IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 27, 2019, accepted April 16, 2019, date of current version May 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913432

A Behavior Based Trustworthy Service Composition Discovery Approach in Cloud Environment

SHANCHEN PANG[1], (Member, IEEE), QIAN GAO[1], (Member, IEEE), TING LIU[2], (Member, IEEE), HUA HE[3], (Member, IEEE), GUANGQUAN XU[4], (Member, IEEE), AND KAITAI LIANG[5], (Member, IEEE)

[1]College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, China
[2]Weihai Science and Technology Bureau, Weihai 264200, China
[3]Mathematics and Statistics Department, Shandong University of Technology, Zibo 255000, China
[4]Tianjin Key Laboratory of Advanced Networking, College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
[5]Surrey Centre of Cyber Security, University of Surrey, Guildford GU2 7XH, U.K.

Corresponding author: Shanchen Pang (pangsc@upc.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61572522, Grant 61572523, Grant 61402533, and Grant 2015020031.

ABSTRACT With the development of cloud computing, more and more individuals and organizations have moved local application resources into the cloud computing resource pool in the form of Web services so that users can choose to invoke. This paper proposes a trustworthy service composition discovery approach to satisfy user requirements including behavioral constraints in a cloud environment. First, we build a global service composition discovery model based on sematic relationships between attributes and present a service composition discovery method. Then, we propose a method for detecting behavioral consistency. We validate the feasibility and effectiveness of the proposed solutions through a case study.

INDEX TERMS Cloud computing, service discovery, behavioral constraints, service behavior, service matching, web service.

I. INTRODUCTION

As a new computing method, cloud computing stores rich Web resources for users. With the expansion of the cloud computing resource pool, it is especially important to improve the resource utilization and meet users' needs. With the development of Internet and the popularity of Web services technology, Service Oriented Computing (SOC) and Service-Oriented Architecture (SOA) have become ~~an~~ efficient ways for automatically discovering and compositing Web services in a distributed, dynamic, and heterogeneous environment. From the perspective of science and technology research, to discovery comprehensive, timely, accurate Web services required by users and perform existing service combinations and integrations are of great significance for improving quality of service. Therefore, the study of trustworthy Web service composition discovery in cloud environment is important.

Service discovery is dependent on the user requirements in cloud computing. Web service discovery is the most basic technology in SOA, and it is a prerequisite for technical applications such as service selection and service composition. Currently, the user requirements considered in service discovery are mainly classified into two categories: 1) Functional requirements (and QoS requirements); 2) The execution process model for the given service composition. In fact, the above classifications of user requirements have certain limitations in real world applications. Users requirements are not limited to functional requirements, but often has behavioral constraints to the execution of service compositions. For example, in online shopping, the user may require cash on delivery or a specific courier delivery for safety reasons. Therefore, the users' requirements including only functional requirements can hardly describe users' behavioral requirements which can easily lead to inconsistency between

The associate editor coordinating the review of this manuscript and approving it for publication was Weizhi Meng.

service execution and user expectations, thereby it will reduce the trustworthiness of the service composition. Further, it is unrealistic in practice to require users to give the full service execution process model. For example, when shopping on ebay, to protect the interests of both parties, ebay as a third party designs a secure transaction mechanism. The buyers do not know and do not need to know what security mechanisms are and how process executes. Thus, it is impractical to require users to provide the data flow and control flow. How to find a solution to the web service portfolio optimization problem in the cloud computing environment combined with the advantages of cloud computing has important research value. Therefore, this paper proposes a service composition discovery approach is proposed to satisfy user requirements behavioral constraints in cloud computing.

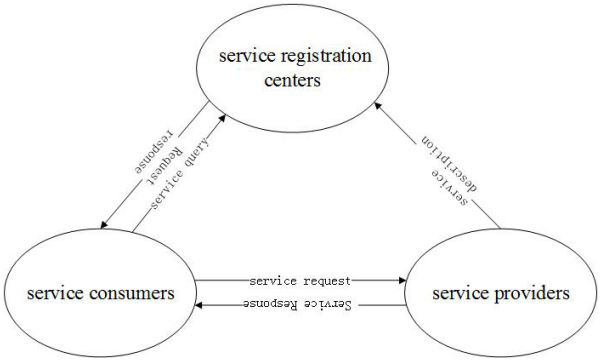


FIGURE 1. Architecture diagram of Web service.

II. RELATED WORK

According to the SOA architecture design specification, the architecture of Web services can be described as shown in Figure 1. The entire architecture consists of three parts: service providers, service consumers, and service registration centers. Service providers mainly use WSDL (Web Services Description Language) and other languages to describe specific services, and they are responsible for publishing Web services to service registration centers for service consumers to invoke. The service registry uses mechanisms such as UDDI (Universal Description, Discovery and Integration) to manage and register available Web service description information. At the same time, it is responsible for receiving query requests from service consumers. After retrieving suitable candidate services, the service registration center establishes a supply and demand relationship between service requesters and service providers. The service consumer, as a user of the Web service, makes an application request to the service registration center, returns the result of the search at the service registration center, and communicates with the service provider through SOAP (Simple Object Access Protocol) to complete the final service call.

Web service discovery is to find the set of service compositions using user requirements based service matching algorithms. To solve this problem, researchers have been working on this area from different perspectives. It can be seen that the description model of available services plays an important role in service discovery. A service in WSDL (Web Service Description Language) based on Web services description is viewed as a collection of operations while operations are mainly described by input and output parameters.

Therefore, this stage mainly discovers services via keyword based syntax matching according to functional requirements. For example, the UDDI (Universal Description, Discovery and Integration) based service discovery method, the vector space model based service discovery [1], and AI (Artificial Intelligence) based WSPR (Web Service Planner) method [2] discover available services based on functional requirements. However, with the increase of available services, more and more services with the same functionality but different QoS (Quality of Service) has been provided. To satisfy users' personalized requirements for QoS of service compositions, QoS information of services has been extended to the UDDI (e.g. UX [3]). QoS aware service discovery approaches have been proposed. In QoS based rich sematic WSDM [4], a Web service discovery method comprising function matching and QoS matching is proposed. In [5], a service discovery method is proposed to meet users' preferences based on functional requirements, transaction nature and QoS features.

Neither keyword syntax matching service discovery approaches nor QoS based service discovery approaches supports machine-readable Web service description. Thus, those approaches have low automation degree and low accuracy. Consequently, ontology is introduced to describe Web services, e.g., WSDL_S [6], WSMO (Web Services Modeling Ontology) [7] and OWL_S [8]. WSDL_S, an extension of WSDL, explains concepts by external ontology. The described sematic information includes the required preconditions, input, output and effects of Web service operations. WSMO is a WSML (Web Services Modeling Language) based Web service description model, which includes ontology, Web services, goals and mediators. OWL_S based Web service description model includes: service profile, process models and grounding. In [9], an ontology-based method is proposd to organize processes according to their functions, and reasoning rules are defined base on the ontology. In this method, a service can be determined whether it can satisfy a process by the vector space model and cosine similarity of function descriptions, and reasoning rules are employed to discover services which have no relationships with the process literally. In [10], a context based semantic Web service matching method is presented to solve the service discovery and sequencing problems. It is introduced in [11] that a matching algorithm for SAWSDL,which adapts and extends known concepts with novel strategies. Effective logic-based and syntactic strategies are introduced and combined in a novel hybrid strategy, targeting an envisioned well-defined, real-world scenario for matching. In [12], it is proposed a keyword syntax matching based sematic service discovery method, which first evaluates the similarity of the

Web service interfaces and then adds semantic annotations by WSDL description model to obtain SAWSDL (Semantic Annotations for WSDL) model. In [13], a high efficient service discovery method based on the inclusion relations and OWL ontology structural information is presented. In [14], a sematic Web service discovery method based on the bipartite graph is proposed. In [15], a semantic Web service fuzzy matching method by adding multiple fuzzy sets with multi-granularity information to OWL-S description model is proposed. In [16], it is believed that the previous OWL-S-based service matcher could not completely capture the functional information of the semantic service. There is a method that combines multiple matching criteria with user feedback to further improve the results of the matchmaker.

Semantic-aware Web service discovery method has increased the accuracy of service discovery. However, a Web service is viewed as a ''black box'' software, which neglects service implementation during service discovery and leads to inconsistency between composite service behaviors and behaviors expected by users, resulting reduced trustworthiness of service composition. It is discussed in [17] that the potential advantages of service activity information for service discovery and composition. In [18], a model is proposed for reliability prediction of atomic Web services that estimate the reliability for an ongoing service invocation based on the data assembled from previous invocations. In [19], there is an internal process model detection based service matching method. Based on service structure and function clustering. In [20], they propose a service discovery method according to functional requirements and process requirements. In [21], researchers present an OWL_S Web service description model, which first change s service process model into a tree, then builds service profile based BF dependency hypergraph, finally achieves service discovery through the graph search algorithm. In [22], they propose a service - oriented behavioral model, which solves service discovery problem by graph matching. And in [23], it is proposed that a similar graph based algorithm for discovering similar service operations and operation compositions. In [24], a service query algebra based composite service process model is proposed to select services based on functions and QoS. In [25], a path based tree matching method is designed to check behavioral consistency. In [26], a flow similarity based automatic service discovery method is proposed, which considers both static structure and dynamic behavior of services. In [27], a systematic approach is proposed to calculate QoS for composite services with complex structures, taking into consideration of the probability and conditions of each execution path. The above process based service discovery methods can be divided into two categories. The first one is to assume that a complete detailed composite service execution process model is given by users, and then design a suitable matching algorithm to find suitable services for service compositions. Such service discovery methods can be seen as a process of instantiating the abstract tasks in given requirement models. In fact, neither service consumers nor system designers can have a complete and clear understanding of service granularity and service behavior characteristics. Thus, it is not realistic to require a service composition execution process model. Another class of service discovery methods are to discover a single service and do not involve service compositions. Taking into account the degree of services reuse, the majority of services are relatively small size and with limited features. Therefore, it is difficult to discover a single service to meet most complex users' requirements. Obviously, the existing methods can hardly satisfy users' requirements including behavioral constraints.

In addition, Web service portfolio discovery is also closely related to the IoT (Internet of Things). With the development of IoT technology, IoT devices need to use cloud resources to expand business needs. The cloud platform uses virtualization technology to provide services for IoT devices, but there are some malicious attacks. In [28], an efficient E-AUA protocol for anonymous user authentication is proposed to resist attacks, which can further ensure the security of Web services for users in cloud computing. In [29], it is proposed a non-abuse contract signing agreement (CSP) for the energy consumption problem of the IoT inappropriate protocol, which guarantees fairness and further maintains the credibility and security of the Web service. In [30], it is proposed that a performance aware cost-effective resource provisioning for future grid IoT-cloud system.

The remainder of the paper is organized as follows. In section 3, a motivating example is introduced. In section 4, a trustworthy composite service discovery method for satisfying behavioral constraints is presented. In section 5, a service composition model based on sematic relationships between concepts is built. In section 6, it is presented that a composition planning method for satisfying requirements. In last section, we conclude the paper and validate the feasibility and effectiveness of the proposed solutions.

III. MOTIVATING EXAMPLE

An online book purchasing service as a motivating example to illustrate the user's requirements and composite service discovery process are presented in this section. The specific user requirements are as follows: a user wants to buy a book online, but do not know the exact book title. He/she only provides the keyword information which can determine the book, the required login user name and password (assume the user has the same login information for every online shopping websites), the bank card information including bank card number and password (the card does not have online payment feature. You must create a short-term credit card based on the bank card to make online payments), as well as delivery types and delivery addresses. In the meantime, for shopping safety, the user requires cash on delivery. All input messages, the desired output messages and behavioral constraints during the execution are described below:

Input message: {keyword, username, password, delivery_type, address, info_bank, card_number, card_password};

Output message:{confirmation};

Assuming that there are three related services in the Web service registry: ebay book purchasing service (EB_book_store), Zhuoyue book purchasing service (ZY_book_store) and online banking service (Online_bank). The input and output message sets are{keyword, username, password, name, email, c_card_type, valid_date, c_card_number, delivery_type, address} and {book_title, account_receipt, buy_receipt, confirmation} respectively. The input and output message sets are {book_title, username, password, name, email, delivery_type, address, cash, C_card_type, C_card_number, valid_date} and {price, account_receipt, confirmation} respectively. The input and output message sets of Online_baning service are{info_bank, card_number, card_password, amount_$, amount_¥} and {C_card_type, C_card_number, valid_date}.

To ease the description of composite service discovery, the process model for the above example is built in Petri net[31], [32]. The process models for EB_book_store, ZY_book_store and Online_bank are as follows.

IV. COMPOSITE WEB SERVISE DISCOVERY

The above is modeling the analysis of the instance to find the number of candidate services. With the increase in the number of Web services, the service portfolio cannot blindly search for a combination of solutions in a large-scale service collection of the service registration. You must first obtain a set of candidate Web service collections that can be used for grouping by functionality (such as the input, output, preconditions, and service effects of the service.), and this process requires service discovery technology to implement.

There are various behavioral constraints given by users. For the motivating example, the given behavioral constraint for transaction safety is cash on delivery; the given behavioral constraint for convenience of delivery checking is to deliver books by Shentong express delivery, the given behavioral constraint for user preference is not deliver by Yuantong express delivery. Thus, this paper considers three types of behavioral constraints: the relationship between the sequence of atomic services, the function that must be performed by a certain service (duty bound), and a function that cannot be performed by a service (separation of duties).

The user requirements are formalized as follows:

Definition 1 (user requirements) (I_r, O_r, B_r) represents the user requirements model, which

I_r denotes a set of user-supplied input messages;

O_r denotes a set of user desired output messages;

B_r denotes the behavioral constraints for service composition, and $B_r = \{B_r^s, B_r^+, B_r^-\}$ where B_r^s is the order constraint between behaviors, B_r^+ is the behavior binding constraint, B_r^- is the behavior separation constraint.

We only consider three types of behavioral constraints for user requirements. In practical applications, it can be extended if necessary while the constraint behavior detection method can be improved based on the new behavioral constraints. However, the composite service discovery approaches are not required to be modified.

We propose a Web service discovery method which takes into account functional properties and behavioral constraints. In the matching process, the proposed method not only considers the semantic information in the profile of Web services which contains the semantic relationships between functional

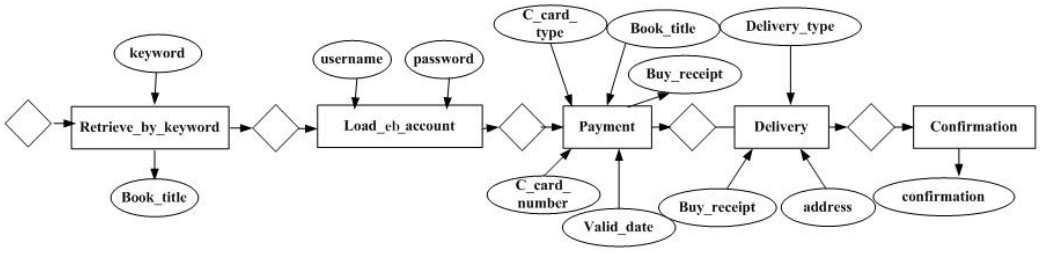


FIGURE 2. Process model of EB_book_store.

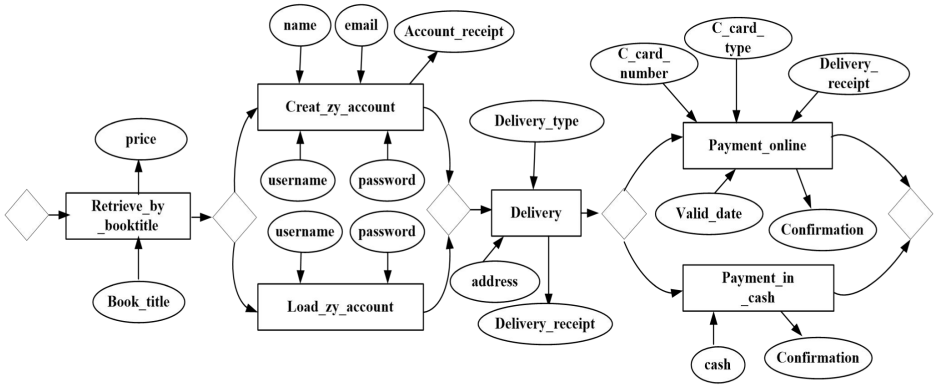


FIGURE 3. Process model of ZY_book_store.

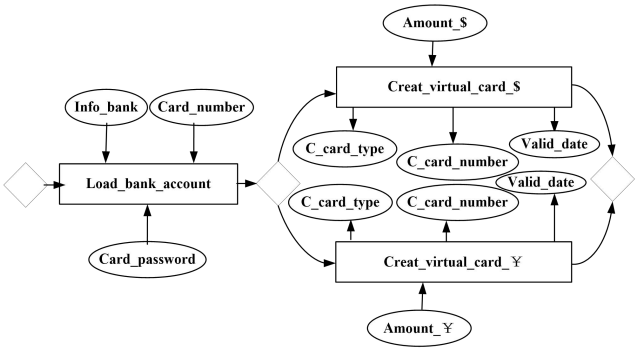


FIGURE 4. Process model of Online_bank.

attributes (the sub-concept is formally defined in [33]), but also requires the Web service execution behavior to be consistent with behavioral constraints. There are usually four situations in sematic concept matching: exact matching (exact), alternative matching (plug-in), included matching (subsume), and mismatching (fail). We only consider the exact matching relationships and alternative matching relationships for functional properties.

As it is difficult for the users to give all the required input messages before compositing services, while balancing the discovery rate and success rate, it is defined that a composite service matching succeeds if and only if one of the following conditions are satisfied:

At least one input message m is available, which is: there is a message m as the sub-concept or equivalent concept in the user input message or the output message of the matching services;

At least one output message n is available, which is: there is a message in the output message required by users or the input message of the matching service, so the message n is the sub-concept or equivalent concept.

According to the above definition of matching, a service which meets the functional requirements is required that at least one input message or an output message is useful. In fact, the message number of matching process can be dynamically adjusted based on the number of the service options, the number of service input messages and the number of output messages.

Function and behavior-oriented composite Web service discovery need to enter the following parameters : the service registry (Registry) for storing available Web services, including an input message set of a Web service (s.I), an output message set of a Web service (s.O), a process model of a Web service (s.PM); user queries (Q), including an input message set of a query (Q.I), a desired output message set of a query (Q.O) and behavioral constraints (Q.Br); the discovered available service alternatives (ServiceSet) which include services that meet the user requirements and must be performed, as well as the services that match successfully but are useless; available message set (message_available), including Q.I and the output message generated in the service execution process (initially include Q.I); the required produced message set (message_needed), including Q.O and the input messages required in the service execution process to produce Q.O (initially only include Q.O). The composite service discovery method is summarized as follows:

Algorithm 1 Service_Discovery (Registry,Q,message _available,message_needed,ServiceSet)

1. message_available=Q.I;
2. message_needed=Q.O;
3. $ServiceSet = \Phi$;
4. **For** $s \in Registry$
5. **If** $(\exists m \in message\_available, \exists i \in s.I : m \prec i \vee m \equiv i) \vee (\exists o \in s.O, \exists m \in message\_needed : o \prec m \vee o \equiv m)$
6. $ServiceSet = ServiceSet \cup \{s\}$;
7. $Registry = Registry - \{s\}$
8. **End if**
9. $message\_available = message\_available \cup s.O - Q.O$;
10. $message\_needed = message\_needed \cup s.I - message\_available - s.O$;
11. **End for**

The above service discovery algorithm is based on semantic relationships between property concepts, which concurrently matches the service based on the available message set for service matching and the required services to generate required messages. Using the ontologies referenced in the Web service and request description documents to find the number of candidate services. For the given user requirements and services in the service registry in the motivating example, the candidate service discovery result is {EB_book_store, ZY_book_store, Online_bank}.

V. BUILDING A GLOBAL COMPOSITE SERVICE MODEL

Service composition technology integrates loosely coupled Web services that are independent of each other into complex, value-added services. The goal of service composition is to improve the reusability and utilization of service components and basic services. On the basis of the composite service discovery, the method first determines whether the service composition can generate all output information desired by the user. If not, then the composition fails; Secondly, taking into account the syntax differences between service concepts and inexact matching, the method builds semantic relationships between concepts to obtain a global composite service model, Its core is to match semantic similarity to candidate services. In order to get all service alternatives that satisfies user requirements.

First, the method determines whether or not all the user desired output messages can be produced by the service in the composite service set. If they can produce, then the method determines the service set for each desired output message. The determination method is summarized as follows:

According to whether the Web service information is processed, Web services are divided into two categories. One is read-only Web services. After service execution, the message

Algorithm 2 Q_Generated_OrNot (Q, ServiceSet, LeafSet)

1. $LeafSet = \Phi$;
2. **For** $\forall m \in Q.O$
3. $LeafSet[m] = \Phi$; *//LeafSet[m] represents the atomic process set producing m*
4. **For** $\forall s \in ServiceSet$
5. **If** $\exists o \in s.O : o \prec m \vee o \equiv m$
6. $LeafSet[m] = LeafSet[m] \cup \{s.ap[m]\}$; *//s.ap[m] represents the atomic process of service s which outputs the message m*
7. **End if**
8. **End for**
9. **If** $LeafSet[m] = \Phi$
10. Return "composition fails for there is one parameter in Q.O which can not been generated by any service in the registry";
11. **Break**;
12. **End if**;
13. $LeafSet = LeafSet \cup LeafSet[m]$;
14. **End for**

does not change, and the message does not consume after being used. Another one is message-processing Web service. After service execution, the message is changed and may be consumed after being used. In the paper, we only consider read-only Web services. The user given input messages and generated messages are reusable.

The built global composite service model is built as below in order to determine the alternative service composition that satisfies the user requirements. In the global composite service model building process, one needs to first consider the concept semantic relationship between the input messages given by the user and input messages of the service composition. Secondly, one needs to consider the concept semantic relationships between the output messages and input messages, and adds semantic transactions for the messages which have equivalence relations. Thirdly, one needs to add a root control placement for a token, and adds control transactions between the root control placement and the composite service placement.

The process for building the global service composition model can be divided into five steps.

Step 1: Input parameter set.

Step 2: The method determines whether or not all the user desired output messages (*LeafSet*=(m1, m2, ……, mn)) can be produced by the service in the composite service set (*ServiceSet*=(s1, s2, ……, sn)), that is,

$$LeafSet = LeafSet \cup ServiceSet$$

Step 3: Add the service composition have been found (*ServiceSet*=(s1, s2, ……, sn)) to the global composite service model (*Dependency_PN*=(P,T;F)). In the meantime, the algorithm expresses user input (Q.I) and output messages (Q.O) as message placements and adds the placements to the model. Each input message placement has one token. To ease identification, the placement with the "slash" shading represents the placement for the input message given by users, while the placement with "grid" shading represents the output messages desired by users. that is,

$$Dependency\_PN = ServiceSet \cup Dependency\_PN$$
$$Dependency\_PN = Q.I \cup Q.O \cup Denpendency\_PN$$

Step 4: The algorithm mainly considers the semantic relationships among three placements. Which adds necessary semantic transactions between Q.I and input message placement according to the semantic concept relationships. Then, the algorithm adds necessary semantic transactions between Q.O and output message placement according to semantic concept relationships.

$$\begin{aligned} Post\_place &= Post\_place \cup s.I \\ Dependency\_PN &= Dependency\_PN \cup t \\ {}^{\bullet}t = Q.I \wedge t^{\bullet} &= Post\_place \\ Pre\_place &= Pre\_place \cup s.O \\ {}^{\bullet}T = Pre\_place \wedge T^{\bullet} &= Q.O Dependency\_PN \\ &= Dependency\_PN \cup T \end{aligned}$$

Next, the algorithm adds necessary semantic transactions between the output message placement and input message placement.

Step 5: The algorithm adds a root control placement (root) and a control transaction in the global composite service model (Dependency_PN = (P,T;F). Each control placement has one token (t). The input placement of the control transaction is the root control placement. The output placement is the start placement. that is,

$$\begin{aligned} PN_s &= (P_s, T_s, F_s, M_{0s}) \\ P_s &= P_{si} \cup P_{sp} \cup P_{sc} \\ Dependency\_PN &= Dependency\_PN \cup t \\ b \in P_{sc} \wedge {}^{\bullet}b & \\ {}^{\bullet}t = \{root\} \wedge t^{\bullet} &= \{b\} \end{aligned}$$

Based on the above building algorithm, the produced global candidate composite service model is shown in Figure. 5.

The algorithm for constructing the global composite service model is summarized as follows:

VI. COMPOSITION PLANNIG

Definition 2 (A predecessor Service) service S_1 is called the predecessor service of S_2, if and only if : $\exists o \in s_1.O, \exists i \in s_2.I : o \prec i \vee o \equiv i$

If S_1 is the predecessor service of S_2, then it denotes $predecessorService(s_2) = \{s_1\}$.

If S_1 is the predecessor service of S_2 and S_2 is the predecessor services of S_1 then S_1, and S_2 are mutually predecessor services.

Definition 3 (A parallel predecessor services of a message) $s, s_i (1 \leq i \leq n)$ is a service, if $\exists o_i \in s_i.O, o_i \prec i \vee o_i \equiv i$ is established, then $s_i (1 \leq i \leq n)$ is the parallel predecessor

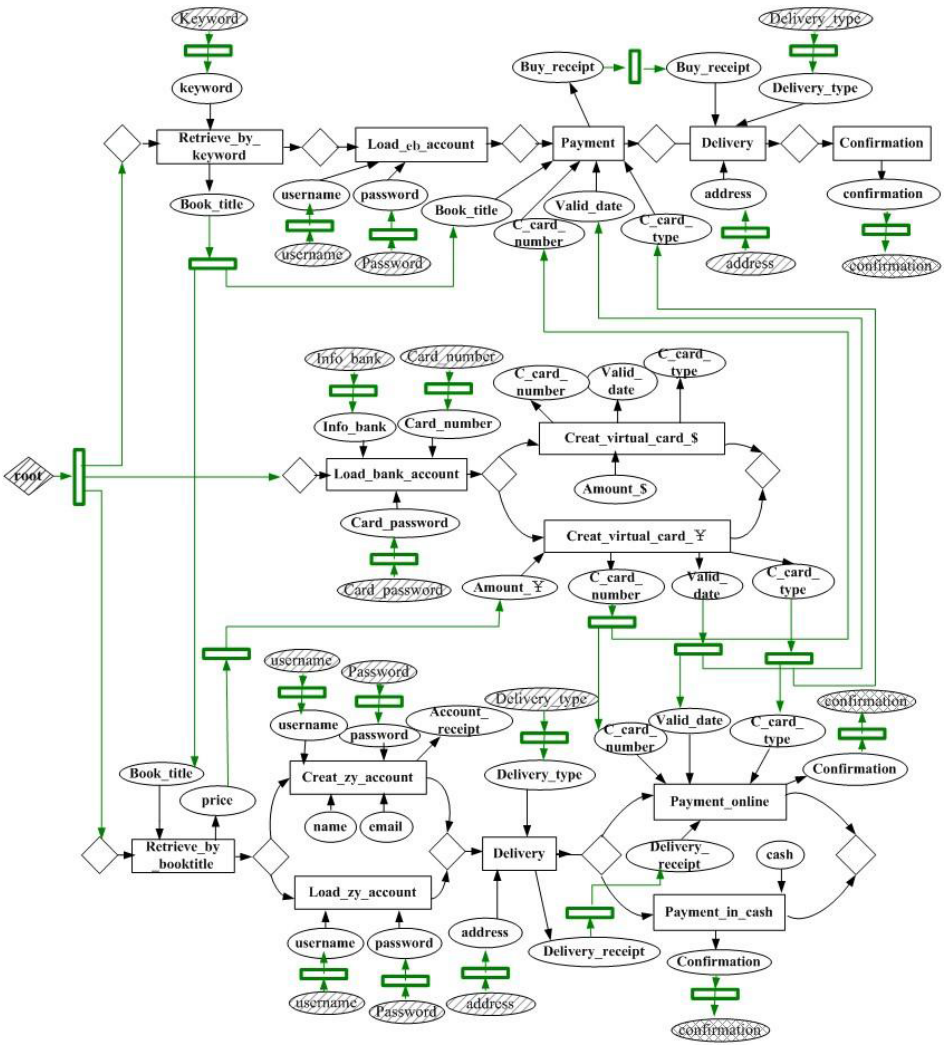


FIGURE 5. Global composite service model.

service for input message I, which is denoted as $PPM(i) = \{s_i/1 \leq i \leq n\}$.

Definition 4 (A parallel predecessor service set $s.I$ for the message set) s is a service, is the para $PPM(i_k)$ llel service set for input message $i_{k,ik} \in s.I$ and $1 \leq k \leq n$, then $PPI(s.I) = \{\{s_1, s_2 \cdots s_n\}/s_k \in PPM(i_k) \wedge (1 \leq k \leq n)\}$ is the parallel predecessor service set for message set of s.

Definition 5 (A parallel predecessor service set of the service set) $S = \{s_i/1 \leq i \leq n\}$ is the service set and $PPI(s_i.I)$ is the parallel predecessor service set of input message set $s_i.I$ of s_i, then $PPS(S) = \{S_1 \cup S_2 \cup \cdots \cup S_n/S_k \in PPI(s_k.I) \wedge (1 \leq k \leq n)\}$ is the parallel predecessor service set of S.

Definition 6 (A projection operation) For the petri net $PN = (P, T, F, M_0)$ and the transaction set Tr, $project(PN, Tr) = (P', T', F', M')$ is the projection of PN on Tr, wherein:

$\bullet P' = \{p/p \in {}^\bullet Tr \cup Tr^\bullet\} \cup root(PN)$;

$\bullet T' = Tr \cup T_s \cup T_c$,

$T_s = \{t/(t \in T) \wedge (\bullet t \subseteq P) \wedge (t^\bullet \cap P' \neq \Phi)\}$,

$T_c = \{t/t = root(PN)\}$;

$\bullet F' = \{f/f \in (P' \times F' \cup F' \times P') \cap F\}$;

• For $p \in P'$, $M'(p) = M_0(P)$.

The detailed composite service set discovery method is summarized as follows:

As shown in line 2-9, the algorithm first determines the service set $Q.O.LeafServiceSet$ which can directly generate the output messages desired by users according to $LeafSet$. As the output messages generated by each composite service may have inclusion relationships, there may be redundant composite service set in $Q.O.LeafServiceSet$. The algorithm deletes the redundancy (line 4-10) and traverses each composite service in composite service set until the root placements in the global composite service model (line 10-30) to extend each element in $Q.O.LeafServiceSet$ to a service set that satisfies user requirements. To further analyse if the functions of each composite service are aligned with user requirements, it requires to generate composite service process model (see line 32-36 for details).In figure. 6-8, it is showed the three alternative service compositions calculated by the algorithm.

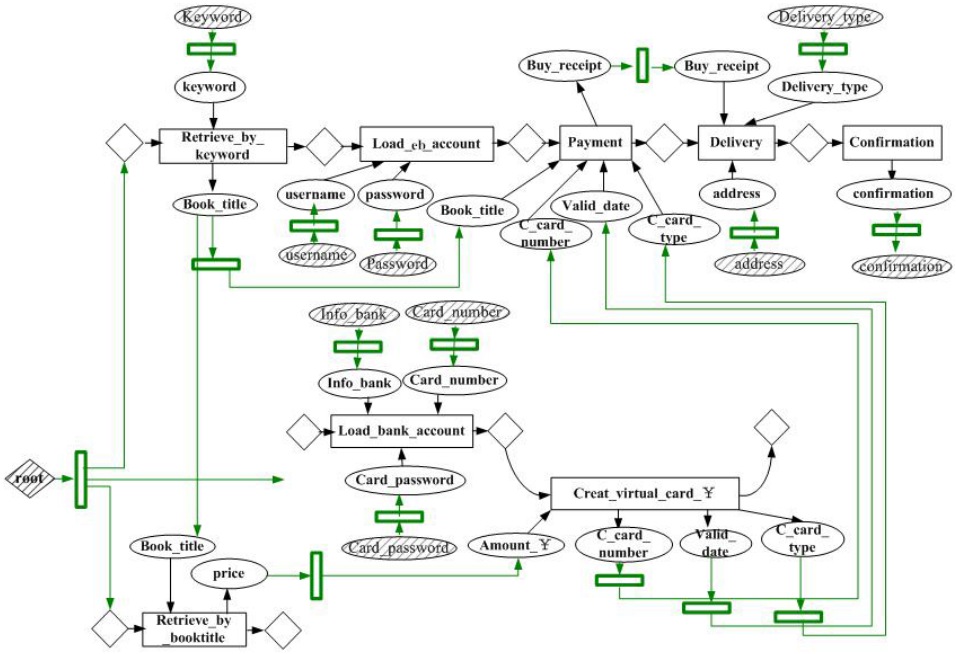


FIGURE 6. Possible composite service 1.

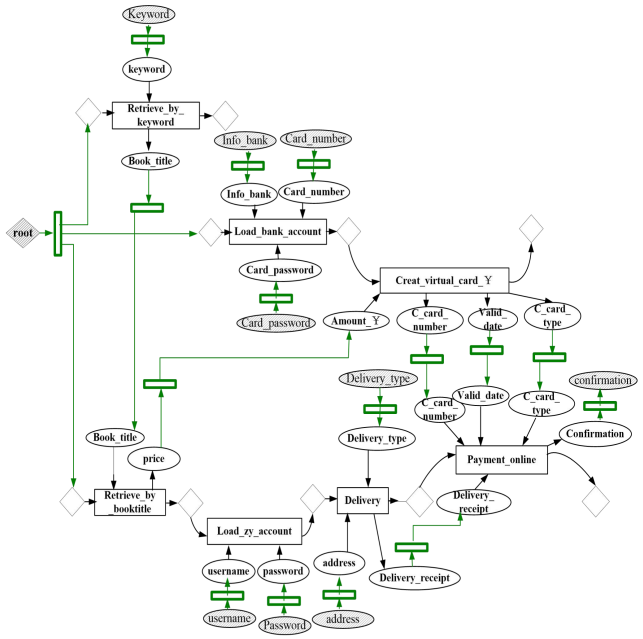


FIGURE 7. Possible composite service 2.

In order to determine whether the discovered service composition meets user requirements, the process model needs to perform further analysis. According to [24], based on the input service composition process model PM and user behavioral constraint model $B_r = \{B_r^s, B_r^+, B_r^-\}$, the detailed method is summarized as follows.

There is a constraint that is cash on delivery. We can see the service composition in Figure.6 does not satisfy the user behavioral constraint as the execution sequence of Payment and Delivery is not consistent with the behavioral constraint.

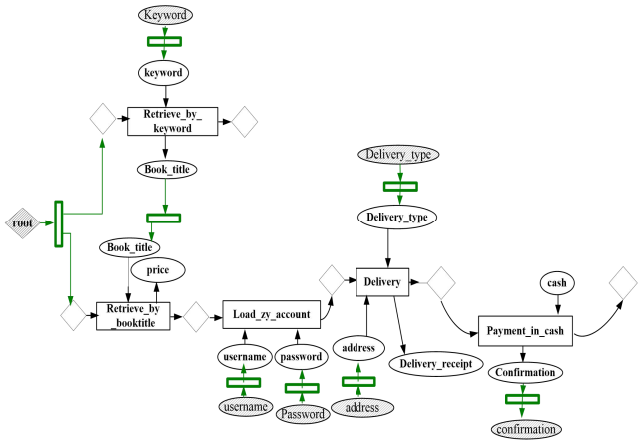


FIGURE 8. Possible composite service 3.

The service compositions in Figure.7 and Figure.8 satisfy the given behavioral constraint.

The service composition which satisfies behavioral constraint may not be executed successfully due to lack of input messages. Thus, it requires to functional validation. The detailed method is summarized as follows.

The service composition shown in Figure. 7 can be executed successfully, while the service composition shown in Figure. 8 is failed due to the lack of input message cash and needs to supply input message by interaction. The core of this part is to eliminate logically inferences that the input and output parameters do not logically meet the requirements or not satisfy the user behavioral constraint.

The matching objects considered by the aforementioned method in their service matching are almost all parameters such as the input, output, preconditions, and effectiveness

Algorithm 3 Creat_Dependency_PN (ServiceSet, Q, Dependency_PN)

1. **For** $\forall s \in ServiceSet$
2. Add s to Dependency_PN;
3. **End for**
4. **For** $\forall m \in Q.I \cup Q.O$
5. Add a message place p_i or p_o to Dependency_PN, and $M(p_i) = 1$;
6. **End for**
7. //Add semantic transition between Q.I and s.I
8. **For** $\forall m \in Q.I$
9. $Post\_place = \Phi$;
10. **For** $\forall s \in ServiceSet$
11. **If** $(\exists i \in s.I : m \prec i \vee m \equiv i)$
12. $Post\_place = Post\_place \cup \{i\}$;
13. **End if**
14. **End for**
15. **If** $|Post\_place| \geq 1$
16. Add t to Dependency_PN, and ${}^\bullet t = \{m\} \wedge t^\bullet = Post\_place$;
17. **End if**
18. **End for**
19. //Add semantic transition between s.O and Q.O
20. **For** $\forall m \in Q.O$
21. $pre\_place = \Phi$;
22. **For** $\forall s \in ServiceSet$
23. **If** $(\exists o \in s.O : o \prec m \vee o \equiv m)$
24. $pre\_place = pre\_place \cup \{o\}$;
25. **End if**
26. **End for**
27. **If** $|pre\_place| \geq 1$
28. Add T to Dependency_PN, and ${}^\bullet T = pre\_place \wedge T^\bullet = \{m\}$;
29. **End if**
30. **End for**
31. //Add semantic transition between s.I and s.O
32. **For** $\forall s_1 \in ServiceSet$
33. **For** $\forall o \in s_1.O$
34. $post\_place = \Phi$
35. **For** $\forall s_2 \in ServiceSet - \{s_1\}$
36. **For** $\forall i \in s_2.I$
37. **If** $o \prec i \vee o \equiv i$
38. $post\_place = post\_place \cup \{i\}$;
39. **End if**
40. **End for**
41. **End for**
42. **If** $|post\_place| \geq 1$
43. Add t to Dependency_PN, and ${}^\bullet t = \{o\} \wedge t^\bullet = post\_place$;
44. **End if**
45. **End for**
46. **End for**
47. //Add control transition between control places
48. Add control place *root* to Dependency_PN and $M(root) = 1$;

Algorithm 3 *(Continued.)* Creat_Dependency_PN (ServiceSet, Q, Dependency_PN)

49. **For** $\forall s \in ServiceSet$
50. $PN_s = (P_s, T_s, F_s, M_{0s})$ is the improved open Petri net of service s, and $P_s = P_{si} \cup P_{sp} \cup P_{sc}$;
51. $b \in P_{sc} \wedge {}^\bullet b = \Phi$;
52. Add control transition t to Dependency_PN, and ${}^\bullet t = \{root\} \wedge t^\bullet = \{b\}$
53. **End for**

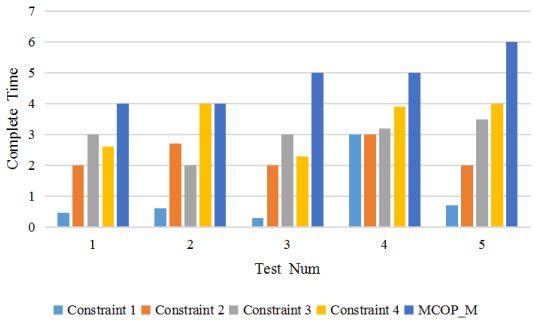


FIGURE 9. Tests results on completing time.

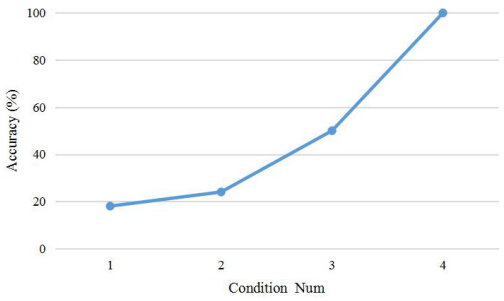


FIGURE 10. Tests results on accuracy.

of the service. These parameters essentially belong to the functional attributes of the candidate service. When there are many services, if only the functional attributes of the service are used as matching objects in the service discovery process, a group of candidate services with similar functions will often appear in the results. Due to the lack of parameter basis outside the functional attributes, it is generally difficult to select the best one among them. Therefore, it is a good choice to consider non-functional attributes to implement Web service discovery in addition to functional attributes. Service quality (QoS)-based service discovery becomes an important basis for finding and selecting candidate services.

In order to verify the feasibility and effectiveness of the proposed method, satisfy the user requirements, a set of tests have been conducted based on three services and 14 atomic services in the paper. Compared with the algorithm —— MCOP_M (Multi-Constrained Optimal Path problem for Multistage graph) proposed in [33], the completing time and the accuracy of the composition all vary with the number of constraints, and the results are shown in figure. 9 - figure. 10.

Algorithm 4 Determine_All_Possible_CompositeServiceSet (LeafSet, ServiceSet, Dependency_PN, Q, CompositeService)

1. *// Determine all possible composite service*
2. *//For each atomic process set which can produce all messages in Q.O*
3.

$$Q.O.LeafServiceSet = \{LSS_i/(1 \le i \le \prod_{m \in Q.O} |LeafSet[m]|) \wedge (LSS_i = \{s_j/(1 \le j \le |Q.O|) \wedge (\forall m \in Q.O : \exists s \in LeafSet[m] \wedge (s = s_j))\})\}$$

4. //Delete the duplicate LSS_i
5. **For** $\forall LSS_i, LSS_j \in Q.O.LeafServiceSet$
6. If $LSS_i \subseteq LSS_j$ Global composite service model consists of a set of service composition including useless services (e.g., input matching is successful, but it does not produce a service with useful output) and services that produce the same and/or partly same messages. There may be multiple alternative service compositions which satisfy user requirements but have different behaviors. Thus, the model first determines the service set which can directly produce output messages desired by users, then tranverse its predecessor services until root services to determine the service compositions that satisfy user requirements. The related concepts are defined as follows:
7. $Q.O.LeafService = Q.O.LeafService - \{LLS_j\}$;
8. **End if**
9. **End for**
10. *//Determine all possible composite service sets*
11. $CompositeServiceSet = \Phi$;
12. **For** $\forall LSS \in Q.O.LeafServiceSet$
13. *//the following function is used to determine all composite service set CS corresponding to the LeafServiceSet LLS,*
14. $CS = LLS$;
15. $COMPOSITESERVICESET(LLS, CS)$
16. **If** $LLS \neq \{root(Dependency_PN)\}$
17. Compute $PPS(LLS)$;
18. *//Delete those redundancy service set to make each service set in PPS only include those service must being executed.*
19. **For** $\forall S_1, S_2 \in PPS(LLS)$
20. **If** $S_1 \subseteq S_2$
21. $PPS(LLS) = PPS(LLS) - \{S_2\}$
22. **End if**
23. **End for**
24. **For** $\forall Set \in PPS(LLS)$
25. $COMPOSITESERVICESET(Set, CS \cup Set)$;
26. **End for**
27. **Else**

Algorithm 4 *Continued.* Determine_All_Possible_CompositeServiceSet (LeafSet, ServiceSet, Dependency_PN, Q, CompositeService)

28. $CompositeServiceSet = CompositeServiceSet \cup \{CS\}$;
29. **End if**
30. **End for**
31. *//Determine all possible composite service Petri net*
32. $CompositeService = \Phi$;
33. **For** $\forall CSS \in ComponentServiceSet$
34. $CompositeService(CSS) = project(Dependency_PN, CSS)$;
35. $CompositeService = CompositeService \cup \{CompositeService(CSS)\}$;
36. **Endfor**

Algorithm 5 Satisfy (PM, B_r)

1. Compute $L(PM)$, $L(B_r^s)$, $L(B_r^+)$, and $L(B_r^-)$;
2. *//L(PM) represent the language set generated based on the process model PM.*
3. **For** $l_1 \in L(PM)$
4. **For** $l_2 \in L(B_r^s)$
5. *//*$T(l_1)$ *is the transition set consisting of the transitions in* l_1;
6. **If** $T(l_1) \supseteq T(l_2) \&\& \Gamma_{l_1 \to T(l_2)} \neq l_2$
7. Return FALSE;
8. **End if**
9. **End for**
10. **For** $l_3 \in L(B_r^+)$
11. **If** $T(l_1) \cap T(l_3) \neq T(l_3)$
12. Return FALSE;
13. **End if**
14. **End for**
15. **For** $l_4 \in L(B_r^-)$
16. **If** $T(l_1) \supseteq T(l_4)$
17. Return FALSE;
18. **End if**
19. **End for**
20. **End for**

To evaluate the superiority of our algorithm, we compare the completion time of MCOP_M and the algorithm presented in this paper when increasing the number of constraints. It is showed in figure.9 that the changes of the composition completing time before and after changing the number of constraints. It can be seen that the values of time is short and stable to every case, and it can also be seen that the time gets a little larger with the rise in the number of constraints. But the comleting time of the algorithm presented in this paper increased very slowly compared with MCOP_M. Figure.10 shows the changes of the accuracy with the number of constraints. It can be seen that the accuracy rises significantly while the number of constraints becomes larger. That is, it costs a little time to improve the accuracy significantly, and the accuracy can reach one hundred percent finally.

Algorithm 6 //Determine the Composite Service CS Is Absent of Input Parameters or Not. Determine_Absent_Para (CS, AdditionalPara)

1. *// CS is the process model of composite service.*
2. Assume APS is the process set of composite service CS=(P,T,F);
3. $TotalInputPara = \Phi$;
4. **For** $\forall ap \in APS$
5. $TotalInputPara = TotalInputPara \cup \text{ap}.I$;
6. $APS = APS - \{ap\}$;
7. **End for**
8. **For** $\forall m \in TotalInputPara$
9. **If** $\exists t \in T : m \in t^{\bullet}$
10. $TotalInputPara = TotalInputPara - \{m\}$;
11. **End if**
12. **End for**
13. $AdditionalPara = TotalInputPara - Q.I$
14. **If** $AdditionalPara = \Phi$
15. Return TRUE;
16. **Else**
17. Return *AdditionalPara*;
18. **End if**

VII. CONCLUSION

In this paper, we proposes a behavior based service oriented process model in cloud environment. First, we designed a behavior based composite service discovery method is built. Then the paper built a global service composition model based on semantic relations between property concepts and a service composition planning method. Finally, a behavioral consistency detection method and validated feasibility and effectiveness of the proposed method from the functional perspective through a motivating example are presented. The proposed service discovery method effectively improves the user expected behavior and consistency of service composition, so that the service composition discovered by the proposed method in cloud computing is more trustworthy. In our future work, we aim at further improving the service combination method. This will make our approach more practical and effective.

REFERENCES

[1] Y. N. Hao, Y. C. Zhang, and J. L. Cao, "Web services discovery and rank: An information retrieval approach," *Future Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1053–1062, Apr. 2010.
[2] S.-C. Oh, D. Lee, and S. R. Kumara, "Web service planner (WSPR): An effective and scalable Web service composition algorithm," *Int. J. Web Services Res.*, vol. 4, no. 1, pp. 1–22, Jan. 2007.
[3] C. Zhou, L. T. Chia, and B. S. Lee, "QoS-aware and federated enhancement for UDDI," *Int. J. Web Service Res.*, vol. 1, no. 2, pp. 58–85, Apr. 2004.
[4] K. Kritikos and D. Plexousakis, "Requirements for QoS-based web service description and discovery," *IEEE Trans. Services Comput.*, vol. 2, no. 4, pp. 320–337, Oct./Dec. 2009.
[5] J. E. Haddad, M. Manouvrier, and M. Rukoz, "TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition," *IEEE Trans. Services Comput.*, vol. 3, no. 1, pp. 73–85, Jan./Mar. 2010.
[6] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller, "Adding semantics to web services standards," in *Proc. 1st Int. Conf. Web Services*, Jun. 2003, pp. 23–26.
[7] S. Sandhya, P. Pabitha, and M. Rajaram, "Enhanced semantic Web service discovery using machine learning on mapped WSMO services," *Int. J. Eng. Technol.*, vol. 6, no. 2, pp. 982–991, May 2014.
[8] D. Martin *et al.*, "Bringing semantics to Web services with OWL-S," *World Wide Web*, vol. 10, no. 3, pp. 243–277, Sep. 2007.
[9] R. Li, K. He, and S. Wang, "An ontology-based process description and reasoning approach for service discovery," in *Proc. Int. Conf. Comput. Sci. Netw. Technol.*, Dalian, China, Oct. 2014, pp. 320–325.
[10] A. Segev and E. Toch, "Context-based matching and ranking of Web services for composition," *IEEE Trans. Services Comput.*, vol. 2, no. 3, pp. 210–222, Jul./Sep. 2009.
[11] P. Plebani and B. Pernici, "URBE: Web service retrieval based on similarity evaluation," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 11, pp. 1629–1642, Nov. 2009.
[12] T. G. Stavropoulos, S. Andreadis, N. Bassiliades, D. Vrakas, and I. Vlahavas, "The tomaco hybrid matching framework for SAWSDL semantic Web services," *IEEE Trans. Services Comput.*, vol. 9, no. 6, pp. 954–967, Nov./Dec. 2015.
[13] G. Meditskos and N. Bassiliades, "Structural and role-oriented Web service discovery with taxonomies in OWL-S," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 2, pp. 278–290, Feb. 2010.
[14] S. G. Deng, J. W. Yin, Y. Li, J. Wu, and Z. Wu, "A method of semantic Web service discovery based on Bipartite graph matching," *Chin. J. Comput.*, vol. 31, no. 8, pp. 1364–1375, Nov. 2008.
[15] G. Fenza, V. Loia, and S. Senatore, "A hybrid approach to semantic Web services matchmaking," *Int. J. Approx. Reasoning*, vol. 48, no. 3, pp. 808–828, Aug. 2008.
[16] A. Averbakh, D. Krause, and D. Skoutas, "Exploiting user feedback to improve semantic Web service discovery," in *Proc. Int. Semantic Web Conf. (ISWC)*. Berlin, Germany: Springer, Oct. 2009, pp. 33–48.
[17] A. Brogi, "On the potential advantages of exploiting behavioural information for contract-based service discovery and composition," *J. Logic Algebr. Program.*, vol. 80, no. 1, pp. 3–12, Jan. 2011.
[18] M. Silic, G. Delac, and S. Srbljic, "Prediction of atomic Web services reliability for QoS-aware recommendation," *IEEE Trans. Services Comput.*, vol. 8, no. 3, pp. 425–438, May/Jun. 2015.
[19] A. Günay and P. Yolum, "Service matchmaking revisited: An approach based on model checking," *J. Web Semantics*, vol. 8, no. 4, pp. 292–309, Nov. 2010.
[20] Y. M. Wang, Z. Yingjun, and X. Binhong, "Semantic Web service discovery based on fuzzy clustering optimization," *Comput. Eng.*, vol. 39, no. 1, pp. 219–223, Jul. 2013.
[21] A. Brogi and S. Corfini, and R. Popescu, "Semantics-based composition-oriented discovery of Web services," *ACM Trans. Internet Technol.*, vol. 8, no. 4, p. 19, Sep. 2008.
[22] D. Grigori, J. C. Corrales, M. Bouzeghoub, and A. Gater, "Ranking BPEL processes for service discovery," *IEEE Trans. Services Comput.*, vol. 3, no. 3, pp. 178–192, Jul./Sep. 2010.
[23] X. Z. Liu, G. Huang, and H. Mei, "Discovering homogeneous Web service community in the user-centric Web environment," *IEEE Trans. Services Comput.*, vol. 2, no. 2, pp. 195–202, Apr./Jun. 2009.
[24] Q. Yu and A. Bouguettaya, "Framework for Web service query algebra and optimization," *ACM Trans. Web*, vol. 2, no. 1, p. 6, Feb. 2008.
[25] Y. Dai, X. W. Hao, and L. Yang, "Service discovery for composition process through matching of behavioral consistency," *J. Northeastern Univ.*, vol. 29, no. 10, pp. 1410–1413, Oct. 2008.
[26] Z.-C. Huang, J.-P. Huai, X. D. Liu, X. Li, and J. J. Zhu, "Automatic service discovery framework based on business process similarity," *J. Softw.*, vol. 23, no. 3, pp. 489–503, Mar. 2012.
[27] H. Y. Zheng, W. Zhao, J. Yang, and A. Bouguettaya, "QoS analysis for Web service compositions with complex structures," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 373–386, Jul./Sep. 2013.
[28] X. J. Zeng, G. Xu, X. Zheng, Y. Xiang, and W. Zhou "E-AUA: An efficient anonymous user authentication protocol for mobile IoT," *IEEE Internet Things J.*, to be published.
[29] G. Q. Xu *et al.*, "CSP-E^2: An abuse-free contract signing protocol with low-storage TTP for energy-efficient electronic transaction ecosystems," *Inf. Sci.*, vol. 476, pp. 505–515, Feb. 2019.
[30] W. L. Li *et al.*, "Performance-aware cost-effective resource provisioning for future grid IoT-cloud system," *J. Energy Eng.*, to be published.
[31] C. J. Jiang, *Petri Net, Theory and Applications*. Beijing, China: Higher Education Press, 2003, pp. 15–35.

[32] S. Pang, W. Zhang, T. Ma, and Q. Gao, "Ant colony optimization algorithm to dynamic energy management in cloud data center," *Math. Problems Eng.*, vol. 2017, no. 3, Jul. 2017, Art. no. 4810514.
[33] D. D. Wang, "Study on Web service discovery and composition in cloud computing," Ph.D. dissertation, School Comput. Commun. Eng., Univ. Sci. Technol., Beijing, China, 2017.

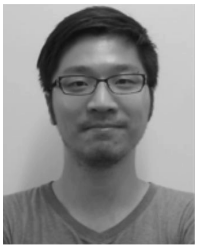
SHANCHEN PANG received the B.E., M.E., and Ph.D. degrees from Tongji University, Shanghai, China. He is currently a Professor with the School of Computer and Communication Engineering, China University of Petroleum, Qingdao, China. His current research interests include Petri nets, service engineering, and formal language theory.

QIAN GAO is currently pursuing the M.S. degree with the College of Computer and Communication Engineering, China University of Petroleum (East China). Her current research interests include web service, cloud computing, and machine learning.

TING LIU received the M.S. degree from the Shandong University of Science and Technology, Shandong, China, 2015. Her current research interests include Petri nets, software quality, performance evaluation, and cloud computing system dependability.

HUA HE, photograph and biography not available at the time of publication.

GUANGQUAN XU received the Ph.D. degree from Tianjin University, in 2008. He is currently a Ph.D. Supervisor and a Full Professor with the Tianjin Key Laboratory of Advanced Networking (TANK), College of Intelligence and Computing, Tianjin University, China. His research interests include cyber security and trust management. He is a member of CCF.



KAITAI LIANG received the Ph.D. degree in computer science (applied cryptography direction) from the City University of Hong Kong, in 2014. He was a Lecturer with the School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University; a Postdoctoral Researcher with the Department of Computer Science, Aalto University, Finalnd; a Visiting Scholar with the Department of Computer Science, UCL; a Visiting Scholar with KU Leuven, Belgium, the Sapienza University of Rome, Italy, and the University of Wollongong, Australia; and a Research Intern with the Institute for Infocomm Research, Singapore. He is currently an Assistant Professor in secure systems with the University of Surrey, U.K., and also a member of the Surrey Centre for Cyber Security and a GCHQ recognized UK Academic Centre of Excellence in Cyber Security Research. He has been involved (as CI and PI) in several European funded projects, e.g., SPEAR, SECONDO, CUREX, and Academic of Finland. His current research interests include data security, user privacy, cybersecurity, blockchain security, and privacy-enhancing technology.