Constraint-Aware Approach to Web Service Composition

PengWei Wang, ZhiJun Ding, ChangJun Jiang, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—The creation of value-added services by automatic composition of existing ones is gaining significant momentum as the potential silver bullet in service-oriented computing. A large number of composition methods have been proposed, and most of them are based on the matching of input and output parameters of services only. However, most services in the real world are not universally applicable, and some applicable conditions or restrictions are imposed on them by their providers. Such constraints have a great impact on service composition, but have been largely ignored by the existing methods. In this paper, they are discussed and defined, and a simple formal expression is adopted to describe them. Two novel concepts, called service intension and service extension, are presented, which allow one to divide the basic elements of a web service definition into two parts. Consequently, their use allows us to propose a constraint-aware service composition method in which service constraints are well taken care. The proposed solution includes a graph search-based algorithm and two novel preprocessing methods. A publicly available test set from ICEBE05 is used to evaluate and analyze the proposed methodology.**

***Index Terms*—Service composition, service constraint, service extension, service intension, web services.**

I. Introduction

SERVICE-ORIENTED COMPUTING (SOC) is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost, interoperable, evolvable, and easy composition of distributed applications even in heterogeneous environments. It promotes the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes, and agile applications that span organizations and computing platforms [33], [34]. Service-oriented architecture (SOA) is its fundamental architectural model that supports it from the architecture perspective [61]. Therefore, SOC is a computing paradigm and SOA is its conceptual-level architectural model, which means that they need to be implemented by some specific technologies. Web service technology is such one that is widely accepted and very promising. Web services are well-defined, self-contained, platform-independent, reusable modules that provide standard business functionality. They can be published, discovered, located, invoked, and loosely coupled across the web, and facilitate the integration of newly built and legacy applications both within and across organizational boundaries [2]. With the emergence and development of SOA and web services technology, more and more companies and organizations expose their business applications in the manner of well-defined services. However, to improve the reusability and simplify the application logic, atomic web services are often simple, and can only provide limited functionality, which cannot always satisfy the personalized and diversified needs of users and appropriately reflect the intricate and flexible business processes. Thus, the ability to efficiently and effectively select and integrate interorganizational and heterogeneous services is important toward the development of web service applications [38].

The creation of value-added services by composing existing ones is a significant trend that has triggered considerable research efforts on web service composition in both academia and industry. Web service description language (WSDL) [56], universal description, discovery, and integration [50], simple object access protocol [42], and part of OWL-S [31] ontology (ServiceProfile and ServiceGrounding) define standards for service description, discovery, and messaging protocols. Note that OWL stands for Ontology Web Language. However, these standards do not deal with the composition of existing services. One of industrial initiatives to address this issue is web services business process execution language (WS-BPEL, originally known as BPEL4WS, BPEL for short) [55], which is an XML-based language supporting process-oriented composition. Another one is OWL-S ServiceModel, which comes from the semantic web community. These initiatives mainly handle service composition where the process flow and bindings between services are known in advance.

Manuscript received January 11, 2013; revised April 10, 2013, accepted May 30, 2013. Date of publication October 18, 2013; date of current version May 13, 2014. This work was supported in part by the National Basic Research Program of China (973 Program) under Grant 2010CB328101, in part by the National Natural Science Foundation of China under Grant 91218301 and Grant 61173042, in part by the Shanghai Science and Technology Research Plan of China under Grant 10DJ1400300, in part by the Rising-Star Program of the Shanghai Science and Technology Commission under Grant 12QH1402300, and in part by the Shu Guang Project supported by the Shanghai Municipal Education Commission and the Shanghai Education Development Foundation. This paper was recommended by Associate Editor N. T. Nguyen. *(Corresponding author: C. J. Jiang.)*

P. W. Wang, Z. J. Ding, and C. J. Jiang are with the Department of Computer Science and Technology, Tongji University, Shanghai 201804, China, and also with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Shanghai 201804, China (e-mail: pwei.wang@gmail.com; zhijun_ding@hotmail.com; cjjiang@tongji.edu.cn).

M. C. Zhou is with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, China, and also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2013.2280559

Despite all these efforts, web service composition is still a highly complex task. Especially, web services are created and updated on the fly nowadays, and it is thus beyond our ability to analyze them and generate the composition plan manually [38]. A large number of methods have been proposed to tackle this problem [6], [10], [23], [25], [26], [29], [30], [35]–[37], [39], [41], [48], [51], [57], [65]–[71]. Among them, dynamic composition methods can generate a process model automatically. In the existing methods for automatic service composition, the requirements of users mainly include the available inputs that they can provide, the outputs that they expect, and the quality of service (QoS) standards that they require. Moreover, some studies [43], [44], [51]–[53] have considered user preferences in the requirements, which are a key component of web service composition. These requirements, including available inputs, required outputs, QoS standards, and preferences, are all provided explicitly by users at the beginning, and the composition of services is guided by this information. In fact, some other user-related information can also greatly affect the composition of services, but often is not given explicitly by users at the beginning. This information is mainly some context conditions, which are often related to users and implicit in the service composition scenarios. For example, in an online shopping scenario, the user's membership level, the total amount of the order, and the user's credit card balance are likely to affect the composition of services. Because of their different values or results, different services may be selected to participate in the subsequent composition. In such situations, a common one is that some available services have the same or similar functions, and may even have the same inputs and outputs, but they can adapt to different results or values of these user-related conditions or constraints implied in the context of such business scenarios.

As a matter of fact, the fundamental reason of such situations is that many available services have applicable conditions or use restrictions, i.e., they are not universally applicable. Most of the available services in our real world are actually like this, and some constraints are often imposed on them by their providers, such as their effective time and available areas. These constraints specify the context conditions that must be met to ensure the correct execution of the service or the proper interaction with it. In this situation, even if the given inputs can match its data interface, a service may still be unable to execute correctly. Therefore, only matching parameters between service interfaces is not enough, and more factors should be given full consideration when we are dealing with service composition. However, most of the existing methods are basically based on input and output interfaces of services. For them, services are merely represented by their input and output parameters, and service compositions are implemented through their matching. Thus, they must be improved.

In this paper, we concentrate our attention on the topic of composing services with constraints. First, through some real world business scenarios, we describe the problem. Then, service constraints and formal expressions are defined. We divide the elements of a web service into two parts: service intension and service extension. Finally, a graph search-based algorithm and two different preprocessing methods are presented to advance the field.

The contributions of our work are as follows.

1) Service constraints are presented and defined for a web service, which specify the conditions that must be met to ensure the correct execution of a service.
2) The proposed solution can solve the problem of automatic service composition while considering service constraints, which have gone beyond what the existing techniques can handle.
3) The proposed graph search-based algorithm can be applied to the general service composition problems. Different from the previous methods, it can generate all the feasible solutions according to a user's request.

Section II discusses the related work. Section III describes a real world business scenario in detail to show the issues. Section IV provides basic concepts and definitions and discusses research issues. Section V describes the proposed solutions. Experimental validation and analysis are given in Section VI. Finally, Section VII concludes this paper.

II. Related Work

A large number of methods have been proposed for automatic service composition. According to the techniques adopted, these methods can be divided into three categories, i.e., graph search-based, formal methods-based, and artificial intelligence (AI) planning techniques-based. In the following, we review and discuss them, respectively.

A. Graph Search-Based Methods

In this category, services are represented by their inputs and outputs. For a registry of available services, a service dependency graph (SDG) can be constructed to show all possible input–output dependencies among the services in this registry. Then, the web service composition (WSC) problem can be converted into a search problem in a graph, i.e., traversing the SDG to find a feasible path either from inputs to outputs or from outputs to inputs, which represents a composite service that can satisfy a user's request.

Hashemian and Mavaddat [11] use a formalism and modeling tool, called interface automata, to represent the behavior of web services, and then convert a WSC problem into a general graph problem. Oh *et al.* [27] present a novel solution called BF* to solve the WSC problem, which is an A* based graph search algorithm. Brogi *et al.* [7], [9] present an algorithm called SAM to determine whether a query can be satisfied by a (composition of) service(s). By building a tree for each process model stored in the registry, they construct a graph representing the dependencies among atomic processes of the services, and then analyze such a dependency graph to determine a service composition satisfying the query. Lang and Su [14] present an AND/OR graph representation of an SDG and its search algorithm for the discovery of composite web services.

Graph search-based methods entirely use inputs and outputs to model services, and the constructed SDG for a service community only reflects the data interface relationships among services. As mentioned earlier, we cannot guarantee the proper execution of a service even if the provided inputs can meet the type requirements of its data interface, because of the existence of service constraints. Thus, these methods fail to address the issue about which we are concerned.

B. Formal Method-Based Techniques

Formal methods, such as Petri net, automaton, process algebra, have been widely used in the studies of service composition. Especially, Petri net, as a formal tool not only with an intuitive graphical expression but also with a strict mathematical foundation, has been widely used in this field.

Berardi *et al.* [4]–[6] develop a finite state machine (FSM)-based framework for automatic service composition, and provide effective techniques for computing service composition where the behavioral description of a service is expressed as FSM. Narayanan and McIlraith [25] use DAML-S ontology to provide semantic markup for web services, and encode service descriptions in Petri nets and provide decision-making procedures for web service simulation, verification, and composition. Brogi and Corfini [8] present a matchmaking system based on OWL-S and Petri nets for discovering deadlock-free compositions of web services. A global Petri net model is generated for a service registry through the data dependencies between services, and then the Petri net state equation technique is used to determine whether there is a composite service satisfying a request. Ding *et al.* [10] present a method to synthesize Petri nets for modeling and verifying composite web services based on the OWL-S specification. The control flow and data flow of a composite service are modeled by Petri nets separately, and then an integrated service net can be constructed by synchronous composition. Tan *et al.* [47] present a compatibility-based method to analyze and compose web services. Services are converted into colored Petri nets and their compatibility is analyzed. In the case of partial compatibility, the method for mediator generation is proposed to assist the automatic composition of partially compatible services. Tan *et al.* [48] propose a novel framework to compose web services from the perspective of data. Both data relations and service composition rules are represented by colored Petri nets, and then a net based approach is proposed to compose services.

The methods in this category are often complicated and difficult to implement, and the computational complexity is also relatively high. Moreover, service constraints about which we care are largely ignored by these methods. Because of their complete theoretical system and rich supporting tools, such formal methods are mainly used for composite service modeling and verification to assure that the composite service can be executed correctly and meet the expectations of service designers and planners.

C. AI Planning Techniques-Based Methods

In this category, the automatic WSC problem is reduced to the well studied AI planning problem. Web services are described by inputs, outputs, preconditions, and effects (IOPE), and regarded as actions in the planning problem. Given an initial state and a goal state, a sequence of actions can be acquired automatically through planning, which represents a solution of service composition. Most of the existing methods for automatic WSC rely on AI planning techniques, and they can be further classified into the methods based on situation calculus, planning domain definition language, rules, hierarchical task network, theorem proving, and so on.

McIlraith *et al.* [21], [22] adapt and extend the Golog language for automatic construction of web services. They address the WSC problem through the provision of high-level generic procedures and customizing users' constraints. In SWORD [35], a service is represented in the form of a Horn rule, and then a rule-based expert system is used to automatically determine whether a desired composite service can be realized by using existing services. Medjahed *et al.* [23] define formal safeguards for meaningful composition through the use of composability rules, which compare the syntactic and semantic features of web services to determine whether two services are composable. In [26], [41], and [57], the SHOP2 planner is applied for automatic WSC. A detailed description is given on the process of translating OWL-S to SHOP2. Rao *et al.* [36], [37], [39] introduce a method for automatic composition of semantic web services by using linear logic theorem proving, where the process model for a composite service can be generated directly from the proof. Oh *et al.* [29], [30] present a WSC algorithm that is an AI planning-based heuristic algorithm. It activates two-step search for a request: forward search to compute the cost of achieving individual parameters starting from the inputs, and regression search to approximate the optimal sequence of services that connects inputs to outputs.

The majority of AI planning techniques-based methods support the use of IOPE information to describe services, but looking through them, the precondition and effect information is not fully utilized. They are still based only on input and output information, which is modeled as preconditions and effects of actions, and used for reasoning. In addition, classical AI planning techniques can derive only linear sequences of actions. Such a 1-D linear process model cannot be used to solve the problems concerned in this paper.

In summary, despite a relatively large body of research in the area of WSC, whether based on graph search, formal methods, or AI planning techniques, the constraints imposed on available services have been largely ignored. Such constraints are used to guarantee the correct execution of services, which have a great impact on service composition. Especially, few efforts have specifically focused on the situation concerned in this paper, i.e., some available services in a business scenario have the same inputs and outputs, and can complete the same task, but have different constraints to meet.

In recent years, there are some studies considering the mismatches and heterogeneities among composed web services in composition. Through annotating WSDL descriptions such that web services are described with contextual details, Mrissa *et al.* [24] propose a context-based mediation approach to solve data heterogeneities among composed web services. Kongdenfha *et al.* [13] propose a taxonomy of common mismatches regarding the service interfaces and business protocols. Based on this, an adaptation methodology to capture and formalize the recurring differences between business interfaces and protocols is presented. Focusing on the data-level semantic heterogeneity including inconsistent data naming, representation, precision, scaling, and unit, which widely exists among the real-world services and hampers the correct interoperability and composition of services, Li *et al.* [15]–[17] propose a solution to automatic determination and reconciliation of context conflicts in web service composition. They address various semantic differences from the context perspective and use a lightweight ontology to describe the related concepts. Similar to our work, these efforts also focus on context-related information, which has a great impact on automatic WSC. However, the constraints concerned in this paper specify the use restrictions, which are explicitly defined by its provider to clarify its function and guarantee its proper invocation; while the above efforts pay attention to the mismatches and conflicts among services engaged in composition, which are caused by the facts that multiple web services are developed by different providers and they may have different specifications or implicit assumptions no matter from interface level or protocol level.

In addition, owing to the market competition, independent providers develop several web services that can offer the same functionality such as currency exchange [20]. To ease and improve the process of web service discovery in an open environment such as the Internet, Maamar *et al.* [3], [19], [20], [45] suggest gathering web services with similar functionality into groups known as communities, which is similar to the idea of grouping SIDE services as a set in this paper. Different from our work, they focus on designing, developing, and managing communities of web services, such as how to initiate, set up, and specify a community, and how to specify and manage web services in a community. By contrast, we focus on the constraints imposed on available services, as well as the resulting scenario where several services can perform the similar functionality but with different applicable conditions. We deal with the issue from the composition level to avoid the execution failure of a composite service due to the existence of such constraints. From the perspective of grouping SIDE services as set and management of the evolution of such sets, the above discussed prior efforts can provide us with useful help and support.

III. Motivating Scenario

A. Shipping and Delivery Scenario

Today, there are many online e-commerce websites such as Amazon, Newegg, Ebay, and Alibaba. Here, for clarity and conciseness, we abstract and simplify some real-world scenarios to form the following example. It includes a series of core tasks: searching products, submitting an order, paying for the order, and shipping/delivery. Take B2C website Icson [54] for example. It provides three different shipping/delivery services: one-day, two-day, and standard shipping. These three services have the same input parameters as $\{$*OrderInformation, PaidNotification*$\}$, and the same output parameters as $\{$*ShippedNotification*$\}$. They can complete the same task, namely, the shipping/delivery of goods. According to the existing methods for automatic service composition [30], [35], [39], [51], we can obtain three composite solutions as shown in Fig. 1.

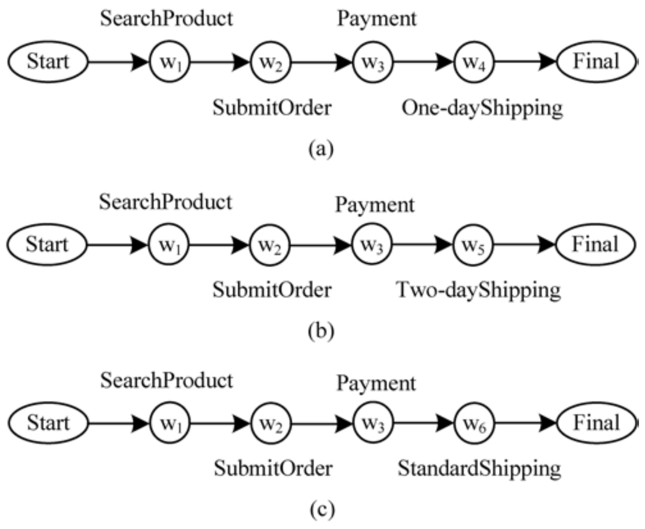


Fig. 1. Three composite services with (a) one-day shipping, (b) two-day shipping, and (c) standard shipping.

However, these three services have different applicable conditions, i.e., they are available only for some special situations. One-day shipping service is available only for the orders whose delivery addresses are located in Shanghai; two-day one is available only for the orders whose delivery addresses are located in provinces of Zhejiang and Jiangsu; while standard shipping service is available for the orders located in other regions of China. Obviously, all of them can perform the shipping/delivery function, but are applicable to different situations. Thus, if we just select one of the three solutions in Fig. 1 randomly to use, it will easily lead to failure when we execute the selected composite service.

Service composers are required to avoid the occurrence of above situations by meeting their application conditions to ensure the correct execution of their composite service by selecting the appropriate component services. In other words, for the above scenario, one possible expected process model of the composite service may be similar to the one illustrated in Fig. 2. When we execute the composite service, the appropriate shipping/delivery service is selected to participate in the composition according to the context at run time.

In Fig. 2, condition C_1 represents that the delivery address of the order is in Shanghai, C_2 means that it is in Zhejiang or Jiangsu province, and C_3 represents other addresses.

B. Discussions

From this scenario, we know that some available services may only be applied to some special situations, i.e., they are not universally applicable, but have their own conditions of application. Especially, we often encounter such scenarios that several available services can complete the same task

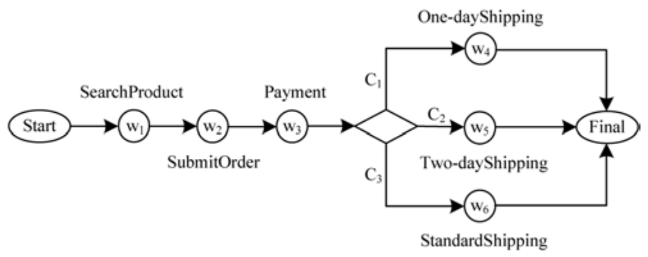


Fig. 2. One possible process model of composite service for the shipping and delivery scenario.

but have different applicable conditions, such as the three different shipping/delivery services in the above example. As a matter of fact, such cases are very common in a real world. Such services often accept the same inputs, produce the same outputs, and can perform the same or similar functions. However, their applicable conditions are different form each other. In addition, the same service provider may provide similar services to adapt to different groups of customers or markets. For example, a web hosting service provider may provide several different hosting services that offer different storage space sizes and file transfer rates according to different customers, such as gold, silver, and bronze members. An online shopping website may provide several different discount services or gift services according to the total amount of orders or the membership level of customers.

However, most of the existing methods for automated service composition do not consider this problem. Obviously, in the above example, the solutions generated from the existing composition methods cannot reflect the facts that some available services may have their specific applicable conditions. Especially, these methods do not differentiate such services that can accomplish the same task, but applicable to fully different conditions. Thus, it will easily result in failure when we execute a randomly selected one of the generated solutions. In a word, most of the existing methods for WSC fail to address the issue in which we are interested.

IV. Preliminaries

The motivating scenario shows the problem of real services having constraints, which hampers the automatic service composition. Before the solution is proposed, in this section, we first explain the basic concepts and definitions to be used later. Furthermore, for the motivating example, we articulate the key concepts and problem description.

A. Basic Concepts

From the above examples, we know that considering input and output parameters only is not enough when we are dealing with service composition. In reality, it is not always the case that a service can be executed to obtain the correct result even if the given input is in accordance with the interface type of the service [49], owning to the presence of service constraints, such as the range of input parameters, available time, and coverage area of the service. Take zip code inquiry services that are common on the Internet for example. They accept a city name and an address as inputs, and return the corresponding zip code as output. However, some of these services may provide zip code information only for the cities and addresses in China, while some others only for the cities in the U.S. Thus, for a specific zip code inquiry service, even if the service requestor enters a city name and an address as inputs, the service cannot get the correct result if the inputs do not satisfy the constraints of the service. The example in Section III shows such scenarios. In the following, we first introduce the definition of service constraints, which is originated and extended from [49].

Definition 1 (Service Constraints): Service constraints specify a set of context conditions that must be met to ensure the correct execution of a service. These restrictions are often imposed by service providers through the way of limiting the range of some attributes of a service.

Each web service is associated with one or more semantic concepts in an ontology, and every concept may have multiple attributes. Thus, service providers can impose constraints on some attributes by restricting their ranges so as to make service functions clear and ensure their correct execution [49]. The range of attribute values can be represented by numerical values or a set of semantic concepts. Formal expression of service constraints is as follows.

Definition 2 (Constraint Expression): Formal expression of service constraints is given in the form of *Attribute opr InstanceData*, where:

1) *Attribute* represents a service attribute;
2) *opr* represents operators such as $=$, $\neq$, $<$, $\leq$, $>$, $\geq$, $\in$, $\subseteq$, $\supseteq$, is, not, in, and $not\ in$;
3) *InstanceData* represents data instances, including traditional numeric data, set of semantic concepts, data set, etc.

Based on these concepts and definitions, we give the definition of the following web service.

Definition 3 (Web Service): An atomic service can be represented by a tuple $WS = (op, I, O; SC, QoS)$, where:

1) op is a semantic concept, and provides semantic description of this service operation;
2) I is the set of semantic concepts that are referenced by the input parameters of the service;
3) O is the set of semantic concepts that are referenced by the output parameters of the service;
4) SC represents the constraints of a service, which specify the context conditions that must be met to ensure the correct execution of the service; and
5) QoS is the set of quality parameters of the service, such as execution time, price, availability, and reputation.

In SAWSDL [40], the op and I/O of a web service can reference the concepts of external semantic models (e.g., service ontology) through extension attribute *modelReference*. A service ontology consists of a common language agreed by a community, e.g., insurance industry. It defines a terminology that is used by all participants in the community. Within a community, service providers describe their services by using the terms of the community's ontology, while service requesters use the terms of the ontology to formulate

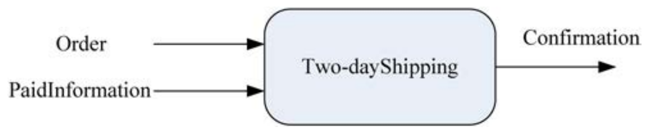
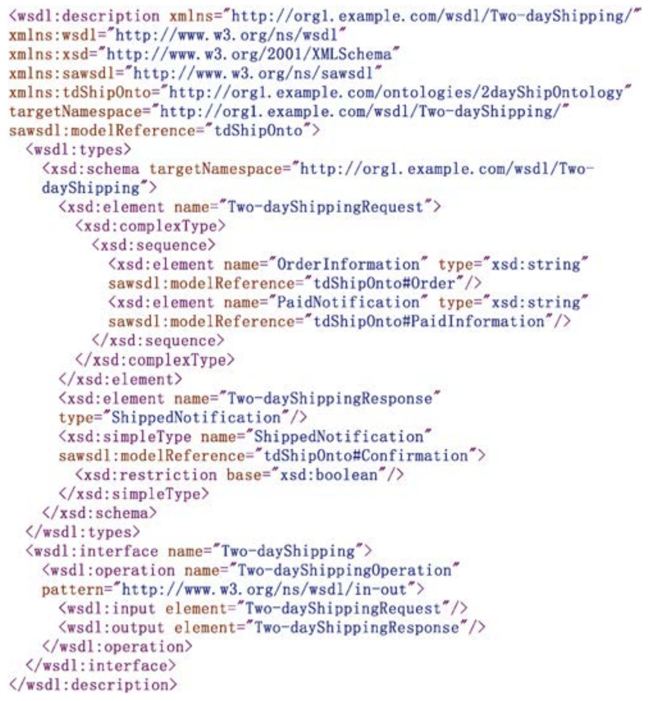


Fig. 3. Intuitive description of two-day shipping service.



```
<wsdl:description xmlns="http://org1.example.com/wsdl/Two-dayShipping/"
xmlns:wsdl="http://www.w3.org/ns/wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sawsdl="http://www.w3.org/ns/sawsdl"
xmlns:tdShipOnto="http://org1.example.com/ontologies/2dayShipOntology"
targetNamespace="http://org1.example.com/wsdl/Two-dayShipping/"
sawsdl:modelReference="tdShipOnto">
  <wsdl:types>
    <xsd:schema targetNamespace="http://org1.example.com/wsdl/Two-
    dayShipping">
      <xsd:element name="Two-dayShippingRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="OrderInformation" type="xsd:string"
            sawsdl:modelReference="tdShipOnto#Order"/>
            <xsd:element name="PaidNotification" type="xsd:string"
            sawsdl:modelReference="tdShipOnto#PaidInformation"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="Two-dayShippingResponse"
      type="ShippedNotification"/>
      <xsd:simpleType name="ShippedNotification"
      sawsdl:modelReference="tdShipOnto#Confirmation">
        <xsd:restriction base="xsd:boolean"/>
      </xsd:simpleType>
    </xsd:schema>
  </wsdl:types>
  <wsdl:interface name="Two-dayShipping">
    <wsdl:operation name="Two-dayShippingOperation"
    pattern="http://www.w3.org/ns/wsdl/in-out">
      <wsdl:input element="Two-dayShippingRequest"/>
      <wsdl:output element="Two-dayShippingResponse"/>
    </wsdl:operation>
  </wsdl:interface>
</wsdl:description>
```

Fig. 4. SAWSDL description of two-day shipping service.

queries over the service registry [60]. All the constraints of a service can be represented by a rule, having the form of $SC_1 \wedge SC_2 \wedge \ldots \wedge SC_n \rightarrow WS$, $SC_i \in SC, 1 \leq i \leq n$. This rule can be represented by semantic web rule language (SWRL) [46] or other languages, which can be referenced into SAWSDL through *modelReference*. Note that SAWSDL does not provide support for QoS description of a web service. Service providers and users can utilize other protocols, e.g., service level agreement, to do so.

Take the aforementioned *Two-dayShipping* for example. Its intuitive description is shown in Fig. 3, and the SAWSDL description in Fig. 4. In Fig. 3, *Order, PaidInformation* and *Confirmation* are all semantic concepts defined in some ontology.

SWRL extends OWL axioms through Horn-like rules, thereby combining Horn rules and OWL knowledge base. Therefore, with respect to the constraint of *Two-dayShipping*, it can be represented in the form of *Order.DeliveryAddress* $\in$ {*Zhejiang, Jiangsu*} $\rightarrow$ *Two-dayShipping*, in which *DeliveryAddress* is an attribute of the semantic concept *Order*. Then, this rule can be referenced into the SAWSDL description of the service as shown in Fig. 4.

Inputs and outputs of a service specify the data/information transformation produced by the service. They contain the underlying functional knowledge of a service [32]. As the core of a service they reveal things of the service itself. In addition, in Definition 3, *op* is the semantic description of this service operation, and describes the functions that the service can provide. Based on this consideration, we call $INT = (op, I, O)$ service intension. By contrast, we call $EXT = (SC, QoS)$ service extension. Because these properties reflect more on the nonfunctional aspects of a service. Service constraints are restrictions imposed on a service to ensure its correct execution and obtain the functions it claims. They always specify the conditions or environments to which the service can be applied, and they have nothing to do with what a service can do or provide, i.e., the functions of a service. In essence, these constraints are imposed not on the service function but the environments supporting it. For QoS properties with which we are familiar, although they are the properties of a service itself, they also have nothing to do with its function. They encompass a number of nonfunctional properties such as price, availability, and reliability, and they do not affect the functionality of a service. Therefore, the constraints and QoS information are viewed as the extensions of a service. To sum up, service intension represents what a service can do, i.e., the core of the service, while service extension reflects the noncore aspects, such as the environment under which the service can be applied, and QoS information. This division and notation will contribute to the classification, identification, and resolution of some service composition related issues.

Next, we give the definition of a service request from users.

Definition 4: A user's request can be represented by a tuple $Req = (I_A, O_E; P, Q)$, where:

1) I_A represents the set of semantic concepts that are referenced by the available inputs that a service requester provides;
2) O_E represents the set of semantic concepts that are referenced by the outputs that a service requester expects;
3) P represents the set of personalized preferences and constraints defined by a service requester;
4) Q represents the set of standards of services' quality parameters defined by a service requester.

Here, the third component of the tuple is user preferences, which include personalized preferences and service constraints designated by service requesters except for QoS-related standards. We have discussed the personalized preference in [51]. An example is that Lucy prefers to go by air over car if the driving time is greater than 4 h. Via service constraints, requesters can limit the range of some attributes to help themselves find the most desired and really needed services. Usually, requesters impose restrictions on the range of attributes associated with their defined inputs and outputs [49].

B. Problem Description and Definition

From the motivating scenario and basic concepts defined above, it can be known that there exist some services with identical intensions but different extensions. How can one deal with such circumstances in WSC? In this section, we articulate the key concepts and definitions to answer it.

For a certain task in a specific business scenario, all the available Same-Intension Different-Extension (SIDE) services that can accomplish the same task constitute a set.

Definition 5: A SIDE service set is $SS = \{WS_1, WS_2, \ldots, WS_k\}$, $WS_i = (op_i, I_i, O_i; SC_i, QoS_i)$, $1 \le i \le k$, where:

1) $op_i \equiv op_j$, $I_i \equiv I_j$, and $O_i \equiv O_j, \forall i, j \in \{1, 2, \ldots, k\}$;
2) $\forall i, j \in \{1, 2, \ldots, k\}, i \not\equiv j : SC_i.InstanceData \not\equiv SC_j.InstanceData$.

$\equiv$ represents the semantic equivalence between two semantic concepts. The first item ensures that the functionalities, inputs, and outputs of these services are equivalent or the same. $SC_i.InstanceData$ represents the attribute values of service WS_i specified by its constraint SC_i.

Definition 6: Suppose that a web service community $C = \{WS_1, WS_2, \ldots, WS_n\}$ where $WS_i = (op_i, I_i, O_i; SC_i, QoS_i)$, $1 \le i \le n$, and a user request $Req = (I_A, O_E; P, Q)$. A WSC problem is to find a finite set of services from C such that:

1) these selected services can be organized in a certain mode of construction to form a new value-added service WS_c;
2) WS_c can accept inputs I_A and produce outputs O_E;
3) WS_c should meet a user's constraint requirements and personalized preferences (i.e., P);
4) the quality of selected component services and their composite service WS_c should meet the standards defined by the user (i.e., Q); and
5) the constraints of all participating component services should be fully enforced, thereby ensuring their correct execution, and guaranteeing the valid interaction and collaboration among them.

From the motivating scenario and problem descriptions, what we are concerned about is that there exist SIDE services, while we are dealing with service compositions. When we are composing existing services to form value-added services, this circumstance needs to be dealt with properly to assure the correct execution of each component service, thereby ensuring the correct execution of the composite service.

In addition, according to the definition of service extension, we know that it includes two parts: service constraints and QoS. Thus, different extensions consist of some different cases under the condition that the intensions are identical. In this paper, this notion refers specifically to those services with different service constraints, regardless of whether the QoS information is the same or not. QoS is a broad concept that encompasses a number of nonfunctional properties of services, such as price, reputation, availability, and reliability. Usually, the QoS information is used to select some specific services for the purpose of their composition in a predefined way that maximizes the performance of the composite service and user satisfaction. This selection is based on the condition that some available services have the same intensions and service constraints. It is based on the predefined process model of a composite service, and cannot change the structure of the process. QoS related work has been widely discussed [1], [18], [58]–[60], [63]–[64].

V. Constraints-Aware Service Composition

In this section, we present a solution for service composition considering service extensions to address the discussed issues. First, a graph search-based algorithm is proposed as a main algorithm, which can also be used for general WSC problems. Then, two different preprocessing techniques to handle SIDE services are presented, namely, a prepackaging method and abstraction and refinement one.

A. Graph Search-Based Algorithm

A large number of methods have been proposed for automated WSC problems, such as graph search-based, formal methods-based, and AI planning techniques-based ones. The proposed main algorithm is graph search-based. Several studies have successfully applied graph search to WSC problems [7], [9], [11], [14], [27]. They represent services as their inputs and outputs. An SDG can be constructed to show all possible input–output data relationships among the available services in a given registry. Then, the WSC problem can be converted into a search problem in the constructed SDG. However, these methods either generate only one composite solution each time [7], [9], [11], [14], or every step of the solution includes a set of executable services, some of which may not belong to this solution, even do not appear in any feasible solution [27], [62], and such a result cannot be considered a real solution. Here, we give an algorithm to find out all the feasible composite solutions that can satisfy a user's request. AND/OR graph is used to represent SDG, and this is similar to the work [14], where only one composite service template can be generated by their algorithm, which is computationally easier.

AND/OR graphs can be seen as a generalization of directed graphs, and it is commonly used in automatic problem solving and problem decomposition. It contains two kinds of nodes: AND and OR, and they are connected by generalized edges, which are called connectors. Each connector in an AND/OR graph connects a set of nodes $\{v_i | i = 1, \ldots, n\}$ to a single node v_o. If there is a logical AND relationship among $\{v_i\}$, this connector represents an AND operation and it is said to be an AND connector. Similarly, if there is a logical OR relationship among them, the connector represents an OR operation and is called to be an OR connector [14]. Here, we adopt AND/OR graphs to represent SDG for service composition. In this representation, since a service can be executed only when all of its input parameters are available, there is a logical AND relationship among those data nodes that are connected to this service node directly, and all the services nodes in this representation are AND nodes. In contrast, there is a logical OR relationship among those service nodes that can produce a certain data parameter because any one of them can generate this output and make the parameter available. Thus, all the data nodes in this representation are OR nodes. With this representation for SDG, we present a search algorithm based on it to find all the feasible composite service solutions.

The algorithm first constructs an SDG according to the given service community C (line 1), which is in the form of AND/OR graph. Then, SDG is initialized based on the information provided by the service requestor (i.e., user request Req) to facilitate the search (line 2). In this step, two dummy nodes are added to the original SDG. One is connected to all the input data nodes that are provided by the service requestor with direct edges. This node is considered to be

Algorithm 1 Graph Search-based Algorithm for WSC Problem

```
Input: A set of available services (service community)
    $C = (WS_1, WS_2, \dots, WS_n)$, and a user request $Req =$
    $(I_A, O_E; P, Q)$.
Output: A set of feasible composite services.
 1: $SDG = ConstructServiceDependencyGraph(C)$;
 2: $G = InitializeSDG(SDG, Req)$;
 3: for each OR node $n \in G$ do
 4:     Label $n$ as "Unknown";
 5: end for
 6: $SearchAllPaths(G, path, n_s, n_t, 1)$;

Procedure $SearchAllPaths(G, path, v, n_t, length)$
 1: $path[length] = v$;
 2: if $v$ is an AND node then
 3:     Label all $v$'s child nodes as "Known";
 4: end if
 5: if $v == n_t$ AND all $v$'s parents are labeled as "Known"
    then
 6:     $ConstructSolutionSubgraph(G, path)$;
 7: else
 8:     for each child node $n_i$ of $v$ do
 9:         if $n_i$ is an OR node then
10:             $SearchAllPaths(G, path, n_i, n_t, length + 1)$;
11:         else
12:             if all $n_i$'s parents are labeled as "Known" then
13:                 $SearchAllPaths(G, path, n_i, n_t, length + 1)$;
14:             end if
15:         end if
16:     end for
17: end if
18: $v_{tem} = path[length]$;
19: if $v_{tem}$ is an AND node then
20:     Label all $v_{tem}$'s child nodes as "Unknown";
21: end if
22: $length$=$length$-1;
```

solved, and used as the starting node for the search process in our algorithm. The other dummy node is connected with all the output data nodes that are expected by the service requestor through an AND connector. It is marked as an AND node and represents the WSC problem to be solved; and it is used as the termination node (or goal node) for the search process. Based on the modified AND/OR graph G, all the data nodes (i.e., OR nodes) are initialized as unknown to represent that these data parameters have not been acquired (lines 3–5). Finally, we call procedure *SearchAllPaths* to find all the paths from the starting node n_s to the termination node n_t (line 6), which is depth-first search-based.

In *SearchAllPaths*, we first record the node currently being visited (line 1). If it is an AND node, we label all its child nodes as known (lines 2–4), which represents that once a service node (AND node) is identified, all its output data nodes can be known. Then, we judge whether n_t has been accessed or not. If yes, and all of its parent nodes have been identified, this means that we have found a path from the starting node to the termination node in G, which represents a feasible solution. Thus, we can easily construct the process model of this solution according to the information (i.e., nodes and their orders) recorded in *path* (lines 5, 6). If not, we call *SearchAllPaths* itself for all the child nodes of the current node, and then complete the search for the entire graph G in a recursive manner (lines 7–17). In the search process by recursive calls, when some conditions are not satisfied, or a path has been found, or a recursive process has been completed (i.e., the completion of a for-loop in the procedure), we need to backtrack to complete the entire graph traversal and find out all the paths from n_s to n_t (lines 18–22). It is worth noting that when we backtrack a service node, all of its child nodes need to be relabeled as unknown, because these parameters become no longer available after the backtracking.

The characteristics of an AND/OR graph make it suitable to describe and solve WSC problems. A service can be invoked only when all its inputs are available, while a data parameter can be made available by any service who takes it as output, as well expressed by an AND/OR graph. With this well structured SDG in hand, we can perform solution search for WSC problems. Even so, finding all the possible solutions is still quite difficult and complex, to be discussed in detail later. In Algorithm 1, we adopt a recursive strategy to try to find all the possible solutions. The algorithm is just a high-level description, in which there are many details needed to handle. For example, when backtracking a service node, we cannot directly make all its child data nodes unavailable, because the data might have been generated several times by different services earlier. Thus, it requires special handling for the known mark of a data node. As a result, we build a list for every data node to record what services have previously generated the data and their order so as to discriminate the data nodes when backtracking. There are several other difficulties to be overcome in this recursive searching process. Therefore, although Lang and Su [14] use AND/OR graphs as the representation for SDG, they do not take a recursive strategy, thus avoiding a number of difficulties in this recursive process. Note that their algorithm only finds one solution each time, which is much easier to handle.

B. Service Composition Through Prepackaging

In an available service community, all the SIDE services corresponding to the same task can constitute a set called a SIDE service set. These services can accomplish the same task, but have different applicable conditions. Thus, in order to ensure the correct execution of these services and the resulting composite service, this situation needs to be dealt with properly and carefully. In this section, we first propose a method called prepackaging to address this problem.

In this method, before the main algorithm for service composition is applied to the available service community C, all the SIDE service sets in C are packaged, respectively. In other words, they are composed together in advance. For each SIDE service set, all of its services are composed together to form a new composite service first, which is a real service and can be invoked and executed directly. Then, these composite services are put in the service community C to replace their corresponding SIDE set of services, and they are regarded as

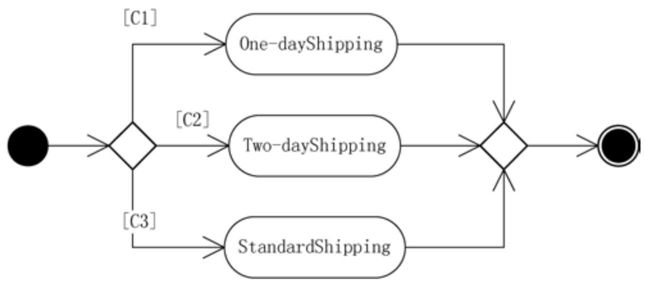


Fig. 5. Process model of composite shipping service by three shipping/delivery services.

component services to participate in the subsequent service compositions.

According to the definition of a SIDE service set, we can easily know that the process model of each composite service generated by a SIDE set contains a branch structure, and each available service from the SIDE set is located at one branch of this structure. This composite service can complete the same task with its component services, but has a wider range of application. When this composite service is actually invoked and executed, an appropriate component service on some branch will be selected automatically to execute according to the truth values of branch conditions, i.e., select the service whose constraints can be satisfied in the execution context.

In the shipping/delivery example, we have three different shipping/delivery services: one-day, two-day, and standard shipping. They have the same intensions, and can complete the same task, but have different constraints, i.e., their application conditions are different from each other. Thus, in this business scenario, they can together constitute a SIDE set. By the prepackaging method, they are composed together in advance to form a new composite shipping service, whose process model is shown in Fig. 5.

From Fig. 5, the process model of this composite service contains a conditional branch structure. Three available services are located at different branches, and the branch conditions are just the constraints of the corresponding services, which represent the scope of their application, respectively. Specific to this scenario, condition $C1$ represents the delivery address of the order is located in Shanghai, i.e., $DeliveryAddress \in \{Shanghai\}$; $C2$ represents the delivery address is located in Zhejiang or Jiangsu province, i.e., $DeliveryAddress \in \{Zhejiang, Jiangsu\}$; $C3$ represents the delivery address is located in other regions of China, i.e., $DeliveryAddress \in \{China\}$ and $DeliveryAddress \notin \{Shanghai, Zhejiang, Jiangsu\}$. This new composite service can complete the same task as its three component services, i.e., the shipping/delivery function. However, it has a wider range of applications because it can handle the orders whose delivery addresses are located in any region of China.

C. Service Composition Through Abstraction and Refinement

In the prepackaging method, the available services in the same SIDE set are composed together in advance to form a new composite service, and then it is used to participate in service compositions in replacement of their corresponding set of SIDE services. Different from this one, in this section, we propose another method by using abstraction and refinement techniques to deal with this problem. In this method, we first abstract all its elements for each SIDE service set.

Given a set SS of SIDE services, we define an abstract service for it, which is actually the intensional abstraction of a group of services that can provide the same functionality and have the same input/output interfaces. It is a conceptual service defined next.

Definition 7 (Abstract Service): Given a SIDE service set $SS = \{WS_1, WS_2, \ldots, WS_k\}$, $WS_i = (op_i, I_i, O_i; SC_i, QoS_i)$, $1 \le i \le k$, the abstraction of SS is $WS_{abs} = (op_{abs}, I_{abs}, O_{abs})$, where:

1) $op_{abs} \equiv \{op_i \mid 1 \le i \le k\}$, which is semantic description of this abstract service;
2) $I_{abs} \equiv \{I_i \mid 1 \le i \le k\}$, which is the set of semantic concepts of inputs; and
3) $O_{abs} \equiv \{O_i \mid 1 \le i \le k\}$, which is the set of semantic concepts of outputs.

According to the definition of a SIDE service set, the services in the same set have the same intensions but different extensions. Thus, the abstract service as defined above is just the abstraction of their common intensions, which preserves their input and output information, and also the functional description. So, in fact, such an abstract service represents a group of available services that can provide the same functionality, with the same interfaces, but have different applicable conditions.

With this definition, we can easily process the existing SIDE services in a given service community C automatically. After all the SIDE services sets are substituted by their corresponding abstract services, we can apply the graph search-based algorithm to WSC problems on the modified service community C_{abs}, and obtain all the feasible solutions. Since each abstract service WS_{abs} preserves the intensions of the replaced real services, this substitution in the available service community would not affect the solution of the WSC problem, and it satisfies soundness and completeness. In other words, the solutions generated from the modified services community C_{abs} must have solutions to corresponding the original WSC problem. Furthermore, all the solutions of the original problem can be generated from C_{abs} directly or indirectly.

In addition, WS_{abs} is just a virtual service. Therefore, when it is part of a certain composite service solution that will be actually executed, we need to dynamically select and bind the appropriate real service from the corresponding SIDE service set according to the actual execution context of this composite service, i.e., select the specific service whose constraints can be satisfied in the current context, and bind it to the corresponding abstract service WS_{abs}. Algorithm 2 simply shows this process.

In summary, the basic idea of this method is to perform abstraction on the services whose intentions are identical but with different extensions. Then, the resulting abstract services are used for service composition instead of their corresponding real services. Last, the abstract services in a composite service solution are specified by the appropriate real services when the composite service is practically executed. Abstraction and refinement techniques are successively adopted in this method.

Algorithm 2 Run-time selection and binding of component web service for WS_{abs}

```
Input: A composite service solution $WS_{ca}$ with an abstract service $WS_{abs}$, service community $C$, and the SIDE services set $SS = \{WS_1, WS_2, \dots, WS_k\}$ corresponding to $WS_{abs}$.
Output: An execution plan of this composite service.
1:  for All the tasks except $WS_{abs}$ in the process model of $WS_{ca}$ do
2:      Select and bind appropriate speific web services to these tasks according to existing methods for service selection and binding.
3:  end for
4:  while $SS \neq \emptyset$ do
5:      $\forall WS_i \in SS$;
6:      if Its constraint $SC_i$ can be satisfied in the current execution context of this composite service then
7:          Bind $WS_i$ to the abstract service $WS_{abs}$;
8:          Exit the while loop;
9:      end if
10:     $SS = SS - \{WS_i\}$;
11: end while
12: if $SS == \emptyset$ then
13:     return failure
14: end if
```

Generally speaking, abstraction and refinement are a pair of complementary steps, and the idea of abstracting first and refining afterwards is a commonly used and effective way to cope with complex problems.

VI. Experiments and Analysis

A. Validation of Graph Search-Based Algorithm

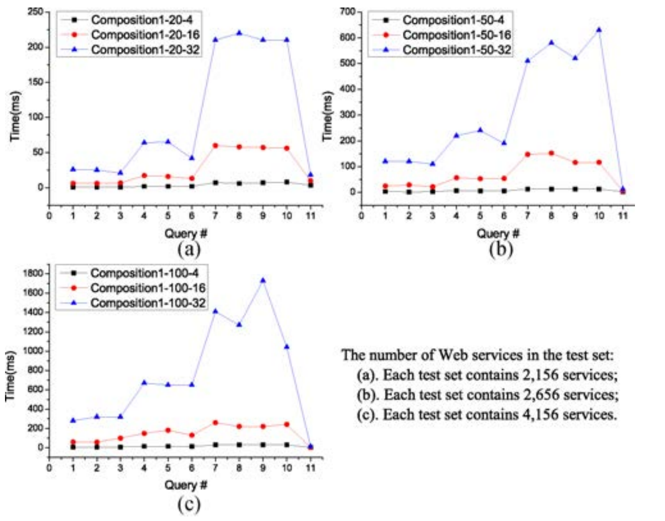
With regard to the general WSC problem, a large number of methods have been proposed to tackle it. Some generate only one composite service [4]–[7], [9], [14], [29], [30], [36], [39]. Some produce a solution that cannot be considered a real composite service template, because every step of such a solution is a set of services, which contains not only those necessary services but also a large number of useless ones; and all the different feasible composite services are mixed together in such solutions [27], [62]. Finding all the possible composite solutions for the general WSC problem is a complicated and difficult task, because this can be reduced to the satisfiability problem, which is a well-known NP-complete problem [28]. Such difficulties as the huge search space, the identification and removal of redundant services, and the low efficiency of finding solutions restrict the methods that can generate all the feasible composite solutions for a general problem, and thus previous approaches [6], [9], [14], [29], [30], [39] mainly focused on finding one (optimal) composite service template. Different from the past work, we propose an algorithm that can generate all the feasible composite services for the general WSC problem in this paper as a main method.

Despite the fact of complexity and difficulties, the proposed graph search-based algorithm is admissible. In order to evaluate it, extensive experiments have been performed via a publicly available test set: ICEBE05 [12], which has been used as the benchmark test data for web services challenge at ICEBE 2005. There are two groups of test data for the composition challenge, namely, Composition1 and Composition2. Both of them have nine test sets respectively, each of which is a repository of web service specifications that contains different numbers of WSDL files. Each test set has 11 queries which is represented by the provided input messages and required output messages.

Here, we used Composition1 for our evaluation. The nine test data sets in it are varied from different aspects, which are described as follows. First, the number of web services in a test set is set to three levels, namely, 2156, 2656, and 4156. Second, the number of input and output parameters of services is also set to three levels, which are differentiated in the range of 4–8, 16–20, and 32–36. Third, from the perspective of composition results with regard to the provided queries, there are four levels of this number, which is 1, 25, 125, and 625. All the nine test sets have the same number of results for the request of the same number, which will be learned from the experimental results. All of the experiments were performed on a PC platform with Intel Core i3 CPU 550 (at 3.2 GHz), Windows 7, and 4-GB RAM. The algorithms are implemented by Visual C# in Microsoft Visual Studio 2010, and all the reported results are based on five runs.

Considering the different size of the nine test sets in Composition1, the comparison of time performance is shown in Fig. 6. The three test sets whose names are started by Composition1–20 all contain 2156 web services, and thus they are grouped together for comparison as shown in Fig. 6(a). Similarly, Fig. 6(b) shows the comparison over the three test sets containing 2656 services and Fig. 6(c) shows the same case of three test sets with 4156 services. From Fig. 6, we know that the time performance is very closely related to the size of test sets. For example, the time spent by the three test sets with 2156 services over each request is between 0.78 ms to 220 ms, while the longest case reaches 630 ms for the test sets with 2656 services and 1730 ms for the test sets with 4156 services.

Meanwhile, in addition to the size of test sets, the number of service parameters is another very important factor that can affect the time performance, which can also be seen from Fig. 6 and more explicitly from Fig. 7. In Fig. 7, at the x-axis, three parameter count ranges are specified, and the average time taken by each test set over its related 11 queries as the y-axis. With the increase of the parameter count, the time taken to solve the problem presents a process of accelerated growth, rather than proportional to the growth of service parameters. This can be derived from the change in slope of the two sections of each line. From this point of view, the impact on problem solving from the factor of service count is similar to that of parameter count, and this can be seen from the comparison on slope of the three lines in Fig. 7, which represent three different levels of test set size. This coincides with the direct analysis of the algorithm, which is based on AND/OR graphs. Two types of nodes are included in this structure: AND nodes and OR nodes, which represent services



The number of Web services in the test set:
(a). Each test set contains 2,156 services;
(b). Each test set contains 2,656 services;
(c). Each test set contains 4,156 services.

Fig. 6. Comparison of test sets grouped by their different sizes. (a) Comparison of time in Composition1-20. (b) Comparison of time in Composition1-50. (c) Comparison of time in Composition1-100.

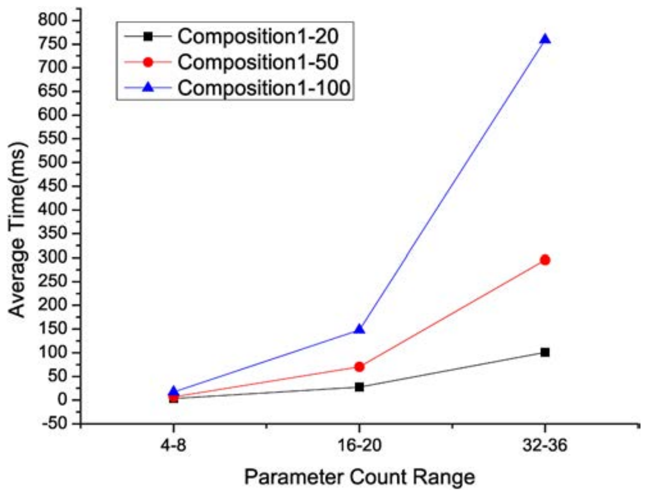


Fig. 7. Comparison of test sets considering the different levels of service parameters.

and parameters, respectively. With the growth of service nodes and data nodes (i.e., parameters), the search space increases dramatically.

In addition, it can also be seen from Fig. 6, the time spent on different requests varies greatly even for the same test set (i.e., both test size and parameter range are identical). For all nine test sets, the 11th request is the simplest, while requests from seventh to tenth are the hardest, followed by requests from fourth to sixth and then requests from first to third. Through the experiments, we have learned that the 11th request has only one solution, while there are 625, 125, and 25 solutions for requests from seventh to tenth, fourth to sixth, and first to third, respectively. In order to more clearly demonstrate the impact of the factor on the number of resulting solutions, we give Fig. 8 from the experimental results. In it, we take the nine test sets as the x-axis. The time values generated by the request of the same number over different test sets are connected by line, and those requests having the same number of resulting

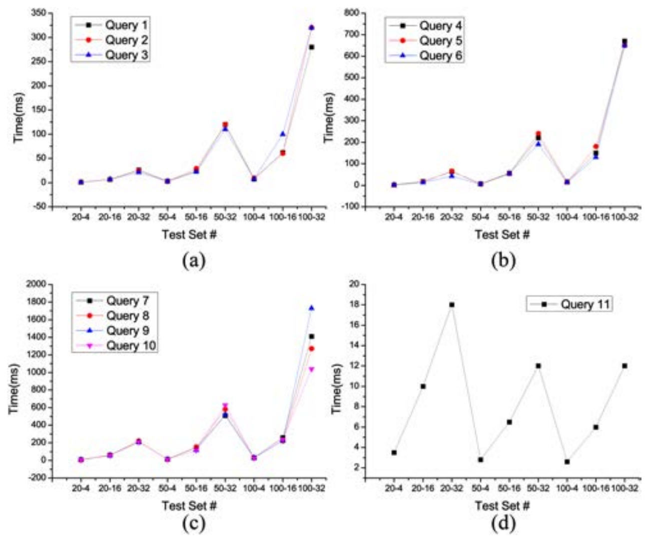


Fig. 8. Comparison of 11 queries considering different numbers of result solutions. (a) Comparison of requests from first to third over nine test sets. (b) Comparison of requests from fourth to sixth over nine test sets. (c) Comparison of requests from seventh to tenth over nine test sets. (d) Comparison of time for request 11 over nine test sets.

solutions are grouped together to form a subgraph of Fig. 8. In addition to some of the above conclusions, we can see that the lines in one subgraph are basically overlapped, which means that over the same test set, the difficulty of those WSC problems with the equal number of solutions is basically the same. With respect to Fig. 8(d), the 11th request for every test set has one solution only which consists of only one service, and thus the case degenerates into a simple service discovery problem. Compared to the service composition problem, the service discovery problem is not so sensitive to the factors such as service count in the registry and parameter range.

B. Comparison Between Two Preprocessing Methods

From the process of problem solving, the prepackaging method and abstraction/refinement one are similar as two preprocessing methods. That is, all the SIDE service sets are processed in advance; then, the resulting services (composite services or abstract services) are used to participate in the subsequent service compositions. Next, the graph search-based algorithm is applied to the modified services community for the WSC problems. These two preprocessing methods both belong to the abstract category in essence, but work at different levels. They are different in the specific implementation process, thereby leading to the difference between the resulting composite solutions, which will be reflected clearly in the actual execution of these composite services. Both methods have their advantages and disadvantages as discussed next.

The results of preprocessing by the prepackaging method are composite services, which can be invoked and executed directly. By contrast, the results of preprocessing by the abstraction/refinement method are abstract services, which are virtual and cannot be invoked directly.

In addition, the results of service composition are composite service solutions, which are flow structures of service models, and each structure defines a process model. According to the process model of a composite service, an execution plan can be generated by selecting good-quality service providers so as to instantiate the model, i.e., an assignment of component services to the tasks in the process model of this composite service is obtained. Then, the execution engine can orchestrate these component services to execute the instance of this composite service [60].

Here, different results of preprocessing lead to the difference among the final composite solutions. The prepackaging method results in ordinary composite service solutions, because each SIDE service set is assembled into a real composite service in the preprocessing stage, which is then registered as a component web service in the service registry. The resulting composite solutions by this method can be directly instantiated as execution plans through existing methods for QoS-based service selection and service binding [18], [58], [60]. However, the composite solutions by the abstraction/refinement method cannot be directly instantiated as execution plans by the above methods when some previously defined abstract services are involved, because such services can only be instantiated in the practical execution stage of the composite service. Unlike traditional QoS-based methods, the selection and binding of specific services here should be dynamical and at run-time, since it depends on the run-time context of the composite service, and only the component service whose constraints can be satisfied in the current run-time context will be selected from the SIDE services set and bound to the corresponding abstract service.

Undoubtedly, the run-time selection and binding of component web services in the actual execution process of a composite service is bound to increase the algorithm complexity and computational cost. Because the composite solutions generated by the prepackaging method can be directly instantiated and then executed by an execution engine, while the solutions by the abstraction/refinement method cannot be instantiated and executed directly. The selection and instantiation of such abstract service takes extra cost and time.

In this process, for an abstract service WS_{abs} in a solution, we need to scan the SIDE services set corresponding to WS_{abs}, and parse these WSDL documents to determine whose constraints can be satisfied in the run-time context. Therefore, compared to the prepackaging method, it costs additional time at least on search and parsing WSDL documents in the given SIDE services set without considering other factors. Suppose that the number of services in the SIDE set SS is n, and the average time to parse a WSDL document and check whether the conditions can be met is $\bar{t}_p$. Since the average search time in set SS is $n/2$, the average time spent by this selection process is $\frac{n}{2}\bar{t}_p$.

In addition, we perform experiments to evaluate this process. Given a number n, a set of simulative SIDE services can be generated through the way of writing n different constraints into a certain WSDL document, respectively. Then, each time we randomly generate a condition and use it to search for the suitable service. In this way, we run the experiment 10 000 times for each test set so as to obtain the average search time. The result is shown in Fig. 9 in which we compare the two proposed methods. The prepackaging method does not need such a process for run-time selection, since its resulting

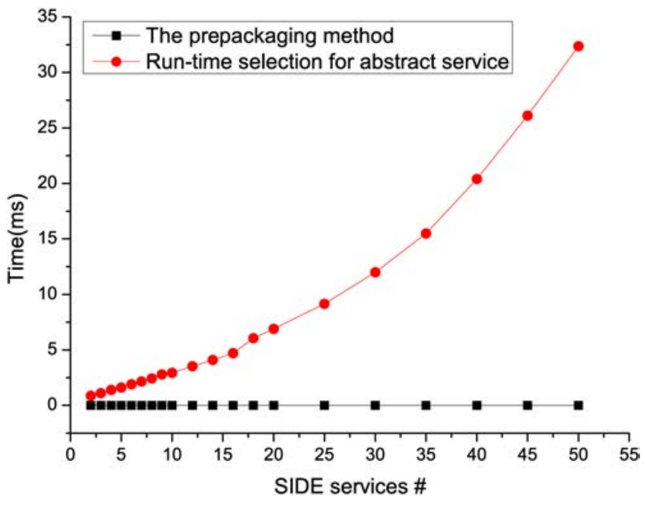


Fig. 9. Comparison of two methods for the dynamic selection process.

solutions can be directly instantiated and executed, and when the composite service assembled by a set of SIDE services is executed actually, the time spent on selecting an appropriate branch to execute is negligible.

Despite the higher complexity and extra cost, the abstraction/refinement method has higher flexibility and scalability. In a specific business scenario, the SIDE service set corresponding to a certain task is not static; on the contrary, it may be dynamically changing. There are many reasons to this changeability; for example, a service provider may add new similar services to expand its business, or cancel some existing services to meet the market needs. Take the shipping/delivery scenario in Section III for example, the B2C cooperation just provides one-day shipping service for the Shanghai area at the beginning. With the expansion of business, two-day shipping service for Jiangsu and Zhejiang provinces is added later, and now its business has been expanded to the whole country. Thus, they increase a new service, standard shipping service, for these newly added areas to meet the demands. In this business scenario, the SIDE service set corresponding to the shipping task is dynamically changing with the expansion of the company's business. The abstraction/refinement method is easier to deal with this changeability than the prepackaging method. In the former method, the process of abstraction in the preprocessing stage is very simple and easy to implement. Thus, as the SIDE service set changes, a new abstract service that captures this change can be generated easily and automatically. By contrast, in the prepackaging method, the results of preprocessing are real composite services. For each SIDE service set in the given service community, a composite service is generated to replace these component services. This includes a series of subprocesses, such as the transformation from high-level process models to executable code of a programming language for service composition, deployment, and publishing it to the service registry as a new service. This prepackaging process is much more cumbersome and labor-intensive than the abstraction process of the abstraction/refinement method. Moreover, once this process is completed, the composite service is fixed, and when some changes occur on a SIDE service set, we have to compose them again through such a cumbersome process. Obviously, in dealing with changes, the abstraction/refinement method is more extensible and flexible than the prepackaging method.

In the previous section, extensive experiments and their results show that the proposed graph search-based algorithm is admissible. Nevertheless, generating all the possible solutions for the general WSC problem is still a complicated and difficult task, owing to the nature of this problem as mentioned earlier. From the above experiments and results, we know that there are three important factors that can significantly affect the algorithm performance: service count in the registry, parameter range of every service, and the number of resulting solutions. With respect to our concerns in this paper, the proposed SIDE concepts and two preprocessing methods can improve the performance of the graph search-based algorithm through influencing two of the three above factors. First, they can reduce the size of service registry. In a certain service registry, supposing that there are two other SIDE services on average for each service of the registry, via the proposed techniques one can reduce the registry size to $1/3$ of the original by using a resulting service (whether it is composite or abstract one) instead of the corresponding SIDE services to participate in the subsequent composition. Second, their use can greatly reduce the number of resulting solutions. When a solution contains a service that has three other SIDE services, there must be three other similar solutions generated along with this one, and their only difference lies in these four SIDE services. If two different such services are contained simultaneously, $4 \times 4 = 16$ similar solutions will be generated. In contrast, only one solution can be generated by the proposed techniques for the two cases, respectively. The more such SIDE services in solutions, the more performance improvements gained from the use of the proposed concepts and methods.

VII. Conclusion

Service constraints are used to ensure the correct execution of the service or proper interaction with other services, thus having a significant impact on service composition. However, they were not put into account in the previous work. This work deals with automated service composition while considering them. Through some real-world business scenarios, we show that some available services in a specific business scenario have the same inputs and outputs, and can perform the same task, but have different constraints. This is common, but has been largely ignored by previous efforts for service composition. A novel solution to tackle this problem is proposed in this paper, which includes a graph search-based algorithm and two different preprocessing methods. The graph search-based algorithm can be applied to the general WSC problem, and generates all the possible solutions. For the two preprocessing methods, despite their differences, they both introduce conditional branch structures (explicitly or implicitly) into the process model of a solution in order to solve the problems brought by service constraints, and ensure the correct execution of the resulting composite service. Extensive experiments are conducted via a publicly available test set, and experimental results show the effectiveness and admissibility of our approach.

In addition, conditional branch structures are introduced into the process models of composite services to reflect SIDE services and their respective constraints so as to ensure the proper execution of composite services in different circumstances. In [51], we consider user preferences in service composition as given explicitly by users in their requests. Similarly, branch structures are introduced into the process models of composite services to satisfy such diverse and personalized preferences. The issue is often related to users too, and it is exactly the user-related information that determines the different choice of SIDE services. However, such information is often implicit in the composition scenarios, and not explicitly stated in the user request at the beginning. From this perspective, these efforts are made to achieve the same goal, i.e., automatic WSC supporting complex control constructs, in order to meet the complex needs under changing environments. This can overcome the deficiencies of the existing methods that rely on sequential structures, because the linear composite model can be applicable to only simple cases with a deterministic environment.

At present, whether web services existing in reality, or public test sets for service composition, most of them are based on WSDL descriptions, which do not support the definition and description of service constraints discussed in this paper. With the acceptance and application of semantic web service description languages such as SAWSDL and OWL-S, we are working to establish a test set that fully considers service constraints and other factors. This can help us perform more extensive experiments for the proposed methods, and also provide support for other future work that considers such factors and integrate with other research outcomes [72]–[87]. In addition, QoS information is planned to be used in order to directly get rid of the services and paths that do not meet the QoS standards, thereby reducing the search space and improving the efficiency of the proposed algorithm.

References

[1] D. Ardagna and B. Pernici, “Adaptive service composition in flexible processes,” *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 369–384, Jun. 2007.

[2] M. Beek, A. Bucchiarone, and S. Gnesi, “A survey on service composition approaches: From industrial standards to formal methods,” IEEE Comput. Soc. Press, New York, NY, USA, Tech. Rep. 2006TR-15, Istituto, 2006, pp. 15–20.

[3] D. Benslimane, Z. Maamar, Y. Taher, M. Lahkim, M. C. Fauvet, and M. Mrissa, “A multilayer and multiperspective approach to compose web services,” in *Proc. AINA*, May 2007, pp. 31–37.

[4] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, “Automatic composition of e-services that export their behavior,” in *Proc. ICSOC*, Dec. 2003, pp. 43–58.

[5] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, “Synthesis of underspecified composite e-services based on automated reasoning,” in *Proc. ICSOC*, Nov. 2004, pp. 105–114.

[6] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, “Automatic service composition based on behavioral descriptions,” *Int. J. Coop. Inf. Syst.*, vol. 14, no. 4, pp. 333–376, Dec. 2005.

[7] A. Brogi, S. Corfini, and R. Popescu, “Composition-oriented service discovery,” in *Proc. SC*, Apr. 2005, pp. 15–30.

[8] A. Brogi and S. Corfini, “Behaviour-aware discovery of web service compositions,” *Int. J. Web Serv. Res.*, vol. 4, no. 3, pp. 1–25, Jul.–Sep. 2007.

[9] A. Brogi, S. Corfini, and R. Popescu, "Semantics-based composition-oriented discovery of web services," *ACM Trans. Internet Technol.*, vol. 8, no. 4, article 19, Sep. 2008.

[10] Z. J. Ding, J. L. Wang, and C. J. Jiang, "An approach for synthesis Petri nets for modeling and verifying composite web services," *J. Inf. Sci. Eng.*, vol. 24, no. 5, pp. 1309–1328, Sep. 2008.

[11] S. V. Hashemian and F. Mavaddat, "A graph-based approach to web services composition," in *Proc. SAINT*, Feb. 2005, pp. 183–189.

[12] ICEBE 2005 organization committee (Oct. 2005). *Test Data for Web Services Challenge at ICEBE 2005* [Online]. Available: http://www.comp.hkbu.edu.hk/~ctr/wschallenge/

[13] W. Kongdenfha, H. R. Motahari-Nezhad, B. Benatallah, F. Casati, and R. Saint-Paul, "Mismatch patterns and adaptation aspects: A foundation for rapid development of web service adapters," *IEEE Trans. Serv. Comput.*, vol. 2, no. 2, pp. 94–107, Apr.–Jun. 2007.

[14] Q. A. Lang and S. Y. W. Su, "AND/OR graph and search algorithm for discovering composite web services," *Int. J. Web Serv. Res.*, vol. 2, no. 4, pp. 46–64, Oct.–Dec. 2005.

[15] X. Li, S. Madnick, H. Zhu, and Y. Fan, "An approach to composing web services with context heterogeneity," in *Proc. ICWS*, Jul. 2009, pp. 695–702.

[16] X. Li, S. Madnick, H. Zhu, and Y. Fan, "Reconciling semantic heterogeneity in web services composition," in *Proc. ICIS*, paper 20, Dec. 2009.

[17] X. Li, S. Madnick, and H. Zhu, "A context-based approach to reconciling data interpretation conflicts in web services composition," *ACM Trans. Internet Technol.*, to be published.

[18] Y. Liu, A. H. H. Ngu, and L. Zeng, "QoS computation and policing in dynamic web service selection," in *Proc. WWW*, May 2004, pp. 66–73.

[19] Z. Maamar, M. Lahkim, D. Benslimane, P. Thiran, and S. Subramanian, "Web services communities: Concepts and operations," in *Proc. WEBIST*, Mar. 2007, pp. 323–327.

[20] Z. Maamar, S. Subramanian, P. Thiran, D. Benslimane, and J. Bentahar, "An approach to engineer communities of web services: Concepts, architecture, operation, and deployment," *Int. J. E-Bus. Res.*, vol. 5, no. 4, pp. 1–21, Oct.–Dec. 2009.

[21] S. McIlraith, T. C. Son, and H. Zeng, "Semantic web services," *IEEE Intell. Syst.*, vol. 16, no. 2, pp. 46–53, Mar.–Apr. 2001.

[22] S. McIlraith and T. C. Son, "Adapting Golog for composition of semantic web services," in *Proc. KR*, Apr. 2002, pp. 482–493.

[23] B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid, "Composing web services on the semantic web," *VLDB J.*, vol. 12, no. 4, pp. 333–351, Nov. 2003.

[24] M. Mrissa, C. Ghedira, D. Benslimane, Z. Maamar, F. Rosenberg, and S. Dustdar, "A context-based mediation approach to compose semantic web services," *ACM Trans. Int. Tech.*, vol. 8, no. 1, article 4, p. 23, Nov. 2007.

[25] S. Narayanan and S. McIlraith, "Simulation, verification and automated compostion of web service," in *Proc. WWW*, May 2002, pp. 77–88.

[26] D. Nau, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN planning system," *J. Artif. Intell. Res.*, vol. 20, pp. 379–404, Dec. 2003.

[27] S. C. Oh, B. W. On, E. J. Larson, and D. Lee, "BF*: Web services discovery and composition as graph search problem," in *Proc. EEE*, Mar. 2005, pp. 784–786.

[28] S. C. Oh, D. Lee, and S. R. T. Kumara, "A comparative illustration of AI planning-based web services composition," *ACM SIGecom Exch.*, vol. 5, no. 5, pp. 1–10, Dec. 2005.

[29] S. C. Oh, D. Lee, and S. R. T. Kumara, "Web service planner (WSPR): An effective and scalable web service compostion algorithm," *Int. J. Web Serv. Res.*, vol. 4, no. 1, pp. 1–23, Jan.–Mar. 2007.

[30] S. C. Oh, D. Lee, and S. R. T. Kumara, "Effective web service composition in diverse and large-scale service networks," *IEEE Trans. Serv. Comput.*, vol. 1, no. 1, pp. 15–32, Jan.–Mar. 2008.

[31] W3C Member Submission. (2004, Nov. 22). *OWL-S: Semantic Markup for Web Services* [Online]. Available: http://www.w3.org/Submission/OWL-S/

[32] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics based automated service discovery," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 260–275, Apr.–Jun. 2012.

[33] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, Oct. 2007.

[34] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: A research roadmap," *Int. J. Coop. Inf. Syst.*, vol. 17, no. 2, pp. 223–255, Jun. 2008.

[35] S. R. Ponnekanti and A. Fox, "SWORD: A developer toolkit for web service composition," in *Proc. WWW*, May 2002, pp. 1–19.

[36] J. Rao, P. Küngas, and M. Matskin, "Application of linear logic to web service composition," in *Proc. ICWS*, Jun. 2003, pp. 3–9.

[37] J. Rao, P. Küngas, and M. Matskin, "Logic-based web service composition: From service description to process model," in *Proc. ICWS*, Jul. 2004, pp. 446–453.

[38] J. Rao and X. Su, "A survey of automated web service composition methods," in *Proc. SWSWPC*, Jul. 2004, pp. 43–54.

[39] J. Rao, P. Küngas, and M. Matskin, "Composition of semantic web services using linear logic theorem proving," *Inf. Syst.*, vol. 31, nos. 4–5, pp. 340–360, Jun.–Jul. 2006.

[40] W3C Recommendation. (2007, Aug. 28). *Semantic Annotations for WSDL and XML Schema (SAWSDL)* [Online]. Available: http://www.w3.org/TR/sawsdl/

[41] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau, "HTN planning for web service composition using SHOP2," *J. Web Semant.*, vol. 1, no. 4, pp. 377–396, Oct. 2004.

[42] W3C Recommendation. (2007, Apr. 27). *Simple Object Access Protocol (SOAP), Version 1.2* [Online]. Available: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

[43] S. Sohrabi, N. Prokoshyna, and S. McIlraith, "Web service composition via generic procedures and customizing user preferences," in *Proc. ISWC*, Nov. 2006, pp. 597–611.

[44] S. Sohrabi, N. Prokoshyna, and S. McIlraith, "Web service composition via the customization of Golog programs with user preferences," in *Conceptual Modeling: Foundations and Applications* (LNCS, vol. 5600), A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. Yu, Eds. New York, NY, USA: Springer-Verlag, 2009, pp. 319–334.

[45] S. Subramanian, P. Thiran, Z. Maamar, and D. Benslimane, "Engineering communities of web services," in *Proc. iiWAS*, Dec. 2007, pp. 57–66.

[46] W3C Member Submission. (2004, May 21). *SWRL: A Semantic Web Rule Language Combining OWL and RuleML* [Online]. Available: http://www.w3.org/Submission/SWRL/

[47] W. Tan, Y. S. Fan, and M. C. Zhou, "A Petri net-based method for compatibility analysis and composition of web services in business process execution language," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 94–106, Jan. 2009.

[48] W. Tan, Y. S. Fan, M. C. Zhou, and Z. Tian, "Data-driven service composition in enterprise SOA solutions: A Petri net approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 686–694, Jul. 2010.

[49] X. Tang, C. Jiang, and M. Zhou, "Automatic Web service composition based on Horn clauses and Petri nets," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13024–13031, Sep. 2011.

[50] OASIS Standard. (2004, Oct. 19). *Universal Description, Discovery and Integration Specification (UDDI), Version 3.0.2* [Online]. Available: http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

[51] P. W. Wang, Z. J. Ding, C. J. Jiang, and M. C. Zhou. "Automated web service composition supporting conditional branch structures," *Enterp. Inf. Syst.* DOI:10.1080/17517575.2011.584132, pp. 1–26, Jun. 2011, (published online).

[52] P. W. Wang, Z. J. Ding, C. J. Jiang, and M. C. Zhou, "Web service compositio techniques in a health care service platform," in *Proc. ICWS*, Jul. 2011, pp. 355–362.

[53] P. W. Wang, Z. J. Ding, C. J. Jiang, and M. C. Zhou. "Design and implementation of a web-service-based public-oriented personalized health care platform," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 4, pp. 941–957, Jul. 2013.

[54] The Icson Website [Online]. Available: http://www.icson.com/

[55] OASIS Standard. (2007, Apr. 11). *Web Services Business Process Execution Language (WS-BPEL), Version 2.0* [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

[56] W3C Recommendation. (2007, Jun. 26). *Web Service Description Language (WSDL), Version 2.0* [Online]. Available: http://www.w3.org/TR/2007/REC-wsdl20-20070626

[57] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau, "Automating DAML-S web services composition using SHOP2," in *Proc. ISWC*, Oct. 2003, pp. 195–210.

[58] P. C. Xiong, Y. S. Fan, and M. C. Zhou, "QoS-aware web service configuration," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 888–895, Jul. 2008.

[59] P. C. Xiong, Y. S. Fan, and M. C. Zhou, "Web service configuration under multiple quality-of-service attributes," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 2, pp. 311–321, Apr. 2009.

[60] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middlware for web service composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[61] L. J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Beijing, China: Tsinghua Univ. Press, 2007.

[62] X. Zheng and Y. Yan, “An efficient web service composition algorithm based on planning graph,” in *Proc. ICWS*, Sep. 2008, pp. 691–699.

[63] Y. Wu, C. G. Yan, Z. J. Ding, G. J. Liu, P. W. Wang, C. J. Jiang, and M. C. Zhou, “A novel method for calculating service reputation,” *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 634–642, Jul. 2013.

[64] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu, “Predicting quality of service for selection by neighborhood-based collaborative filtering,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 43, no. 2, pp. 428–439, Mar. 2013.

[65] P. Sun, C. J. Jiang, and M. C. Zhou, “Interactive web service composition based on petri net," *Trans. Inst. Meas. Contr.*, vol. 33, no. 1, pp. 116–132, Feb. 2011.

[66] P. Xiong, C. Pu, and M. C. Zhou, “Protocol-level service composition mismatches: A Petri net siphon based solution," *Int. J. Web Serv. Res.*, vol. 7, no. 4, pp. 1–20, Oct.–Dec. 2010.

[67] P. Xiong, Y. Fan, and M. C. Zhou, “A Petri net approach to analysis and composition of web services," *IEEE Trans. Syst., Man Cybern. A, Syst. Human*, vol. 40, no. 2, pp. 376–387, Mar. 2010.

[68] W. Tan and M. C. Zhou, *Business and Scientific Workflows: A Service-Oriented Approach*. Hoboken, NJ, USA: Wiley, 2013.

[69] M. B. Blake, W. Tan, and F. Rosenberg. “Composition as a service,” *IEEE Internet Comput.*, vol. 14, no. 1, pp. 78–82, 2010.

[70] X. Li, Y. Fan, Q. Z. Sheng, Z. Maamar, and H. Zhu, “A Petri net approach to analyzing behavioral compatibility and similarity of web services,” *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 3, pp. 510-521, May 2011.

[71] X. Li, Y. Fan, S. Madnick, Q. Z. Sheng, “A pattern-based approach to protocol mediation for web service composition,” *Inform. Softw. Technol.*, vol. 52, no. 3, pp. 304–323, 2010.

[72] J. Cao, W. Zhang, and W. Tan. “Dynamic control of data streaming and processing in a virtualized environment,” *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no 2, 365–376, Apr. 2012.

[73] M. Dotoli, M. P. Fanti, C. Meloni, and M. C. Zhou, “A multi-level approach for network design of integrated supply chains,” *Int. J. Prod. Res.*, vol. 43, no. 20, pp. 4267–4287, Oct 15, 2005.

[74] M. Dotoli, M. P. Fanti, C. Meloni, and M. C. Zhou, “Design and optimization of integrated e-supply chain for agile and environmentally conscious manufacturing,” *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 36, no. 1, pp. 62–75, Jan. 2006.

[75] Y. Du, C. Jiang and M. C. Zhou, “Modeling and analysis of real-time cooperative systems using Petri nets,” *IEEE Trans. Syst., Man, Cybern. A, Syst, Humans*, vol. 37, no. 5, pp. 643–654, Sep. 2007.

[76] K. Huang, Y. Fan, W. Tan, and M. Qian. “BSNet: A network-based framework for service-oriented business ecosystem management,” *Concurrency Comput.: Practice Exper.*, vol. 25, no. 13, pp. 1861–1878, 2013.

[77] Z. W. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*, New York, NY, USA: Springer, 2009.

[78] W. Tan, P. Missier, I. Foster, R. Madduri, D. De Roure, and C. Goble, “A comparison of using taverna and BPEL in building scientific workflows: the case of cagrid,” *Concurrency Comput.: Practice Exper.*, vol. 22, no. 9, pp. 1098–1117, 2010.

[79] W. Tan, I. Foster, R. Madduri, “Combining the power of taverna and cagrid: scientific workflows that enable web-scale collaboration,” *IEEE Internet Comput.*, vol. 12, no. 6, pp. 61–68, Nov.–Dec. 2008.

[80] W. Tan, M B. Blake, I. Saleh, and S. Dustdar, “Social-network-sourced big data analytics,” *IEEE Internet Comput.*, vol. 17, no. 5, pp. 62–69, Sep.–Oct. 2013.

[81] W. Tan, J. Zhang, and I. Foster, “Network analysis of scientific workflows: A gateway to reuse,” *IEEE Computer*, vol. 43, no. 9, pp. 54–61, Sep. 2010.

[82] N. Q. Wu and M. C. Zhou, *System Modeling and Control With Resource-Oriented Petri Nets*, New York, NY, USA: CRC Press, 2010.

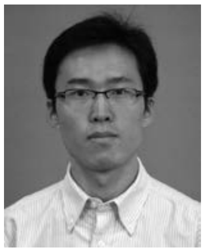
[83] Y. Zheng, Y. Fan, and W. Tan, “Towards workflow simulation in service-oriented architecture: An event-based approach,” *Concurrency Comput.: Practice Exper.*, vol. 20, no. 4, pp. 315–330, 2008.

[84] M. C. Zhou and F. DiCesare, “Petri net synthesis for discrete event control of manufacturing systems.” London, I.K.: Kluwer Academic, 1993.

[85] M. C. Zhou and K. Venkatesh, *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach.* Singapore: World Scientific, 1998.

[86] H. Zhu and M. C. Zhou, “Roles in information systems: A survey,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 3, pp. 377–396, May 2008.

[87] Zhu, H., M. C. Zhou and P. Seguin, “Supporting software development with role,” *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 36, no. 6, pp. 1110–1123, Nov.



PengWei Wang received the B.S. and M.S. degrees in computer science from the Shandong University of Science and Technology, Qingdao, China, in 2005 and 2008, respectively. He is currently pursuing the Ph.D. degree at the Department of Computer Science and Technology, Tongji University, Shanghai, China.

His current research interests include services computing, web services, and Petri nets.

ZhiJun Ding received the M.S. degree from the Shandong University of Science and Technology, Taian, China, in 2001, and the Ph.D. degree in computer science from Tongji University, Shanghai, China, in 2007.

He is currently an Associate Professor with the Department of Computer Science and Technology, Tongji University. He has authored more than 50 papers in domestic and international academic publications. His current research interests include service computing, semantic web, formal engineering, Petri nets, and workflows.

ChangJun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1995.

He finished his post-doctoral work at the Institute of Computing, Chinese Academy of Sciences, in 1997, and continued his research at the City University of Hong Kong, New Kowloon, Hong Kong, as a Visiting Professor in 1998. Currently, he is a Professor with the Department of Computer Science and Technology, Tongji University, Shanghai, China. He is currently the Leading Scientist with the National Basic Research Program of China (973 Program) on the project “Model and theory in Internet information service.” His current research interests include concurrency theory, Petri nets, formal verification, services computing, and application research on massive information services and wireless networks.

MengChu Zhou (S’88–M’90–SM’93–F’03) received the B.S. degree in control engineering from Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and is a Distinguished Professor of electrical and computer engineering. He is currently a Professor with Tongji University, Shanghai, China. His interests include intelligent automation, Petri nets, sensor networks, semiconductor manufacturing, web services, and workflows. He is a fellow of the American Association for the Advancement of Science and IFAC.