the fruitfulness of service interoperation. Orchestration is the most widespread workflow management system which coordinates disparate autonomous services [6]. In orchestration, a composite service is created through multiple steps. First, the requirements of users are analyzed. Second, the business logic is represented at a highly abstract level. Finally, concrete services are bound to abstract tasks statically at design-time or dynamically during run-time. These research achievements make great sense to Web services, but there still exist some open problems. In this paper, we focus on the trust issues of service composition.

In the domain of Web service, trust can be briefly explained as following: from the user's perspective, the services he invokes can return the correct results and satisfy his non-functional contracts; from the integrator's perspective, the composite service should satisfy some requirements such as connectivity, correctness, non-functional properties and

The associate editor coordinating the review of this manuscript and approving it for publication was Wajahat Ali Khan.

scalability [7]. However, in the open runtime environment, cheating and malicious services are often mixed with honest and good ones. How to distinguish the untrustworthy services from the trustworthy ones, and then use the trustworthy services to create value-added composite service becomes a critical research area [8]. The trust evaluation for both atomic services and composite service is always needed to be considered because of the open and dynamic circumstance of Web service.

As a composite service is actually a conglomeration of outsourced Web services, it is created through multiple steps from the perspective of integrators or the service composition engines. The risk of exposing business data to untrustworthy trading partners may be high. So, the trustworthiness of composite services is extremely important. Following the steps of service composition, trustworthy service composition should appropriately solve the trust issues in the following aspects: service discovery, service evaluation, service selection, transport protocols, system architectures, topology and so on [9]. Based on trustworthy service selection and composition, recent researches for Web service composition and optimization have attracted more and more attention. We classify the prior research into four categories as following.

A. TRUST DEFINITION AND FACTORS

In order to study the problems about trustworthy service composition and optimization, it is necessary to understand the definition of trust first. There are several definitions of trust in the literatures. We list some of the definitions in the following.

Ying *et al.* define trust as entity's competence, and the connotation of competence is whether an entity acts dependably, securely, and reliably in a specified context [10].

Mui *et al.* define trust as a subjective expectation, which is an agent's expectation of the future behavior of others according to their historical behavior [11]. The trust of an entity is related to its reputation.

Viriyasitavat *et al.* view trust as a subjective measurement of dependability, security, and reliability of mutual relationship between interactive services in a specific context at a given time [12].

From the above definitions, we observed that trust can be defined according to various categories and a general definition can be quite elusive. The definitions of trust are different in different domains and contexts. As a result, a trust definition should be given in the context of Web service composition, which is the focus research problem of this paper.

Trust can be reflected by multi-factors such as reliability, utility, availability, reputation, risk, confidence, quality of services and other attributes [13]. As trust is an abstract concept and is influenced by many factors, the factors included in trust evaluation should be customizable according to the user's requirements and the specific context.

There are different metrics and different evaluation methods to evaluate trust. Continuous or discrete values are used to measure the level of trust. For example, trust is expressed as a real number in the intervals in [14], [15]. Probability models [16] and fuzzy logics [17] have also been used to measure trust.

B. TRUST EVALUATION OF ATOMIC SERVICES

Use own or others' experiences or both to infer the trust of atomic services, a great number of approaches have been proposed to evaluate trust and reputation of atomic services [18]. These ratings from experiences and recommendations are denoted as real numbers in the interval [0, 1].

Many researchers propose to use complex rating aggregation algorithms to filter out the ''bad'' ratings. Thus, the key to the success of ratings is the aggregation algorithm, in other words, how to integrate others' ratings into the trustworthiness derivation. Many algorithms and models have been used to aggregate ratings [19].

C. TRUST EVALUATION OF COMPOSITE SERVICES

Petri-nets is used to describe the logic of composite service, and then calculate the trust of composition based on aggregation topology [20]. Two-tier model is used to schedule Web service workflow [21]. Social network is used to analyze the trust of composition [22]. Reduction rules set can be also used to evaluate the trust of composition [23]. Subjective logic, Bayesian networks and other methods based on beta-mixture model have also been used to evaluate the global trust of composite services [24], [25]. Compared with atomic services, there are fewer approaches for trust evaluation of composite services [26].

D. TRUST-BASED OPTIMIZATION OF BINDING SCHEMES

With the development of internet and software engineering, a huge number of functionally equivalent services appear on the internet. As a result, distinguishing and selecting services with respect to non-functional properties become essential for both atomic and composite services.

Some researchers evaluate trust based on a set of QoS attributes [27]. Thus, the trust-based optimization is often based on QoS attributes. In other words, the optimization of binding schemes is carried out according to a set of QoS attributes, such as price, availability, reliability and response time, to find the best binding scheme with the highest QoS value. Some other studies combine trust with other traditional QoS attributes by a weight setting method to discover the best binding scheme [28]. Calculate the trust value of composite services through Bayesian networks, and the optimization is fulfilled by ant colony algorithm. Because of the large number of concrete services, the time used for optimization may be intolerable.

In any case, finding the optimal binding scheme is an NP-hard problem, requiring a significant amount of time and effort to find the optimal binding scheme from a huge number of possible ones. In order to find the optimal binding scheme, the classic algorithms for solving NP-hard problems, such as linear programming, dynamic programming and knapsack

problem, have also been adapted to the research area of trust-based binding optimization [29].

On the other hand, the correctness of composition process is a crucial part of trustworthiness, which requires the compatibility of the message passing between component services. There are multiple service implementations available from various providers for the same task, and a selection between the different binding schemes according to non-functional properties needs to be made. A binding scheme with high QoS properties is an effective trust evidence for the composite service [30].

From the present researches for service composition and optimization described above, a few open problems can be summarized in the following.

First, a clear definition of trust about service composition is needed. Although there exist many trust definitions, none of them can accurately describe the definition of trust and its connotation in the context of service composition. To be specific, the factors which will influence the trustworthiness of composite service have not been clearly identified. As composite services have their own characteristics, the trust definitions used in other domains need to be revised.

Second, the interface-matching problem among component services approaches is often ignored by approaches described above, while interface-matching is important for an executable composite service. An optimization does not make any sense if the optimal binding scheme is unable to be executed. An interface-matching check is essential for trustworthy service composition, which is used to ensure the correct interactions behavior of a composite service.

Last, it takes a significant amount of time and effort to find the optimal binding scheme among many possible ones. However, many of these binding schemes are not executable, either because of the interfaces among component services are incompatible, or because of the global non-functional attribute cannot meet the integrator's expectation. The integrator urgently needs a method which can reduce the number and complexity of the binding solution.

In our previous paper [31], we have tried to research the trustworthiness of composite services. In this paper, we propose a novel framework TWSCO for trustworthy Web service composition and optimization. Compared with previous work, we give formal definition and connotation of trustworthy service composition and optimization, optimize algorithms of component services clustering and template generator, extend binding optimizer form the integrator's preference. In addition, experiments based on real Web services are presented to illustrate the proposed framework TWSCO can effectively and efficiently guarantee the user preference trust of the composite service.

The contributions of this paper are manifold. First, a definition of trustworthy service composition is proposed. Then, a trust-based service composition and optimization framework called TWSCO is proposed, and each process of the framework TWSCO is elaborated in the following. In order to increase the efficiency of finding the best binding scheme, the concrete services are clustered based on interface similarity. At the cluster level, a composite template is constructed, and then an optimization for binding scheme is carried out according to the user specific trust metrics. Generally speaking, the composite template formed on component service clusters and binding optimizer based on the integrator's preference are the key innovations of this work, which can guarantee the trust of composition processes and improve the efficiency of optimization.

The remainder of this paper is organized as followings. Section II gives definition and denotation of trustworthy service composition and optimization. Section III gives our proposed Trustworthy Web Service Composition and Optimization (TWSCO) framework for composite services. Section IV presents an empirical study based on real Web services to verify the proposed method; Section V concludes the paper and describes the future work.

II. DEFINITION AND CONNOTATION OF TRUSTWORTHY SERVICE COMPOSITION AND OPTIMIZATION

Trust has received much attention in many domains. An appropriate definition of trust should contain the relevant context information from a particular perspective. Considering trust in service composition from the integrator's perspective. We define trust as following.

Definition 1: Trust is a subjective mutual belief that the component services act dependably, securely and reliably and the interacting behaviors between components act correctly in a specific context for a specified period.

According to the definition, trust is a comprehensive concept which may be influenced by many factors such as the trust of component services, and the trust of interacting behaviors. Trust is also a dynamic value which may change over time. From the perspective of service integrator, we focus on the trust of a composite service at design-time instead of run-time. As shown in Figure 1, we extract three factors of composite service: trustworthy components, correct interacting behavior and optimal binding scheme, which are the connotation of trust and influence the composite service's trust at design-time.

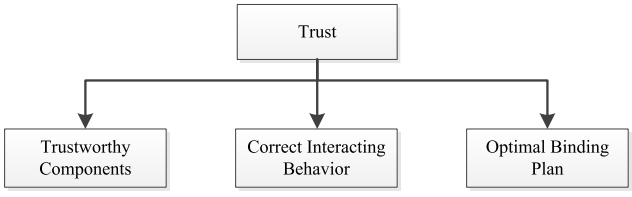


FIGURE 1. The connotation of trust.

A. SELECT TRUSTWORTHY COMPONENT SERVICE

An untrustworthy component service participating in a composite service will lower the overall trust inevitably. Selecting a trustworthy component service is rather important to the composite service. Trust of a composite service is multifold,

which includes a behavior aspect and a security aspect. Behavior aspect describes whether the target service performs reliably as expected, and a security aspect includes the credibility and authority of the composite service. The study of trust evaluation and measurement mechanisms of components are out of the scope in this paper. Current trust measure models for atomic services can be easily adapted to evaluate the trust of component services in composite service.

B. GUARANTEE CORRECTNESS OF INTERACTING BEHAVIORS IN COMPOSITION PROCESSES

Service composition involves more than a simple superposition of component services. It is greatly influenced by the interactive behavior among components. The correct interactions among components are the basis of an executable composite service, namely, the successful execution is the reflection of the trust of the composite service. End-user's feeling of trust towards the composite service will decrease rapidly if it is unable to be invoked. A guarantee of successful interactions at design-time is the key point to guarantee trust in composition processes. The interface matching between components is critically important to the interactions of components.

Keywords based approaches are used to collect Web services from resources, and the amount of returned results could be extremely large [32].The integrator often relies solely on the WSDL documents to learn more about the returned services. In order to guarantee the correctness of interacting behaviors among component services, the service integrator needs to check the WSDL documents of candidate services one by one, which is a complex and time-consuming task.

Based on the interactive requirements of composite services and similarity among service interfaces, we propose a method to cluster the retrieved services. Interface compatibility checking is carried out at the cluster level. A candidate service needs to receive inputs parameters from other services or send outputs parameters to other services. These services which the parameters receive from or send to have data exchange relation with the target candidate service. In the interface checking step, the services which cannot exchange data with the target service are filtered out. Then, a compatibility composition template is generated. In the end, the trust of composition is guaranteed, and the efficiency of service composition process is increased.

C. OPTIMIZE BINDING SCHEME VIA TRUST EVALUATION

Based on relationship of control-flow and data-flow, a composite service is described as a combination of generic service tasks in terms of service ontologies [33]. The generic service tasks are also called abstract services. Each abstract service can be realized by a great number of alternative concrete services. After binding all concrete services to corresponding abstract services, a concrete binding scheme is generated. Since functionally equivalent services have different non-functional attributes including different trust values, the trustworthiness of various binding schemes is different. The purpose of trust evaluation is to optimize the binding of concrete services to generate the most trustworthy composite service. Atomic service usually uses experience based method to calculate trust. Before execution, a composite service cannot be invoked and no experience is available. Therefore, the trust evaluation methods proposed for atomic services cannot be used directly for composite services.

Most proposed methods evaluate trust of composite services by aggregating trust values of component services based on topology, and the trust of component services are calculated according to historical interaction experiences. The trust values of component services are expressed as real numbers, representing the ratings from users. However, trust is a comprehensive concept, which may be influenced by many factors. Historical interaction experience is one of the factors, but it cannot represent the trust of composite services. On the other hand, a trustworthy composite service should have the ability to execute as well as meeting the expectation of the service integrator. Different integrators may have different subjective belief according to his preference. This subjective belief is the concrete representation of his expectation. Thus, an appropriate trust evaluation method for composite services binding scheme must be able to describe the features of trust according to the integrator's specification.

We use trust metrics to present the trust of binding schemes from the integrator's perspective. The trust metrics are defined as following.

Definition 2: Trust metrics are defined as the information of an entity that is required and used to evaluate the trust of that entity [34].

An entity can be a service in this paper. Trust metric is the first party information provided by a service to evaluate its trustworthiness. For example, a service can present its reliability as a trust metric. The term 'metric' represents the need to quantify the information and is defined as a set of measures (or input) necessary for trust evaluation.

The trust metrics are integrator specific because different integrators may select different evaluation criteria. For example, some integrators evaluate trust based on selected QoS attributes [27], thus, the trust-based optimization is often based on QoS attributes; others may consider trust as a special QoS attribute, and combine it with other traditional QoS attributes to evaluate trust of composite service. In this paper, based on the topology of the abstract composite process, the integrator's preference trust metrics of concrete component services, we calculate the trust value of a binding scheme.

III. TWSCO: A FRAMEWORK FOR TRUSTWORTHY WEB SERVICE COMPOSITION AND OPTIMIZATION

The present approaches for composite service check service composability on every binding scheme, which is an NP-hard problem. In addition, the interface-matching algorithms based on a series of multileveled match checking are complex. So, it is difficult for present approaches to find the optimal binding plan among many possible ones. Inspired by the problems of the present approaches, a novel trustworthy

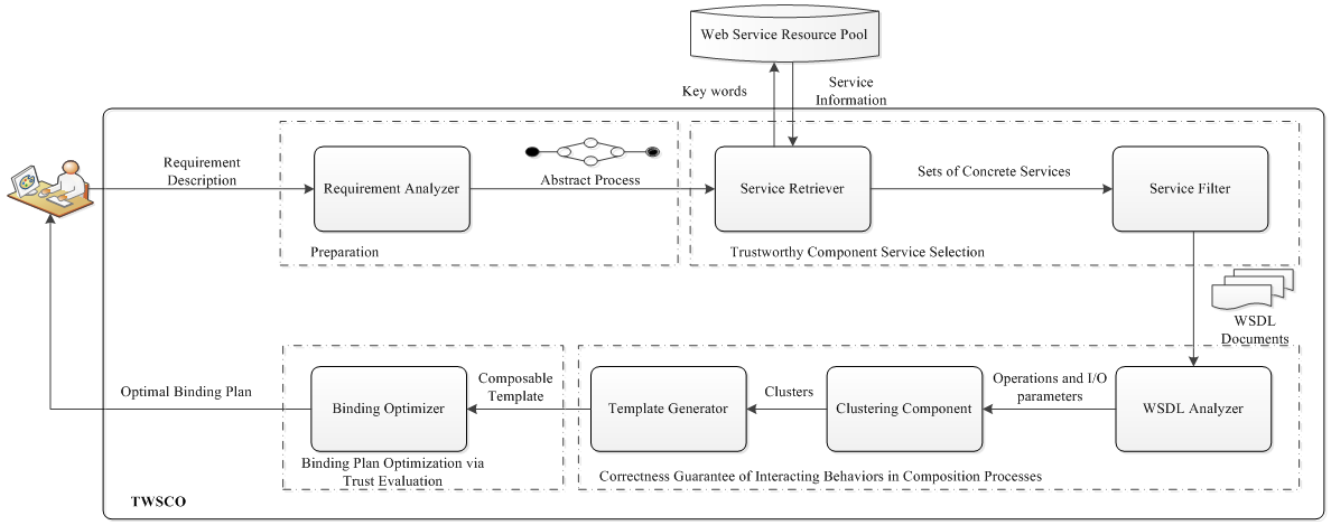


FIGURE 2. Architecture of TWSCO.

service composition and optimization framework TWSCO is proposed in this paper, where interface-matching problem among component services and optimization from integrator's preference are considered. TWSCO define trust as a multifactor concept described in Section II. Interface-based service clustering method is used to guarantee the trust of composition processes and a composite template is formed at the cluster level. Trust evaluation method from an integrator's preference is included in the framework to find out the most trustworthy binding scheme. Generally speaking, the composite template formed on component service clusters and binding optimizer from the integrator's preference are the key innovations of this work, which will guarantee the trust of composition processes and improve the efficiency of optimization.

In the TWSCO framework, a service integrator first specifies composition requirements, including functional and non-functional requirements. Then the requirement analyzer translates the requirements to an abstract composition process. The service retriever searches for concrete services related to abstract services from Web service resource pool and the retrieved results with related information, such as WSDL documents and non-functional attribute values, are returned together to the retriever. Then, service filter will remove services without well-defined WSDL documents, without existing endpoint URIs and with trust values lower than the specified thresholds. Next, the WSDL documents of candidate services are passed to the WSDL analyzer. The WSDL analyzer filters the services, and parses the WSDL documents to get the operations along with correlated input and output parameters. Next, the clustering component classifies candidate services into clusters according to the similarity between operations. Then, a composition template is generated at the cluster level. The last but one step, the optimization operation via trust evaluation is carried out based on the generated template. Finally, the optimal binding scheme is returned to the integrator. Figure 2 shows the architecture of TWSCO framework, and the followings are the details of each framework component.

A. REQUIREMENT ANALYZER

A service integrator sends requirement description documents to the requirement analyzer. The requirement analyzer formalizes the requirements, denoted as UR shown in (1), where F and NF denote functional requirements, non-functional requirements respectively, I and O denote expected input parameters and output parameters respectively.

$$UR = (F, NF, I, O) \tag{1}$$

$$AC = (AS, CR, DR) \tag{2}$$

$$AS = \{AS_0, AS_1, AS_2, \cdots AS_n\} \tag{3}$$

Then according to these requirements, an abstract composition process denoted as AC is generated and shown in (2). Abstract services set denoted as AS is an abstract service set and shown in (3). Each element in AS can be denoted as $AS_i = (F)$, $0 \leq i \leq n$, where F denotes the function of AS_i. Set of control relations among abstract services is denoted as CR which is another presentation of control-flow of service. Pairwise abstract services AS_m, AS_n may have one of the following four control relations:

1) SEQUENCE RELATION

Denote as $ControlRelation(AS_m, AS_n)$.

2) CHOICE RELATION

Denote as $ControlRelation(AS_m + AS_n)$.

3) PARALLEL RELATION

Denote as $ControlRelation(AS_m \times AS_n)$.

4) LOOP RELATION

Denote as $ControlRelation(!AS_m)$.

Set of data relations among abstract services is denoted as DR which is another presentation of data-flow of service. For instance, if data relation between AS_m, AS_n is $DataRelation(AS_m, AS_n, O, I)$ which means the output parameters of AS_m are the input parameters of AS_n.

B. SERVICE RETRIEVER

For abstract service, the service retriever retrieves concrete services from the service resource pool by sending a query request message. The retrieve process is a keyword based searching process. The keywords are determined manually by the integrator according to the functional requirements of each abstract task. The results with related information, such as WSDL documents locations and non-functional attribute values, are returned together to the retriever. Equation (4) represents a concrete service, where f denotes the functional information, nf denotes non-functional information, and $wsdldoc$ denotes the corresponding WSDL document. The service retriever can organize the returned concrete services corresponding to each abstract service as sets denoted in (5).

$$s = (f, nf, wsdldoc) \tag{4}$$

$$CS = \{CS_{AS_i} | AS_i \in AS\}, \quad \text{where } CS_{AS_i} = \{s | s.f = AS_i.F\} \tag{5}$$

C. SERVICE FILTER

The service filter removes the component services according to the specific filter criterion. The specific criterion can be denoted as the services without well defined WSDL documents, or those without endpoint URI or trust values less than the predefined thresholds τ or operations without the required functions. The service filter can reduce the number of concrete services set CS for each abstract task, and the reduced set concrete services set CS' is shown in (6). The WSDL documents of concrete services in CS' will be returned to the WSDL analyzer after filtering.

$$CS' = \{CS'_{AS_i} | AS_i \in AS\} \tag{6}$$

where $CS'_{AS_i} = \{s | s \in CS_{AS_i} \wedge s \text{ passed filter process}\}$

D. WSDL ANALYZER

A WSDL document consists of a set of operations, numerous input and output parameters, which provides a definition of the structure and acceptable values for XML requests and responses. Based on WSDL4J [35], the WSDL analyzer parses the WSDL document and labels the services formally denoted as (7) by extending (4). Each operation opt_i is denoted as name, input parameters I and output parameters O.

$$s = (f, nf, OPT),$$

where

$$\forall opt_i \in OPT, \quad opt_i = (Name, I, O)$$

$$\forall I_i \in I, \quad I_i = \{name, type\}$$
$$\forall O_i \in O, \quad O_i = \{name, type\} \tag{7}$$

E. CLUSTERING COMPONENT

Based on the similarity among component services, a component clustering method is proposed to cluster component services and can greatly reduce the number of services in the optimization process. Since the operation is the basic unit of a component service, the interface similarity between pairwise component services can be defined as similarity between the input and output parameters of operations. The clustering algorithm is shown in Figure 3. For simplicity, we assume only one operation of a service provides the required function. In practice, there may be more than one operation in the same service that can provide the same function. In this case, we can do preprocessing by eliminating unnecessary or redundant operations in the service filter step.

Algorithm 1: Component Service Cluster
Input: Filtered Concrete Services *CS'*, Similarity Threshold λ
Output: Interface Similarity based Clusters *CL*
Begin:

```
while( CS' != Φ)
 { int k = 1;
   FirstCS = getItem(CS');
   Initialize CL_1;
   C_source = getItem(FirstCS);
   Append C_source To CL_1;
   DeleteItem C_source From FirstCS;
   while (FirstCS != Φ)
     { int Simi = 0;
     int flag = 0;
     C_target = getItem(FirstCS);
     for(int i= 1; i <= k; i ++)
       { int simi = 0;
         C_source = getItem(CL_i);
         simi = Simi_operations(C_source, C_target);
         if (simi > Simi)
           { Simi = simi;
             flag = i;
           }}
       if(Simi > λ)
           { Append C_target To CL_flag;}
       else
           { k = k + 1;
             Initialize CL_k;
             Append C_target To CL_k;}
       DeleteItem C_target From FirstCS;
       if(FirstCS == Φ)
         { for(int i = 0; i <= k; i ++)
           { Output CL_i;} }
     }
    DeleteItem FirstCS From CS';}
  return;
End
```

FIGURE 3. Algorithm of component services clustering.

The concrete service sets and threshold of similarity are input parameters of algorithm 1. The value of threshold λ ($\lambda \in [0, 1]$) affects the number of clusters. Each service forms its own cluster with $\lambda = 1$, while all concrete services form a single cluster with $\lambda = 0$. All sets appearing in the algorithm

are organized as a list. The function *getItem* gets the first element of each list at a time. The function *Append* appends the element to the end of the list, and the function *DeleteItem* deletes a specified element from the list. Lines 2 to 7 fulfill the initialization of the clustering, and the first concrete service of the first service set forms the first cluster. Lines 8 to 29 implement the main steps of the clustering. Line 11 define a service as the target service, then, compare the interface similarity between pairwise clusters (lines 12 to 19) and record the maximal similarity value and the corresponding serial cluster number. If the maximal similarity value is bigger than the threshold, add the target service to the most similar cluster (line 20 and 21), otherwise the target service forms a new cluster (lines 22 to 25). When all concrete services have been classified, output the result of clustering (lines 27 to 29). The clustering process is iteratively executed until every concrete service set has been clustered. The output of the component clustering algorithm is denoted as (8).

$$CL = \{CL_{AS_i} | AS_i \in AS\}$$

where

$$CL_{AS_i} = \{CL_{i0}, CL_{i1}, \cdots CL_{ik}\} \tag{8}$$

and

$$CS'_{AS_i} = \cup CL_{ij}, \quad 0 \le j \le k$$

$Simi_{operations}(C_{source}, C_{target})$ shown in (9) is used to calculate the interface similarity between pairwise services, where m and n are the number of input parameters of the operation pairwise services, k and l are the number of output parameters respectively. In (9) I_i combines I_j means assume that the i^{th} parameter of the source service is similar to the j^{th} parameter of the target service, and then the similarity can be calculated. It is worth noting that one input parameter of the source service can only combine to one input parameter of the target service. O_i combines O_j has the same meaning. The similarity of operation pairs is calculated by identifying the pair-wise correspondences of their input/output parameter lists. Similarity calculation aims to maximize the sum of the similarity scores. The similarity of input parameters is calculated according to (10), where the lexical similarity and the data type similarity can be calculated based on WordNet [36] respectively. Similarly, the similarity of output parameters can be calculated by (11).

$$\begin{aligned} &Simi_{operation}(C_{source}, C_{target}) \\ &= \frac{Max \sum_{i=1}^{m} \sum_{j=1}^{n} Simi_{input}(C_{source}.OP.I_i, C_{target}.OP.I_j) \times x_{ij}}{2 \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}} \\ &+ \frac{Max \sum_{i=1}^{k} \sum_{j=1}^{l} Simi_{output}(C_{source}.OP.O_i, C_{target}.OP.O_j) \times y_{ij}}{2 \sum_{i=1}^{k} \sum_{j=1}^{l} y_{ij}} \end{aligned} \tag{9}$$

where

$$x_{ij} = \begin{cases} 1, & if\ I_i\ combines\ I_j \\ 0, & else \end{cases}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2 \cdots m, \quad \sum_{i=1}^{m} x_{ij} = 1, \ j = 1, 2 \cdots n$$

and

$$y_{ij} = \begin{cases} 1, & if\ O_i\ combines\ O_j \\ 0, & else \end{cases}$$

$$\sum_{j=1}^{l} y_{ij} = 1, \quad i = 1, 2 \cdots k, \quad \sum_{i=1}^{k} y_{ij} = 1, \ j = 1, 2 \cdots l$$

$$\begin{aligned} Simi_{input}(I_i, I_j) = &\omega_1 Simi_{lexical}(I_i.name, I_j.name) \\ &+ \omega_2 Simi_{datatype}(I_i.type, I_j.type) \end{aligned} \tag{10}$$

where

$$\omega_1 + \omega_2 = 1$$

$$\begin{aligned} Simi_{output}(I_i, I_j) = &\omega_1 Sim_{lexical}(O_i.name, O_j.name) \\ &+ \omega_2 Simi_{datatype}(O_i.type, O_j.type) \end{aligned}$$

where

$$\omega_1 + \omega_2 = 1 \tag{11}$$

F. TEMPLATE GENERATOR

After component services clustering process, we get service clusters based on the interface similarity. The interface matching operation at the cluster level becomes possible. A valid interaction is simply a sequence of messages which does not contain errors, and whose operations correspond to the behavior of the services. Due to the matching operation, the concrete services which cannot really be combined together are filtered so that the number of concrete services during binding optimization is greatly reduced. The pseudo code is shown in Figure 4. The first service of each concrete cluster list represents the input and output parameters of the cluster and the interface matching method $InterfaceMatching(CL_{source}, CL_{target})$ resembles the method used in [37]. At the cluster level, a best matching template will be generated through template generating process.

G. BINDING OPTIMIZER

Although the composite template can help to greatly reduce the number of concrete services, each abstract service may be implemented by many alternative services. As functionally equivalent services have different non-functional attributes, the trust values of different binding schemes are different. The purpose of optimization is to find the best binding scheme. In this paper, the best means most trustworthy. As mentioned in section II, trust metrics are user specific. Based on control-flow, non-functional attributes of component services and user preference, we evaluate the trust value of a binding scheme. There are four kinds of control-flow described earlier

Algorithm 2: TemplateGeneration
Input: Interface Similarity based Clusters *CL*
Data Relations *AP.DR*
Output: Composite Template
Begin:

```
while (AP.DR != Φ)
  {
    int Simi=0; int simi=0; int lab1=0; int lab2=0;
    DR = getItem(AP.DR);
    AS_source = DR.AS_m;
    AS_target = DR.AS_n;
    for(int i = 0; i <= | CL_ASsource|; i++)
     {
       CL_source = getItem(CL_ASsource);
       for(int j = 0; j <= | CL_AStarget|; j++)
        {
          CL_target = getItem(CL_AStarget);
          simi = InterfaceMatching(CL_source, CL_target);
          if (simi>=Simi)
           { Simi = simi;
             lab1 = i;
             lab2 = j;
           }
        }
     }
    Append CL_ASsourcelab1, CL_AStargetlab2 To Template;
    DeleteItem DR From AP.DR;
  }
  return Template;
End
```

FIGURE 4. Algorithm of template generator.

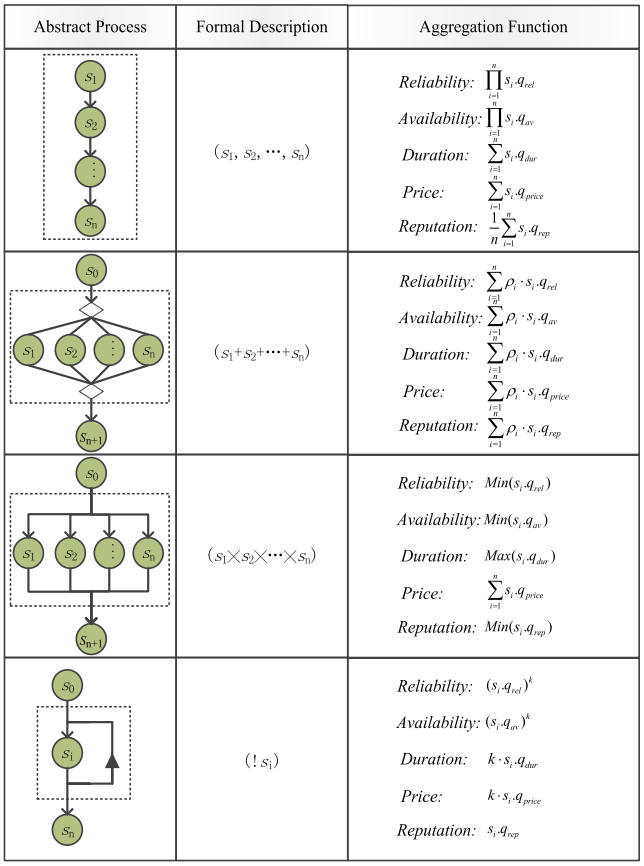


FIGURE 5. Aggregation function for trust value evaluation of binding schemes.

in part A of section III. The non-functional attributes contain multidimensional values such as QoS values. We briefly describe the aggregating method according to the topology of the abstract composite process in Figure 5, where ρ_i is the execution probability of branch i, and k is the iteration number of the loop. The trust metrics of a binding scheme can be adjusted by adding integrator's preference non-functional attributes. Some of the attributes could be of cost type, which means the higher the value, the lower the quality. Some other attributes could be of benefit type, which means the higher the value, the higher the quality. The following is the main steps to evaluate trust value of each binding scheme.

1) NORMALIZE EACH TRUST METRIC

Different trust metrics attributes may have different measurement units and numeric intervals. For this reason, the attribute values must be normalized before calculating the QoS value. For negative attributes, values are normalized according to (12). For positive attributes, values are normalized according to (13).

$$q = \begin{cases} \dfrac{q_{max} - q}{q_{max} - q_{min}} & if\ q_{max} - q_{min} \neq 0 \\ 1 & if\ q_{max} - q_{min} = 0 \end{cases} \tag{12}$$

$$q = \begin{cases} \dfrac{q - q_{min}}{q_{max} - q_{min}} & if\ q_{max} - q_{min} \neq 0 \\ 1 & if\ \ q_{max} - q_{min} = 0 \end{cases} \tag{13}$$

2) CALCULATE RELATIVE WEIGHT OF EACH TRUST METRIC

Element a_{ij} of matrix A is the comparison of importance of pairwise trust metrics, and can be approximately denoted as $a_{ij} = \omega_i / \omega_j$, where ω_i is the weight of the i-th trust metric. Matrix A multiply weight vector $\omega = [\omega_1, \omega_2, \cdots, \omega_n]^T$, we get (14). According to definition of eigenvalue and eigenvector, it can be found from (15) that weight vector ω is exactly the eigenvector of A and n is exactly the eigenvalue of A. So, Equation (14) can be rewrite as (16). The corresponding weight for each trust metric is given by the eigenvector ω corresponding to the maximum eigenvalue λ max.

$$A\omega = \begin{bmatrix} \omega_1/\omega_1 & \omega_1/\omega_2 & \cdots & \omega_1/\omega_n \\ \omega_2/\omega_1 & \omega_2/\omega_2 & \cdots & \omega_2/\omega_n \\ \vdots & \vdots & \ddots & \vdots \\ \omega_n/\omega_1 & \omega_n/\omega_2 & \cdots & \omega_n/\omega_n \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix} = n\omega \tag{14}$$

$$(A - nI)\omega = 0 \tag{15}$$

$$A\omega = \lambda_{\max}\omega \tag{16}$$

3) EVALUATE TRUST VALUE OF EACH BINDING SCHEME

The trust value of each binding scheme can be calculated according to (17), where each ω_i is the weight of the i-th trust metric and Q_i is the value of the i-th trust metric. The best

binding scheme with the largest trust value can be discovered by calculating each trust value of each binding scheme. In addition, we can sort the binding schemes and generate a ranking sequence of binding schemes according to calculated trust value of each binding scheme.

$$T = \omega_1 Q_1 + \omega_2 Q_2 + \cdots + \omega_n Q_n$$

where

$$\sum_{i=1}^{n} \omega_i = 1 \tag{17}$$

IV. EMPIRICAL STUDY

To demonstrate how the proposed TWSCO framework works, and to verify that the trust and efficiency of a composite service can be improved by considering the three aspects described in section II (trustworthy component service selection, correctness guarantee of interacting behaviors in composition processes, and binding scheme optimization via trust evaluation), empirical study based on real Web services is presented in this section.

A. ANALYZE INTEGRATOR'S REQUIREMENTS

Suppose an integrator integrates a composite service with the function of obtaining weather information from a given IP address, and expects the composite service with integrator inference based trust values as high a as possible.

As shown in Figure 6. IPtoCity, CitytoZipcode, and ZipcodetoWeather are three abstract services, denoted as AS_0, AS_1 and AS_2 respectively. The control-flow among three component services is $AS_0 \rightarrow AS_1 \rightarrow AS_2$ and the composite service is acquired by a simple sequence composition. Data relation $DataRelation(AS_0, AS_1, O, I)(AS_1, AS_2, O, I)$ means the output parameters of AS_0 are the input parameters of AS_1, and the output parameters of AS_1 are the input parameters of AS_2. The Start and End services are virtual services that provide the input and output of the composite service respectively. IPtoCity returns the city name corresponding to a given IP address. CitytoZipcode returns the zip code for a given city name. ZipcodetoWeather returns the weather information of an area represented by the zip code.

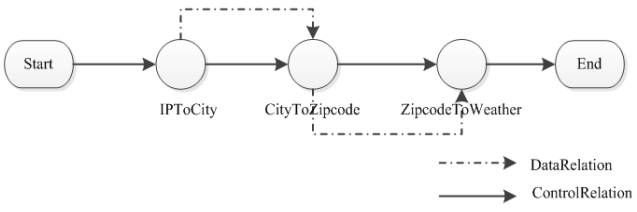


FIGURE 6. Abstract composite process.

B. RETRIEVE SERVICES

We have retrieved services from seekda which crawler has collected 28606 services from 7739 providers all over the world [38]. The keywords used for searching are shown in column 2 of Table 1 and the number of returned results is shown in columns 2 and 3 of Table 1.

TABLE 1. Number of returned services and filtered services.

	Key words	Returned	Filtered		
			a	*b*	*c*
AS_0	IP+city	99	35	11	8
AS_1	city+zipcode	66	28	7	4
AS_2	zip+weather	17	10	5	3

C. FILTER SERVICES

The service filter removes services which do not meet the specified demands. Services will be filtered out following four sequential steps: 1) the services without the required function; 2) the services without endpoint URIs or trust values are lower than threshold λ; 3) the services without well defined WSDL documents.

We assume the trust values of all candidate services are bigger than threshold λ. The reasons for making this assumption are as following: the services returned do not have trust values and using trust models to evaluate the trust of component services needs a long time running log, and the number of candidate services is small when filtering the services whose endpoint URIs do not exist. The results of filtering are shown in Table 1. The results under columns 4 to 6 of Table 1 are generated after filtering with steps *a*, *b* and *c*. The number under filtered step *a* is obtained after applying filter step a, then *b* by filter step *b*, *c* by filter step *c*.

D. PARSE WSDL DOCUMENTS

A WSDL document describes the interface information of a concrete service. It consists of a set of operations and each operation has numerous input and output parameters. After parsing, the concrete services are listed in Table 2.

E. CLUSTER CONCRETE SERVICES

We apply the interface similarity based service clustering algorithm 1 proposed in section III to check the comparability among component services and reduce the number of services involved in the next optimization process. The followings are the clusters of services in Table 2 in our empirical study.

$$\begin{aligned} CL_{AS_0} &= \{\{CS_{01}, CS_{04}, CS_{07}\}, \{CS_{05}\}, \\ &\quad \{CS_{02}, CS_{03}, CS_{06}CS_{08}\}\} \\ CL_{AS_1} &= \{\{CS_{11}, CS_{14}\}, \{CS_{12}, CS_{13}\}\} \\ CL_{AS_2} &= \{\{CS_{21}, CS_{22}\}, \{CS_{23}\}\} \end{aligned}$$

F. GENERATE THE COMPOSITE TEMPLATE

At the component services clusters level, we use algorithm 2 proposed in section III to generate the most feasible composite template as $\{CS_{01}, CS_{04}, CS_{07}\} \rightarrow \{CS_{12}, CS_{13}\} \rightarrow \{CS_{21}, CS_{221}\}$.

TABLE 2. Information of concrete services.

AS	*CS*		*OP*		
			name	*I*	*O*
AS_0	CS_{01}	IP2Geo	ReselveIP	{ipAddress,string} {license,string}	{City,string}
	CS_{02}	Showmyip_lookupPortType	Showmyip_lookup	{lookup_ip,string}	{IP_Address_Lookup_Properties,array of strings}
	CS_{03}	IpAddressSearchWebService	Getcountrycitybyip	{the ipAddress, string}	{response,array of strings}
	CS_{04}	Ip2LocationWebService	ip2location	{ipAddress,string} {license,string}	{city,string}
	CS_{05}	Ip2loc	IPLocation	{ipAddress,string} {license,string}	{code,string}
	CS_{06}	Wsip	GetLocation	{ip,string}	{Location,array of strings}
	CS_{07}	Geometry	IPQuery	{ip,string} {key,string}	{city,string} {state,string} {zipcode,strin
	CS_{08}	GeoCoder	IPAddressLookup	{ipaddress,string}	{city,string} {country,string}
AS_1	CS_{11}	AddressLookup	CityStateToZipCodeMatcher	{city,string} {stateAbbrev,string} {IgnoreBadCitySpelling,string} {licenseKey,string}	{zip, array of strings}
	CS_{12}	ZipcodeLookupService	cityto1zip	{city,string}	{cityTo1ZipResult,string}
	CS_{13}	ZipRpc	cityto1zip	{city,string}	{cityTo1ZipResult,string}
	CS_{14}	ZipCodeLookup	GetZIPCodeForCityState	{city,string} {state,string}	{zip, array of strings}
AS_2	CS_{21}	Weather	GetWeather	{zip,string}	{GetWeatherForZipCodeResponse,array of strings}
	CS_{22}	Weather	GetCityWeatherByZip	{zip,string}	{weather return,array strings}
	CS_{23}	WeatherBugWebService	GetLiveWeatherByUSZipCode	{zipcode,string} {unittype,array of strings} {Acode,string}	{LiveCompactWeatherDataarray of strings}

G. OPTIMIZE BINDING SCHEMES

The real-world Web services found in seekda include non-functional attributes of duration and availability only. To illustrate the user preference based binding scheme optimization of framework TWSCO, we add three trust metrics (i.e., Price, Reliability, Availability) and assign each a random value. Candidate concrete services generated in the step *F* with 5 dimensional trust metrics are shown in Table 3.

Trust value of each binding scheme is decided by trust metrics of components, the topology structure and the user preference weights of trust metrics. Formulas for sequence topology are used for calculating the values of trust metrics for each binding scheme in this case study. Composite service template generated in the step *F* consists of 12 binding schemes shown in Table 4, each column of table IV shows the values of the corresponding dimensional trust metric of each binding scheme. The following is the three steps to calculate trust value of composite service of each binding scheme.

TABLE 3. Trust metric values of candidate concrete services.

AS	*CS*	Dur./ms	Pri./yuan	Rel./%	Avail./%	Rep./[0,1]
AS_0	CS_{01}	170	129	72.40	90.06	0.82
	CS_{04}	465	164	80.85	84.19	0.77
	CS_{07}	675	291	90.87	96.38	0.93
AS_1	CS_{12}	787	397	74.34	90.06	0.98
	CS_{13}	563	318	75.31	80.13	0.76
AS_2	CS_{21}	211	416	90.02	80.26	0.78
	CS_{22}	600	131	70.93	72.34	0.81

TRUST METRIC VALUES OF CANDIDATE CONCRETE SERVICES

TABLE 4. Candidate binding schemes and the corresponding dimensional trust metrics.

No.	Binding scheme			Dur./ms	Pri./yuan	Rel./%	Avail./%	Rep./[0，1]
1	CS_{01}	CS_{12}	CS_{21}	1168	942	48.45	65.10	0.86
2	CS_{01}	CS_{12}	CS_{22}	1557	657	38.18	58.67	0.87
3	CS_{01}	CS_{13}	CS_{21}	944	863	49.08	57.92	0.79
4	CS_{01}	CS_{13}	CS_{22}	1333	578	38.67	52.20	0.80
5	CS_{04}	CS_{12}	CS_{21}	1463	977	54.11	60.85	0.84
6	CS_{04}	CS_{12}	CS_{22}	1852	692	42.63	54.85	0.85
7	CS_{04}	CS_{13}	CS_{21}	1239	898	54.81	54.14	0.77
8	CS_{04}	CS_{13}	CS_{22}	1628	613	43.19	48.80	0.78
9	CS_{07}	CS_{12}	CS_{21}	1673	1104	60.81	69.67	0.90
10	CS_{07}	CS_{12}	CS_{22}	2062	819	47.92	62.79	0.91
11	CS_{07}	CS_{13}	CS_{21}	1449	1025	61.60	61.98	0.82
12	CS_{07}	CS_{13}	CS_{22}	1838	740	48.54	55.87	0.83

TABLE 5. Normalized trust metric of each binding scheme.

No.	Dur./ms	Normalized	Pri./yuan	Normalized	Rel./%	Normalized	Avail./%	Normalized	Rep./[0,1]	Normalized
1	1168	0.7996	942	0.3080	48.45	0.4385	65.10	0.7810	0.86	0.6429
2	1557	0.4517	657	0.8498	38.18	0	58.67	0.4729	0.87	0.7143
3	944	1	863	0.4582	49.08	0.4654	57.92	0.4370	0.79	0.1429
4	1333	0.6521	578	1	38.67	0.0209	52.20	0.1629	0.80	0.2143
5	1463	0.5358	977	0.2414	54.11	0.6802	60.85	0.5774	0.84	0.5000
6	1852	0.1878	692	0.7833	42.63	0.1900	54.85	0.2899	0.85	0.5714
7	1239	0.7361	898	0.3916	54.81	0.7100	54.14	0.2559	0.77	0
8	1628	0.3882	613	0.9335	43.19	0.2139	48.80	0	0.78	0.0714
9	1673	0.3479	1104	0	60.81	0.9663	69.67	1	0.90	0.9286
10	2062	0	819	0.5418	47.92	0.4159	62.79	0.6703	0.91	1
11	1449	0.5483	1025	0.1502	61.60	1	61.98	0.6315	0.82	0.3571
12	1838	0.2004	740	0.6920	48.54	0.4424	55.87	0.3388	0.83	0.4286

1) NORMALIZE EACH TRUST METRIC

Equation (12) and (13) are used to normalize each trust metric in Table 4. The corresponding normalized trust metrics are shown in Table 5. where the normalized values are shown in the column next to the original values.

2) CALCULATE WEIGHT OF EACH TRUST METRIC

To verify framework TWSCO can effectively express preference of integrator, assume Integrator_1 and Integrator_2 need to optimize and select optimal binding scheme. Integrator_1 and Integrator_2 set the preferences to trust metrics of component services as following.

$$A_1 = \begin{bmatrix} 1 & 1/2 & 3 & 2 & 1/5 \\ 2 & 1 & 4 & 3 & 1/4 \\ 1/3 & 1/4 & 1 & 2 & 1/6 \\ 1/2 & 1/3 & 1/2 & 1 & 1/7 \\ 5 & 4 & 6 & 7 & 1 \end{bmatrix}$$

Integrator_1 sets reputation as the most important trust metric, followed by price, duration, reliability and availability. Saaty's 1-9 scale [39] is used for pairwise comparison: 1 means the equal importance, 9 means the highest level of importance. We present the following pairwise comparison matrix A_1.

The maximum eigenvalue of matrix A_1 is $\lambda_{\max} = 5.2097$, and matrix A_1 has passed consistent check [39]. The eigenvector ω corresponding to the eigenvalue $\lambda_{\max} = 5.2097$ is $\omega = [0.128, 0.202, 0.071, 0.058, 0.541]$, which is the weight vector of the corresponding trust metrics for Integrator_1's preference.

$$A_2 = \begin{bmatrix} 1 & 4 & 3 & 1/3 & 5 \\ 1/4 & 1 & 1/4 & 1/6 & 1/2 \\ 1/3 & 4 & 1 & 1/4 & 3 \\ 3 & 6 & 4 & 1 & 7 \\ 1/5 & 2 & 1/3 & 1/7 & 1 \end{bmatrix}$$

TABLE 6. Trustworthiness of each binding scheme of two integrators.

	1	2	3	4	5	6	7	8	9	10	11	12
Integrator$_1$	0.5888	0.6433	0.3562	0.4123	0.4696	0.5217	0.2386	0.2921	0.6735	0.7189	0.4014	0.4483
Integrator$_2$	0.6681	0.3941	0.5602	0.3019	0.5303	0.2591	0.4338	0.1772	0.7186	0.4184	0.5993	0.3161

Integrator$_2$ set availability as the most important trust metric, followed by duration, reliability, reputation and price. We present the following pairwise comparison matrix A_2.

The maximum eigenvalue of matrix A_2 is $\lambda_{\max} = 5.2542$, and matrix A_2 has passed consistent check. The eigenvector ω corresponding to the eigenvalue $\lambda_{\max} = 5.2542$ is $\omega = [0.258, 0.050, 0.140, 0.488, 0.006]$, which is the weight vector of the corresponding trust metrics for Integrator$_2$'s preference.

3) EVALUATE TRUST VALUE OF EACH BINDING SCHEME

Equation (17) is used to calculate trustworthiness of each binding scheme of Integrator$_1$ and Integrator$_2$. The result is shown in Table 6. From Table 6, the optimal binding scheme for Integrator$_1$ is binding scheme 10: $CS_{07} \rightarrow CS_{12} \rightarrow CS_{22}$, while the optimal binding scheme for Integrator$_2$ is binding scheme 9: $CS_{07} \rightarrow CS_{12} \rightarrow CS_{21}$. In addition, we can generate a rating sequence of binding schemes for each integrator by sorting trust values of all binding schemes.

The optimal binding scheme is generated according to integrator preferences, so that the binding scheme is the most trustworthy one from the integrator's perspective.

H. EVALUATION FOR TWSCO FRAMEWORK

The above TWSCO case illustrates how the proposed TWSCO framework can effectively improve the trustworthiness of service composition.

For each abstract service, there are a great number of concrete services. Evaluating each concrete service corresponding to abstract service is a NP-hard problem. The framework TWSCO proposed in this paper extends current works on service composition and optimization by classifying candidate services into clusters according to the similarity between operations, and generating a composition template for the cluster level. The optimization operation via trust evaluation is carried out based on the generated template which is a key contribution of TWSCO.

1) GENERATE A NEW LEVEL OF COMPONENT CLUSTERS

According to the operation process of TWSCO, the services are organized in different levels as shown in Figure 7. Compared with the traditional service composition methods, the cluster level is a new level added to the service composition and optimization process. Applying TWSCO, trustworthiness of service composition and optimization in this case study is guaranteed because of optimization for both component services and binding schemes.

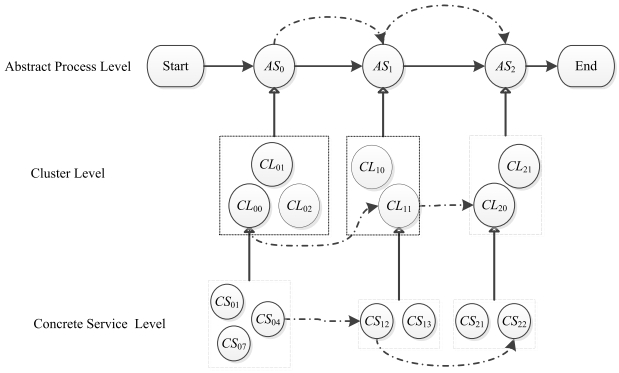


FIGURE 7. Service levels according to TWSCO.

2) OPTIMIZE COMPONENT SERVICES AND BINDING SCHEMES

The services returned by the Web service resource pool are filtered. The services without well-defined WSDL documents or without endpoint URIs or trust values less than the given threshold or operations without required functions are rejected. The trust of remaining component services is improved, and the number of candidate services is reduced.

Trust and efficiency of composition and optimization processes is guaranteed. Firstly, candidate services which have data relations have passed interface matching checking. Secondly, in binding optimization process, the number of concrete services has been greatly reduced. Observed that in our case study, the number of original binding schemes is 111078 (99*66*17), and most of them are inexecutable. If the optimization is carried out right after service filtering, the number of binding schemes is 96 (8*4*3), which is still quite large. Using our template based method, the number is reduced to 12 (3*2*2), which is much smaller than the original one. Fig. 8 shows the number of component services in different process of TWSCO in our empirical study.

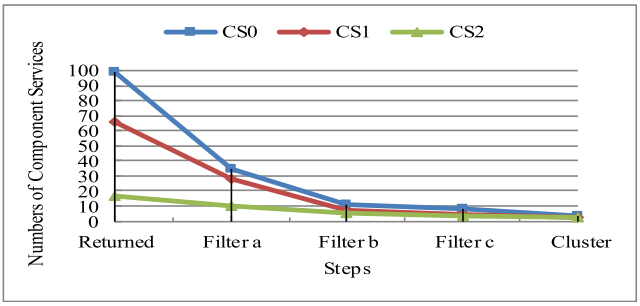


FIGURE 8. Number of different process of component services of TWSCO.

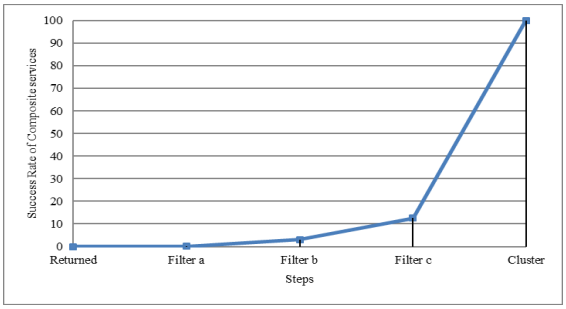


FIGURE 9. Success rate of different process of services composition.

The success rate of each step of services composition can be defined as the ratio of the number of successful binding schemes and the number of all binding schemes needed optimization. Fig. 9 shows the success rate of different step in services composition of framework TWSCO in our case study. It can be found that the success rate of services composition can be greatly increased through services filtering and component services clustering and matching.

3) SUPPORT USER PREFERENCE-BASED BINDING OPTIMIZATION

Different services integrator's preference to trust metric is different. Integrator will think the composite service is trustworthy if the composite service executes according to his expectation and has the expected service quality. Therefore, the integrator's trust metrics preference plays an important role in service composition and optimization.

Since different integrators have different trust evaluation criteria, the trust metrics selection is user specific. To verify framework TWSCO can effectively represent integrator preference, according to Table 6, we compare the two integrators' service ranking sequence of binding schemes shown in Fig. 10. Through two integrators demand the same fundamental function, and use same composite template and same component services for composite service, the inference-based binding scheme sequences are obviously different. For Integrator1, the ranking sequence of binding schemes based on trust values is 10→9→2→1→6→5→12→4→11→3→8→7. For Integrator2, the ranking sequence of binding schemes based on trust values is 9→1→11→3→5→7→10→2→12→4→6→8. The main reason is their preference to trust metric is different, and the weights of trust metrics are user specific.

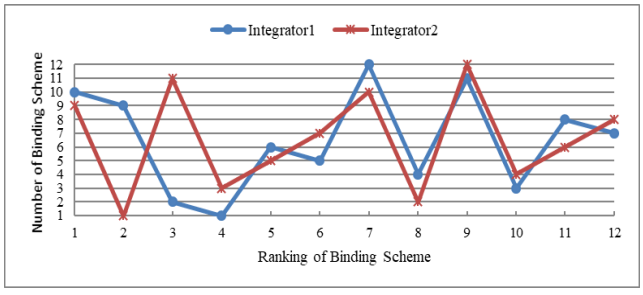


FIGURE 10. Ranking sequence of binding schemes based on different user preferences.

To verify the framework TWSCO can express integrator preference to trust metrics, top 25% binding schemes of each ranking sequence of all binding schemes of each integrator are selected to calculate average values of all five trust metrics. The average trust metrics of all five trust metrics of all 12 candidate binding schemes are also calculated as benchmark reference values. To show comparison results in just one Figure 11, the trust metric duration and price are narrowed 10 times while reputation is magnified 100 times.

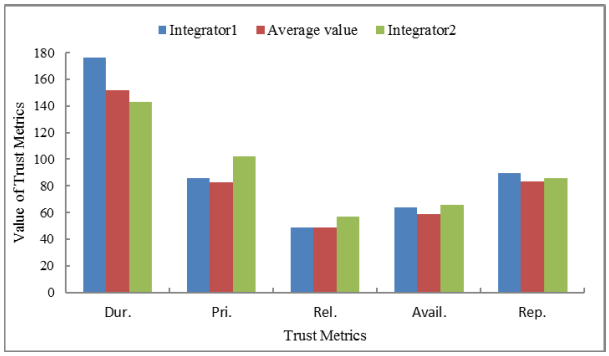


FIGURE 11. Comparison of trust metrics values among binding schemes based on different integrator preference.

From Figure 11, it can be found that the average values of reputation and price of the selected top 25% binding schemes according to Integrator1's preference are better than the corresponding values of the selected top 25% binding schemes according to Integrator2's preference, and better than the corresponding average values of all 12 candidate services. While the average values of availability and duration of the selected top 25% binding schemes according to Integrator2's preference are better than the corresponding values of the selected top 25% binding schemes according to Integrator1's preference, and better than the corresponding average values of all 12 candidate services. The experimental results show that the proposed method can effectively express user preferences for multi-dimensional trust metrics.

4) COMPARISON WITH CURRENT WORKS

Probability, reduction and product are three main methods for evaluating trust of composite services [27]–[29]. For probability and production based methods, the trust value of a composite service is sensitive to the number of component and will be sharply decreased with the growing number of component services. In our framework TWSCO, a composite template among component services is formed at the cluster level to greatly reduce the number of services. In addition, a user specific trust metric is defined and an optimization method is included to efficiently find out the most trustworthy binding scheme.

Based on Vector-Space Model, WordNet and semantic similarity metrics is proposed in an interface-matching algorithm [36], [37]. Text document and semi-structured document matching methods are proposed to calculate the similarity among text descriptions, signatures and logical constraints. Composability is checked through a set of rules based on ontology [33]. These approaches check service composability on every binding scheme, which is an NP-hard problem. On the other hand, the interface-matching algorithms are based on a series of multileveled match checking, which makes the algorithm complex. Our framework TWSCO classifies candidate services into clusters according to the similarity between operations. Then, a composition template is generated at the cluster level. As a result, the scale of optimization is greatly reduced.

5) THREATS TO VALIDITY

Like any empirical validation, ours has its limitations as following.

First, the empirical study involves a simple sequence composition, and the candidate service set is not big because of the nature of the specific real world Web service. The proposed framework should work more efficiently for composite services with complex topology and large component service sets.

A second concern is about the interaction behavior between component services. In this empirical study, we only consider the interface matching problem which represents the interdependency of data between components. A successful interaction demands messages to be exchanged under additional, sequential constraints. We did not examine these sequential constraints in our case study.

V. CONCLUSION AND FUTURE WORK

Trust plays an important role in service composition and optimization processes due to the open and dynamic environment of Web services. In this paper, we first define a trust concept for composite services. Then, we propose TWSCO framework for service composition and optimization. In this framework TWSCO, service filter is used to guarantee the trust of component service selection processes, and interface-based service clustering method is used to guarantee the trust of composition processes. In addition, a preference based trust evaluation method is also included in the framework to find out the most trustworthy binding scheme. Experimental results show that the framework TWSCO can effectively guarantee the trust and efficiency of service composition and optimization. There are a few directions for future research.

A. The empirical study presented in this paper is a simple sequence composition, and the candidate service set is relatively small. We expect the clustering method will work better with larger service sets. In the future, we will apply the proposed framework TWSCO for more complex composite services.

B. The inter-service dependencies and conflicts should be taken into consideration during service optimization to improve TWSCO in the future work.

C. Most of the processes of TWCSO are achieved manually. As automated service composition is essential for practical usage, we should adapt TWCSO to the automated execution engines in the future.

D. Follow the tracks of the latest research process of Web service composition and Optimization, continuously improve and optimize the proposed framework TWSCO.

ACKNOWLEDGMENT

The authors would like to thank Q. Shanshan and G. Guoqing for their work during the paper correction.

REFERENCES

[1] A. L. Lemos, F. Daniel, and B. Benatallah, "Web service composition: A survey of techniques and tools," *ACM Comput. Surveys*, vol. 48, no. 3, pp. 1–41, 2015.

[2] S. Bansal, A. Bansal, G. Gupta, and M. B. Blake, "Generalized semantic Web service composition," *Service Oriented Comput. Appl.*, vol. 10, no. 2, pp. 111–133, Jun. 2016.

[3] P. Wang, Z. Ding, C. Jiang, M. Zhou, and Y. Zheng, "Automatic Web service composition based on uncertainty execution effects," *IEEE Trans. Services Comput.*, vol. 9, no. 4, pp. 551–565, Jul. 2016.

[4] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling, "Untrusted business process monitoring and execution using blockchain," in *Proc. Int. Conf. Bus. Process Manage. (BPM)*, Rio de Janeiro, Brazil, Sep. 2016, pp. 329–347.

[5] C. Yu and L. Huang, "A Web service QoS prediction approach based on time- and location-aware collaborative filtering," *Service Oriented Comput. Appl.*, vol. 10, no. 2, pp. 135–149, Jun. 2016.

[6] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization," *IEEE Commun. Mag.*, vol. 54, no. 1, pp. 98–105, Jan. 2016.

[7] S. Chen, G. Wang, and W. Jia, "Cluster-group based trusted computing for mobile social networks using implicit social behavioral graph," *Future Gener. Comput. Syst.*, vol. 55, pp. 391–400, Feb. 2016.

[8] W. Viriyasitavat and A. Martin, "A survey of trust in workflows and relevant contexts," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 3, pp. 911–940, 3rd Quart., 2012.

[9] T. Zhang, J.-F. Ma, N. Xi, X.-M. Liu, and J.-B. Xiong, "Trust-based decentralized service composition approach in service-oriented mobile social networks," *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 44, no. 2, pp. 258–267, 2016.

[10] S. Ying and T. Grandison, "Big data privacy risk: Connecting many large data sets," in *Proc. IEEE 2nd Int. Conf. Collaboration Internet Comput. (CIC)*, Pittsburgh, PA, USA, Nov. 2016, pp. 86–91.

[11] L. Mui, M. Mohtashemi, and A. Halberstadt, "A computational model of trust and reputation," in *Proc. 35th Annu. Hawaii Int. Conf. Syst. Sci.*, Big Island, HI, USA, 2002, pp. 2431–2439.

[12] W. Viriyasitavat, L. D. Xu, and W. Viriyasitavat, "Compliance checking for requirement-oriented service workflow interoperations," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1469–1477, May 2014.

[13] Z. Lu, G. Qu, and Z. Liu, "A survey on recent advances in vehicular network security, trust, and privacy," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 760–776, Feb. 2019.

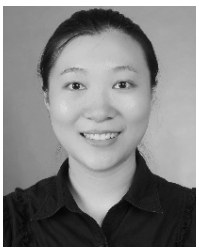
[14] S. Villata, G. Boella, D. M. Gabbay, and L. van der Torre, "A socio-cognitive model of trust using argumentation theory," *Int. J. Approx. Reasoning*, vol. 54, no. 4, pp. 541–559, Jun. 2013.

[15] P. R. Vamsi and K. Kant, "Trust and reputation aware geographic routing method for wireless ad hoc networks," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 27, no. 2, pp. 121–137, 2018.

[16] X. Luna Dong, "Knowledge-based trust: Estimating the trustworthiness of Web sources," *Proc. VLDB Endowment*, vol. 8, pp. 1–15, Feb. 2015.

[17] A. Selvaraj and S. Sundararajan, "Evidence-based trust evaluation system for cloud services using fuzzy logic," *Int. J. Fuzzy Syst.*, vol. 19, no. 2, pp. 329–337, Apr. 2017.
[18] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A survey on trust and reputation models for Web services: Single, composite, and communities," *Decis. Support Syst.*, vol. 74, pp. 121–134, Jun. 2015.
[19] W.-F. Tung and S.-H. Tseng, "Hybrid approach of trust inference and expert social search," *J. Converg. Inf. Technol.*, vol. 7, no. 7, pp. 342–351, 2012.
[20] A. Abdelli, W. Serrai, L. Mokdad, and Y. Hammal, "Time Petri nets for performance evaluation of composite Web services architectures," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Larnaca, Cyprus, Jul. 2015, pp. 122–127.
[21] J. A. Garay, R. Gelles, D. S. Johnson, A. Kiayias, and M. Yung, "A little honesty Goes a long way: The two-tier model for secure multiparty computation," in *Proc. 12th Theory Cryptogr. Conf., (TCC)*, Warsaw, Poland, Mar. 2015, vol. 9014. Springer Verlag, pp. 134–158.
[22] L. Liu and H. Jia, "A novel optimal social trust path selection algorithm for large-scale complex social networks," *IEICE Trans. Commun.*, vol. E97.B, no. 9, pp. 1910–1920, 2014.
[23] P. Sun, "Service composition and optimal selection with trust constraints," in *Proc. IEEE Asia–Pacific Services Comput. Conf.*, Dec. 2010, pp. 645–653.
[24] L. Li and Y. Wang, "A subjective probability based deductive approach to global trust evaluation in composite services," in *Proc. IEEE Int. Conf. Web Services*, Jul. 2011, pp. 604–611.
[25] N. Sasikaladevi, "Trust based cloud service composition framework," *Int. J. Grid Distrib. Comput.*, vol. 9, no. 1, pp. 99–104, Jan. 2016.
[26] J.-J. Guo, J.-F. Ma, X.-X. Guo, X.-H. Li, J.-W. Zhang, and T. Zhang, "Trust-based service composition and selection in service oriented architecture," *Peer-to-Peer Netw. Appl.*, vol. 11, no. 5, pp. 862–880, Sep. 2018.
[27] W. Li, J. Wu, Q. Zhang, K. Hu, and J. Li, "Trust-driven and QoS demand clustering analysis based cloud workflow scheduling strategies," *Cluster Comput.*, vol. 17, no. 3, pp. 1013–1030, Sep. 2014.
[28] W. Zhan and Q. Li, "An improved collaborative filtering based on a weighted network and triadic closure," in *Proc. IEEE 12th Intl Conf Ubiquitous Intell. Comput.*, Beijing, China, Aug. 2015, pp. 1760–1763.
[29] Y. Wang, I.-R. Chen, J.-H. Cho, A. Swami, and K. S. Chan, "Trust-based service composition and binding with multiple objective optimization in service-oriented mobile ad hoc networks," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 660–672, Jul. 2017.
[30] Y. Kim and K. G. Doh, "Use-case driven service modelling with XML-based tailoring for SOA," *Int. J. Web Grid Services*, vol. 9, no. 1, p. 35–38, 2013.
[31] X. Wu, B. Li, R. Song, C. Liu, and S. Qi, "Trust-based service composition and optimization," in *Proc. 19th Asia–Pacific Softw. Eng. Conf.*, Hong Kong, Dec. 2012, pp. 67–72.
[32] E. Al-Masri, "A quality-driven approach for ranking Web services," *Lect. Notes Electr. Eng.*, vol. 312, pp. 599–606, 2015.
[33] E. D. C. Bezerra, D. Lopes, and Z. Abdelouahab, "Dynamic Web service composition with MDE approaches and ontologies," in *Proc. 6th Int. Joint Conf. Comput., Inf., Syst. Sci., Eng., (CISSE)*, Bridgeport, CT, USA, Dec. 2010, vol. 151. Springer Verlag, pp. 661–675.
[34] Z. M. Aljazzaf, M. A. M. Capretz, and M. Perry, "Trust-based Service-Oriented Architecture," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 28, no. 4, pp. 470–480, 2016.
[35] WSDL4J. *IBM*. [Online]. Available: http://sourceforge.net/projects/wsdl4j
[36] E. Rudnicka, F. Bond, L. Grabowski, M. Piasecki, and T. Piotrowski, "Lexical perspective on Wordnet to Wordnet mapping," in *Proc. 9th Global WordNet Conf. (GWC)*, Singapore, Jan. 2018.
[37] F. Lécue, S. Salibi, P. Bron, and A. Moreau, "Semantic and syntactic data flow in Web service composition," in *Proc. IEEE Int. Conf. Web Services*, Singapore, Sep. 2008, pp. 211–218.
[38] J. Scicluna, N. Steinmetz, and M. Zaremba, "Service bundling with seekda! Dynamic shop," in *Information and Communication Technologies in Tourism*. Vienna, Austria: Springer, 2010, pp. 369–380.
[39] T. L. Saaty, "Decision making—The analytic hierarchy and network processes (AHP/ANP)," *J. Syst. Sci. Syst. Eng.*, vol. 13, no. 1, pp. 1–35, Mar. 2004.

CHUNLING HU received the Ph.D. degree of computer science from the Hefei University of Technology, in 2011. She is currently a Professor with the School of Artificial Intelligence and Big Data, Hefei University, China. Her research interests include web service, security, trust management, reliability and performance analysis, software repository mining, and intelligence analysis, and so on. She is a CCF and ACM member.



XIAONA WU received the master's degree from the School of Computer Science and Engineering, Southeast University, in 2017. She is currently a Teacher with Southeast University. Her main research interests include the trust evaluation of composite web service, software testing and verification, and so on.

BIXIN LI (Member, IEEE) received the Ph.D. degree of software engineering from Nanjing University, in 2001. He is currently a Full Professor with the School of Computer Science and Engineering. He is also the Director of the Software Engineering Institute, Southeast University (ISEU), China. His main research interests include web service, software analysis, testing and verification of complex system, evolving systems, and so on. He is a Senior CCF Member.