A Petri Net Approach to Analyzing Behavioral Compatibility and Similarity of Web Services

Xitong Li, Yushun Fan, Quan Z. Sheng, *Member, IEEE*, Zakaria Maamar, and Hongwei Zhu

***Abstract*—Web services have become the technology of choice for service-oriented computing implementation, where Web services can be composed in response to some users' needs. It is critical to verify the compatibility of component Web services to ensure the correctness of the whole composition in which these components participate. Traditionally, two conditions need to be satisfied during the verification of compatibility: reachable termination and proper termination. Unfortunately, it is complex and time consuming to verify those two conditions. To reduce the complexity of this verification, we model Web services using colored Petri nets (PNs) so that a specific property of their structures is looked into, namely, *well structuredness*. We prove that only reachable termination needs to be satisfied when verifying behavioral compatibility among well-structured Web services. When a composition is declared as valid and in the case where one of its component Web services fails at run time, an alternative one with similar behavior needs to come into play as a substitute. Thus, it is important to develop effective approaches that permit one to analyze the similarity of Web services. Although many existing approaches utilize PNs to analyze behavioral compatibility, few of them explore further appropriate definitions of behavioral similarity and provide a user-friendly tool with automatic verification. In this paper, we introduce a formal definition of context-independent similarity and show that a Web service can be substituted by an alternative peer of similar behavior without intervening other Web services in the composition. Therefore, the cost of verifying service substitutability is largely reduced. We also provide an algorithm for the verification and implement it in a tool. Using the tool, the verification of behavioral similarity of Web services can be performed in an automatic way.**

***Index Terms*—Behavioral compatibility, behavioral similarity, Petri net (PN), service composition, Web service.**

Manuscript received February 15, 2009; revised December 27, 2009 and May 25, 2010; accepted July 29, 2010. Date of publication January 9, 2011; date of current version April 15, 2011. This work was supported in part by the National Natural Science Foundation of China under Grants 60674080 and 61033005, by the National High Technology Research and Development (863) Program of China under Grant 2007AA04Z150, and by the National Basic Research Development (973) Program of China under Grant 2006CB705407. The work of Q. Z. Sheng was supported in part by the Australian Research Council Discovery Grant DP0878367. This paper was recommended by Associate Editor M. P. Fanti.

X. Li is with the MIT Sloan School of Management, Cambridge, MA 02142 USA, and also with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: xitongli@mit.edu; lxt04@mails.tsinghua.edu.cn).

Y. Fan is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: fanyus@tsinghua.edu.cn).

Q. Z. Sheng is with the School of Computer Science, The University of Adelaide, Adelaide, S.A. 5005, Australia (e-mail: qsheng@cs.adelaide.edu.au).

Z. Maamar is with the College of Information Technology, Zayed University, Dubai, United Arab Emirates (e-mail: zakaria.maamar@zu.ac.ae).

H. Zhu is with the College of Business and Public Administration, Old Dominion University, Norfolk, VA 23529 USA (e-mail: hzhu@odu.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/TSMCA.2010.2093884

I. Introduction

THE DEVELOPMENT of Web applications are cumbersome, costly, and time consuming due to various reasons such as the increasing complexity of end-users' needs, heterogeneity of various IT systems, and continuous pressure to deliver up-to-date and reliable systems to end users [1], [2]. Web services are nowadays among the technologies of choice for the development of Web applications. Web services are universally accessible software components/applications that can be discovered and invoked using open-standard Internet protocols [3]. Composition, whereby multiple independent Web services are assembled to accomplish a more complex task, is one of the key motivations to embrace Web services [4], [5].

To guarantee the successful execution of composition scenarios, component Web services from independent providers need to be verified in order to ensure that mutual interactions between them do not lead to any conflicts or deadlocks [6]. Specifically, we need to verify the *compatibility* of the participating Web services. There are three aspects of compatibility: *syntactic*, *semantic*, and *behavioral* [7]–[9]. Syntactic compatibility means that the structural interfaces of the interacting Web services are consistent, e.g., the labels of corresponding messages and data types are the same. Currently, the Web Service Description Language (WSDL) provides a standardized way to specify these structural interfaces. Semantic compatibility means that the interacting Web services exchange information that can be understood in a consistent and unambiguous way. Finally, behavioral compatibility means that the interacting Web services agree on what to expect from each other in terms of operations to execute, outcomes to deliver, and messages to send and receive. Behavior herein refers to the possible sequences of messages that Web services mutually exchange in response to some triggers or business dependences. Although syntactic and semantic compatibilities are critical, this paper focuses on Web services' behaviors in terms of behavioral compatibility and similarity. Intuitively, behavioral similarity means that the behaviors of two Web services overlap to a certain extent and one Web service can be substituted by the other in a composition.

There exist a good number of approaches to verifying behavioral compatibility of Web services. Different formalisms are used, including finite-state machine (FSM) [8], [10], [11], process algebra/pi calculus [12], [13], and Petri nets (PNs) [14]–[17]. These formalisms reflect the different ways of addressing the compatibility issue. In [8], behavioral compatibility means that the order (i.e., sequence) of messages exchanged with external partners and observed from one Web service is

preserved if it is observed from another Web service. Unfortunately, this notion of behavioral compatibility cannot be used to verify whether the respective processes of the interacting Web services can successfully terminate, as another Web service may exhibit additional behaviors that cause certain errors. The work presented in [18] focuses on high-level units of collaborative business processes and presents a scenario-based technique for the compatibility verification among the collaborations. The scenario-based modeling technique is different from the message-based modeling approach [14]–[17] that is adopted in our work. In [14], the concept of *weak soundness* is adopted to define behavioral compatibility. However, the weak soundness derived from the traditional workflow theory is too narrow; it does not reflect the properties of Web services like autonomy, loosely coupledness, and asynchronous communication. For example, a correct composition may still contain unprocessed internal/external messages even when the component Web services have successfully terminated. However, this situation is not allowed under the condition of weak soundness. Furthermore, the internal choices of Web services are not addressed by the models in [14]. To this end, this paper aims to address these shortcomings and provides an appropriate notion of behavioral compatibility for Web services composition. Specifically, we develop the Service Workflow Net (SWN) formalism based on colored PNs (CPNs) in order to model Web services. The formalism depicts not only the observable message exchange sequences but also the internal and external messages buffered and the choices made by the component Web services. These details are useful in the analysis of behavioral compatibility and similarity of Web services. For example, the condition of behavioral compatibility in this paper is relaxed to allow unprocessed internal messages when the processes of interacting Web services complete.

Traditionally, two conditions need to be checked in order to verify behavioral compatibility of Web services: *reachable termination* and *proper termination* [14], [15], [17]. Reachable termination means that each Web service in a composition can eventually reach a terminal state to announce its completion. Proper termination means that all control flows [19] of the Web service should terminate when the composition reaches a terminal state. Both conditions can be used to address the behavioral adaptation of Web services [20]. Unfortunately, verifying proper termination requires examining all reachable terminal states of the composition. In the case of a large number of states, the enumerative examination of these states is very costly and time consuming. It is thus deemed appropriate to relax the condition of proper termination so that the complexity of verifying compatibility can be reduced. To this end, we look into the structure of the Petri-net-based models of Web services and introduce the concept of *well structuredness* which characterizes a property of the structure of Web services. We show that only the requirement of reachable termination needs to be satisfied when verifying behavioral compatibility of well-structured services.

Due to the dynamic environment in which Web services run, a Web service involved in a composition may fail to respond to client's requests or to maintain a satisfying quality-of-service level [21]. In both situations, an alternative Web service of similar behavior needs to be identified and used to substitute the failed one. This becomes another significant research issue in service-oriented computing (SOC), i.e., Web services substitution [22], [23]. Analyzing behavioral similarity, also known as substitutability or replaceability [12], [23], [24], is thus of great importance to Web services substitution and goes along with the general idea of behavioral compatibility between the substituting Web service and the partners of the substituted Web service.

In [8], the notion of similarity based on execution traces is proposed, which states that Web service ws_2 is similar to Web service ws_1 if each possible trace in ws_1 must be preserved in ws_2. The notion of simulation is used in [10] to compare protocols of Web services with respect to their complete execution trees. However, neither trace equivalence nor simulation is exactly suitable for Web services, because the notion based on execution traces is too broad while the notion based on simulation is too narrow [12], [25]. Based on the principle of behavior subtyping [26], the concept of substitutability is presented to analyze protocols of software components [24]. In [12], the notion of replaceability of Web services is proposed by using behavior subtyping. However, neither the formal condition of substitutability (i.e., replaceability) nor an algorithm for verifying substitutability is presented in [12] and [24].

Intuitively, ws_2 is similar to ws_1 if ws_2 behaves like ws_1 in a comparable situation. Several notions of behavioral similarity are reported in the literature with different interpretations of the terms "behave like" and "comparable situation" [25], [27]. "Behave like" means that all possible sequences of exchanged messages of ws_1 are preserved by ws_2 and ws_2 does not extend the behavior of ws_1. "Comparable situation" means that ws_1 is substituted by ws_2 in a transparent way to the partners of ws_1, i.e., ws_2 should be compatible with any Web service that is compatible with ws_1.

With these two interpretations in mind, we propose the concept of behavioral similarity which is considered to be context independent. The context of ws_1 herein refers to the interacting partners of ws_1 in a composition. In the case where the composition is valid, the interacting partners of ws_1 should be compatible with ws_1. In the sense of context-independent similarity, substituting ws_1 in the composition with a similar service ws_2 can be directly performed, since ws_2 is compatible with other services in this composition. The concept of context-independent similarity guarantees the compatibility of ws_2 with the partners of ws_1 in the composition. Informally, the sufficient conditions that ws_2 is similar to $ws1$ are as follows: 1) ws_2 preserves the behavior of ws_1; 2) ws_2 does not extend the behavior of ws_1; and 3) the process of ws_2 terminates when ws_1 terminates. Similar concepts of context-independent similarity of Web services can be found in [23], [25], and [27]. However, none of these efforts formalize the conditions that context-independent similarity should satisfy. Without a formal definition of the conditions, the verification of behavioral similarity cannot be performed automatically. Furthermore, for analyzing the behaviors of Web services, the global and internal choice branches play a significant role and need to be represented in Web service models [28]. However, the choice branches are ignored in [27]. The reason is that the

formalism adopted in [27], i.e., labeled transition systems, has weaker capability than PNs with respect to depicting internal choice branches or asynchronous communications of Web services. To address these weaknesses of prior research works, we have investigated the observable message sequences of Web services and the relations of similarity between the message sequences. Based on our investigation, we provide the sufficient conditions of context-independent similarity and a corresponding algorithm for verifying the similarity. We develop a tool to automate the verification of similarity between Web services using this algorithm. The tool is integrated within the Platform-Independent PN Editor (PIPE) [29].

The rest of this paper is organized as follows. Section II presents the formalism for modeling the behaviors of Web services. Section III describes our approach to the compatibility analysis of the behaviors of Web services. Section IV proposes our approach to similarity analysis and the algorithm to verify the similarity. Section V describes the tool for verifying the behavioral similarity and equivalence of Web services. Finally, Section VI concludes this paper.

II. Formalism of SWNs

We develop SWNs as the underlying formalism for depicting the behaviors of Web services. An SWN extends a CPN [30]. CPNs have a well-defined semantics for describing both states and actions of concurrency systems and a graphical representation which is easy to understand. The behavior of a CPN model can be analyzed by means of either several formal analysis methods or simulation. CPN-based models not only provide a formalism to depict the internal logics and message exchange sequences (i.e., behaviors) but also encompass the mechanisms of compatibility and similarity analysis of Web services. Therefore, in the literature, many initiatives propose to model the behaviors of Web services based on CPNs [14]–[17]. The definition of the CPN is presented hereinafter. Detailed definitions can be found in [30].

Definition 1 (CPN): A CPN is a tuple $CPN = (\Sigma, P, T, F, C, G, E, M_i)$, where its elements are defined as follows.

1) Σ is a finite set of nonempty types, also called color set. It is a multiset that allows duplicate elements.
2) P is a finite set of places.
3) T is a finite set of transitions, disjoint from P.
4) $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation F. C is a color function, defined from P into Σ.
5) G is a guard function, i.e., $G : T \to BoolExpression$, and $\forall t \in T : Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \Sigma$.
6) E is the arc expression function, i.e., $E : F \to BoolExpression$, and $\forall f \in F : Type(E(f)) = C(p)_{\mathrm{MS}} \wedge Type(Var(E(f))) \subseteq \Sigma$, where p is the place that is connected to f.
7) M_i is an initiation function, i.e., $M_i : P \to \Sigma$.

In our work, we model the behavior of a Web service using an SWN defined as follows.

Definition 2 (SWN): An SWN is an extended CPN with labels, denoted as $SWN = (CPN, A, L)$, where the CPN distinguishes three disjoint types of places.

1) $P = P^{\mathrm{IC}} \cup P^{\mathrm{IM}} \cup P^{\mathrm{EM}}$, where P^{IC} is the set of internal control places, P^{IM} is the set of internal message places, and P^{EM} is the set of external message places. $P^{\mathrm{IC}} \cap P^{\mathrm{IM}} = \emptyset$, $P^{\mathrm{IC}} \cap P^{\mathrm{EM}} = \emptyset$, and $P^{\mathrm{IM}} \cap P^{\mathrm{EM}} = \emptyset$.
2) P^{IC} has two special places, i.e., i and o. The initial place i satisfies $\bullet i = \emptyset$, and the final place o satisfies $o\bullet = \emptyset$.
3) If all places in P^{EM} and all arcs connecting to these places are removed, the workflow net of the Web service is obtained. It is denoted as $CPN = \Phi(SWN)$. If a transition t^* to a CPN that connects o to i (i.e., $\bullet t^* = \{o\}, t^*\bullet = \{i\}$) is added, then the resulting PN becomes strongly connected.
4) $M_i(i) \neq 0$ and $\forall p \in P$, $p \neq i$, we have $M_i(p) = 0$. M_i is the initial marking.
5) A is a finite set of labels. $\forall a \in A$ is a label of string type, and $\exists \tau \in A$ is a "silent" label.
6) L is a labeling function, i.e., $L : T \to A$.

In Definition 2, the places in P^{IC} represent the control flow [19] of the Web service. The places in P^{IM} and P^{EM} represent the internal and external message buffers, respectively. Each transition connecting to P^{IM} or P^{EM} is an interaction transition representing an action that sends or receives messages. A contains the operating semantics of the Web service, and every label in A represents an operation name. Each transition of the SWN is mapped onto an operation name by L. Silent label τ is mapped onto those transitions that represent internal actions that are unobservable from the external environment. To differentiate the transitions of incoming/outgoing messages, the polarity function for these transitions is defined and denoted as $\varphi : T \to \{+1, 0, -1\}$, i.e., $\varphi(t) = -1$, if t depicts an action of sending messages; $\varphi(t) = +1$, if t depicts an action of receiving messages; and $\varphi(t) = 0$, if t depicts an action without sending or receiving messages. Apparently, $\varphi(\tau) = 0$.

Various specification languages for Web services composition exist, such as Business Process Execution Language (BPEL), Web Services Choreography Description Language, and Web Service Choreography Interface. Among them, BPEL obtains the dominance and has been proposed by OASIS as an industrial standard supported by major software vendors such as IBM, Oracle, and SAP. The behaviors of Web services described in BPEL specification can be smoothly transformed into SWNs. Detailed approaches to the transformation are given in [16] and [17]. It should be noted that the Petri-net-based service models used by those approaches [16], [17] are different from the models of Definition 2 in the sense that only external message places are depicted in their models while internal message places are ignored. Hence, the requirements of behavioral compatibility proposed in their works are too narrow: They require that there exists no token in the internal message places when Web services processes successfully terminate. In fact, the internal message places may retain tokens in such cases which reflect the properties of Web services, like loosely coupledness and asynchronous communication.

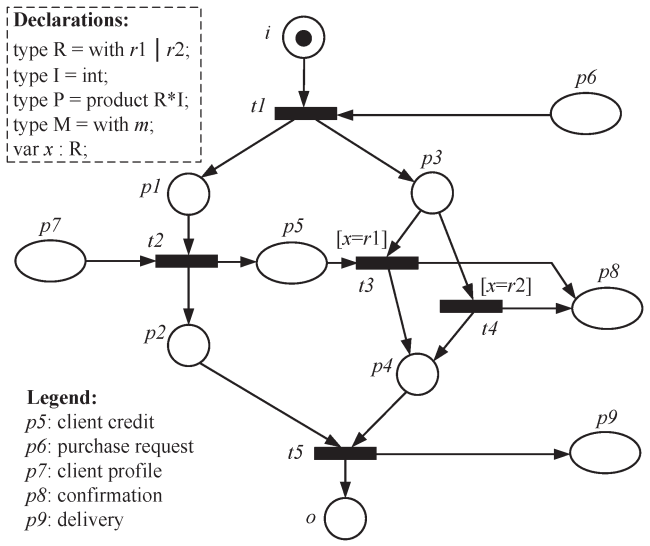


Fig. 1. SWN of Web service ws_1.

Fig. 1 shows a graphical representation of Web service ws_1 using an SWN. For the declarations in Fig. 1, the color set R has two elements $r1$ and $r2$ corresponding to two different request categories processed by ws_1. The color set M has one element m representing the messages. Type I represents the integers, and type P is the Cartesian product of the two color sets R and I—these declarations will be used throughout the examples in this paper. In Fig. 1, we use circles to represent the internal control places having the type P as the color set and ovals to represent the message places having the type M. Note that $p5$ is an internal message place, while $p6 \sim p9$ are external message places. After receiving the purchase request from a client, ws_1 initiates two parallel branches. One branch expects to receive the client's profile and provides the client's credit information (modeled by place $p5$) to the other branch. According to the request category (e.g., $r1$ or $r2$), the other branch decides whether the client's credit information is required. If the purchase request x is $r1$ with specials, ws_1 needs to check the client's credit and then sends the confirmation. Otherwise, ws_1 sends confirmation directly in the case that the request x is $r2$. Both branches are synchronized by the delivery action. As shown in Fig. 1, transition $t3$ has a guard "$[x = r1]$," and transition $t4$ has a guard "$[x = r2]$."

When Web services are composed together, some of their external message places are merged to form internal message places of the composite Web service. These internal message places should be identified in the SWN models of Web services, such as $p5$ in Fig. 1. Apparently, internal message places (e.g., $p5$) are different from control-flow places (e.g., $p1 \sim p4$) and external message places (e.g., $p6 \sim p9$)—we will show later how internal message places affect the behavioral compatibility of Web services. As a result, we consider that our service model as in Definition 2 is more reasonable than other service models presented in [16] and [17], since they are developed based on the traditional workflow theory and ignore internal message places that result from the composition of multiple Web services.

At the initial marking M_i, only the initial place i has tokens, i.e., $M_i = [i]$. A transition t is enabled in marking M_j, denoted as $M_j[t >$. After t fires, the marking changes from M_j to M_k, denoted as $M_j[t > M_k$. After a sequence of transitions (i.e., $\sigma \in T^*$) fires, the marking changes from M_i to M_k, denoted as $M_i[\sigma > M_k$. The set of markings that are reachable from a marking M is denoted as $R(M) = \{M' | \exists \sigma \in T^*, M[\sigma > M'\}$. A marking M_f is a final marking if $M_f \in R(M_i)$ and $M_f(o) \neq 0$. The set of all final markings is denoted as $\mathcal{F}$. Detailed notations of CPN can be found in [30].

With the capability of detecting all completion incompatibilities (e.g., deadlocks), the concept of completion compatibility has become the most recognized notion of behavioral compatibility of Web services [17], [23], [31]. We thus adopt it as the foundation of our formal definition of behavioral correctness of SWN. Intuitively, an SWN is behaviorally correct if it always interacts properly with its partners, i.e., its process can successfully terminate without any deadlock. We distinguish the final markings of SWN into correct and false ones.

Definition 3 (Correct Final Marking): A final marking M_f of SWN is correct, denoted as M_{cf}, iff $\forall p \in P^{\text{IC}}$, $p \neq o$: $M_f(p) = 0$. Otherwise, M_f is false. The set of all correct final markings is denoted as $\mathcal{F}_C$.

According to Definition 3, when an SWN is in a correct final marking, the final place o must contain some token(s), and any other internal control place contains no token. Note that there may remain some token(s) in internal and/or external message places. In such a case, we consider that the process of the SWN successfully terminates. Take ws_1 in Fig. 1 as an example. The client credit is provided in place $p5$ regardless of the request category. The message token in $p5$ will not be consumed in the case of nonspecial purchase request, i.e., the guard $[x = r2]$ is true. However, the processes of both branches of ws_1 can still successfully terminate in this situation. Based on the notion of Definition 3, $[p5, o]$ is one of the correct final markings of ws_1, and $[o]$ is the other correct final marking.

Definition 4 (Reachable Transition Sequence): A transition sequence $\sigma = \langle t_1, t_2, \ldots, t_n \rangle \in T^*$ is a reachable transition sequence of an SWN iff $\exists M : M_i[\sigma > M$. $\Sigma(SWN)$ is the set of all reachable transition sequences of the SWN, i.e., $\Sigma(SWN) = \{\sigma | \sigma \in T^*, \exists M : M_i[\sigma > M\}$.

Definition 5 (Behavioral Correctness): The behavior of an SWN is correct iff, when the environment satisfies the requirement of external messages, for any reachable nonfinal marking of SWN, there exists a transition sequence that leads to a final marking, and all final markings must be correct, i.e., as follows.

1) $\forall M \in R(M_i)$, $M \notin \mathcal{F}$, and $\sigma \in T^*$, such that $M[\sigma > M_f \in \mathcal{F}$.
2) $\mathcal{F}_C = \mathcal{F}$.

By the environment satisfying the requirement of external messages, we mean that the environment always sends/receives a message whenever the SWN is expected to receive/send such a message. The requirement of the "perfect" environment removes the possibility that the external partners are the causes of critical errors (e.g., deadlocks) during the execution of the SWN. Thus, the reachability of the correct final markings can

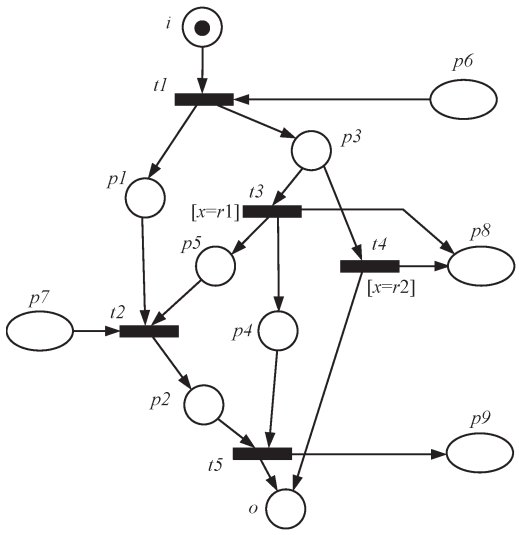


Fig. 2. SWN of Web service ws_2 with incorrect behavior.

be used to verify the behavioral correctness of the SWN. We assume that such a "perfect" environment in Definition 5 exists, which is a reasonable assumption, as discussed in [14]. When performing verification, we just remove all the places in P^{EM} and the arcs connecting to them. In other words, the verification of correctness only needs to be performed on the workflow net of the SWN, i.e., $CPN = \Phi(SWN)$. The first condition of Definition 5 requires that every marking that is reachable from the initial marking can reach a final marking, which is referred to as *reachable termination* (also known as deadlock freeness). The second condition of Definition 5 requires that the reachable final markings are correct in the sense of Definition 3, which is referred to as *proper termination*.

It should be pointed out that the existing Petri-net-based models of Web services only consider external message places rather than internal ones, such as in [14]–[17]—in those models all internal places are control places. According to the traditional workflow theory, the existence of any token in the internal places is not allowed when the interacting Web services processes complete. Otherwise, the Web services are considered to be incompatible. In contrast, Definition 5 requires a more relaxed condition than the notions of soundness of Web services proposed in [14] and [17], as Definition 5 allows the possible existence of token(s) in the internal and/or external message places after the process of the Web service terminates. Again, take ws_1 in Fig. 1 as an example. $[p5, o]$ and $[o]$ are both correct final markings of ws_1 that are reachable from its initial marking $[i]$, which is verified using the methods of reachable analysis. In fact, the behavior of ws_1 is correct according to Definition 5.

Fig. 2 shows the SWN of another Web service ws_2 with incorrect behavior. For the sake of simplicity, the declarations and business semantics of ws_2 (and the other examples in the rest of this paper) are omitted. Note that transition $t4$ fires when the guard $[x = r2]$ is true. In such a case, ws_2 reaches the marking $[p1, o]$, which is an incorrect final marking, because $p1$ is a control place and cannot contain token(s) in a correct final marking. Thus, ws_2 has incorrect behavior according to Definition 5.

III. Compatibility Analysis of Web Services

If a Web service with incorrect behavior participates in a composition, it may result in serious errors, e.g., deadlocks, during the execution of the composition. In the following discussion, we only consider the case where the component Web services in a composition have correct behaviors (according to Definition 5). In this case, whether the composition can properly execute or not depends on the behavioral compatibility of its participating Web services.

Definition 6 (Behavioral Compatibility): Two SWNs SWN_1 and SWN_2 are compatible, denoted as $SWN_1 \sim SWN_2$, iff the behavior of their composite Web service $SWN = SWN_1 \oplus SWN_2$ is correct.

In the aforementioned definition, the composition operator $\oplus$ is used to indicate a composite Web service consisting of two component Web services. In other words, $SWN = SWN_1 \oplus SWN_2$ means that SWN is a composite Web service composed of SWN_1 and SWN_2. The formal definition of Web services composition is based on place fusion and presented in our previous work [17]. It is straightforward to see the symmetry property of behavioral compatibility, i.e., $SWN_1 \sim SWN_2 \Leftrightarrow SWN_2 \sim SWN_1$.

As mentioned in Definition 5, the notion of behavioral correctness requires two conditions, i.e., *reachable termination* and *proper termination*. According to Definition 6, analyzing the behavioral compatibility is subject to verifying these two conditions. To simplify the verification, we investigate a set of Web services with specific structures, i.e., those that are *well structured*. For well-structured Web services, we will conclude later that only the condition of reachable termination (i.e., deadlock freeness) is sufficient to claim the behavioral compatibility.

Definition 7 (Well-Structured SWN): An SWN is well structured iff, after removing all places in P^{EM} and all arcs connecting to these places, the remaining workflow net, i.e., $CPN = \Phi(SWN)$, is well structured.

The definition of well-structured Web services is based on the concept of well-structured workflow nets [26], [32]. Intuitively, well structuredness characterizes a property of the structure of Web services. That is, two parallel branches of a Web service initiated by an AND split should not be joined by an OR joint; they should instead be synchronized by an AND joint. Similarly, two alternative branches created by an OR split should not be synchronized by an AND joint; they should be joined by an OR joint.

Fig. 3 shows the SWN of a well-structured Web service ws_3, where the OR split at $p1$ is joined by the OR joint at o. However, the SWN of ws_2 in Fig. 2 is not well structured because the AND-split transition $t1$ is complemented by the OR-joint place o. There are two disjoint paths leading from $t1$ to o: one via $p1$, $t2$, $p2$, and $t5$ and the other via $p3$ and $t4$. The SWN of ws_1 shown in Fig. 1 is another example of an SWN, which is not well structured, as the AND split $t1$ is complemented by the OR joint $p4$.

Theorem 1: Given two well-structured services SWN_1 and SWN_2 and their composite Web service $SWN = SWN_1 \oplus SWN_2$, each final marking of SWN that is reachable from the initial marking is correct, i.e., $\mathcal{F}_C = \mathcal{F}$.

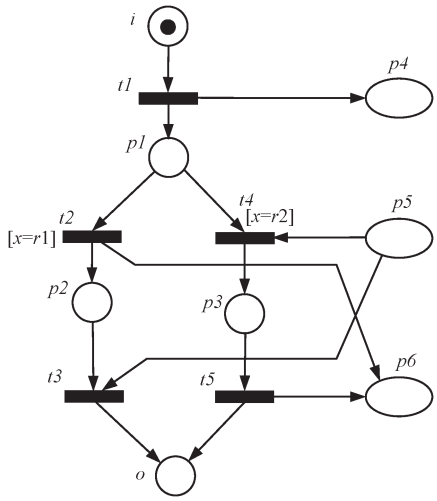


Fig. 3. SWN of a well-structured Web service ws_3.

Proof (Proof by Contradiction): If we assume that $\mathcal{F}_C \neq \mathcal{F}$, then $\exists M_f \in \mathcal{F}$ and $M_f \notin \mathcal{F}_C$ since $\mathcal{F}_C \subseteq \mathcal{F}$. With $M_f \in \mathcal{F}$, we have $M_f \in R(SWN, M_i)$ and $M_f(o) \neq 0$. With $M_f \notin \mathcal{F}_C$, we have $\exists p \in P^{\mathrm{IC}}$, where $p \neq o$, and $M_f(p) \neq 0$. According to the definition of Web service composition, $P^{\mathrm{IC}} = P_1^{\mathrm{IC}} \cup P_2^{\mathrm{IC}} \cup \{i, o\}$. If $p = i$, then $\exists M'_f \in \mathcal{F}$, such that $M'_f(i_1) \neq 0$ and $M'_f(i_2) \neq 0$. Hence, we can assume that $p \in P_1^{\mathrm{IC}}$ or $p \in P_2^{\mathrm{IC}}$. Without loss of generality, let $p \in P_1^{\mathrm{IC}}$. As M_f is a final marking of SWN, there exists a final marking M_{1f} of SWN_1 that is reachable, such that $M_{1f}(o_1) \neq 0$ and $M_{1f}(p) \neq 0$. Because of the second condition of Definition 2, there is an elementary path C_1 of SWN_1 between i_1 and o_1, such that $p \neq C_1$. Similarly, there is another elementary path C_2 of SWN_1 between i_1 and o_1, such that $p \in C_2$, and $C_2 \neq C_1$. Since both C_1 and C_2 start from i_1 and terminate at o_1, and $M_{1f}(o_1) \neq 0$ and $M_{1f}(p) \neq 0$, then there exists a transition $t \in C_1 \cap C_2$. Moreover, we know that $o_1 \in C_1 \cap C_2$. This situation is contrary to the assumption that SWN_1 is well structured. Thus, Theorem 1 is proved. □

As mentioned before, two conditions need to be verified for checking the behavioral compatibility of Web services, i.e., reachable termination and proper termination. Owing to Theorem 1, we conclude that only the condition of reachable termination (i.e., deadlock freeness) needs to be considered when verifying the behavioral compatibility of well-structured services. Based on Theorem 1, we have the following result.

Theorem 2: Given two well-structured services SWN_1 and SWN_2 and their composite Web service $SWN = SWN_1 \oplus SWN_2$, SWN_1 and SWN_2 are compatible, i.e., $SWN_1 \sim SWN_2$, iff $\forall M \in R(SWN, M_i)$, $M \neq \mathcal{F}$, $\exists \sigma \in \Sigma(SWN)$, such that $M[\sigma > M_f \in \mathcal{F}$.

Proof: According to Definition 6, $SWN_1 \sim SWN_2 \Leftrightarrow$ the behavior of SWN is correct. Owing to Theorem 1, we know that $\mathcal{F}_C = \mathcal{F}$ for well-structured services. Hence, the behavior of SWN is correct $\Leftrightarrow \forall M \in R(SWN, M_i)$, $M \notin \mathcal{F}$, $\exists \sigma \in \Sigma(SWN)$, such that $M[\sigma > M_f \in \mathcal{F}$, according to Definition 5. □

It is worth mentioning that the composition of three or more Web services is similar to the composition of two Web services. Thus, the aforementioned two theorems can be easily extended to the situations of multiple Web services. Hence, Theorem 2 can be applied to reduce the complexity of behavioral compatibility analysis of Web services.

IV. Similarity Analysis of Web Services

From the perspective of facilitating Web services substitution, the behavioral similarity analysis of Web services is closely dependent on compatibility analysis—they are the two flip sides of the *substitution* coin [12], [27]. Two similar Web services do not have to require that their internal processes are exactly the same. Generally, only the observable behaviors of these two Web services are required to be similar. Thus, the focus of analyzing their similarity is on their observable behaviors. Informally, the observable behavior refers to the message exchange sequence of a Web service that depicts the interactions of the Web service with its external environment, e.g., client applications and partner services. To analyze behavioral similarity, several formal definitions of observable behavior are proposed as follows.

Definition 8 (Interaction Transition Sequence): Given a reachable transition sequence $\sigma \in \Sigma(SWN)$, after removing the internal transitions of σ, we call the remaining sequence $\varepsilon \in \langle t'_1, t'_2, \ldots, t'_k \rangle$ the interaction transition sequence of σ, where $L(t'_l) \neq \tau$, $l = 1, \ldots, k$. Accordingly, we denote the operation of obtaining an interaction transition sequence as an operator ε.

Definition 9 (Similar Transition Sequence): Given two reachable transition sequences $\sigma_1 \in \Sigma(SWN_1)$ and $\sigma_2 \in \Sigma(SWN_2)$, where $\varepsilon_1 = \varepsilon(\sigma_1) = \langle t_1^1, t_2^1, \ldots, t_k^1 \rangle$ and $\varepsilon_2 = \varepsilon(\sigma_2) = \langle t_1^2, t_2^2, \ldots, t_l^2 \rangle$, σ_2 is similar to σ_1, denoted as $\sigma_2 \approx \sigma_1$, iff $k = l \; \forall j = \{1, \ldots, k\}$, $L(t_j^1) = L(t_j^2)$, $\varphi(t_j^1) = \varphi(t_j^2)$, and $G(\sigma_1) = G(\sigma_2)$, where $G(\sigma_1) = G(t_1^1) \wedge G(t_2^1) \wedge \cdots \wedge G(t_k^1)$ and $G(\sigma_2) = G(t_1^2) \wedge G(t_2^2) \wedge \cdots \wedge G(t_l^2)$.

Definition 10 (Opposite Transition Sequence): Given two reachable transition sequences $\sigma_1 \in \Sigma(SWN_1)$, $\sigma_2 \in \Sigma(SWN_2)$, and $\varepsilon_1 = \varepsilon(\sigma_1) = \langle t_1^1, t_2^1, \ldots, t_k^1 \rangle$, $\varepsilon_2 = \varepsilon(\sigma_2) = \langle t_1^2, t_2^2, \ldots, t_l^2 \rangle$, σ_1 is the opposite transition sequence of σ_2, denoted as $\sigma_1 \approx \overline{\sigma_2}$, iff $k = l \; \forall j = \{1, \ldots, k\}$, $L(t_j^1) = L(t_j^2)$, $\varphi(t_j^1) = -\varphi(t_j^2)$, and $G(\sigma_1) = G(\sigma_2)$, where $G(\sigma_1) = G(t_1^1) \wedge G(t_2^1) \wedge \cdots \wedge G(t_k^1)$ and $G(\sigma_2) = G(t_1^2) \wedge G(t_2^2) \wedge \cdots \wedge G(t_l^2)$.

Based on the aforementioned definitions, we propose the definition of behavioral similarity of two Web services and the definition of behavioral equivalence.

Definition 11 (Behavioral Similarity): Given two Web services SWN_1 and SWN_2, SWN_2 is similar to SWN_1, denoted as $SWN_2 \succ SWN_1$, iff $\forall \sigma_2 \in \Sigma(SWN_2)$, $M_{2i}[\sigma_2 > M_2$, then $\exists \sigma_1 \in \Sigma(SWN_1)$ and $M_{1i}[\sigma_1 > M_1$, where $\sigma_1 \approx \sigma_2$, such that either of the following conditions holds.

1) If $M_1 \in \mathcal{F}_1$ or $M_1[\tau^* > M_{1f} \in \mathcal{F}_1$, then $M_2 \in \mathcal{F}_2$ or $M_2[\tau^* > M_{2f} \in \mathcal{F}_2$.
2) If $\exists t_1 \in T_1$, $t_1 \neq \tau$, $M_1[\tau^* > M'_1 \notin \mathcal{F}_1, M'_1[t_1 >$, then $\exists t_2 \in T_2$, $t_2 \neq \tau$, such that $L(t_2) = L(t_1)$, $\varphi(t_2) = \varphi(t_1)$ and $M_2[\tau^* > M'_2 \notin \mathcal{F}_2, M'_2[t_2 >$.

Definition 12 (Behavioral Equivalence): Given two Web services SWN_1 and SWN_2, SWN_1 is equivalent to SWN_2,

denoted as $SWN_1 \approx SWN_2$, iff $SWN_1 \succ SWN_2$ and $SWN_2 \succ SWN_1$.

Complying with the standard similarity notions, the behavioral similarity defined in Definition 11 is a specific type of relation between the states of two SWNs. According to Definition 11, SWN_2 being similar to SWN_1 does not mean that SWN_1 is similar to SWN_2. Hence, the symmetry property of behavioral similarity does not hold in the sense of Definition 11. Definition 12 requires that two SWNs are behaviorally equivalent only if one is similar to the other and *vice versa*. Apparently, we have the symmetry property of behavioral equivalence of Web services, i.e., $SWN_1 \approx SWN_2 \Leftrightarrow SWN_2 \approx SWN_1$. It is easy to prove that both the similarity and the equivalence of Web services hold the transitivity property, i.e., as follows.

1) $SWN_1 \succ SWN_2$, $SWN_2 \succ SWN_3 \Rightarrow SWN_1 \succ SWN_3$.
2) $SWN_1 \approx SWN_2$, $SWN_2 \approx SWN_3 \Rightarrow SWN_1 \approx SWN_3$.

As discussed earlier, substitution using a similar Web service needs to be independent of (i.e., transparent to) the context of the substituted Web service; context herein refers to other partner Web services in the composition regarding the substituted Web service. Therefore, substituting a Web service in a composition with another similar one can be performed in a transparent way—we do not need to verify whether the substituting Web service is compatible with other partner Web services in the new composition. Hence, the cost of Web services substitution in the sense of context-independent similarity is largely reduced. Herein, we will prove that if one Web service A is behaviorally similar to another Web service B in the sense of Definition 11, then A is compatible with any partners of B in the composition.

Definition 13 (Projection): Given two reachable transition sequences $\sigma_1 \in \Sigma(SWN_1)$, $\sigma_2 \in \Sigma(SWN_2)$, and $\varepsilon_1 = \varepsilon(\sigma_1) = \langle t_1^1, t_2^1, \ldots, t_k^1 \rangle$, $\varepsilon_2 = \varepsilon(\sigma_2) = \langle t_1^2, t_2^2, \ldots, t_l^2 \rangle$, for $i = \{1, \ldots, k\}$ and t_i^1, if $\forall t_j^2$, $j = \{1, \ldots, l\}$, $L(t_i^1) \neq L(t_j^2)$, then we remove t_i^1 from σ_1. The remaining sequence is regarded as the projection of σ_1 on σ_2, denoted as $\mathbb{P}_{\sigma_2}(\sigma_1)$. Similarly, we get the projection of σ_2 on σ_1, denoted as $\mathbb{P}_{\sigma_1}(\sigma_2)$.

Definition 13 is used to simplify the representation of the following lemma. Informally, $\mathbb{P}_{\sigma_2}(\sigma_1)$ is the remaining transition sequence of σ_1 in which the transitions that are irrelevant to σ_2 are removed from σ_1—only the transitions of σ_1 that are relevant to σ_2 need to be considered. $\mathbb{P}_{\sigma_1}(\sigma_2)$ is the remaining transition sequence of σ_2 in which the transitions that are irrelevant to σ_1 are removed from σ_2.

Lemma: Given two well-structured Web services SWN_1 and SWN_2, $SWN_1 \sim SWN_2$, iff $\forall \sigma_1 \in \Sigma(SWN_1)$, $\forall \sigma_2 \in \Sigma(SWN_2)$, $M_{1i}[\sigma_1 > M_1$, $M_{2i}[\sigma_2 > M_2$, $\sigma_1' = \mathbb{P}_{\sigma_2}(\sigma_1)$, $\sigma_2' = \mathbb{P}_{\sigma_1}(\sigma_2)$, if $\sigma_1' \approx \overline{\sigma_2'}$, then either of the following conditions holds.

1) $M_1 \in \mathcal{F}_1$ or $M_1[\tau^* > M_{1f} \in \mathcal{F}_1$, and $M_2 \in \mathcal{F}_2$ or $M_2[\tau^* > M_{2f} \in \mathcal{F}_2$.
2) $\exists t_1 \in T_1$, where $t_1 \neq \tau$, and $\exists t_2 \in T_2$, where $t_2 \neq \tau$, such that $L(t_1) = L(t_2)$, $\varphi(t_1) = -\varphi(t_2)$, and $M_1[\tau^* > M_1' \notin \mathcal{F}_1, M_1'[t_1 >, M_2[\tau^* > M_2' \notin \mathcal{F}_2, M_2'[t_2 >$.

Proof: We denote the composite Web service $SWN = SWN_1 \oplus SWN_2$. On one hand, if $SWN_1 \sim SWN_2$ and $\sigma_1' \approx \overline{\sigma_2'}$, then the composite Web service SWN can reach a marking M. If M is a final marking, then we know that $M_1 \in \mathcal{F}_1$ and $M_2 \in \mathcal{F}_2$. Otherwise, neither SWN_1 nor SWN_2 reaches a final marking, and $\exists \sigma \in \Sigma(SWN)$, such that $M[> M_f \in \mathcal{F}$, according to Theorem 2. Hence, $\exists t_1 \in T_1$, where $t_1 \neq \tau$, and $\exists t_2 \in T_2$, where $t_2 \neq \tau$, such that $L(t_1) = L(t_2)$, $\varphi(t_1) = -\varphi(t_2)$, and $M_1[\tau^* > M_1' \notin \mathcal{F}_1$, $M_1'[t_1 >$, $M_2[\tau^* > M_2' \notin \mathcal{F}_2$, $M_2'[t_2 >$.

On the other hand, since $\sigma_1' \approx \overline{\sigma_2'}$, we know that the composite Web service SWN can reach a marking M. If M is not a final marking, then either $M_1[\tau^* > M_1' \in \mathcal{F}_1$, $M_2[\tau^* > M_2' \in \mathcal{F}_2$, or $\exists t_1 \in T_1$, $t_1 \neq \tau$, $\exists t_2 \in T_2$, $t_2 \neq \tau$, such that $L(t_1) = L(t_2)$, $\varphi(t_1) = -\varphi(t_2)$, and $M_1[\tau^* > M_1' \notin \mathcal{F}_1$, $M_1'[t_1 >$, $M_2[\tau^* > M_2' \notin \mathcal{F}_2$, $M_2'[t_2 >$. In both cases, we have $\exists \sigma \in \Sigma(SWN)$, such that $M[\sigma > M_f \in \mathcal{F}$. According to Theorem 2, we know that $SWN_1 \sim SWN_2$. □

Theorem 3: Given well-structured services SWN_1, SWN_2, and SWN, we have

$$SWN_2 \succ SWN_1, SWN \sim SWN_1 \Rightarrow SWN \sim SWN_2.$$

Proof: $\forall \sigma \in \Sigma(SWN)$, $\sigma_2 \in \Sigma(SWN_2)$, $M_i[\sigma > M$, $M_{2i}[\sigma_2 > M_2$, $\sigma' = \mathbb{P}_{\sigma_2}(\sigma)$, $\sigma_2' = \mathbb{P}_{\sigma}(\sigma_2)$. Suppose that $\sigma_2' \approx \overline{\sigma'}$. Since $SWN_2 \succ SWN_1$, then $\exists \sigma_1 \in \Sigma(SWN_1)$, such that $M_{1i}[\sigma_1 > M_1$ and $\sigma_1 \approx \sigma_2$, according to Definition 11. Suppose that $\sigma_1' = \mathbb{P}_{\sigma}(\sigma_1)$. Thus, we have $\sigma_1' \approx \sigma_2'$. Therefore, $\sigma_1' \approx \overline{\sigma'}$. According to $SWN \sim SWN_1$ and the Lemma, one of the following two conditions holds.

1) $M \in \mathcal{F}$ or $M[\tau^* > M_f \in \mathcal{F}$, and $M_1 \in \mathcal{F}_1$ or $M_1[\tau^* > M_{1f} \in \mathcal{F}_1$.
2) $\exists t \in T$, where $t \neq \tau$, and $\exists t_1 \in T_1$, where $t_1 \neq \tau$, such that $L(t) = L(t_1)$, $\varphi(t) = -\varphi(t_1)$, and $M[\tau^* > M' \notin \mathcal{F}, M'[t >, M_1[\tau^* > M_1' \notin \mathcal{F}_1, M_1'[t_1 > tT$.

According to $SWN_2 \succ SWN_1$ and Definition 11, if $M_1 \in \mathcal{F}_1$ or $M_1[\tau^* > M_{1f} \in \mathcal{F}_1$, then $M_2 \in \mathcal{F}_2$ or $M_2[\tau^* > M_{2f} \in \mathcal{F}_2$. If $\exists t_1 \in T_1$, $t_1 \neq \tau$, $M_1[\tau^* > M_1' \notin \mathcal{F}_1$, $M_1'[t_1 >$, then $\exists t_2 \in T_2$, $t_2 \neq \tau$, such that $M_2[\tau^* > M_2' \notin \mathcal{F}_2$, $M_2'[t_2 >$, and $L(t_2) = L(t_1)$, $\varphi(t_2) = -\varphi(t_1)$. According to the Lemma, we have $SWN \sim SWN_2$. □

Theorem 3 presents an interesting fact regarding the well-structured Web services. If a well-structured Web service SWN_2 is similar to SWN_1, then for any other well-structured Web service SWN that is compatible with SWN_1, SWN_2 is also compatible with SWN. This is very useful for Web services substitution. Suppose that SWN_1 is a component Web service in a certain composition. When SWN_1 fails to work or could not satisfy the nonfunctional requirements, it needs to be substituted by an alternative well-structured Web service SWN_2 with similar behavior in the sense of Definition 11. The conclusion of Theorem 3 guarantees that SWN_2 is compatible with other partner well-structured Web services in the composition. The new composite Web service can execute successfully without the need of verifying the new composition again. Hence, Web services substitution can be directly

1. **Input:** $CRG(ws_1)$, $CRG(ws_2)$
2. **Output:** IsSimilar (ws_2, ws_1)
3. Set $T(ws_2, M_2)$; // for all transitions of service ws_2
   // that are enabled in marking M_2
4. Set $U_2 = \emptyset$; // the set of markings of service ws_2
   // that has been checked
5. Set M_1=M_{1i}, M_2=M_{2i};
6. if $M_2 \in \mathcal{F}_2$
7. if $M_1 \in \mathcal{F}_1$ or $M_1[\tau^* > M_{1f} \in \mathcal{F}_1$
8. then return TRUE;
9. else return FALSE;
10. else // else-if line 6
11. for each $t_2 \in T(ws_2, M_2)$ and $M_2 \notin U_2$
12. if $t_2 = \tau$, $M_2[\tau > M_2'$
13. then add M_2 to U_2, $M_2 = M_2'$,
14. return IsSimilar (ws_2, ws_1);
15. else $M_2[t_2 > M_2'$ // else-if line 12
16. if $\exists t_1 \in T_1$, $t_1 \neq \tau$, $M_1[\tau^* > M_1'$, $M_1'[\, t_1 > M_1''$,
17. and $L(t_2) = L(t_1)$, $\varphi(t_2) = \varphi(t_1)$
18. then if $M_1'' \in \mathcal{F}_{1F}$, or $M_1''[\tau^* > M_{1f} \in \mathcal{F}_{1F}$
19. then if $M_2' \in \mathcal{F}_{2F}$, or $M_2'[\tau^* > M_{2f} \in \mathcal{F}_{2F}$
20. then return TRUE;
21. else return FALSE;
22. else $\exists t_1' \in T_1$, $t_1' \neq \tau$, $M_1''[\tau^* > M_1^{(3)}$,
23. $M_1^{(3)}[t_1' > M_1^{(4)}$
24. if $\exists t_2' \in T_2$, $t_2' \neq \tau$, $M_2'[\tau^* > M_2''$,
25. $M_2''[t_2' > M_2^{(3)}$, and $L(t_2') = L(t_1')$,
26. $\varphi(t_2') = \varphi(t_1')$,
27. then add M_2 to U_2, $M_1 = M_1^{(4)}$,
28. $M_2 = M_2^{(3)}$,
29. return IsSimilar (ws_2, ws_1);
30. else return FALSE; // else-if line 24
31. else return FALSE; // else-if line 16
32. end // end-for line 11

Fig. 4. Algorithm for verifying the behavioral similarity of well-structured Web services.

performed in the sense of context-independent similarity of Web services presented in Definition 11.

To automatically perform the verification of behavioral similarity, we developed an algorithm that checks if ws_2 is similar to ws_1, as shown in Fig. 4. The rationale of the algorithm is to search the communicating reachability graphs (CRGs) of two Web services and verify their similarity in the sense of Definition 11. As a kind of reachability graph, the CRG of a Web service is used to analyze the behavior of its SWN based on the state analysis. Details on the definition and the constructing method of CRGs are presented in our previous work [17], [33].

The concept of opposite Web services is useful for analyzing the compatibility of Web services [12], [27]. For a Web service ws, its opposite Web service $\overline{ws}$ is obtained when the actions of sending messages are changed to that of receiving messages and *vice versa*. It is easy to see that ws is compatible with its opposite Web service $\overline{ws}$, i.e., $ws \sim \overline{ws}$. We present the following useful corollaries regarding the opposite Web service that can be used to facilitate the verification of behavioral compatibility.

Corollary 1: Given well-structured Web services ws, ws_1, and $\overline{ws}$, we have $ws_1 \succ ws \Rightarrow ws_1 \sim \overline{ws}$.

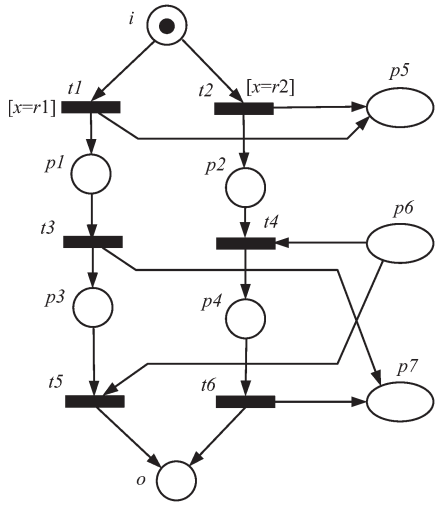


Fig. 5. SWN of Web service ws_4.

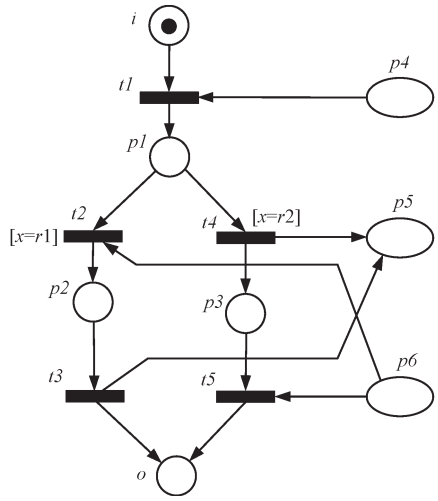


Fig. 6. SWN of Web service ws_5.

Proof: According to Theorem 3 and $ws \sim \overline{ws}$, it is easy to have $ws_1 \sim \overline{ws}$. □

Corollary 2: Given the well-structured Web services ws, ws_1, ws_2, and $\overline{ws}$, we have

$$ws_1 \succ ws \qquad ws_2 \succ \overline{ws} \Rightarrow ws_1 \sim ws_2.$$

Proof: According to Corollary 1, we have $ws_1 \sim \overline{ws}$. According to Theorem 3 and $ws_2 \succ \overline{ws}$, we have $ws_1 \sim ws_2$. □

Fig. 5 shows the SWN of a Web service ws_4. By using the algorithm of Fig. 4, it is verified that ws_4 is behaviorally similar to ws_3 in Fig. 3. Fig. 6 shows the opposite service (i.e., ws_5) of ws_3. Both ws_3 and ws_4 are compatible with ws_5, which exemplifies the conclusions presented in this paper.

V. Tool for Similarity and Equivalence Verification

A tool has been developed to automate the verification of behavioral similarity and equivalence of Web services using our approach. The tool was developed as an analysis module of the PIPE [29]. PIPE is an open-source implemented in Java, which

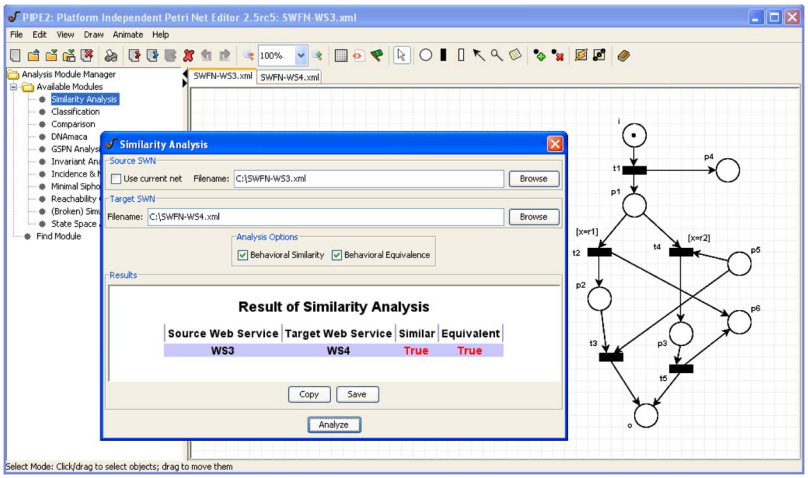


Fig. 7. Snapshot of the tool for similarity analysis, which is integrated into the PIPE.

provides a platform-independent GUI for creating, editing, and analyzing PNs. The PNs created by PIPE conform to an XML-based interchange format, i.e., PN Markup Language (PNML) [34].

At the top left of the interface in Fig. 7, PIPE v2.5 offers several analysis modules to check behavioral properties such as reachability graph and state space analysis. We implemented a new module to perform the functionality of similarity analysis of two SWNs. At the right side of the interface is the workspace of PIPE, where SWNs can be modeled and represented as intuitive diagrams. Each SWN is modeled in one tab of the workspace. Internally, the SWN models are specified and stored in PNML files. By double clicking the module label of Similarity Analysis, a dialogue box pops up and asks users to select the *source* SWN (left-hand side operand of the similar operator "$\succ$") and the *target* SWN (right-hand side operand of the similar operator "$\succ$"). For demonstration purposes, Fig. 7 shows that the source SWN is ws_3 and the target SWN is ws_4, which are also shown in Figs. 3 and 5, respectively. The algorithm given in Fig. 4 has been implemented inside the module. To perform the analysis of behavioral similarity and/or equivalence using the algorithm, the user simply presses the ANALYZE button, and then, the result appears as a formatted report. The result shows that ws_3 and ws_4 are behaviorally equivalent. In fact, on one hand, each interaction transition sequence of ws_3 corresponds to a similar one of ws_4. On the other hand, each interaction transition sequence of ws_4 corresponds to a similar one of ws_3.

We further validated our approach using several real-world Web services. Fig. 8 shows the behavior of a weather forecast Web service provided by the U.S. National Oceanic and Atmospheric Administration (NOAA) (i.e., National Digital Forecast Database XML/SOAP service www.nws.noaa.gov/xml).

As shown in Fig. 8, after receiving a place (either a zip code or a city name) from a user, the NOAA's weather forecast Web service first performs the availability check of the place. It then converts the available place into longitude and latitude coordinates. Using the coordinates, the service provides the map in which the place is located. The service also retrieves the weather forecast by querying the NOAA National Digital

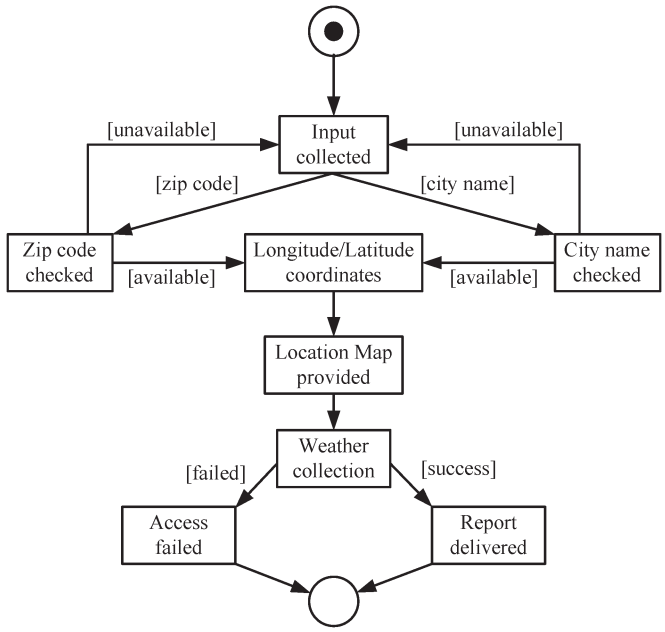


Fig. 8. Behavior of NOAA's weather Web service [19].

Forecast Database. If the access to the database succeeds, it delivers the weather forecast report. Otherwise, it sends a message about access failure. The details of the Web service are described in one of our recent papers [19].

Fig. 9 shows the SWN of the behavior of NOAA's weather Web service. For the sake of simplicity, the declarations and business semantics of the SWN are defined in the PNML file and omitted in the graph. The places $p1$, $p7$, $p8$, and $p9$ in Fig. 9 are external message places representing the input message (e.g., a zip code or a city name), the location map, the weather forecast report, and the access-failed message, respectively. World Weather Online (WWO) is another Web service providing free weather forecast data (http://www.worldweatheronline.com/). The behavior of WWO's weather service can be also modeled as the SWN model using our tool. Fig. 10 is the screenshot that shows the SWN of WWO's weather service. The similarity analysis using

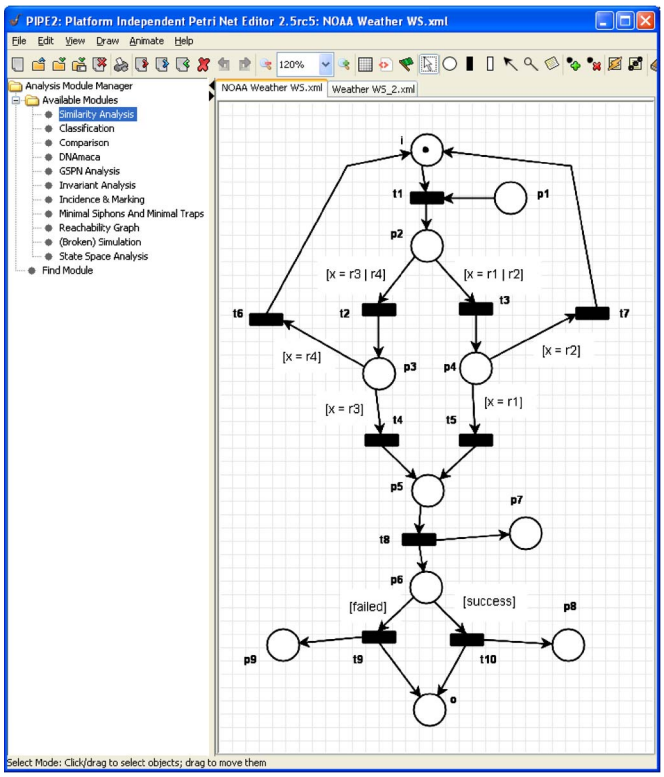


Fig. 9. SWN of NOAA's weather Web service.

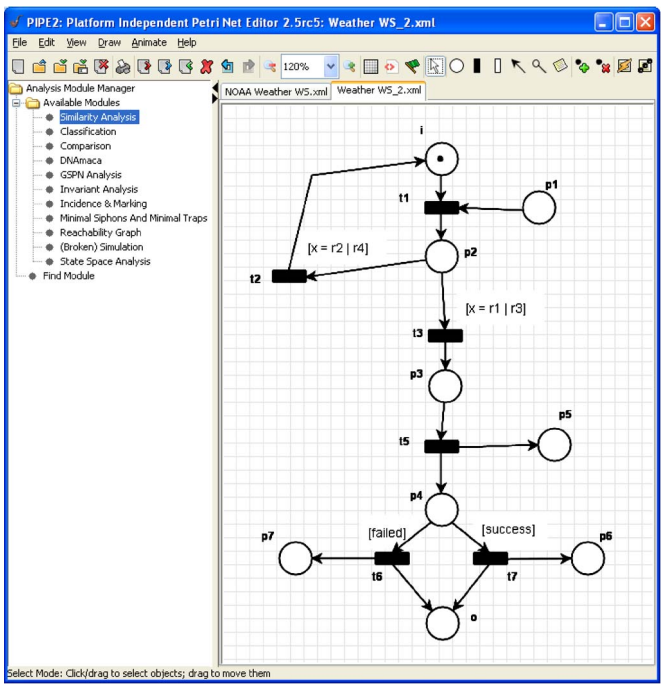


Fig. 10. SWN of WWO's weather Web service.

our tool reveals that the behaviors of the two weather Web services shown in Figs. 9 and 10 are actually similar.

VI. Discussions and Conclusion

SOC is a promising paradigm for the integration of Web applications. Web services composition is one of the key objectives for adopting such a computing paradigm. Traditionally, it is very costly and time consuming to verify behavioral compatibility for Web services composition; both conditions, i.e., reachable termination and proper termination, need to be checked.

This paper has provided a good number of specification languages for formally describing Web services behaviors, such as FSM [8], [10], [11], process algebra/pi calculus [12], [13], and PNs [14]–[17]. Compared to FSM and process algebra, the Petri-net-based approaches have a well-defined semantics for describing both states and actions of Web services, as well as a graphical representation. Web services behaviors based on PN models can be analyzed by means of either formal analysis methods or simulation. Therefore, many research works propose to model the behaviors of Web services based on PN models [14]–[17], [33], [35]. As a high-level PN model, the CPN model has a more compact and expressive power [30] than basic PN models. Detailed comparison between CPN models and other specification languages of Web services can be found in [17]. Based on our previous work [17], [33], we model Web services behaviors using SWNs which are labeled CPNs with three disjoint types of places representing control flows, external messages, and internal messages, respectively. Compared to other similar specification languages, the advantages of the SWN model are as follows: 1) It distinguishes different message types of Web services using different color sets; 2) it depicts different branches of Web services processes using guard expressions; and 3) it represents Web services messages and actions using labels. These advantages help describe and perform the compatibility and similarity analysis of Web services behaviors. Specifically, the SWN model not only provides a formalism to depict the internal logics and message exchange sequences (i.e., behaviors) but also encompasses the mechanisms of compatibility and similarity analysis of Web services.

Based on the SWN model, we have presented a formal definition of behavioral compatibility, which requires a more relaxed condition than the existing approaches. To reduce the complexity of verifying behavioral compatibility, we have looked into the structural property of SWNs and have proposed a novel notion of well structuredness of Web services. Based on this notion, we proved that only reachable termination needs to be checked when verifying behavioral compatibility among well-structured Web services.

Service substitution is another significant issue with Web services composition. Unfortunately, most existing notions of behavioral similarity of Web services are inappropriate for performing service substitution. Although many existing approaches utilize PNs to analyze behavioral compatibility, few of them further explore appropriate definitions of behavioral similarity and provide a user-friendly tool for the automatic verification. To this end, we have proposed the formal definition of context-independent similarity that is in line with our definition of behavioral compatibility. We concluded that substituting a Web service in a composition can be performed in a transparent way as long as the new Web service is similar to the substituted one. We have provided an algorithm for the verification of behavioral similarity based on our definitions. Furthermore, we have developed a tool, which has been integrated into the

open-source project PIPE, to automatically perform the verification of behavioral similarity.

In our current work, the rationale of verifying behavioral compatibility and similarity of Web services is to search the CRGs. Based on the reachability analysis, our approach is still not efficient, particularly with the increase of the state space. The future work is to use the recently developed technique based on elementary siphon [35]–[40] to reduce the complexity of the problem. In particular, the recent work reported in [35] has developed the method of using elementary siphons to deal with BPEL-based services and overcome the complexity problem. We also plan to extend our work to quantify to what degree a Web service is similar to another. Aside from that, we will apply our research results to two important research areas in SOC, i.e., Web service discovery and Web services communities. For service discovery, developers can use our approach to search for candidate Web services of similar behaviors according to a specified service profile. On the other hand, Web services communities, which are a means that permits gathering Web services of similar functionalities into groups, can be used to accelerate the process of discovering and composing Web services [41], [42]. Our proposed work can be used to facilitate the establishment of these communities.

ACKNOWLEDGMENT

The authors would like to thank Prof. M. Zhou at the New Jersey Institute of Technology, Newark, for his valuable suggestions in improving the quality of this paper.

REFERENCES

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

[2] J. Zhang, C. K. Chang, L. J. Zhang, and P. C. K. Hung, "Toward a service-oriented development through a case study," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 6, pp. 955–969, Nov. 2007.

[3] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed, "Deploying and managing Web services: Issues, solutions, and directions," *Int. J. Very Large Data Bases*, vol. 17, no. 3, pp. 537–572, May 2008.

[4] N. Srini and A. M. Sheila, "Simulation, verification and automated composition of Web services," in *Proc. 11th Int. Conf. WWW*, 2002, pp. 77–88.

[5] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan, "Automated discovery, interaction and composition of semantic Web services," *Web Semantics: Sci., Serv. Agents World Wide Web*, vol. 1, no. 1, pp. 27–46, Dec. 2003.

[6] M. P. Fanti and M. C. Zhou, "Deadlock control methods in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 1, pp. 5–22, Jan. 2004.

[7] X. Li, Y. Fan, and F. Jiang, "A classification of service composition mismatches to support service mediation," in *Proc. 6th Int. Conf. GCC*, 2007, pp. 315–321.

[8] H. S. Chae, J.-S. Lee, and J. Bae, "An approach to checking behavioral compatibility between Web services," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 18, no. 2, pp. 223–241, Mar. 2008.

[9] Z. Maamar, D. Benslimane, G. K. Mostefaoui, S. Subramanian, and Q. H. Mahmoud, "Toward behavioral Web services using policies," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1312–1324, Nov. 2008.

[10] B. Benatallah, F. Casati, and F. Toumani, "Analysis and management of Web service protocols," in *Proc. Conceptual Modeling—ER*, 2004, pp. 524–541.

[11] H. Foster, S. Uchitel, J. Magee, J. Kramer, and I. C. London, "Compatibility verification for Web service choreography," in *Proc. IEEE ICWS*, 2004, pp. 738–741.

[12] A. Brogi, C. Canal, E. Pimentel, and A. Vallecillo, "Formalizing Web service choreographies," *Electron. Notes Theor. Comput. Sci.*, vol. 105, pp. 73–94, Dec. 2004.

[13] S. Deng, Z. Wu, M. Zhou, Y. Li, and J. Wu, "Modeling service compatibility with pi-calculus for choreography," in *Proc. Conceptual Modeling—ER*, 2006, pp. 26–39.

[14] A. Martens, "On compatibility of Web services," *Petri Net Newslett.*, vol. 65, pp. 12–20, 2003.

[15] A. Martens, S. Moser, A. Gerhardt, and K. Funk, "Analyzing compatibility of BPEL processes," in *Proc. Int. Conf. Internet Web Appl. Serv./Adv. Int. Conf. Telecommun.*, 2006, pp. 147–155.

[16] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede, "Formal semantics and analysis of control flow in WS-BPEL," *Sci. Comput. Program.*, vol. 67, no. 2/3, pp. 162–198, Jul. 2007.

[17] 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.

[18] M. De Backer, M. Snoeck, G. Monsieur, W. Lemahieu, and G. Dedene, "A scenario-based verification technique to assess the compatibility of collaborative business processes," *Data Knowl. Eng.*, vol. 68, no. 6, pp. 531–551, Jun. 2009.

[19] Q. Z. Sheng, Z. Maamar, H. Yahyaoui, J. Bentahar, and K. Boukadi, "Separating operational and control behaviors: A new approach to Web services modeling," *IEEE Internet Comput.*, vol. 14, no. 3, pp. 68–76, May/Jun. 2010.

[20] H. R. M. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, "Semi-automated adaptation of service interactions," in *Proc. 16th Int. Conf. WWW*, 2007, pp. 993–1002.

[21] 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.

[22] Y. Taher, D. Benslimane, M. C. Fauvet, and Z. Maamar, "Towards an approach for Web services substitution," in *Proc. IDEAS*, 2006, pp. 166–173.

[23] J. Pathak, S. Basu, and V. Honavar, "On context-specific substitutability of Web services," in *Proc. IEEE ICWS*, 2007, pp. 192–199.

[24] N. Hameurlain, "Flexible behavioural compatibility and substitutability for component protocols: A formal specification," in *Proc. 5th IEEE Int. Conf. SEFM*, 2007, pp. 391–400.

[25] A. Martens, "Simulation and equivalence between BPEL process models," in *Proc. Des., Anal., Simul. Distrib. Syst. Symp. (DASD)*, 2005.

[26] B. H. Liskov and J. M. Wing, "A behavioral notion of subtyping," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 6, pp. 1811–1841, Nov. 1994.

[27] L. Bordeaux, G. Salaün, D. Berardi, and M. Mecella, "When are two Web services compatible?," in *Proc. Technol. E-Serv.*, 2005, pp. 15–28.

[28] S. Moser, A. Martens, K. Gorlach, W. Amme, and A. Godlinski, "Advanced verification of distributed WS-BPEL business processes incorporating CSSA-based data flow analysis," in *Proc. IEEE Int. Conf. SCC*, 2007, pp. 98–105.

[29] P. Bonet, C. M. Llado, R. Puijaner, and W. J. Knottenbelt, "PIPE v2.5: A Petri net tool for performance modelling," in *Proc. 23rd Latin Amer. Conf. Informat. (CLEI)*, 2007.

[30] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. Vol. 1, Basic Concepts*. New York: Springer-Verlag, 1997, ser. Monographs in Theoretical Computer Science.

[31] R. M. Dijkman, "Notions of behavioral compatibility and their implications for BPEL processes," Centre Telematics Inf. Technol., Univ. Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-06-41, 2006.

[32] W. M. P. van der Aalst, "The application of Petri nets to workflow management," *J. Circuits, Syst. Comput.*, vol. 8, no. 1, pp. 21–66, 1998.

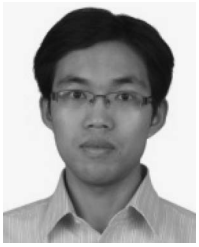
[33] X. Li and Y. Fan, "Modeling and logical correctness verification of Web service processes," *Comput. Integr. Manuf. Syst.*, vol. 14, no. 4, pp. 675–682, 2008, in Chinese.

[34] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber, "The Petri Net Markup Language: Concepts, technology, and tools," in *Proc. 24th Int. Conf. Appl. Theory Petri Nets*, *Lecture Notes in Computer Science*, 2003, pp. 483–506.

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

[36] Z. Li and M. C. Zhou, "On siphon computation for deadlock control in a class of Petri nets," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 3, pp. 667–679, May 2008.

[37] Z. Li and M. C. Zhou, "Clarifications on the definitions of elementary siphons in Petri nets," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 6, pp. 1227–1229, Nov. 2006.
[38] Z. Li and M. C. Zhou, "Control of elementary and dependent siphons in Petri nets and their application," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 133–148, Jan. 2008.
[39] Z. Li, H. S. Hu, and A. R. Wang, "Design of liveness-enforcing supervisors for flexible manufacturing systems using Petri nets," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 37, no. 4, pp. 517–526, Jul. 2007.
[40] 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.
[41] J. Bentahar, Z. Maamar, W. Wan, D. Benslimane, P. Thiran, and S. Subramanian, "Agent-based communities of Web services: An argumentation-driven approach," *Serv. Oriented Comput. Appl.*, vol. 2, no. 4, pp. 219–238, Dec. 2008.
[42] 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, 2009.



Xitong Li received the B.S. and Ph.D. degrees in control science and engineering from the Tsinghua University, Beijing, China. He is currently working toward the Ph.D. degree in management science in the Information Technologies (IT) Group, MIT Sloan School of Management, Cambridge, MA.

His research interests include all aspects of effective management and use of data/information, e.g., data/information integration, economics of using data/information, data/information quality, service-oriented computing, Web services, business process analysis, etc.

Yushun Fan received the B.S. degree in automatic control from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 1984 and the M.S. and Ph.D. degrees in control theory and application from Tsinghua University, Beijing, in 1987 and 1990, respectively.

He is currently a Professor with the Department of Automation, the Director of the System Integration Institute, and the Director of the Networking Manufacturing Laboratory, Tsinghua University. He authored ten books in enterprise modeling, workflow technology, intelligent agent, object-oriented complex system analysis, and computer-integrated manufacturing, respectively, and published more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration and integrated platform, object-oriented technologies and flexible software systems, Petri nets modeling and analysis, and workshop management and control.

Quan Z. Sheng (S'03–M'06) received the Ph.D. degree in computer science from the University of New South Wales, Sydney, Australia, in 2006.

He is currently a Senior Lecturer with the School of Computer Science, The University of Adelaide, Adelaide, Australia. His research interests include service-oriented architectures, distributed computing, and pervasive computing. He is the author of more than 70 publications.

Dr. Sheng is a member of the Association for Computing Machinery. He was the recipient of the Microsoft Research Fellowship in 2003. He served on program committees for dozens of conferences and was the Publication Chair of the 2005 International Conference on Web Information Systems Engineering (WISE), the Program Chair of the 2008 IEEE International Conference on Signal-Image Technology and Internet-Based Systems, and the Publicity Cochair of the 2005 International Conference on Service Oriented Computing, WISE 2007, and the 2010 International Conference on Pervasive Services. He is the Program Chair of the International Symposium on Web and Mobile Information Services WAMIS in 2010–2011.

Zakaria Maamar received the Ph.D. degree in computer sciences from Laval University, Quebec City, QC, Canada.

He is currently a full Professor with the College of Information Technology, Zayed University, Dubai, United Arab Emirates. His research interests include Web services, mobile computing, and social networks. He has published several papers in various journals and conference proceedings.

Dr. Maamar is the founder of the annual International Symposium on Web Services.

Hongwei Zhu received the Ph.D. degree in technology, management, and policy from the Massachusetts Institute of Technology, Cambridge, MA.

He is currently an Assistant Professor of information technology with the College of Business and Public Administration, Old Dominion University, Norfolk, VA. Prior to that, he was a Research Scientist with the MIT Information Quality Program. He has also worked in industries as a Consultant and a Senior Software Engineer. His research interests include semantic data integration, data mining, data reuse, quality of data standards, information quality management, and information policy.