Hindawi
Complexity
Volume 2020, Article ID 2751084, 11 pages
https://doi.org/10.1155/2020/2751084

Research Article

A Correctness Checking Approach for Collaborative Business Processes in the Cloud

Qi Mo,[1] **Yuqi Wang**,[1] **Jixiang Xiang**,[1] **and Tong Li**[2]

[1]*School of Software, Yunnan University, Kunming 650091, China*
[2]*School of Big Data, Yunnan Agricultural University, Kunming 650091, China*

Correspondence should be addressed to Tong Li; tli@ynu.edu.cn

Received 5 December 2019; Accepted 18 January 2020; Published 11 February 2020

Guest Editor: Xuyun Zhang

Copyright © 2020 Qi Mo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the increasing popularity of cloud computing, especially the emergence of Business Process as a Service (BPaaS), more and more enterprises construct their process collaborations based on BPaaS services. Indeed, the collaborative business process built by BPaaS services can be seen as a complex system, as it covers multiple business processes (i.e., BPaaS services) and they act independently. Since business processes corresponding to BPaaS services are usually provided by different cloud service providers, and their interactions are unforeseen in advance, in actual execution, some behavioral anomalies (e.g., deadlocks) may occur. To this end, based on BPaaS services, we propose an approach to build process collaborations in the cloud. In this approach, we first model collaborative business processes using open nets. Then, we check their correctness based on stubborn sets. Finally, in case they are partially correct, we generate reliable paths for the coordination execution between business processes. Our approach is implemented in the PIPE (an open tool for Petri nets) and evaluated on actual cases that show its effectiveness and efficiency.

1. Introduction

Currently, the new paradigm cloud computing has received great attention, as it can deliver shared services (e.g., computing capacity, storage, and software applications) to clients over the Internet in a ubiquitous, convenient, and on-demand way with a minimal management effort [1–3].

With the widespread application of cloud computing, especially the emergence of BPaaS (a cloud service that can be delivered to clients in the form of processes) [4], more and more enterprises deploy their business processes to the cloud to achieve value-added services. Based on the cloud platform, such as keyword search [5], enterprises residing on the cloud can find and compose some business processes with complementary competencies and knowledge (i.e., BPaaS services) into their business processes to build collaborative business processes to achieve business success [6–8]. For example, for a retailer residing on the cloud, its transportation process can be outsourced to a BPaaS service that coordinates the actual transportation to improve the effectiveness and efficiency. In practice, this model brings at least two benefits to enterprises. First, enterprises can more easily build their collaborative business processes than ever before, as BPaaS services can be directly invoked in the cloud without having to develop them independently [9]. Second, collaborative business processes built by BPaaS services are more scalable and reliable because they are deployed in the cloud.

In fact, the collaborative business process built by composing business processes (i.e., BPaaS services) gathered in the cloud can be seen as a complex system, as it covers multiple business processes and these business processes act independently [10]. In the cloud, these business processes are usually developed by different organizations and their interactions are unforeseen by interactive parties. Consequently, behavioral anomalies (e.g., deadlocks) may be caused and eventually have an adverse impact on the execution of their composition.

In order to eliminate these behavioral anomalies, the correctness checking approach is dominant in existing approaches [6, 10–13]. Given a collaborative business process, the approach automatically detects its correctness (e.g.,

soundness [6]) using formal techniques (e.g., model checking). In case it is incorrect, developers can repair it via diagnosis information described by a trace leading to errors.

However, in the actual checking process, existing approaches for correctness checking [6, 10, 11] usually need to construct the full state space (i.e., a direct graph that covers all reachable states and edges between states) of collaborative business processes, and hence, they may suffer from the low efficiency. Some approaches [12, 13] do exist that focus on improving the checking efficiency. Concretely, these approaches first abstract business processes in the collaboration as public views. Then, they compose these public views to build an abstract collaborative process. Finally, they detect the correctness based on the abstract collaborative process. In general, these approaches can improve checking efficiency while considering privacy. Yet, they require business processes in the collaboration to be structured. Note that the term "structured" means that the business process only contains the sequence, concurrency, selection, and loop structures, and it is formed through the composition of these four structures. Since the assumption may not hold in practice [14], the efficiency of their correctness checking may not increase significantly. Additionally, they do not consider the fact that business processes in the collaboration are typically partially correct (there is at least one path in the collaboration process, and from which the collaboration process can be successfully terminated) [15, 16], and hence, they cannot generate reliable paths for each business process. In particular, a reliable execution path can be seen as a collaborative work plan between a set of business processes [17], and from which their collaboration can be successfully terminated.

To address these problems, based on stubborn sets in [18], this paper proposes an approach to build collaboration processes in the cloud (see Figure 1). Concretely, in our approach, we first model collaborative business processes based on BPaaS services using open nets [19]. Afterwards, we check their correctness based on stubborn sets. At last, in case they are partially correct, we generate reliable paths for the coordination execution between business processes.

The main contributions of this paper are summarized as follows:

(1) We present a method for rapid correctness checking based on stubborn sets

(2) We propose a method for generating reliable paths for the coordination execution between business processes

(3) The proposed approach is implemented in the PIPE, and its effectiveness and efficiency are validated with actual cases

The paper has the following organizations. Section 2 gives a motivating example to illustrate our approach throughout the paper. Section 3 introduces open nets and uses them to model collaborative business processes. Section 4 presents the method for checking correctness based on the stubborn set. Section 5 proposes a method for generating reliable paths for each business process. Section 6 introduces our prototype tool called cctool and evaluates our approach based on real-world cases. Section 7 compares the related work. Section 8 concludes this paper.

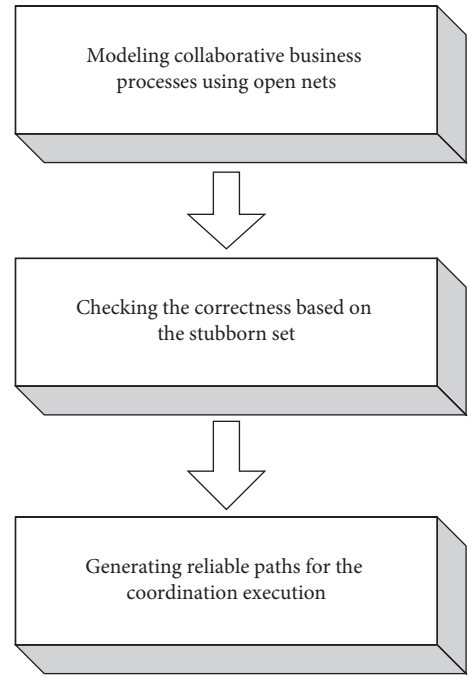

Figure 1: Overview of our approach.

2. Motivating Example

To achieve additional value-added services, a supplier *Supp* deploys its ordering process to the cloud as a BPaaS service for other enterprises to invoke. Based on the cloud platform, a retailer *Reta* residing on the cloud finds the service and composes it into its process to build a collaborative business process OP to achieve business success.

With BPMN, the processes of *Supp* and *Reta* are depicted in Figure 2, where *Supp* can receive ordering requests from *Reta*. In general, the product *A* in *Supp* is sufficient, and hence, *Supp* receives the request for ordering product *A* in any case. However, product *B* in *Supp* is insufficient in some cases. Thus, *Supp* first checks its stock and then receives the request for ordering product *B*. In case order *A* or *B* is received, *Supp* sends the ordered product to *Reta*. After that, *Reta* is able to sell it to customers for an intermediate profit.

Generally, the business processes of *Reta* and *Supp* in the cloud are independently developed by different cloud service providers; thereby, all potential interactions between them are unforeseen in advance. Consequently, some behavioral anomalies, such as deadlocks, may occur during OP's actual execution. For instance, in OP, if *Supp* waits to receive the order of product *B* while *Reta* sends an order of product *A*; then, a deadlock occurs, and it is depicted as red lines in Figure 2.

In order to avoid these behavioral anomalies, we propose a method to check OP's correctness based on the stubborn set in this paper. In case of partially correct, we then present

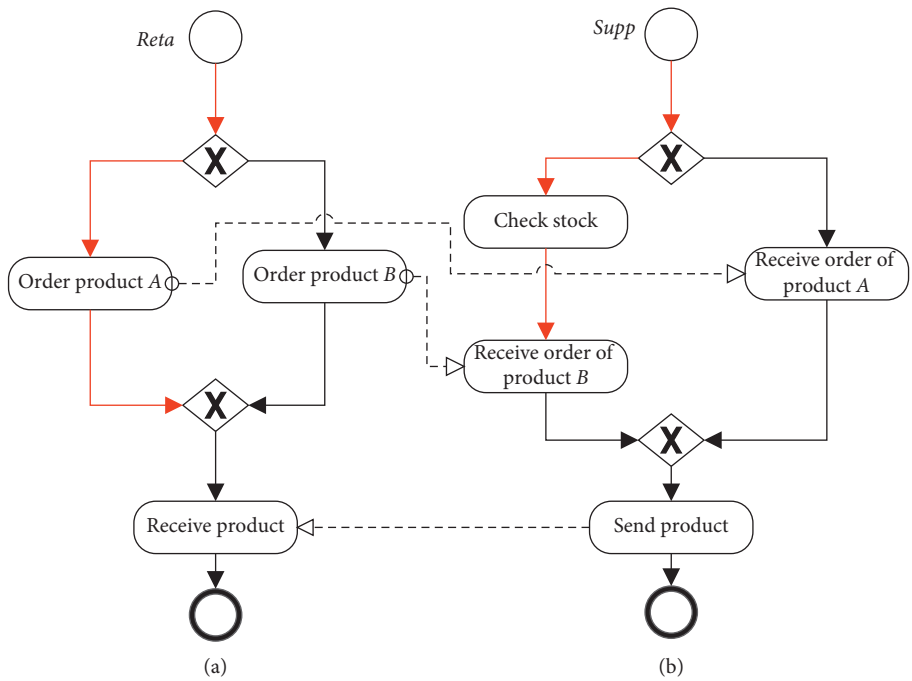

Figure 2: OP.

a method to generate all reliable execution paths for the coordination execution between *Reta* and *Supp*. In actual execution, based on these reliable paths, the interaction between *Reta* and *Supp* can be successfully terminated.

3. Modeling Methods

In this section, we first briefly introduce the concept of open nets [19] and then illustrate the method to model collaborative business processes based on BPaaS services.

3.1. Open Nets. In this paper, open nets will be used to describe business processes and their composition, which can be used to model collaborative business processes in the cloud. Compared with the traditional Petri nets, open nets are enriched with message places to model asynchronous message channels between business processes (e.g., BPaaS services) in the cloud [19]. Open net can be formally defined as follows.

Definition 1 (Open Net). An open net can be defined as a tuple $N = (P, T; F, M_0, M_e)$, where

(1) P is a set of places, $P_I \subseteq P$ are internal places, and $P_M \subseteq P$ are message places, such that $P_I \cap P_M = \varnothing$
(2) T and F are the transitions and flow relation of N
(3) M_0 is the initial marking, such that $M_0(i) = 1$, and $M(p) = 0$ for place $p \neq i$, where i is the source place
(4) M_e is the final marking, such that $M_0(o) = 1$, and $M(p) = 0$ for place $p \neq o$, where o is the sink place

Using the place fusion technique [4, 20], we can compose multiple open nets into an open net. Given two open nets, their composition can be formally defined as follows.

Definition 2 (Composition). The composition of two open nets $N_1 = (P_1, T_1; F_1, M_{01}, M_{e1})$ and $N_2 = (P_2, T_2; F_2, M_{02}, M_{e2})$ can be described as an open net $N_1 || N_2 = (P, T; F, M_0, M_e)$, where

(1) $P = P_1 \cup P_2$
(2) $T = T_1 \cup T_2$
(3) $F = F_1 \cup F_2$
(4) M_0 is an initial marking, such that $M_0(i_1) = 1 \wedge M_0(i_2) = 1$, and $M(p) = 0$ for place $p \neq i_1$ and $p \neq i_2$
(5) M_e is a final marking, such that $M_e(o_1) = 1 \wedge M_e(o_2) = 1$, and $M(p) = 0$ for place $p \neq o_1$ and $p \neq o_2$

Clearly, according to Definition 2, we can conclude that the composition is both associative and commutative. Consequently, given a set of open nets, their composition can be noted as $N_1 \| \cdots \| N_n$.

3.2. Modeling Collaborative Business Processes. In [21], based on open nets, we proposed a method to construct collaborative business processes. Its basic idea is that we first convert all business processes in the collaboration into open nets based on their informal descriptions and then compose these open nets to generate a collaborative business process. The method can also be used in our context. That is, we first map the business process corresponding to each BPaaS service for collaboration to an open net, and then, a collaboration process can be built by composing these open nets.

Note that in this paper, we restrict ourselves to business processes that cover no loops, as the loop in the process model can typically be converted into a sequential structure [22].

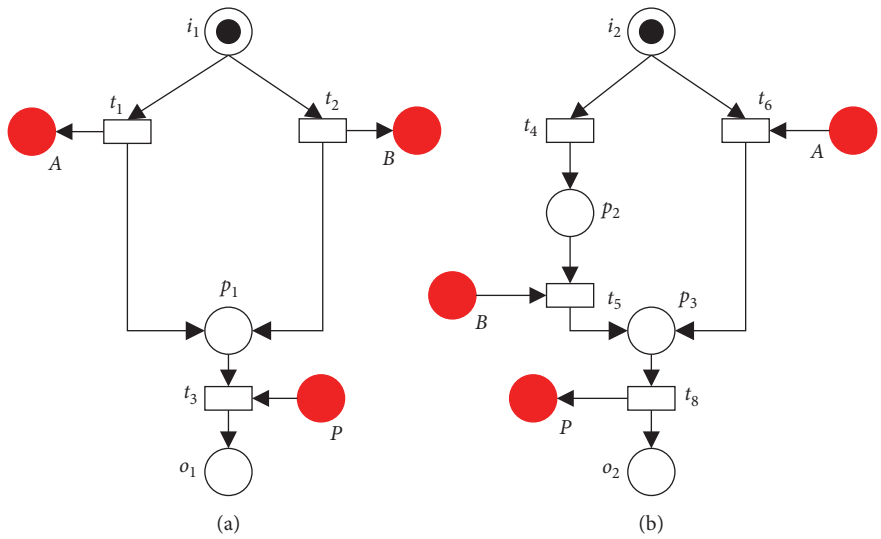

Figure 3: Open nets. (a) Open net N_1 corresponding to *Reta*. (b) Open net N_2 corresponding to *Supp*.

Example 1. To formally construct collaborative business process OP, based on its description, we first convert the business processes corresponding to the BPaaS services of *Reta* and *Supp* into two open nets, i.e., N_1 and N_2 in Figure 3. In particular, in Figure 3, the red places indicate message places, and the label of each transition is described in Table 1.

Then, we can formally construct OP by composing the two open nets, as shown in Figure 4.

Table 1: Labels of N_1 and N_2.

Transition	Label
t_1	Send order *A*
t_2	Send order *B*
t_3	Receive product
t_4	Check stock
t_5	Receive *B*-order
t_6	Receive *A*-order
t_8	Send product

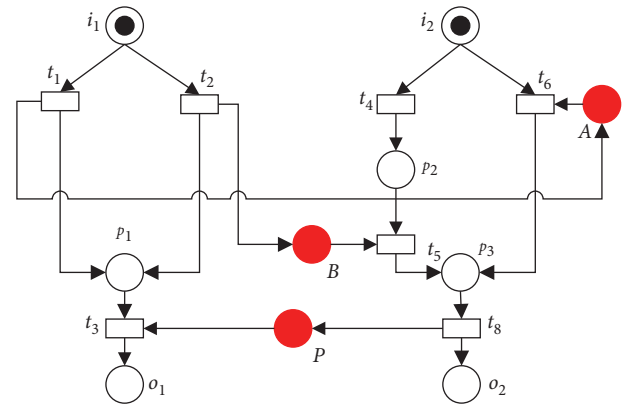

Figure 4: Collaborative business process OP.

4. Checking Correctness

In this section, we first define the correctness of collaborative business processes based on *weak termination* [19]. Then, we define the stubborn set for collaborative business processes and present a method for generating the reduced state space. At last, we propose an algorithm to check the correctness of the collaborative business processes based on its reduced state space.

4.1. Correctness. In this paper, we employ a special variant of *soundness* [4], i.e., *weak termination*, to define the correctness of collaborative business processes, because activities in BPaaS services can be used in different collaborations and the exclusion of some activities in a concrete collaboration may not be a design flow in practice [23].

Definition 3 (Correctness). Let $N = (P, T; F, M_0, M_e)$ be a collaborative business process built by composing BPaaS services in the cloud. Then, for each reachable marking M in N, if M can reach the final marking, then N is called correct.

In essence, Definition 3 implies the fact that each BPaaS service in the collaboration can be successfully terminated, i.e., the final marking can be reached, and the messages generated during the execution of N can be received.

4.2. Stubborn Sets. To check the correctness of collaboration processes, existing approaches (e.g., [6, 10, 11]) typically need to build their full state space at first; thereby, they are inefficient and intractable, even for collaboration processes that are bounded, as the state-space explosion may occur. In order to alleviate the issue, in this paper, we first generate the reduced state space of collaborative business processes using stubborn sets [24] and then check the correctness on it. In this way, our approach can greatly improve checking efficiency for actual collaboration processes.

Currently, multiple stubborn sets for verifying different properties have been proposed, such as the stubborn set for simple linear time logic [25]. In general, the construction of the stubborn set is depended on the properties to be verified [26]. We first present the definition of the stubborn set for

collaborative business processes based on the concepts related to the stubborn set of Petri nets in [18]. Informally, given a marking M, its stubborn set $St(M)$ refers to a set of enabled transitions at M and $St(M)$ remains stubborn when the transitions outside $St(M)$ fire.

Definition 4 (Stubborn Set). Let $N = (P, T; F, M_0, M_e)$ be a collaborative business process and M be its marking. Then, the stubborn set $St(M)$ of M can be described as

(1) If $\exists t \in \text{en}(M)$, then $\exists t \in St(M)$ where $t \in \text{en}(M)$

(2) If $t \in \text{en}(M)$, then $\exists (^\bullet t)^\bullet \in St(M)$

(3) If $t \notin \text{en}(M)$, then $\exists p \in {}^\bullet t$: $M(p) = 0$ and ${}^\bullet p \in St(M)$

In Definition 4, (1) states that the stubborn set is not empty if there are some enabled transitions at marking M; (2) implies that the conflict set of each transition is included in $St(M)$; and (3) implies that the casual set of each transition is covered in $St(M)$.

With Definition 4, below we give an algorithm to calculate the stubborn set corresponding to a special marking.

Algorithm 1 first picks any transition from the currently enabled transitions ($L1$). Then, the algorithm iteratively uses (2) or (3) of Definition 4 until reaching a fixed point ($L3$~$L11$). Assume that N has n transitions, then the time complexity of the algorithm is $O(n)$.

Example 2. Using Algorithm 1, we can generate a possible stubborn set $St(M_0)$ for the initial marking M_0 in OP, i.e., $St(M_0) = \{t_1, t_2\}$. Concretely, if the algorithm chooses the enabled transition t_1 to generate the stubborn set, then transition t_2 is added to $St(M_0)$ as it is in conflict with transition t_1. After that, the algorithm computes the conflict set of transition t_2 as it is enabled. Yet, since the conflict set at this point is empty, the algorithm terminates.

With the concept of the stubborn set, the reduced state space of collaborative business processes can be generated using the following algorithm.

Technically, the basic idea of Algorithm 2 is similar to the process of generating the reachability graph of Petri nets [20]. The only difference between them is that only the stubborn set is used to generate the successors of the marking at each iteration ($L7$~$L15$). Assume that SGG has n nodes, then the time complexity of the algorithm is $O(n)$.

4.3. Checking Correctness. In [21], given a collaboration process, we proposed an effective algorithm for checking its correctness with the concept of transitive closures. Here, we briefly describe its basic idea. That is, the algorithm first generates its state space, i.e., the reachability graph. Then, it calculates the transitive closures corresponding to all nodes in the state space with an algorithm called Floyd–Warshall [27]. Finally, the algorithm determines the correctness based on these generated transitive closures. Concretely, if the transitive closures corresponding to all nodes in the state space cover a final marking, then we derive that the collaboration process is correct, and otherwise, it is incorrect. In the case of incorrectness, two cases exist, i.e., if some nodes (not all) in the state space can reach a final marking, then we define that the collaboration process is partially correct, and otherwise it is fully incorrect. In our context, the algorithm is directly employed to check the correctness of a cloud-based collaboration process. Note that the Floyd–Warshall algorithm is a classic algorithm for computing transitive closures. Its basic idea is that given a directed graph, the algorithm first constructs its adjacency matrix and then calculates transitive closures of nodes based on transitive relations [27]. To save space, the details on the algorithm are not presented in this paper.

Input: $N = (P, T; F, M_0, M_e)$ and marking M
Output: The stubborn set $St(M)$
(1) pick any $t \in \text{en}(M)$;
(2) $St(M) = \{t\}$; $\mathbb{R} = \{t\}$;
(3) **while** $\mathbb{R} \neq \varnothing$ **do**
(4) pick $t \in \mathbb{R}$;
(5) **if** $t \notin \text{en}(M)$ **then**
(6) $\Theta = \{{}^\bullet p \mid \exists p \in {}^\bullet t$ s.t. $M(p) = 0\}$;
(7) **else**
(8) $\Theta = (^\bullet t)^\bullet$;
(9) **end if**
(10) $St(M) = St(M) \cup \Theta$;
(11) $\mathbb{R} = \mathbb{R} \cup (\Theta \backslash St(M))$;
(12) **end while**
(13) **return** $St(M)$;

Algorithm 1: Compute stubborn sets.

Input: $N = (P, T; F, M_0, M_e)$
Output: The reduced state space SSG;
(1) define SSG $= (N, E, M_0, M_e)$;
(2) tag M_0;
(3) add M_0 to Υ and N;
(4) **while** $\Upsilon \neq \varnothing$ **do**
(5) pick any $M \in \Upsilon$;
(6) generate the stubborn set $St(M)$ of M;
(7) **for each** $t \in St(M)$ **do**
(8) compute $M[t > M'$;
(9) add (M, t, M') to E;
(10) **if** M' is untagged **then**
(11) tag M';
(12) add M' to Υ and N;
(13) **end if**
(14) **end for**
(15) **return** SSG;

Algorithm 2: Compute the reduced state space.

Example 3. According to the algorithm described in [21], the fact that OP is partially correct can be derived. Concretely, we first calculate OP's reduced state space SSG, as depicted in Figure 5.

Then, we calculate the transitive closure corresponding to each node in SSG. Clearly, in Figure 5, we can see that $closure(M_5)$ does not contain M_e (i.e., marking M_5 cannot reach the final marking), and hence OP is partially correct.

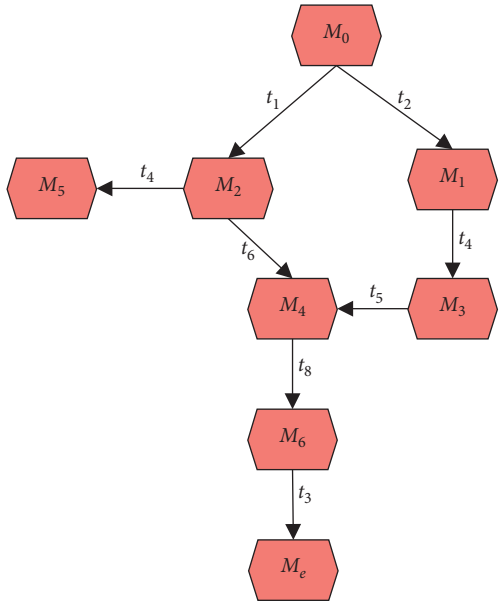

Figure 5: Reduced state space of OP.

5. Generating Reliable Paths

We first present the concept of execution paths. Intuitively, an execution path of an open net refers to a trace from its initial marking to its final marking. Given a business process, its execution path can be formally defined as follows.

Definition 5 (Execution Path). Let $N = (P, T; F, M_0, M_e)$ be a business process; then, its execution path is a sequence of transitions from M_0 to M_e.

Given a collaborative business process, its execution paths, called collaborative execution paths, are formed by the composition of execution paths in multiple business processes.

Definition 6 (Collaborative Execution Path). Let $N = (P, T; F, M_0, M_e)$ be a collaborative business process built by composing BPaaS services in the cloud. Then, one of its execution paths can be defined as $\text{cep} = \{\text{ep}_1, \ldots, \text{ep}_n\}$ where ep_i $(1 \le i \le n)$ is an execution path of N_i.

In collaborative execution paths, not every path can be executed successfully. Based on message places, we can define reliable execution paths from collaborative execution paths.

Definition 7 (Reliable Execution Path). Let $N = (P, T; F, M_0, M_e)$ be a collaborative business process built by composing BPaaS services in the cloud. Then, given one of its execution paths $\text{cep} = \{\text{ep}_1, \ldots, \text{ep}_n\}$, cep is reliable iff for each input message place p_i connected to a transition of ep_i, there exists an output message place p_o connected to a transition of ep_o, such that $i \neq o$.

In essence, a reliable execution path can be seen as a collaborative work plan. In practice, it can guide each business process to act properly, ensuring their collaboration can be successfully terminated.

Input: business processes $N_1, \ldots, N_n$
Output: reliable execution paths $\sum$
(1) generate the state spaces $\text{SG}_1, \ldots, \text{SG}_n$;
(2) obtain the sets of execution paths $\text{EP}_1, \ldots, \text{EP}_n$;
(3) compute $\prod(\text{EP}_1, \ldots, \text{EP}_n)$;
(4) **for each** execution path ep in $\prod(\text{EP}_1, \ldots, \text{EP}_n)$ **do**
(5) **if** ep satisfies Definition 6 **then**
(6) add ep to $\sum$;
(7) **end if**
(8) *end for*
(9) **return** $\sum$;

Algorithm 3: Generate reliable execution paths.

Then, we give an algorithm for generating reliable execution paths for a set of business processes.

Algorithm 3 first generates the state space of each business process ($L1$). Then, the algorithm obtains the execution paths of each state space and their Cartesian product ($L2$~$L3$) is computed. Finally, for each execution path in the cross product, if it satisfies Definition 6, then the algorithm adds it to $\sum$($L4$~$L9$).

By analyzing Algorithm 3, we can derive that its time complexity is $O(|\text{SG}_1| + \ldots + |\text{SG}_n| + |\prod(\text{EP}_1, \ldots, \text{EP}_n)|)$, where $|\text{SG}_i|$ indicates the number of the nodes in SG_i and $|\prod(\text{EP}_1, \ldots, \text{EP}_n)|$ indicates the number of the execution paths in $\prod(\text{EP}_1, \ldots, \text{EP}_n)$.

Example 4. Based on Algorithm 3, we can obtain all reliable execution paths in OP. We first compute the execution paths $\text{EP}_1 = \{t_1{}^\wedge t_3, t_2{}^\wedge t_3\}$ of N_1 depicted in Figure 3(a) and the execution paths $\text{EP}_2 = \{t_4{}^\wedge t_5{}^\wedge t_8, t_6{}^\wedge t_8\}$ of N_2 depicted in Figure 3(b). Then, we compute the Cartesian product between EP_1 and EP_2, i.e., $\prod(\text{EP}_1, \text{EP}_2) = \{(t_1{}^\wedge t_3, t_4{}^\wedge t_5{}^\wedge t_8), (t_1{}^\wedge t_3, t_6{}^\wedge t_8), (t_2{}^\wedge t_3, t_4{}^\wedge t_5{}^\wedge t_8), (t_2{}^\wedge t_3, t_6{}^\wedge t_8)\}$. Finally, according to Definition 6, we derive that the reliable execution paths in OP are $\sum = \{(t_1{}^\wedge t_3, t_6{}^\wedge t_8), (t_2{}^\wedge t_3, t_4{}^\wedge t_5{}^\wedge t_8)\}$. In practice, these reliable execution paths can guide *Reta* and *Supp* to operate in a coordinated manner, ensuring the correct execution of OP. For example, if *Reta* orders product A (i.e., executing t_1), then *Supp* knows that it should choose to receive order A (i.e., executing t_6) instead of checking stock, as the collaborative execution path $(t_1{}^\wedge t_3, t_6{}^\wedge t_8)$ is a reliable execution path, thus avoiding the deadlock described in our motivating example.

6. Implementation and Experiments

In this section, we first introduce our prototype tool called cctool. Then, we validate the proposed approach with actual cases.

6.1. Implementation. The proposed approach is implemented as a module cctool in the PIPE. Currently, the module cctool is submitted to the GitHub (https://github.com/MoqiYNU/cctool). The running interface of cctool is shown in Figure 6, where the motivating example OP is validated.

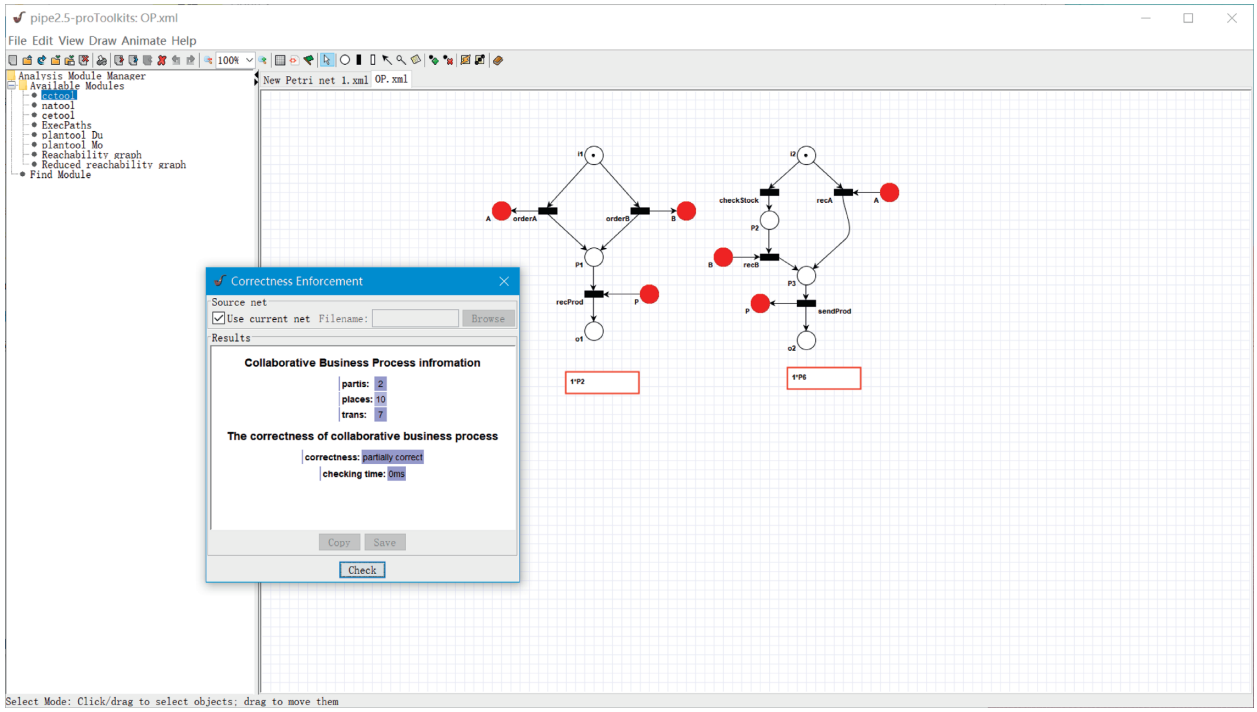

Figure 6: Screenshot of the tool cctool.

As presented in Figure 6, at the right side of the interface is the workspace of the PIPE, where each business process can be modeled separately. By double-clicking the module label of cctool, a collaboration process can be constructed through message place fusion and then its reduced state space is generated. Finally, its correctness will be checked based on the reduced state space. In case it is partially correct, all reliable execution paths in it will be written to the hard disk as a text file. With the help of these reliable execution paths, the collaboration process executes according to the specified paths and can be successfully terminated eventually.

6.2. Experiments. To confirm the effectiveness and efficiency of the proposed approach, we validate it with actual cases. In our experiments, we utilized a PC with Inter(R) CORE i7 CPU 1.80 GHz and 16 GB memory, running Windows 10.

6.2.1. Cases. Since public collaboration processes cannot be available at present [28], in [21], we build a case set that contains 30 diverse and practical collaboration processes from available resources such as research papers (e.g., [6–8, 10, 11, 29]) and other online materials (e.g., the official website of BPMN). Additionally, based on the 7PMG guideline [30], we also confirm that these cases are reasonable for our experiments, as each collaboration process in the case set contains approximately 50 activities and this is roughly consistent with the tasks involved in the actual process. Currently, these cases have been submitted to the GitHub (https://github.com/MoqiYNU/Cases). In this paper, we directly employ these cases to conduct our experiments. To save space, the details (e.g., places and transitions) of each case are presented in [21].

6.2.2. Effectiveness. In this paper, the effectiveness means that the proposed approach can successfully achieve correctness checking. In our experiments, we also compare our approach with two types of typical correctness checking approaches, i.e., the checking approach based on the full state space (called CaF) [6, 10, 11] and the checking approach based on the view (CaV) [12, 13]. For the sake of simplicity, our approach in experiments is denoted as CaS, i.e., a checking approach based on the stubborn set.

Table 2 presents the experimental results of correctness checking for all cases, where "+" means correct, "+/−" means partially correct, and "−" means fully incorrect. Following the experimental results, we can see that our approach (i.e., Cas), and both the CaF and CaV approaches can complete the correctness checking for all cases. Meanwhile, we also observe that the checking results of our approach are consistent with the CaF and CaV approaches, thereby confirming the fact that our approach is effective.

6.2.3. Efficiency. In this paper, the efficiency means that the proposed approach can more efficiently achieve correctness checking compared with existing typical approaches. In our experiments, we also compare our approach with CaF and CaV.

By recording the time that it takes for each approach to detect the case, we obtain the average running overheads of the three approaches, as shown in Figure 7.

In Figure 7, we observe that for large cases (i.e., the case that contains more states), CaF needs to take more time to

Table 2: Results of correctness checking.

Case	Checking results	
	Cas	CaF|CaV
Ca-05~Ca-09, Ca-14~Ca-18, Ca-26~Ca-27	+	+ | +
Ca-01~Ca-04, Ca-11~Ca-13, Ca-19~Ca25, Ca-28~Ca-30	+/−	+/− | +/−
Ca-10	−	− | −

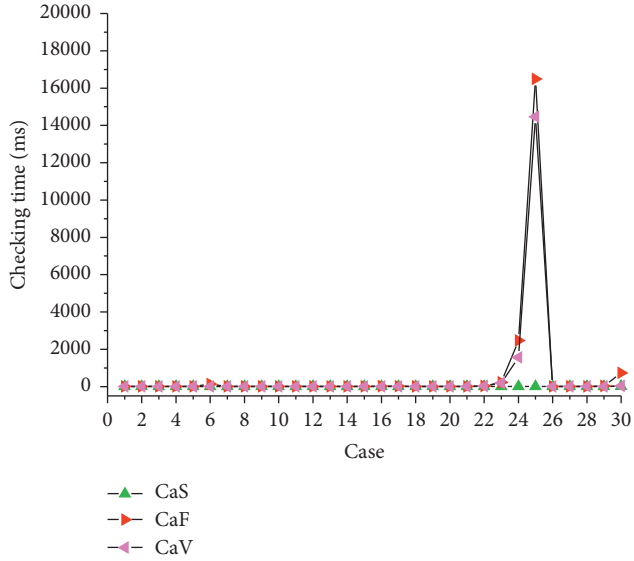

Figure 7: Checking time of three approaches.

complete the correctness checking. For example, for Ca-25, it takes 16488 ms to complete correctness checking. By analyzing CaF, we find that this is mainly caused by the full state space that needs to be exploited during its correctness checking. As far as CaV is concerned, there are some differences. That is, for structured large cases with more internal transitions (i.e., transitions without associated message places), the approach can quickly achieve correctness checking. For example, for Ca-30 which is structured and contains 22 internal transitions (55 transitions in total), it only takes 25 ms for complete checking while 730 ms is taken for CaF. However, for unstructured large cases with less internal transitions, the approach still takes more time to complete correctness checking. For example, for Ca-25, it still takes 14460 ms to complete correctness checking. In practical applications, since most process models are not structured [14], the checking efficiency of the approach may not increase significantly. This is also confirmed in Figure 7, from which we can see that the checking efficiency of CaV, on all cases, is not significantly improved compared to CaF. As far as our approach is concerned, compared with CaF and CaV, its checking efficiency has been greatly improved, as only the reduced state space needs to exploit during correctness checking. For example, for Ca-25, it only takes 5 ms to achieve correctness checking as only 135 nodes and 182 edges are generated in its reduced state space instead of 1939 nodes and 8040 edges in its full state space.

Based on the experimental results above, both the effectiveness and efficiency of our approach are confirmed. In practice, the construction of collaborative business processes in the cloud can benefit from the proposed approach.

7. Related Work

Both the state space-based and view-based checking approaches are related to our approach.

7.1. State Space-Based Checking Approaches. To sum up, state space-based checking approaches can be divided into the following three subtypes: the automata-based checking approach, the Petri net-based checking approach, and the process algebra-based checking approach.

7.1.1. Automata-Based Checking Approaches. In [31], Xu et al. first convert BPEL processes into the guarded automata models based on transformation rules. Then, they translate the generated automata models into Promela processes. Finally, some conversational properties are verified on SPIN. In [32], Zhou et al. proposed a formal technique to verify the interaction among web service-based processes considering requestors' requirements. In [33], Flavio et al. first provide a direct formalization for BPMN based on Labeled Transition Systems. Then, they verify some LTL-based properties (e.g., the reachability property) using the model checking technique.

7.1.2. Petri Net-Based Checking Approaches. In [6], Aalst first define the correctness criterion in terms of soundness. Then, they build the cross-organizational workflow based on synchronous and asynchronous communications. Lastly, they verify its correctness based on the reachability graph. Zhang et al. [29] proposed a Petri net and Pi calculus-based approach to model and analyze business collaborations. In the approach, they first fuse the two formal methods based on a mapping method, in which Petri nets are used to specify the local flow of the business process while the interaction between them is specified by Pi calculus, and the mapping integrates both to obtain a unified model. Then, they generate the state graph of the unified model, and the soundness can be verified on it. Ge et al. [34] proposed an effective method to verify the correctness of cross-organizational workflows based on the invariant analysis. In this approach, they first model cross-organizational workflows using Interaction-Oriented Petri Nets (IOPN). Then, they decompose the model into a set of sequence diagrams. Lastly, the correctness can be verified on these sequence diagrams. Zeng et al. [10] proposed an approach to model and verify cross-department business processes. In this approach, they first employ the RM_WF_Net (a WF-net [6] extended with message and resource factors) to model cross-department business collaborations. Then, they verify their correctness based on their reachability graph. Kheldoun et al. [35] proposed a verification technique for complex business processes based on high-level Petri nets. In this approach,

they first employ Business Process Modeling Notation (BPMN) to describe a collaborative process. Then, they map the generated model to Petri nets based on some proposed rules. Lastly, they use model checking to check its correctness. Du et al. [17] proposed a three-stage approach to analyze the time compatibility via model checking. In this approach, they first model each web service as a fragment described Petri nets. Then, they transform each fragment into a time automata net (TAN). Lastly, by composing these generated TANs, a web service composition is built and its correctness can be analyzed by UPPAAL. To model and verify the emergency response process, Duan et al. [11] proposed a refinement-based approach. They first refine a top-level model into a bottom-level model using some collaboration patterns from different abstraction levels and then verify the refined model at its reachability graph.

7.1.3. Process Algebra-Based Checking Approaches. Wong and Gibbons [36] employ the process algebra CSP (Communicating Sequential Process) to formalize BPMN and then verify some correctness properties using the tool FDR (Failure Divergence Refinement). Mendoza [37] proposed a formal compositional verification approach to specify and verify business processes. In this approach, they employ CSP + T (Communicating Sequential Processes + Time) to model the BPMN model with time and then use model checking to verify its correctness. Based on the idea of model transformation, Zhu et al. [38] first utilize the idea of model transformation to establish a verification framework. Then, they map the composition to CSP processes based on a set of transformation rules. Lastly, the correctness can be validated by the tool FDR.

7.2. View-Based Correctness Checking Approaches. To support B2B collaboration, Norta and Eshuis [12] proposed an approach to describe structurally collaborative processes. Their approach first defines private and public layers based on WF-nets. Then, the approach uses some combination projections such as grey box projection to generate external processes from conceptual processes. Lastly, the correctness of business process collaborations can be verified on the composition of external processes. To more effectively verify the correctness of cross-organizational processes, Mo et al. [13] first use private processes to describe the complete process of organizations. Then, they abstract them into public views based on four rules. Lastly, the collaborative business process is built by composing public processes. Based on the approach, the verification efficiency can be improved. To effectively model cross-organizational emergency response processes, Duan et al. [24] first introduce the TRM_WF_net, i.e., a WF-net with messages and resources. Then, based on TRM_WF_nets, they present a three-layer framework to describe the emergency response process. Lastly, a set of rules are given to reduce the TRM_WF_net model. In practice, these rules can improve the evaluation efficiency of temporal performance while considering privacy.

7.3. Summary of Existing Work. Following the literature review above, we observe that most of the existing approaches (e.g., [6, 10, 17, 29, 31–38]) verify the correctness using the full state space. Hence, these approaches may suffer from a low efficiency, as the state-space explosion exists. Several approaches (e.g., [12, 13, 24]) check correctness based on the public view. To a certain extent, these approaches can improve the checking efficiency while considering privacy. Yet, they require business processes in the collaboration to be structured, and the assumption may not hold in practice [14]. Thus, the efficiency of their correctness checking may not increase significantly. In comparison, since our approach checks correctness based on stubborn sets and puts no restrictions on the business process, the verification efficiency of our approach can be greatly improved.

8. Conclusion

Based on BPaaS services, developers gathered in the cloud can compose them to construct collaborative business processes to achieve value-added services. However, the correctness is considered to be a key problem at their design phase. In this paper, based on the stubborn set, we propose an approach to check their correctness. In practice, our approach can greatly improve the development efficiency compared with the existing approaches.

In general, the time and resource properties are considered to be two important aspects for collaborative business processes in the cloud [9]. Our further work will develop effective techniques to evaluate the temporal performance and to resolve the resource conflict problem. Additionally, edge computing and the Internet of Things (IOT) are two areas closely related to cloud computing [5, 39–44]. Based on the cloud service, our further work will extend the proposed approach to the two areas.

Data Availability

The cases used in our experiments can be accessed at the GitHub https://github.com/MoqiYNU/Cases, and this is described in Section 6.2.1 of our updated manuscript.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Natural Science Foundation of China (nos. 61862065, 61662085, and 61702442).

References

[1] A. Weiss, "Computing in the clouds," *netWorker*, vol. 11, no. 4, pp. 16–25, 2007.

[2] P. M. Mell and T. Grance, "The NIST definition of cloud computing," Technical Report, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2011.

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[4] M. X. Wang, K. Y. Bandara, and C. Pahl, "Process as a service," in *Proceedings of the 2010 IEEE International Conference on Services Computing*, pp. 578–585, IEEE Computer Society, Miami, FL, USA, 2010.

[5] X. Xu, Y. Chen, X. Zhang et al., "A blockchain-based computation offloading method for edge computing in 5G networks," *Software: Practice and Experience*, 2019.

[6] W. Aalst, "Modeling and analyzing interorganizational workflows," in *Proceedings of the 1st International Conference Application of Concurrency to System Design*, pp. 262–272, IEEE Computer Society, Los Alamitos, CA, USA, March 1998.

[7] I. Chebbi, S. Dustdar, and S. Tata, "The view-based approach to dynamic inter-organizational workflow cooperation," *Data & Knowledge Engineering*, vol. 56, no. 2, pp. 139–173, 2006.

[8] Q. Mo, L. Bai, F. Dai, J. Qin, Z. Xie, and T. Li, "A correctness enforcement approach for collaborative business processes," *IEEE Access*, vol. 7, pp. 87069–87084, 2019.

[9] W. Song, C. Zhang, and H. Jacobsen, "An empirical study on data flow bugs in business processes," *IEEE Transactions on Cloud Computing*, 2018.

[10] Q. Zeng, F. Lu, C. Liu, H. Duan, and C. Zhou, "Modeling and verification for cross-department collaborative business processes using extended Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, pp. 349–362, 2015.

[11] H. Duan, C. Liu, Q. Zeng, and M. Zhou, "Refinement-based hierarchical modeling and correctness verification of cross-organization collaborative emergency response process," *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 2018.

[12] A. Norta and R. Eshuis, "Specification and verification of harmonized business-process collaborations," *Information Systems Frontiers*, vol. 12, no. 4, pp. 457–479, 2010.

[13] Q. Mo, F. Dai, R. Zhu et al., "An approach to extract public process from private process for building business collaboration," *Journal of Computer Research and Development*, vol. 54, no. 9, pp. 1892–1908, 2017.

[14] K. Ajay, P. Pascal, and S. Gwen, "Checking business process evolution," *Science of Computer Programming*, vol. 170, no. 15, pp. 1–26, 2019.

[15] W. Tan, Y. S. Fan, and M. Zhou, "A Petri net-based method for compatibility analysis and composition of web services in business process execution language," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 1, pp. 94–106, 2009.

[16] P. C. Xiong, Y. S. Fan, and M. C. Zhou, "A Petri net approach to analysis and composition of web services," *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, vol. 40, no. 2, pp. 376–387, 2010.

[17] Y. Du, B. Yang, and H. Hu, "Model checking of timed compatibility for mediation-aided web service composition: a three stage approach," *Expert Systems with Applications*, vol. 112, no. 1, pp. 190–207, 2018.

[18] A. Valmari, *Stubborn Sets for Reduced State Space Generation*, Springer, Berlin, Germany, 1989.

[19] W. Aalst, N. Lohmann, and M. L. Rosa, "Ensuring correctness during process configuration via partner synthesis," *Information Systems*, vol. 37, no. 6, pp. 574–592, 2012.

[20] W. Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business processes*, Springer, Berlin, Germany, 2011.

[21] Q. Mo, W. Song, F. Dai et al., "Development of collaborative business processes: a correctness enforcement approach," *IEEE Transactions on Services Computing*, 2019.

[22] Y. Du, X. Li, and P. Xiong, "A Petri net approach to mediation-aided composition of web services," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 429–435, 2012.

[23] N. Lohmann, "Compliance by design for artifact-centric business processes," *Information Systems*, vol. 38, no. 4, pp. 606–618, 2013.

[24] H. Duan, C. Liu, Q. Zeng, and M. Zhou, "A package reduction approach to modeling and analysis of cross-organization emergency response processes with privacy protected," *IEEE Access*, vol. 6, pp. 55573–55585, 2018.

[25] B. Lehmann, N. Lohmann, and A. Wolf, "Stubborn sets for simple linear time properties," in *Application and Theory of Petri Nets*, Springer, Heidelberg, Germany, 2012.

[26] L. M. Kristensen and A. Valmari, "Improved question-guided stubborn set methods for state properties," in *Lecture Notes in Computer Science*, vol. 1825, pp. 282–302, Springer, Berlin, Germany, 2000.

[27] H. Thomas, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, USA, 2009.

[28] M. Borkowski, W. Fdhila, M. Nardelli et al., "Event-based failure prediction in distributed business processes," *Information Systems*, vol. 81, pp. 220–235, 2018.

[29] L. Zhang, Y. Lu, and F. Xu, "Unified modelling and analysis of collaboration business process based on Petri nets and Pi calculus," *IET Software*, vol. 4, no. 5, pp. 303–317, 2010.

[30] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, "Seven process modeling guidelines (7PMG)," *Information and Software Technology*, vol. 52, no. 2, pp. 127–136, 2010.

[31] X. Fu, T. Bultan, and J. Su, "Analysis of interacting BPEL web services," in *Proceedings of the ACM International Conference on World Wide Web*, pp. 621–630, New York, NY, USA, May 2004.

[32] Z. Zhou, L. T. Yang, S. Bhiri, L. Shu, N. Xiong, and M. Hauswirth, "Verifying mediated service interactions considering expected behaviours," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1043–1053, 2011.

[33] C. Flavio, F. Fornari, A. Polini, B. Re, and F. Tiezzi, "A formal approach to modeling and verification of business process collaborations," *Science of Computer Programming*, vol. 166, no. 15, pp. 35–70, 2018.

[34] J.-D. Ge, H.-Y. Hu, Y. Zhou, H. Hu, D.-Y. Wang, and X.-B. Guo, "A decomposition approach with invariant analysis for workflow coordination," *Chinese Journal of Computers*, vol. 35, no. 10, pp. 2169–2181, 2012.

[35] A. Kheldoun, K. Barkaoui, and M. Ioualalen, "Formal verification of complex business processes based on high-level Petri nets," *Information Sciences*, vol. 385-386, pp. 39–54, 2017.

[36] P. Y. H. Wong and J. Gibbons, "Formalisations and applications of BPMN," *Science of Computer Programming*, vol. 76, no. 8, pp. 633–650, 2011.

[37] L. E. Mendoza, M. I. Capel, and A. P. Marıa, "Conceptual framework for business processes compositional verification," *Information and Software Technology*, vol. 54, no. 2, pp. 709–730, 2017.

[38] Y. Zhu, Z. Huang, and H. Zhou, "Modeling and verification of web services composition based on model transformation," *Software: Practice and Experience*, vol. 47, no. 5, pp. 709–730, 2017.

[39] X. Xu, Q. Liu, Y. Luo et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522–533, 2019.

[40] L. Qi, Y. Chen, Y. Yuan et al., "A Qos-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web Journal*, 2019.

[41] X. Xu, Y. Xue, L. Qi et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[42] X. Xu, C. He, Z. Xu et al., "Joint optimization of offloading utility and privacy for edge computing enabled IoT," *IEEE Internet of Things Journal*, 2019.

[43] L. Qi, Q. He, F. Chen et al., "Finding all you need: web APIs recommendation in web of things through keywords search," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 5, pp. 1063–1072, 2019.

[44] X. Xu, Y. Li, T. Huang et al., "BeCome: blockchain-enabled computation offloading for IoT in mobile edge computing," *IEEE Transactions on Industrial Informatics*, 2019.