Journal of Intelligent Manufacturing (2020) 31:1737–1748
https://doi.org/10.1007/s10845-018-1422-y

Blockchain-based business process management (BPM) framework for service composition in industry 4.0

Wattana Viriyasitavat[1] · Li Da Xu[2] · Zhuming Bi[3] · Assadaporn Sapsomboon[1]

Received: 7 March 2018 / Accepted: 7 May 2018 / Published online: 14 May 2018

Abstract

Business process management (BPM) aims to optimize business processes to achieve better system performance such as higher profit, quicker response, and better services. BPM systems in Industry 4.0 are required to digitize and automate business process workflows and support the transparent interoperations of service vendors. The critical bottleneck to advance BPM systems is the evaluation, verification, and transformation of trustworthiness and digitized assets. Most of BPM systems rely heavily on domain experts or third parties to deal with trustworthiness. In this paper, an automated BPM solution is investigated to select and compose services in open business environment, Blockchain technology (BCT) is explored and proposed to transfer and verify the trustiness of businesses and partners, and a BPM framework is developed to illustrate how BCT can be integrated to support prompt, reliable, and cost-effective evaluation and transferring of Quality of Services in the workflow composition and management.

Keywords Industry 4.0 · Business process management (BPM) · Block-chain technology (BCT) · Internet of things (IoT) · Trustworthiness · Service selection and composition · Smart contracts · Quality of Services (QoS)

Introduction

Industry 4.0 brings together a series of technological innovations such as Internet of Things (IoT), Cyber Physical System (CPS), Service-oriented Architecture (SoA), Blockchain, and Cloud technologies (Xu and Duan 2018; Xu et al. 2017; Lu 2017a, b). Interoperation and integration of modern industries within for example the manufacturing and service segments rely on cross-organizational business processes (Seok and Nof 2018; Mollahoseini Ardakani et al. 2018; Xie et al. 2017; Lu et al. 2016; Danila et al. 2016). With an increase of the complexity and turbulence of globalized business environment, companies are facing a massive challenge to optimize and innovate their business process so that they can gain business advantages in highly competitive market (Hsieh and Lin 2014). Therefore, business process management (BPM) systems in Industry 4.0 are forced to digitize and automate business process workflows and support the transparent interoperations of service vendors. .

Nowadays, most of business processes operate across organizational boundaries (Camarinha-Matos and Pantoja-Lima 2001); however, many activities are manually performed and many decisions related to business processes are made by people. For instance, a bank or escrow holds a payment before goods are successfully delivered. One of the greatest changes to meet the standard of Industry 4.0 is how to implement digitalized and automated BPM systems underpinned by technological innovations. In this paper, we tackle with the trustiness in BPM, which is the critical and fundamental to any business, and we explore the possibility of using BCT. If BCT can be used in a company supply chain that implements a cyber-physics system (CPS) in its plants, the system can automatically process an order for

✉ Wattana Viriyasitavat
hardgolf@hotmail.com; wattana@cbs.chula.ac.th

Li Da Xu
lxu@odu.edu

Zhuming Bi
biz@ipfw.edu

Assadaporn Sapsomboon
assadaporn@cbs.chula.ac.th

[1] Department of Statistics, Faculty of Commerce and Accountancy, Chulalongkorn University, Bangkok, Thailand

[2] Department of Information Technology & Decision Sciences, Old Dominion University, Norfolk, VA 23529, USA

[3] Department of Civil and Mechanical Engineering, Purdue University Fort Wayne, Fort Wayne, IN 46805, USA

the equipment replacement by selecting the best supplier based on the specified requirements. Moreover, the payment can be triggered by coding conditions as the entities in a Blockchain smart contract. The payment will be realized if only the products are delivered accordingly (Faizod 2006).

In fact, the realization of the supply chain example requires the conglomeration of technologies behind the scene (Mahdavi et al. 2009). The workflow management technologies such as Petri Net, Pi Calculus, Business Process Execution Language (BPEL), Web-Services Business Process Execution Language (WS-BPEL), and Business Process Model and Notation (BPMN) are widely used for business process modeling. The executions of tasks inside a business process are fulfilled by services, which their interfaces are standardized by using SoA technology encapsulation (Zhang et al. 2017). Service selection and composition demand for formal methods that specify Quality of Service (QoS) requirements and conditions for participation, in which workflow modeling and specification technologies play its role (Chhun et al. 2016; Viriyasitavat 2013). Cross-organizational business processes often aggregate distributed services at the phases of decision-making support and runtime operations. Everything as a Service (EaaS), enabled by the Internet of Things (IoT), Service Oriented Architecture (SoA), and Cloud technologies, increases a number of services with similar functionalities but differed in QoS values (e.g. availability, response time) and preferences (e.g. price, reputation) (Puttonen et al. 2016; Mohammad and Thomas 2009). We call for effective methodologies to select services and compose them as service workflows for the specified goals (Viriyasitavat et al. 2012; Lee et al. 2009). Compliance checking technology is a major part to automate the service selection and replacement during design space and runtime, which greatly supports dynamic environment where service attributes are continuously changing. Several techniques (Viriyasitavat and Martin 2012; Viriyasitavat et al. 2014a, b; Xu et al. 2012) are available to select preferable and best-fit services in different contexts.

Although a number of approaches have been proposed to tackle the service selection and composition problem (Coutinho et al. 2016; Dustdar and Schreiner 2005; Rao and Su 2005), one clear obstacle that prohibits the success of BPM is the trustiness of QoS parameters. Some of existing works made the assumption that these values are reliable (Mohammad and Thomas 2009; Huang et al. 2005; Hwang et al. 2008); while the others relied on the third-party authorities such as UDDI to collect, maintain, and provide QoS values for pubic use with or without fee via APIs (Ran 2003; Ali 2005). It was also explored to utilize the attribute-based digital certificates to endorse the values (Ran 2003; Haq et al. 2010). However, the preliminary works discovered some unsolved issues:

1. The assumption that QoS values are inherently trusted is not a good proposition in real-world applications.
2. Using central authorities are facing a number of challenges and risks such as scalability, high maintenance overhead, handling Denial of Service (DoS)/Distributed Denial of Service attack (DDoS), single point of failure, fraud, and trust being bound to one entity.
3. EaaS enabled by IoT, SoA, and Cloud technologies exponentially increases geographically distributed services over the Internet. Managing attribute-based digital certificate is operatively and economically impractical, since QoS values are dynamic and need real-time update and verification.

To overcome the above obstacles, this paper presents the application of BTC that underpins Bitcoin, the first cryptocurrency system launched in 2008, and other several crytocurrencies (Nakamoto 2008). It has shown its great potential to be used an effective solution to service selection and composition in service workflow. We make our effort in integrating BCT into BPM systems to achieve the benefits in particularly in three main aspects shown in Fig. 1.

1. Building trusted relations among previously-unknown services based on QoS values without the involvement of trusted intermediaries.
2. Reducing costs in multiple aspects: (1) manual operations and fee that traditionally associate with trusted intermediaries are no longer visible, (2) no central authorities are required to maintain QoS values, and (3) the attribute-based digital certificate scheme can be omitted but the proposed system still maintains the same level of security.
3. Accelerating transactions by automating activities in service selection and composition with real-time update and verification of QoS values.

In a summary, these benefits promote the overall automation and interoperability of the cross-organizational business processes in Industry 4.0.

Related works

A growing number of distributed, possibly ad-hoc, services that operate autonomously in dynamic environments where some provide the same functionality with different QoS values makes service selection and composition very challenging. Each company fully or partially manages its business process where the strategies of the service selection depend

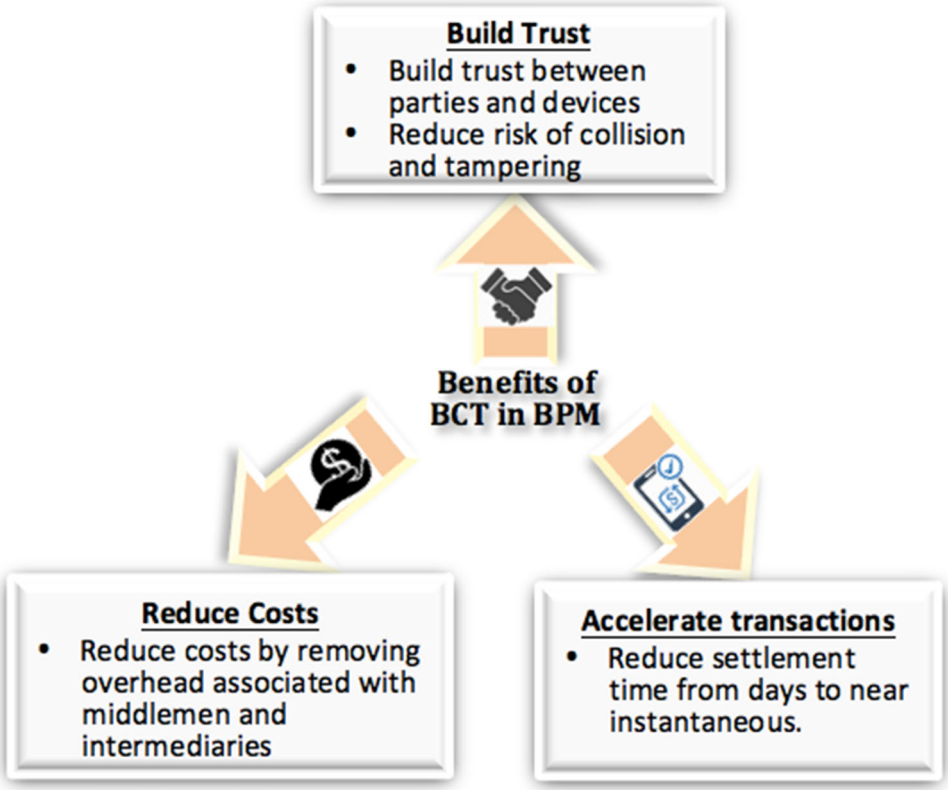

Fig. 1 Three main benefits of BCT in BPM (IBM 2018)

on its own business conditions and requirements (Watanabe et al. 2012). For instance, the reputation, availability, reliability, response time, latency, price, are often used as parameters in selecting services. The nature of these QoS requirements and their compliances is dynamic with respect to time. QoS can change frequently which require collecting, processing, and assessing the information for decision-making activities in a real-time fashion. This section summarizes the prominent works in the area of QoS-based service selection and composition schemes.

Service selection and composition are characterized as a decision and optimization problem (Huang et al. 2005). This process is very complex and takes the consideration of the impact in balancing the performance in the optimization. Zeng et al. (Zeng et al. 2010) presented a process model for service composition with a number of quality criteria including price, duration, reliability, availability, and reputation. However, these QoS values were observed with single entity that utilizes services. Ardagna and Pernici (2005) developed an approach to find the best set of services by transforming the service composition problem into a mixed integer linear problem. Mohammad and Thomas (2009) combined the local and global selection techniques for the scalable solution.

There were a few surveys on the service selection and composition. The traditional approach for the QoS-based service evaluation collected and stored QoS information in the centralized registry, usually through Cloud services. For instance, Liu et al. (2004) designed a QoS registry for a phone service market place. Maximilien and Singh (2004) proposed a solution that delegates QoS evaluation to authorized agents under the pre-defined ontology. Serhani et al. (2005) and Yu and Lin (2005) introduced respectively QoS broker architecture and framework to handle QoS information. Agflow middleware (Zeng et al. 2004) is an implementation of the service composition using service broker being in charge of managing QoS information. These approaches were lacking scalability, and they hardly operate across business boundaries to support global scale business process (Li et al. 2007). Additional overhead of implementing and maintaining registry was another economical factor to its realization. To alleviate this problem, Gibelin and Makpangou (2005) developed the distributed QoS architecture using the authorized proxies to manage QoS information. Q-Peer (Li et al. 2007) is a peer-to-peer prototype to fully manage QoS information without centralized registries. QoS information had several copies to share the load on different peers to promote scalability and performance. Gathering information directly from other agents was another technique to solve scalability. But the integrity of the recommendation could be unreliable and incorrect (Ali et al. 2005).

One raised issue in these QoS schemes is the trustworthiness of QoS information maintained by authorized or peer entities. Several mechanisms were proposed using monitoring QoS techniques (Sahai et al. 2002; Ludwig et al. 2004; Dan et al. 2004; Barbon et al. 2006; Jurca et al. 2007) by periodically probing service with specific criteria. Reliable QoS monitoring was therefore crucial for ensuring trustworthiness. Another technique by Ran (2003) and Haq et al. (2010) employed attribute certificate PKI to endorse the claims of QoS values for a Web service before using. As already mentioned, managing attribute certificate was impractical since QoS values are dynamic and need real-time update and verification. The problem of these techniques falls into the same category of the reliance on the central registries. This hinders the scalability and performance and also requires additional implementation and maintenance cost. The trustworthiness of information relies purely on the monitoring systems and/or attribute certificate.

In a dynamic large-scale environment with the application of IoT and Cloud technologies, the realization of the trusted central authority-based frameworks is facing major challenges regarding scalability, integrity, and trustworthiness, which is far from being satisfactory in the context of cross-organizational business process collaboration.

Backgrounds

Blockchain technology (BCT)

Blockchain is one of core technologies of Industry 4.0, which provides a promising solution to the aforementioned problems. This technology is first revealed in a famous cryptocurrency Bitcoin (Nakamoto 2008). Technically speaking Blockchain is a database that creates a distributed and tamper-proof (immutable) digital ledger of transactions. The chain is auditable as it contains timestamp of blocks binding together and maintained by every participating nodes (Liu et al. 2017). Once transactions are written into a block, they cannot be modified or destroyed. Various efforts have been made to use BCT to decentralize the IoT infrastructure, promoting trust of information that can be shared in decentralized fashion (Kshetri 2017). In business process 4.0, a promising future seems likely to gain Blockchain capabilities to promote:

1. Resilience carried out by distributed ledgers that diminish the single point of failure as appeared in central registry,
2. Scalability as the computation power can be enhanced by the number of distributed peers in the network without using centralized systems,
3. Security provided by strong cryptography such as strong cryptographic hash like SHA-256 or encryption using ECC or RSA, used to create digital signature such as ECDSA,
4. Autonomy since services can carry out functionalities autonomously without the intervention of central authorities or brokers, and
5. Trustworthiness as QoS information is persistent and cannot be modified.

Workflow specification languages

BCT will enable the execution of collaborative business processes involving untrusted parties without requiring a central registry (García-Bañuelos et al. 2017). The incorporation of Blockchain into business process for service selection and compositions will significantly leverage interoperation of services and cross-organizational business processes by promoting scalability, security, and trustworthiness across involved parties. Modern business processes rely on real-time QoS monitoring to check weather services are still complied with requirements. With the increasing number of services supported by IoT and Cloud technologies, industries are facing the big challenge to select preferable services for business process workflow composition. The emergence of service workflows in a large-scale opened environment requires formal languages for specifying QoS of services as participation requirements (Viriyasitavat and Martin 2017). These requirements can be specified by two parties: (1) a company or a workflow owner that specifies requirements of services to be selected as part of its business processes, and (2) services may express their requirements in order to provide functionalities (Viriyasitavat and Martin 2010, 2011a). Making a workflow verifiable by a specification language requires formal and mathematical representation (Viriyasitayat et al. 2018a, b). Service workflow has been employed to support business such as optimizing and automating processes according to workflow requirement specifications (Viriyasitavat and Martin 2017). Compliance checking algorithms (Viriyasitavat and Martin 2012, 2017; Viriyasitavat et al. 2014a, b; Xu et al. 2012; Viriyasitavat 2016) provide automatic verification of the specification.

Various approaches have been proposed to define and abstract workflow processes including Petri Net (Hollingsworth 1994; Workflow Management Coalition 1999; Han et al. 2018), pi calculus (van der Aalst 2005), UML Activity Diagram, BPEL or BPMN, etc. (Fehling et al. 2014). However, there are a few studies on QoS-based service workflow specification languages for service selection and composition. According to the general requirements needed for service (Viriyasitavat and Martin 2011b), related works in this area are present. Altunay et al. (2005) addresses two types of trust relationships in a workflow. Direct trust occurs between adjacent, while indirect trust is the relationship of services that are not immediately connected. Trust estab-

lishment is based on QoS values; however, this work fails to explain how QoS values are used. HENS+ (Viriyasitavat 2009) is a Petri-Net-based framework to define delegation and trust transitivity with QoS attributes. This work emphasizes on the domain level where QoS is adhered to service workflow instead of services. Halle et al. (2007) presented CTL-FO+ for data-aware constraints where service qualities are determined based solely on message exchange. Zeng et al. (2004) developed a middleware called AgFlow to facilitate service selection. One shortcoming of this approach is QoS parameters are limited to price, reputation, execution rate, duration, and availability, where in fact the number and importance of QoS attributes are subjective and vary in different business processes and time. To overcome this limitations, SWSpec (Viriyasitavat et al. 2012) and its compliance checking (Viriyasitavat and Martin 2012; Viriyasitavat et al. 2014; Xu et al. 2012; Viriyasitavat 2016) have been developed based on the concept of whenever better services are available, there should be an efficient way to select and use those services. The better services imply services that better satisfy QoS requirements, for instance, one service has higher reputation score over another. The requirements are subjective and can be specified independently and formally using SWSpec language with its logic-based propositions to specify various QoS requirements in the form of formal formulae. Compliance checking provides a tool based on formal method to verify services candidates that satisfy requirements with extension to evaluate better services as mentioned. Viriyasitavat (2016) employed Fuzzy AHP to evaluate the best alternative among service candidates based on a ranking approach. Formal specification languages for QoS specification and compliance checking are two essential ingredients in the modern business process innovation.

However, all of existing works rely on the trustworthiness of QoS values and how they are managed is left for further implementation. These values are assumed to be inherently trusted and available when needed.

Blockchain-enabled service selection

A large number of Blockchain applications are being implemented across a range of industries, including finance and banking, insurance, supply chain management, and energy management. This section focuses how BCT will be utilized in business processes. The aspects of BCT for business process improvement opportunities are demonstrated via "Value-driven business process management" framework (VBPM) (Franz and Kirchmer 2012) in the following benefits.

1. Efficiency and quality: Blockchain-enabled solutions increase efficiency by minimizing time and cost such as automatically replacing old services when a new one is discovered without the involvement of central registries.
2. Agility and Compliance: Service workflow, its specification languages, and SoA technologies play important roles in automating compliance checking and therefore increase agility of modern business. Everything can be encapsulated and standardized in the form of services using SoA and IoT technologies. Blockchain provides trusted QoS information of services to be selected.
3. Integration and networking: Blockchain enables automation in the integration of cross-organizational business processes by eliminating manual operations carried out from intermediaries. This can apply to service selection and composition, which involve the collaboration of multiple parties such as workflow owner, services, and central registries.

Figure 2 depicts a part of service workflow with a simple demonstration of Blockchain playing its part. Earthquake Detection task requires three composite services with two QoS requirements about reputation and availability rate. Tsunami Detection task requires three composite services with one QoS requirement about availability rate. QoS values are stored in Blockchain ledgers resided in IoT-underlined sensor services and possibly other service consumers or other authorized entities. Service consumers are responsible to provide QoS values (see "A framework of Blockchain-based service selection and composition" section for further details).

BCT is being seen as a substitute of intermediaries by establishing trust from previously-unknown parties. In conjunction with auxiliary technologies such as SoA, Service Workflow, Specification languages, CPS, IoT, and many others will create significant impact on business process improvement.

A framework of Blockchain-based service selection and composition

In this section, we will illustrate the protocol of how our proposed Blockchain enabled business process can be applied. In this framework, QoS Blockchain is available as services. For the sake of simplicity in this demonstration, the following subsection is scoped within reputation from service consumer feedback. Other QoS parameters can be implemented in the same manner but different transaction types and values shown in "Blockchain transactions" section.

Blockchain transactions

The feedback scheme defines two transaction types in creating and storing QoS values. The first transaction is contract

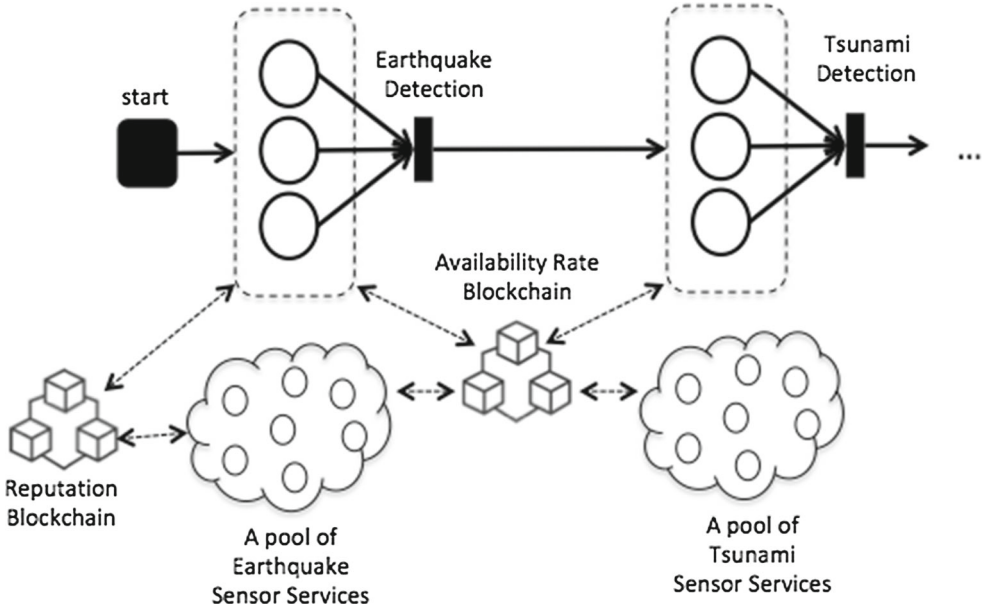

Fig. 2 Example of Blockchain-enabled service selection and composition

TransactionID				Type: Contract	
Transaction Timestamp					
Contract bytecode	Time Release	Service Consumer	Service Provider	Consumer Signature	Secret Hash
0x…	…	0x…	0x…	0x…	sh

TransactionID			Type: Feedback	
Transaction Timestamp				
Service Consumer	Service Provider	Provider Signature	Feedback Value	Secret Hash
0x…	0x…	0x…	5	sh

Fig. 3 Two types of Blockchain transactions

used to initialize contract between service provider and service consumer in giving feedback values. The second transaction is feedback given by a service consumer to a service provider. This transaction is used to store a feedback value from a consumer indexed by Secret Hash (sh). The detail of operations will be explained later in this section. Figure 3. illustrates the format of Blockchain transactions. (Table 1 shows the important notations in these transactions).

Protocol

Assume that each service consumer and provider contain their own public and private key pairs for identification and the proof of identity using digital signature. For the system to be secure, at least SHA-256, used in Bitcoin, or better is utilized for cryptographic hash function, and ECC, RSA, or

Table 1 Notations

Notations	Description
TransactionID	The running number used to indicate transaction
Type	Type of a transaction either contract or feedback.
Transaction timestamp	The time when a transaction is created
Contract bytecode	The code of contract to be stored in Blockchain
Timesrelease	A feedback value can be given after this specified time
Service consumer	The identity associated to a service consumer denoted by the hash of the consumer's public key
Service provider	The identity associated to a service provider denoted by the hash of the provider's public key
Consumer signature	The confirmation of service being invoked with correct timestamp
Provider signature	The credential for the creation of a transactions
Secret hash (sh)	The value used for authentication when a consumer giving feedback
Feedback value (fv)	A feedback value from a service consumer

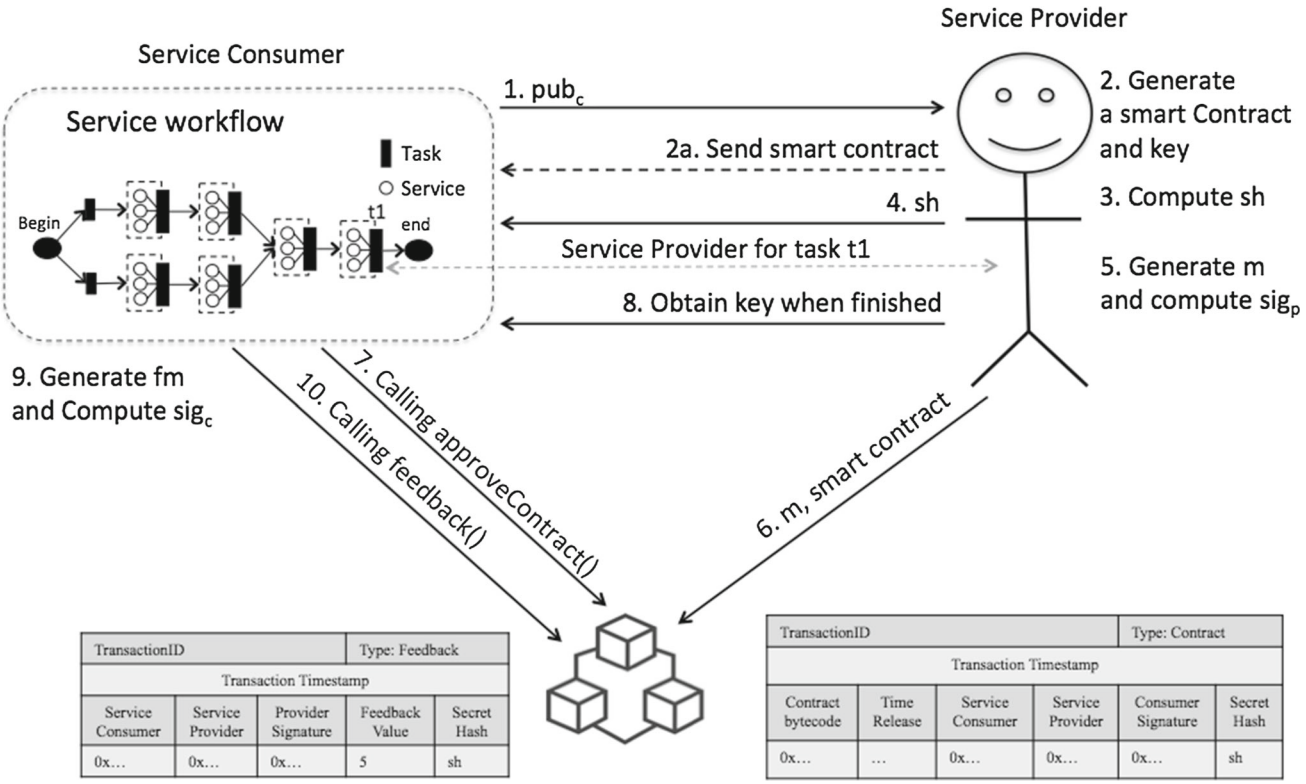

Fig. 4 Usage protocol for transaction creation

other standards by NIST are used for asymmetric cryptography. The following paragraph describes the steps of creating transactions (see Fig. 4). These activities of creating contract occur after service selection is done.

Let (1) pub$_c$ be a public key of a consumer, (2) pub$_p$ be a public key of a provider, (3) sig$_c$ be a digital signature of a consumer, and (4) sig$_p$ be a digital signature a provider. The system relies on smart contract once created will be translated into bytecode and embedded into Blockchain at specific address, which is persistent and immutable. Listing 1 borrows some notations from Ethereum Solidity to demonstrate how QoS feedback can be established in this scheme.

Listing 1. Excerpt of smart contract

```
1:  contract ServiceQoS {
2:      // basic information to verify before getting feedback
3:      struct serviceInfo {
4:          byte32 name;              // just a name of this service
5:          uint timestamp;           // time of contract creating
6:          uint timeRelease;         // feedback can be given after this time
7:          address serviceP;         // service provider public key
8:          address serviceC;         // service consumer public key
9:          byte32 sh;                // secret hash for verifying who giving feedback
10:         bool status;              // status of the contract either "inactive" or "active"
11:     }
12:
13:     serviceInfo public myServiceInfo;
14:
15:     mapping (address => serviceInfo) public allServiceInfo;
16:     mapping (byte32 => sh) public allContract;
17:
18:     event Write(unit _timeRe, address _from, address _to, byte32 _sig, byte32 _sh);
19:     event Write(address _from, address _to, byte32 _sig, uint fv);
20:     // the constructor whose code is run only when the contract is created.
21:     function ServiceQoS(byte32 _name, uint _timeRe,
22:                         address _serviceC, byte32 _sig, byte32 _sh) public {
23:         myServiceInfo.name = _name;
24:         myServiceInfo.timestamp = Time.now;
25:         myServiceInfo.timeRelease = _timeRe;
26:         myServiceInfo.serviceP = msg.sender;
27:         myServiceInfo.serviceC = _serviceC;
28:         myServiceInfo.sh = _sh;
29:         myServiceInfo.status = false; // setting status to be inactive
30:
31:         byte32 h = SHA256 (_timeRe, _serviceC, msg.sender, _sh);
32:         // check if _sig is valid signature
33:         if (!verify(h, _sig)) throw; // assume that function verify() is used to verify signature
34:         // write the contract instance into Blockchain
35:         emit Write(_timeRe, msg.sender, _serviceC, _sig, _sh);
36:     }
37:
38:     function approveContract(byte32 _sh) public return(bool success) {
39:         require(allContract[_sh]));
40:         var thisContract = allContract[_sh];
41:         if(thisContract.serviceC != msg.sender || thisContract.status) throw;
42:         thisContract.status = true;
43:         return true;
44:     }
45:
46:     function feedback(byte32 key, byte32 _serviceP,
47:                       byte32 _sh,  uint _fv,  byte32 _sig) public returns(bool success) {
48:         require(allContract[_sh]));
49:         var thisContract = allContract[_sh];
50:         sh = SHA256 (key, thisContract.timeRelease, msg.sender, _serviceP);
51:         if(sh != _sh || !thisContract.status) throw;
52:         byte32 h = SHA256 (key, serviceP, _sh, _fv);
53:         if (!verify(h, _sig)) throw; // assume that function verify() is used to verify signature
54:         emit Write(msg.sender, _serviceP, _sig, _fv);
55:         kill(thisContract);          //kill contract indexed by sh after feedback is given
56:         return true;
57:     }
58:
59:     function feedback(byte32 _serviceP, byte32 _sh,
60:                       uint _fv, byte32 _sig) public returns(bool success) {
61:         require(allContract[_sh]));
62:         var thisContract = allContract[_sh];
63:         if (thisContract.timeRelease > now || !thisContract.status) throw;
64:         byte32 h = SHA256 (_serviceP, _sh, _fv);
65:         if (!verify(h, _sig)) throw; // assume that function verify() is used to verify signature
66:         emit Write(msg.sender, _serviceP, _sig, _fv);
67:         kill(thisContract);  //kill contract indexed by sh after feedback is given
68:         return true;
69:     }
70: }
```

Step1: After agreeing upon service usage, the service provider obtains pub_c from the service consumer.

Step2: The service provider creates a feedback smart contract and generates a secret key (key) specific to this contract. The purpose of key is to authenticate the service consumer in giving a feedback to this service instance thereafter. This key has not yet been sent to the consumer, and will be released after the end of service execution. For example, sending as the return value of the last instruction of service execution. Optionally, the smart contract is sent to the service consumer for approval.

Step3: The service provider uses key to compute secret hash (sh) where $\text{sh} = \text{SHA}_{256}(\text{key}, \text{TimeRelease}, \text{pub}_c, \text{pub}_p)$. The purpose of TimeRelease is if service execution is unsuccessful, after a specified time indicated by TimeRelease, the service consumer will be able to give feedback without any condition.

Step4: sh is sent to the service consumer as a part of credentials in giving feedback at the end of execution. In security point of view, sh provides two properties. (1) It provides integrity of values $\langle key, TimeRelease, \text{pub}_c, \text{pub}_p \rangle$ as the role of sh is the commitment to the service consumer to verify the correctness of a transaction at a time of giving feedback in **Step9**, and (2) sh is used as a part of credentials to authenticate the service consumer in giving feedback after the execution ends (see **Step10**).

Step5: The service provider computes a digital signature $\text{sig}_p = \text{Sig}(\text{TimeRelease}, \text{pub}_c, \text{pub}_p, \text{sh})$ bundled into message (m) where $\text{m} = \langle name, TimeRelease, \text{pub}_c, \text{sig}_p, \text{sh} \rangle$.

Step6: Then, the provider creates a smart contract transaction and instantiates the contract using information from the message (m) by executing the constructor ServiceQoS() (cf. lines 21-36 in Listing 1). The smart contract constructor will create a contract type transaction and appends into the QoS Blockchain which is disseminates the block to other nodes in the network. This step notifies other nodes that a consumer is invoking the service. Other nodes verify the block correctness by using sig_p.

Step7: The service consumer checks the smart contract in the block and executes function approveContract() (cf. lines 38-44 in Listing 1) if everything looks fine. If approveContract() is not called, the feedback cannot be given. This function changes internal status to be true, which means the contract is active.

Step8: A program logic from the service instance used by the service consumer can be written to reveal key after the execution is done.

Step9: After the execution, the service consumer generates a feedback message (fm) containing $\text{fm} = \langle key, \text{pub}_p, \text{sh}, \text{fv} \rangle$ where fv is a feedback value, and computes a digital signature $\text{sig}_c = \text{Sig}(\text{key}, \text{pub}_p, \text{sh}, \text{fv})$.

Step10: Then the service consumer executes function feedback() (cf. lines 46-57 in Listing 1) to give feedback value to that service. The function searches for the transaction of the contract instance indexed by sh. The contract (1) verifies that the commitment obtained from a transaction indexed by sh in the QoS block is really the same with sh computed by $\text{SHA}_{256}(\text{key}, \text{TimeRelease}, \text{pub}_c, \text{pub}_p)$, (2) checks the contract's status, (3) verifies signature (sig_c), (4) writes the feedback transaction into QoS block, and then (5) kills the contract transaction.

Step10a: If there is an issue that key cannot be obtained, the service consumer optionally executes function feedback() (cf. lines 59-69 in Listing 1) to give a feedback value to that service after a specified time, indicated by TimeRelease. The function searches for the contract transaction indexed by sh. The contract (1) verifies that TimeRelease is less than now, (2) checks the contract's status, verifies signature (sig_c), (3) writes the feedback transaction into QoS block, and then (4) kills the contract transaction. Please note that $\text{sig}_c = \text{Sig}(\text{pub}_p, \text{sh}, \text{fv})$ because key cannot be obtained.

QoS Blockchain

Blockchain contains blocks of several transactions, which are connected using cryptographic hash. Blocks are stored in distributed nodes. The update of transactions contained in a new block requires consensus mechanism for determining the safety and liveness of the block. Nodes in this setting are services that use QoS Blockchain. Unlike Proof-of-Work (PoW) concepts in Bitcoin and Ethereum which experience high delay in storing transactions and chaining a block, the QoS Blockchain needs real-time information that provides the most updated values regarding QoS of services based on feedbacks. In this setting, a new block to be chained to the main Blockchain is done by the execution of the smart contract. The QoS Blockchain has the following characteristics.

1. The QoS Blockchain structure is the same as other Blockchain applications. Blocks are chained together using hash as a pointer to the previous block providing immutability.
2. Each block has the timestamp providing the information about the time of creation. This timestamp much be greater than the timestamps in the transactions contained in the block. A new block is also signed by a creator.
3. The node to be responsible to add a new block is selected pseudorandomly each round, for instance every 10 minutes. To avoid selective transactions when one deliberately avoids including specific transactions in QoS Blockchain, two or more nodes are selected to add a new block. The consensus of a new block is from the voting majority from the nodes in the network. The bigger block, a block with greater number of transactions, will always be voted.

Usage examples and discussions

In dynamic opened environment, services are selected according to participation requirements, often imposed by the workflow owner. In this example, we employ SWSpec language (Viriyasitavat et al. 2012) to specify QoS requirement of service attributes. For example in Fig. 4, one requirement specifies a service attribute for the task t_1 is that the average feedback since the past 3 months is greater than 4, assuming that feedback value ranges from 0 to 5. This requirement is expressed by SWSpec as $\mathcal{P}A_{t1}(\text{average}_{3\text{months}\leq t\leq \text{now}}(\text{feedback}) > 4))$. $\mathcal{P}A_{t1}$ indicates all services selected for the task t1 and the atom inside the brackets specifies the QoS requirement. Suppose a pool of services capable of executing t1 is available, service candidates can be evaluated using information from QoS Blockchain, simply, by querying QoS feedbacks of each service and computing the average from the blocks with the timestamp greater than 3 months. If multiple services satisfy this requirement, ranking technique can be used to select the best alternative (Viriyasitavat 2016).

One security implication in this setting is the collusion to corruptly boost service feedback. This situation occurs when a service provider deliberately selects a service consumer with a malicious agreement to provide high feedback values, with or without actual use of the service. The proposed QoS Blockchain framework is incapable of detecting these transaction frauds. However, permissioned Blockchain can be implemented to alleviate this problem. The Blockchain permits only verified services to be part of the framework and only known service consumers are able to give feedbacks. If collusion is discovered, service providers and consumers will receive penalty such as lowering the feedback values, or just simply unregistering from the QoS Blockchain framework.

Conclusion

The BPM systems in Industry 4.0 are expected to automate service selections and compositions reliably and promptly with transparent interoperations of dynamic organizations. Our thorough review on existing BPM works and tools found that experts' supports or third-party certificates are still essential in selecting and composing services for business projects; this leads to a number of the issues of BPM systems such as (1) the hurdles for the scalability of the growing population of available services, (2) the potential of transition loopholes, (3) the time delays of trustworthiness verifications, and (4) the conflict of openness and security concern. It is our argument that the central issue is to find an effective mechanism to deal with trustworthiness of the organizations in the open business environment, and we propose to look into Blockchain Technology (BCT) in depth since it appears to be an ideal solution to replace experts and third-part authorities. To illustrate the potential application of BCT in BPM systems, we have developed a BCT-based approach where the trustworthiness of businesses are processed as the text in the form of smart contracts. As our plan for continuous study in this field, this framework will be integrated into our previous BPM systems (Viriyasitayat et al. 2018a, b) to automate the evaluation and verification of the trustworthiness in service selection and composition.

References

Ali, A. S., Ludwig, S. A., & Rana, O. F. (2005). A cognitive trust-based approach for web service discovery and selection. In *Third European conference on web services (ECOWS'05)*. https://doi.org/10.1109/ECOWS.2005.2.

Altunay, M., Brown, D., Byrd, G., & Dean, R. (2005). Trust-based secure workflow path construction. In *Proceedings of international conference on service oriented computing*.

Ardagna, D., & Pernici, B. (2005). Global and local QoS guarantee in web service selection. In *Business process management workshops*.

Ardakani, M. R. M., Hashemi, S. M., & Razzazi, M. (2018). A cloud-based solution/reference architecture for establishing collaborative networked organizations. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-017-1387-2.

Barbon, F., Traverso, P., Pistore, M., & Trainotti, M. (2006). Run-time monitoring of instances and classes of web-service compositions. In *Proceedings of ICWS 2006*. https://doi.org/10.1109/ICWS.2006.113.

Camarinha-Matos, L. M., & Pantoja-Lima, C. (2001). Cooperation coordination in virtual enterprises. *Journal of Intelligent Manufacturing*, *12*(2), 133–150.

Chhun, S., Moalla, N., & Ouzrout, Y. (2016). QoS ontology for service selection and reuse. *Journal of Intelligent Manufacturing*, *27*(1), 187–199.

Coutinho, C., Cretan, A., da Silva, C. F., Ghodous, P., & Jardim-Goncalves, R. (2016). Service-based negotiation for advanced collaboration in enterprise networks. *Journal of Intelligent Manufacturing*, *27*(1), 201–216.

Dan, A., Davis, D., Kearney, R., Keller, A., King, R. P., Kuebler, D., et al. (2004). Web services on demand: WSLA-driven automated management. *IBM Systems Journal*, *43*(1), 136–158.

Danila, C., Stegaru, G., Stanescu, A. M., & Serbanescu, C. (2016). Web-service based architecture to support SCM context-awareness and interoperability. *Journal of Intelligent Manufacturing*, *27*(1), 73–82.

Dustdar, S., & Schreiner, W. (2005). A survey on web services composition. *International Journal of Web and Grid Services*, *1*(1), 1–30. https://doi.org/10.1504/IJWGS.2005.007545.

Faizod (2006). Blockchain, industry 4.0 and the internet of things. Faizod. https://faizod.com/blockchain-industrie-4-0-und-das-internet-der-dinge/. Accessed 5 Feb 2018.

Fehling, C., Koetter, F., & Leymann, F. (2014). Compliance modeling-formal descriptors and tools. Report. University of Stuttgart. http://www.iaas.uni-stuttgart.de/institut/ehemalige/fehling/TR-2014-Compliance-Modeling.pdf. Accessed 5 Feb 2018.

Franz, P., & Kirchmer, M. (2012). *Value-driven business process management: The value-switch for lasting competitive advantage*. New York: McGraw Hill Professional.

García-Bañuelos, L., Ponomarev, A., Dumas, M., & Weber, I. (2017). Optimized execution of business processes on blockchain. In J. Carmona, G. Engels, & A. Kumar (Eds.), *Business process management. BPM 2017. Lecture notes in computer science* (Vol. 10445). Cham: Springer.

Gibelin, N., & Makpangou, M. (2005). Efficient and transparent web-services selection. In B. Benatallah, F. Casati, & P. Traverso (Eds.), *ICSOC 2005 (pp. 145–156). LNCS. 3826* (pp. 527–532). Heidelberg: Springer.

Halle, S., Villemaire, R., Cherkaoui, O., & Ghandour, B. (2007). Model checking data-aware workflow properties with CTL-FO+. In *Enterprise distributed object computing conference, 2007. EDOC 2007. 11th IEEE International* (pp. 267–267). Annapolis, MD.

Han, L., Xing, K., Chen, X., & Xiong, F. (2018). A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems. *Journal of Intelligent Manufacturing*, *29*(5), 1083–1096.

Haq, I., Alnemr, R., Paschke, A., Schikuta, E., Boley, H., & Meinel, C. (2010). Distributed trust management for validating SLA choreographies. In P. Wieder, R. Yahyapour, & W. Ziegler (Eds.), *Grids and service-oriented architectures for service level agreements*. Boston, MA: Springer.

Hollingsworth, D. (1994). Workflow management coalition: The workflow reference model. WFMC document WFMCTC00-1003.

Hsieh, F. S., & Lin, J. B. (2014). Context-aware workflow management for virtual enterprises based on coordination of agents. *Journal of Intelligent Manufacturing*, *25*(3), 393–412.

Huang, L., Walker, D. W., Huang, Y., & Rana, O. F. (2005). Dynamic Web Service Selection for Workflow Optimisation. In *Proceedings of 4th UK e-science Programme All Hands Meeting (AHM)*.

Hwang, S.-Y., Lim, E.-P., Lee, C.-H., & Chen, C.-H. (2008). Dynamic web service selection for reliable web service composition. *IEEE Transactions on Service Computing*, *1*(2), 104–116. https://doi.org/10.1109/TSC.2008.2.

IBM (2018), Watson IoT and Blockchain: Disruptor and game changer. https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WW912350USEN. Accessed 5 Feb 2018.

Jurca, R., Faltings, B., & Binder, W. (2007). Reliable QoS monitoring based on client feedback. In *Proceedings of the 16th international conference on World Wide Web (WWW '07)* (pp. 1003–1012). ACM, New York, NY, USA. https://doi.org/10.1145/1242572.1242708.

Kshetri, N. (2017). Can Blockchain strengthen the internet of things? *IT Professional*, *19*(4), 68–72. https://doi.org/10.1109/MITP.2017.3051335.

Lee, M., Yoon, H., Shin, H., & Lee, D. G. (2009). Intelligent dynamic workflow support for a ubiquitous web service-based manufacturing environment. *Journal of Intelligent Manufacturing*, *20*(3), 295–302.

Li, F., Shaung, K., & Su, S. (2007). Q-Peer: A decentralized qos registry architecture for web services. In *Service-oriented computing-ICSOC 2007*. Springer, Berlin, Heidelberg.

Liu, Y., Ngu, A. H., & Zeng, L. Z. (2004). QoS computation and policing in dynamic web service selection. In *Proceedings of the 13th international conference on World Wide Web* (pp. 66–73). ACM Press, New York.

Liu, B., Yu, X. L., Chen, S., Xu, X., & Zhu, L. (2017). Blockchain based data integrity service framework for IoT data. *2017 IEEE International Conference on Web Services (ICWS)*(pp. 468–475). Honolulu, HI. https://doi.org/10.1109/ICWS.2017.54.

Lu, Y. (2017b). Cyber physical system (CPS)-based industry 4.0: A survey. *Journal of Industrial Integration and Management, 2*(3), 1–57.

Lu, Y. (2017a). A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, *6*, 1–10.

Ludwig, H., Dan, A., & Kearney. R., (2004) Cremona: An architecture and library for creation and monitoring of WS-Agreements. In *ICSOC '04. Proceedings of the 2nd international conference on Service oriented computing* (pp. 65–74). ACM Press, New York, NY, USA.

Lu, Y., Wang, H., & Xu, X. (2016). ManuService ontology: A product data model for service-oriented business interactions in a cloud manufacturing environment. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-016-1250-x.

Mahdavi, I., Mohebbi, S., Zandakbari, M., Cho, N., & M ahdavi-Amiri, N. (2009). Agent-based web service for the design of a dynamic coordination mechanism in supply networks. *Journal of Intelligent Manufacturing*, *20*(6), 727–749.

Maximilien, E. M., & Singh, M. P. (2004). A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, *8*(5), 84–93.

Mohammad, A., & Thomas, R. (2009). Combining global optimization with local selection for efficient QoS-aware service composition. In *Proceedings of the 18th international conference on World Wide Web (WWW '09)* (pp. 881–890). ACM, New York, NY, USA. https://doi.org/10.1145/1526709.1526828.

Nakamoto, S., (2008). Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org.

Puttonen, J., Lobov, A., Soto, M. A. C., & Lastra, J. L. M. (2016). Cloud computing as a facilitator for web service composition in factory automation. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-016-1277-z.

Ran, S. (2003). A model for web services discovery with QoS. *SIGecom Exchanges*, *4*(1), 1–10. https://doi.org/10.1145/844357.844360.

Rao, J., & Su, X., (2005). A survey of automated web service composition methods. In *Semantic web services and web process composition* (pp. 43–54). Springer, Berlin.

Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A. P. A., & Casati, F. (2002). Automated SLA monitoring for web services. In *DSOM. Lecture Notes in Computer Science, 2506* (pp. 28-41). Springer, Berlin.

Seok, H., & Nof, S. Y. (2018). Intelligent information sharing among manufacturers in supply networks: Supplier selection case. *Journal of Intelligent Manufacturing*, *29*(5), 1097–1113.

Serhani, M. A., Dssouli, R., & Hafid, A., et al. (2005). A QoS broker based architecture for efficient web services selection. In: *ICWS*

2005. Proceedings of IEEE international conference on web services (pp. 113–120). IEEE Computer Society Press, Los Alamitos.

van der Aalst, W. M. P. (2005). Pi calculus versus petri nets: Let us eat "humble pie" rather than further inflate the "pi hype". *BPTrends*, *3*(5), 1–11.

Viriyasitavat, W. (2009). Modeling delegation in requirements-driven trust framework. In *2009 Congress on services-I* (pp. 522–529). Los Angeles, CA.

Viriyasitavat, W. (2013). A framework of trust in service workflows. Ph.D. dissertation. Department of Computer Science, University of Oxford, Oxford, UK.

Viriyasitavat, W., & Martin, A. (2010). Formal trust specification in service workflows. In *IEEE/IFIP 8th international conference on embedded and ubiquitous computing (EUC)* (pp. 703–710).

Viriyasitavat, W., & Martin, A. (2012). An improvement of requirement-based compliance checking algorithm in service workflows. In *Proceedings of international conference on advances in information technology (AIT)* (pp. 41–45). UACEE, Bangkok, Thailand.

Viriyasitavat, W. (2016). Multi-criteria selection for services selection in service workflow. *Journal of Industrial Information Integration*, *1*, 20–25.

Viriyasitavat, W., & Martin, A. (2011b). In the relation of workflow and trust characteristics, and requirements in service workflows. *Communications in Computer Information Science*, *251*, 492–506.

Viriyasitavat, W., & Martin, A. (2011a). Formalizing trust requirements and specification in service workflow environment. In R. Zhang, J. Cordeiro, X. Li, Z. Zhang, & J. Zhang (Eds.), *ICEIS* (Vol. 3, pp. 196–206). Setúbal: SciTePress.

Viriyasitavat, W., & Martin, A. (2017). The reviews and analysis of the state-of-the-art service workflow specification languages. *Journal of Industrial Information Integration.*, *8*, 1–7.

Viriyasitavat, W., Xu, L., & Martin, A. (2012). SWSpec: The requirement specification language in service workflow environments. *IEEE Transactions on Industrial Informatics*, *8*(3), 631–638.

Viriyasitavat, W., Xu, L. D., & Viriyasitavat, W. (2014a). Compliance checking for requirement-oriented service workflow interoperations. *IEEE Transactions on Industrial Informatics*, *10*(2), 1469–1477.

Viriyasitavat, W., Xu, L. D., & Viriyasitavat, W. (2014b). A new approach for compliance checking in service workflows. *IEEE Transactions on Industrial Informatics*, *10*(2), 1452–1460.

Viriyasitayat, W., Xu, L. D., & Bi, Z. M. (2018a). The extension of semantic formalization of service workflow specification language. *IEEE Transactions on Industrial Informatics*. https://doi.org/10.1109/TII.2018.2807400.

Viriyasitayat, W., Xu, L. D., Bi, Z. M., & Sapsomboon, A. (2018b). Extension of specification language for soundness and completeness of service workflow. *Enterprise Information System*. https://doi.org/10.1080/17517575.2018.1432769.

Watanabe, K., Mikoshiba, S., Tateyama, T., & Shimomura, Y. (2012). Service process simulation for integrated service evaluation. *Journal of Intelligent Manufacturing*, *23*(4), 1379–1388.

Workflow management coalition (1999). Workflow management coalition specification: Terminology & glossary. WFMC document WFMC-TC-1011.

Xie, Y., Chen, S., Ni, Q., & Wu, H. (2017). Integration of resource allocation and task assignment for optimizing the cost and maximum throughput of business processes. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-017-1329-z.

Xu, L. D., & Duan, L. (2018). Big data for cyber physical systems in industry 4.0: A survey. *Enterprise Information Systems*. https://doi.org/10.1080/17517575.2018.1442934.

Xu, L. D., Xu, E. L., & Li, L. (2017). Industry 4.0: State of the art and future trends. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2018.1444806.

Xu, L. D., Viriyasitavat, W., Ruchikachorn, P., & Martin, A. (2012). Using propositional logic for requirements verification of service workflow. *IEEE Transactions on Industrial Informatics.*, *8*(3), 639–646.

Yu, T., & Lin, K. J. (2005). A broker-based framework for QoS-aware web service composition. In *EEE-2005. Proceeding of IEEE international conference on e-Technology, e-Commerce and e-Service, Hong Kong, China*. IEEE Computer Society Press, Los Alamitos.

Zeng, L., Benatallah, B., Lei, H., Ngu, A., Flaxer, D., & Chang, H. (2010). Flexible composition of enterprise web services. *Electronic Markets*, *12*(2), 141–152.

Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., & Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, *30*(5), 311–327.

Zhang, Y., Zhang, G., Liu, Y., & Hu, D. (2017). Research on services encapsulation and virtualization access model of machine for cloud manufacturing. *Journal of Intelligent Manufacturing*, *28*(5), 1109–1123.