# Monitoring Interactions across Multi Business Processes with Token Carried Data

Chuanyi Li, Jidong Ge, Zhongjin Li, Liguo Huang, Hongji Yang, and Bin Luo

**Abstract**—The rapid development of web service provides many opportunities for companies to migrate their business processes to the Internet for wider accessibility and higher collaboration efficiency. However, the open, dynamic and ever-changing Internet also brings challenges in protecting these business processes. There are certain process monitoring methods and the recently proposed ones are based on state changes of process artifacts or places, however, they do not mention defending process interactions from outer tampering, where events could not be detected by process systems, or saving fault-handling time. In this paper, we propose a novel Token-based Interaction Monitoring framework based on token carried data to safeguard process collaboration and reduce problem solving time. Token is a more common data entity in processes than process artifacts and they cover all tasks' executions. Comparing to detecting places' state change, we set security checking points at both when tokens are just produced and to be consumed. This will ensure that even if data is tampered after being created it would be detected before being used. For applying monitoring framework, we develop a collaboration constructing method with token-based process mining techniques to derive global interaction processes as well as organize historical process data in forms of token.

**Index Terms**—web services, collaboration construct, process reengineering methodology, business process monitoring

## 1 Introduction

With the rapid development of various Internet applications, a great deal of daily life services and management facilities are available on and conducted through the Internet (i.e., web services). The growth of online shopping services is a compelling case [1]. Many small companies are moving their locally deployed business processes to the Internet (see Fig. 1) for better supporting the collaboration with other enterprises or among their departments to achieve more business successes. However, the security of business data (e.g., data submitted by users, actual executor of the task, the amount of money for buying material, etc.) faces more risks, since all data are exposed to the open, dynamic, and ever-changing Internet environment [2]. In this paper, business process data security refers to that no risks to the successful execution of the process would be caused by any problems of tasks' input and output. Insecure process data may lead the process instance to an unexpected routine, delay the completeness of the instance, or even result in a process failure. The proposed monitoring framework will alert warning messages while any abnormal process data occur.

In environments with processes to be mainly executed manually, such as health care, financial management, and collaborations among departments, process data are stored in the database or other data management systems after generated and will not be retrieved until the next task starts. This increases the possibility that business data are changed directly in the database by outer tampering through Internet. However, business information systems will fail to detect this kind of data change. Although there are strategies for protecting data in database management systems, it is difficult to judge whether a data change is legal or not without understanding business requirements. As more and more cross-organizational business processes are deployed over the Internet, there is an emergent demand to protect them from this kind of risk at runtime [3]. So, in this paper, we propose a novel process monitoring strategy by setting process data checking points before using the retrieved data as input for the next task. We name it pre-checking. For example, in an Order Management system, there is a task 'Calculate Outlay' (CO) interacting with the task 'Make Payment' (MP) in the Financial Management system and informs it the amount of money it should give to the participant who is in charge of preparing material for the custom's order. Even if the amount of money is tampered after submitted by task CO, it will be detected by pre-checking before executing MP. This will save time and effort from mitigating the effects of taking the tampered data as input. Besides pre-checking on input of tasks, we also propose post-checking on outputs of tasks to prevent the propagation of their illegal behavior. Post-checking is performed after executions of tasks and it will detect risks made by inner process executions as early as possible. By post-checking, whether the just-executed task should be rolled back or not would be known upon the task is executed.

In order to provide legal data standard for pre/post checking on input/output of tasks, process discovery technique is adopted. While small companies move their


- *C. Li, J. Ge, Z. Li and B. Luo are with the State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, China. E-mail: {lcynju@126.com, gjd@nju.edu.cn, lzjnju@126.com, luobin@nju.edu.cn}*
- *L. Huang is with the Department of Computer Science and Engineering, Southern Methodist University, USA. E-mail: lghuang@lyle.smu.edu*
- *H. Yang is with the Centre for Creative Computing, Bath Spa University, Bath, England, UK. E-mail: h.yang@bathspa.ac.uk*

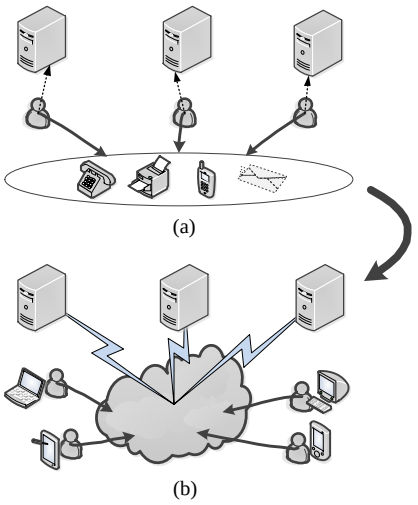



Fig. 1. From traditional process interaction to Internet-based process interaction

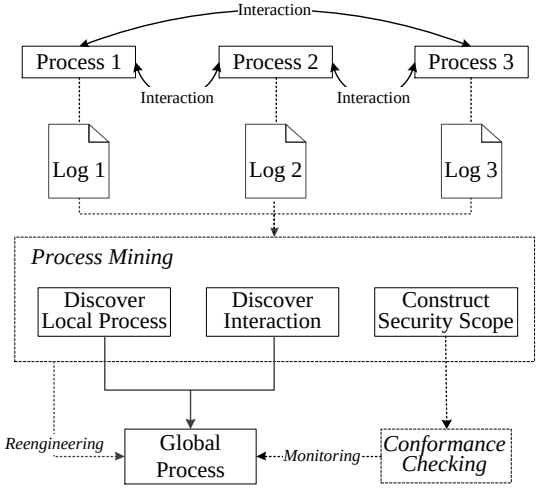



Fig. 2. Framework proposed in this paper for process reengineering and monitoring

locally deployed business processes to the Internet, they need to construct a global collaborating process for making interaction more convenient, as well as applying monitoring framework easier. Process discovery method can integrate a global collaborating model from separated sub-processes among different enterprises or departments, and prepare checking standards for monitoring. Inputs of process discovery are system logs of constituent processes and messages exchanged across them. We adopt the Interorganizational Workflow (IOWF) [4] to model the global interaction, and employ token-log-based process mining algorithm (i.e., $\tau$ [5]) to discover it.

The main contributions of this paper are:

- Constructing collaborating model for small companies with token-log-based mining algorithm $\tau$. Tokens can be viewed as small business artifacts that exist throughout the life cycle of process running instance, and are transferred between connected tasks.
- Developing a novel process interaction monitoring strategy based on pre/post checking on input/output of tasks with token carried data. Pre-checking is to check if data carried by to-be-consumed token is legal. Post-checking is to check if data carried by just produced token is legal. Different from monitoring ideas proposed in [6] and [7], we also pay attention to outer tampering through Internet. Besides, token is a more general data structure in business process than process artifact. Compared to detecting places' state change proposed in [7], our approach details the place state change with token produced and consumed stages. We set the checking points at both when a token is just leaving and being put into a place, which ensures that a possible execution fault or an outer tampering can be detected as early as possible.

Fig. 2 shows the proposed monitoring framework with mining global interaction model (*reengineering*) and checking token carried data using the token-log-based conformance checking theory [8] (*monitoring*).

The rest of this paper is organized as follows. In Section 2, a running example is introduced to illustrate the monitoring framework. Related work including processes collaboration, mining and monitoring are introduced in Section 3. Modeling tool, IOWF, and other related data structures (e.g., Interaction Place, Token and Interaction Token) are defined in Section 4. In Section 5, we illustrate our approach with sub-sections of Generating Token Log, Constructing Algorithm, and Monitoring Strategy. A comprehensive case study based on the running example of Section 2 is analyzed in Section 6. Related results of evaluation experiments conducted on the proposed frameworks are presented in Section 7. Section 8 concludes the paper.

## 2 Running Example

In order to illustrate the monitoring framework clearly, we first present the approach with an example based on a clothing factory in Nantong, Jiangsu Province, China. It is a small business company that has no global business process management system before we build one for them. The initial requirement is to build a system to monitor interactions between different departments, e.g., order management and financial. The inputs are historical interacting messages, such as emails, paper files and short messages, among different departments. We first build collaborating process management system (i.e., Flexible Manufacturing Chain, FMC) using process mining technique, and construct data checking criteria from historical data at the same time. Then the proposed monitoring strategy can be applied. Following is the running example for illustrating pre-checking and post-checking. In FMC, 'Orderprepare' is a task in the order management department, and it interacts with task 'Fileprepare' in the financing department.

- Post-checking: After 'Orderprepare' is executed, *related data fields* that should be checked will be sent to *the connecting point* between 'Orderprepare' and 'Fileprepare'. Upon the connecting point receiving the data, the post-checking, i.e. *check* data fields based

on *historical ones*, is triggered. If there are problems with these data fields, warning messages will be alerted and the process will be *suspended* with the execution of task 'Orderprepare' marked as failed. Otherwise, result business data generated by 'Orderprepare' will be written into database or other data persistence media, and 'Fileprepare' is ready to be fired.

- Pre-checking: Upon 'Fileprepare' is fired, pre-checking is triggered. 'Fileprepare' required input business data would be retrieved from data persistence media and they would be *checked* according to *historical ones* stored in the connecting point between 'Orderprepare' and 'Fileprepare'. If there were problems with the retrieved data, 'Fileprepare' would not be allowed to start and reset to unfired state directly. Otherwise, it can be executed and post-checking will be triggered after its execution.

Note that we mark certain important terms in italics style in the description. *Connecting point* and *certain resources* mean *place* and *tokens* in business process modeling language respectively. In Section 4.1, we will introduce them in detail. Questions related to others will also be answered in following sections. For example, how *related data fields* are sent, how to construct *historical data*, details of *suspending* a task, and how data integrity is actually *checked*, etc.

## 3 RELATED WORK

### 3.1 Business Process Collaboration

There are two kinds of collaboration among business processes. One is among cross-organizational business process that are established between organizations and their business partners to improve their performance and competitiveness [9]. The other is among smaller, autonomous, and interconnected sub-systems that compose a large business process inner an organization [10]. Both kinds of process collaboration can be defined through collaborative business processes (CBPs) [9] as well as Interorganizational Workflow. A methodology for modeling and evolving cross-organizational business processes based on business protocols is proposed in [11]. A data-aware interaction model based on Data-Aware Interaction Nets that uses elements of both Interaction Petri Nets [12] and Workflow Nets with Data [13] is proposed in [14]. How to identify services in the design phrase in building interorganizational process is concerned in [15]. In this paper, we focus on the collaboration among sub-processes inner an enterprise, furthermore propose a framework for constructing a global collaboration model. The process for constructing the collaboration model is like service mashup [16], [17], [18] to some extent.

### 3.2 Process Mining

Process mining consists of three main research directions, i.e., process discovery, conformance checking and process enhancement. Traditionally, process discovery is to mine a process model from the event data on the viewpoint of control flow [19], [20]. There are many process discovery algorithms, such as $\alpha$ [21], $\alpha^+$ [22], $\alpha^{++}$ [23], $\alpha^{\#}$ [24], $\beta$ [25], $\lambda$ [26], $\tau$, region-based [27] and genetic ones [28]. Recently, more and more works related to data perspective have been done and the mined achievements are applied to process data management successfully, e.g., processing queries over workflow executions [29]. Conformance checking, also referred to as conformance analysis, aims at the detection of inconsistencies between a process model and its corresponding execution log [8],[30],[31]. The idea of process enhancement is to extend or improve an existing process model using execution information of the actual process recorded in event logs [19]. Artifact [32] is another research topic emerging in process management that analyzes the business process from the viewpoint of data objects. There are both process discovery and conformance checking researches based on artifacts [33],[34],[35],[36]. However, the token used in this paper is a special artifact that exists between any two interacting tasks and all events or data changes will be recorded by logging token information [5]. Before the global collaborating process model is built, interactions between local processes are implemented through messages. These messages are tagged with sender, receiver and corresponding dates. Tokens defined in token-log-based mining algorithm also consist of fields of producer, consumer and corresponding time. It is appropriate and easy to transform messages into tokens. Then, token-log-based mining algorithm can be applied directly. Besides, the historical business data carried by interacting messages, which will be used as checking criteria in monitoring, can be gathered easily by gathering tokens. At last, the pre-checking and post-checking ideas in the proposed monitoring strategy are inspired by token-log-based conformance checking method, where tokens in the process model are compared with those in the token log. So, in this paper, ideas of both process discovery and conformance checking employed for contribution are token-log-based.

### 3.3 Process Monitoring

Business process monitoring provides well-established means for tracking the history and state of business processes and evaluating their performance [10]. An algorithm for runtime monitoring of message-based workflow was proposed in [37]. Many other process monitoring frameworks are proposed from different perspectives of process models and based on different computing techniques. Some are monitoring the compliance between running behaviors and the designed ones. The running behaviors are represented by system logs while the designed ones are represented by process compliance [38],[39] or business constrains [10]. Others are based on the analysis of history data and they are named with data-driven process monitoring and fault detecting techniques [40],[41], where basic data-driven methods, principle component analysis (PCA) [42], partial least squares (PLS) [43],[44], fisher discriminant analysis (FDA) [45], and independent component analysis (ICA) [46] are used. These techniques are more like fault

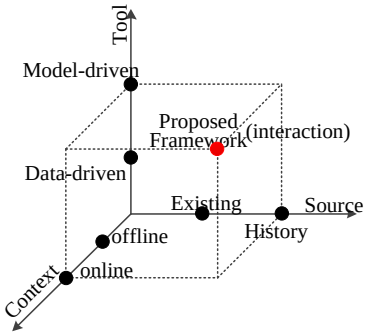



Fig. 3. Classification of process monitoring methods and the proposed framework is a history-data-based online model-driven one.

detecting or process auditing [47],[48] than monitoring since lacking support of online or runtime monitoring. In [10], a monitoring mechanism by determining the state of business processes is proposed, and the idea of using process mining techniques to monitor and guide online running cases is relatively new. The solution towards how to detect actual events in the process environment is presented in [49]. Related to process monitoring in Internet environment where cyber security is concerned [50], a Web service monitoring method is proposed in [51]. Web service monitoring techniques can be found in [52],[53],[54]. In [55], a framework for monitoring processes and interactions among them based on process states changing by detecting global business events is proposed. There are also proposals of monitoring processes with state transition event of data, where data object refers to process artifact [6]. To conclude, classes of existing process monitoring works are shown in Fig. 3. The proposed one is a history-data-based, model-driven, and online multi-process interaction monitoring strategy, which is marked with a red dot in Fig. 3. It checks process interacting data through token transformed events. The highlights are consideration on outer data tampering operations through Internet and saving time in finding process data risks. Besides, different from implementing checking work for each task in different action handling functions, where it is difficult to find what kind of actions are used for different tasks, the proposed checking work is triggered in handling functions for state changing of places. That means the proposed framework is easier to implement since the state changing of places would be caused by execution of any task and there are only two kinds of state changing events, i.e., token added and token removed.

## 4 Interaction Modeling

### 4.1 Modeling Language

Process reengineering is to mine a global interaction process model. First, we define the Interorganizational Workflow (IOWF) [4] for modeling the interactions. IOWF is based on Workflow net [56] which is a sub-class of Petri nets [57]. Petri net $PN$ can be denoted by a 3-tuple, $PN=(P,T,F)$. $P$ is the set of *places* in the net. $T$ is the set of *transitions* and $P \cap T=\emptyset$ and $P \cup T \neq \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is

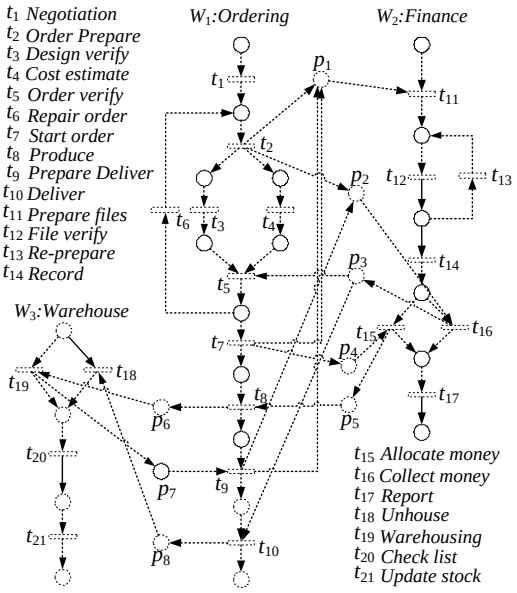



Fig. 4. An example interaction model of a clothing factory

the set of *flow relations* between places and transitions. A WF-net requires the Petri net to have 1) a single *Start* place, 2) a single *End* place, and 3) each node must be on one path from *Start* to *End*. *Tokens* are transferred between transitions and places for determining whether a transition is ready to fire or not.

An IOWF is essentially one 'global' workflow process which consists of several 'local' workflow processes and an interaction structure [4]. There are two kinds of communications for interacting: asynchronous and synchronous. Asynchronous communication corresponds to the exchange of messages between local processes. Synchronous communication forces local processes to execute specific tasks at the same time. Synchronous communication is also implemented by exchanging messages between synchronized tasks, where these messages are synchronous messages, e.g., information exchanged through a phone call [4]. That means a synchronous communication is kind of a combination of asynchronous communications between sub-tasks of synchronized tasks. In this paper, we do not distinguish synchronous communications from asynchronous ones, and only asynchronous communications are considered. An IOWF is defined as a tuple $IOWF=(PN_1, PN_2 \ldots, PN_n, PAC, AC)$, where:

1. $n \in N^+$ is the number of local workflow nets,
2. $\forall\, k \in \{1,\ldots, n\}$: $PN_k=(P_k,T_k,F_k)$ is a WF-net with source place $i_k$ and sink place $o_k$,
3. $\forall k, l \in \{1,\ldots, n\}$, if $k \neq l$ then $(P_k \cup T_k) \cap (P_l \cup T_l)=\emptyset$,
4. $T^{\square}= \cup_{k\in\{1,\ldots,n\}} T_k$, $P^{\square}= \cup_{k\in\{1,\ldots,n\}} P_k$, $F^{\square}= \cup_{k\in\{1,\ldots,n\}} F_k$,
5. $PAC$ is the set of asynchronous communication places,
6. $AC \subseteq PAC \times \mathbb{P}(T^{\square}) \times \mathbb{P}(T^{\square})$ is the asynchronous communication relation ($\mathbb{P}(T^{\square})$ is the set of all non-empty subsets of $T^{\square}$), and
7. $\forall p \in PAC$, $\{(p',x,y) \in AC \mid p'=p\}$ is a singleton.

For example, the IOWF in Fig. 4 is the model of FMC system mentioned in Running Example Section and it is

TABLE 1
AN EXAMPLE THE TOKEN LOGS OF $W_1$ AND $W_2$ IN THE MODEL IN FIG. 4

| C ID | 1 | | | | | | | | | | | 1 | | | | | | | | 2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Process | $W_1$ | | | | | | | | | | | $W_2$ | | | | | | | | $W_2$ | | | | | | | |
| PT | *null* | $t_1$ | $t_2$ | $t_2$ | … | $t_{16}$ | $t_5$ | $t_7$ | $t_{15}$ | … | $t_{10}$ | *null* | $t_2$ | $t_{11}$ | $t_{12}$ | $t_{14}$ | $t_2$ | $t_{16}$ | $t_{17}$ | *null* | $t_7$ | $t_{11}$ | $t_{12}$ | $t_{14}$ | $t_7$ | $t_{15}$ | $t_{17}$ |
| CT | $t_1$ | $t_2$ | $t_3$ | $t_4$ | … | $t_5$ | $t_7$ | $t_8$ | $t_8$ | … | *null* | $t_{11}$ | $t_{11}$ | $t_{12}$ | $t_{14}$ | $t_{16}$ | $t_{16}$ | $t_{17}$ | *null* | $t_{11}$ | $t_{11}$ | $t_{12}$ | $t_{14}$ | $t_{15}$ | $t_{15}$ | $t_{17}$ | *null* |
| PEID | *null* | 1 | 2 | 2 | … | 4 | 5 | 6 | 4 | … | 9 | *null* | 2 | 1 | 2 | 3 | 2 | 4 | 5 | *null* | 6 | 1 | 2 | 3 | 6 | 4 | 5 |
| CEID | 1 | 2 | 3 | 4 | … | 5 | 6 | 7 | 7 | … | *null* | 1 | 1 | 2 | 3 | 4 | 4 | 5 | *null* | 1 | 1 | 2 | 3 | 4 | 4 | *5* | *null* |
| PTCID | \ | \ | \ | \ | … | 1 | \ | \ | 2 | … | \ | \ | 1 | \ | \ | \ | 1 | \ | \ | \ | 1 | \ | \ | \ | 1 | \ | \ |

denoted by $w$=($W_1$,$W_2$,$W_3$,$PAC$,$AC$) where $PAC$={$p_1$,$p_2$,$p_3$,$p_4$, $p_5$,$p_6$,$p_7$,$p_8$} and $AC$={($p_1$,$t_2$,$t_{11}$),($p_1$,$t_7$,$t_{11}$),($p_1$,$t_9$,$t_{11}$),($p_2$,$t_2$,$t_{16}$), ($p_2$,$t_9$,$t_{16}$),($p_3$,$t_{16}$,$t_5$),($p_3$,$t_{16}$,$t_{10}$),($p_4$,$t_7$,$t_{15}$),($p_5$,$t_{15}$,$t_8$),($p_6$,$t_8$,$t_{19}$), ($p_7$,$t_{19}$,$t_9$),($p_8$,$t_{10}$,$t_{18}$)}.

The interaction between local processes is represented by places in *PAC*, but the definition of places in PAC is too simple to be used in the monitoring framework. We make an extension to the definition of place in PAC and name it *Interaction Place*, *IP*. The key element of an IP is *Safe Condition* (*SC*) of interaction variables. SC of one *Interaction Variable* (*IV*) is the value scope of the variable within which the interaction is viewed as a safe one from the viewpoint of this variable. We note value scope of an IV with a set $S$ and the safe condition of it as $sc$=($IV$, $S$). The set for safe condition of all IVs is denoted as $SC$=($IVs$, $Ss$). Hence, an interaction place *IP* of a place $p \in PAC$ in the IOWF $w$=($PN_1$,$PN_2$…,$PN_n$,$PAC$,$AC$) is a tuple $IP$=($PrTS$,$PoTS$,$PrSC$,$PoSC$), where:

1. $PrTS$ is the set of input tasks of $p$: $PrTS$={$x$ | ($p$,$x$,$y$) $\in AC$}.
2. $PoTS$ is the set of output tasks of $p$: $PoTS$={$y$ | ($p$,$x$,$y$) $\in AC$}.
3. $PrSC$=($PrIVs$,$PrSs$) where $PrIVs$={$IV_1$,$IV_2$,…,$IV_m$} and $PrSs$={$S_1$,$S_2$,…,$S_m$}: $\forall\ k \in \{1,...,m\}$, $IV_k$ is a related variable of tasks in $PrTS$ and $S_k$ is the safety scope of $IV_k$.
4. $PoSC$=($PoIVs$, $PoSs$) where $PoIVs$={$IV_1$,$IV_2$,…,$IV_m$} and $PoSs$={$S_1$,$S_2$,…,$S_m$}: $\forall k \in \{1,...,m\}$, $IV_k$ is related variable of tasks in $PoTS$ and $S_k$ is the safety scope of $IV_k$.

For example, if the interaction between $t_2$ and $t_{11}$ in Fig. 4 has two kinds of variables--*sender* and *handler*, and the *sender* can only be one in {*Adam*, *Colin*, *Evan*}, while *handler* can only be one in {*Brian*, *George*}, then the IP $p_1$ is denoted as $p_1$=({$t_2$}, {$t_{11}$}, ({*sender*}, {{*Adam*, *Colin*, *Evan*}}), ({*handler*}, {{*Brian*, *George*}})).

### 4.2 Token Carried Historical Data

Considering that there are only independent supporting systems for local business processes, we need to discover a global one to apply the monitoring framework. The discovering work can be done by process mining. First, we need to construct structured system logs, which can be used as inputs of process mining algorithms directly, from those unstructured interaction records. This will be discussed in Section 5.1 in detail. In this section, we define structured system logs for process mining algorithm.

As mentioned in Section 3.2, the token-log-based mining algorithm $\tau$ will be used in this paper. A token is denoted by a 5-tuple $token$=($CID$,$PT$,$CT$,$PEID$,$CEID$) where:

1. *CID* is the identity of the instance creating *token*. CID is only unique within a local business process, which means two instances from different processes may have a same CID.
2. *PT* is the producer of *token*, and *CT* is the consumer.
3. *PEID* is the *EID* of *PT* when producing *token* and *CEID* is the *EID* of *CT* when consuming *token*.
4. All fields of *token* can be retrieved through '.' function, e.g., CID = *token.CID*

*EID* represents *Execution Identification* [5] that is assigned to each execution of a task by the process management system to identify each single execution of tasks. EID works as a unique timestamp in each process instance. Since different local processes may have tasks of same name in reality, we assume tasks are renamed with the combinations of its process name and its real name in advance. This ensures names of tasks are unique globally. Tokens representing the interacting messages are called *Interacting Token* (*IT*). An IT would be recorded in the log of local process that receives the message, and the PT of this IT is not in the task set of the local process. While generating an IT, the CID of the instance in which its PT is executed is attached to it. This will help to put interacting instances of different local processes together. Besides, the system log consisting of tokens is called *Token Log* (*TL*).

The basic concept of process interaction monitoring is to check if the events happened in temporary interaction are having high compliance with those happened in the past. In other words, history interaction data will be viewed as constraints on interactions, and temporary interaction data will be checked if they are within these constraints. In the previous subsection, Safe Condition (SC) has been defined to represent the constraints of interaction variables. Here, temporary value of interaction variable is defined as *Safe Value* (*SV*) (i.e., variable-value pair) and denoted as $SV$=($IV$, $V$), where $IV$ represents the interaction variable and $V$ represents the value. Eventually, Interaction Token is defined as $itoken$=($token$, $PTCID$, $PSVs$, $CSVs$), where:

1. $token$={$CID$,$PT$,$CT$,$PEID$,$CEID$} is the normal token representing the interaction.
2. $PTCID$ is the CID of the instance where $PT$ of $token$ is executed.
3. $PSVs$={$SV_1$,$SV_2$,…,$SV_n$}: $\forall k \in \{1,...,n\}$, $SV_k$=($IV_k$, $V_k$) where $IV_k$ is $PT$ related variable.
4. $CSVs$={$SV_1$,$SV_2$,…,$SV_m$}: $\forall l \in \{1,...,m\}$, $SV_l$=($IV_l$, $V_l$) where $IV_l$ is $CT$ related variable.

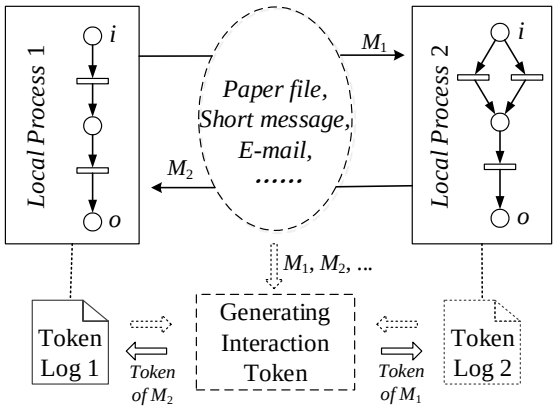



Fig. 5. Process of generating interaction tokens from traditional interaction messages

5. All fields in *token* can be reached by *itoken* through '.' too, e.g., *itoken.CID* = *token.CID*.

For example, Table 1 shows a part of the token logs of local process $W_1$ and $W_2$ in the model shown in Fig. 4 and the tokens are from same interacting instances. Tokens with *null* PT and PEID are initialized tokens putting in the input place of the process. Tokens with null CT and CEID are output tokens remaining in the output place of the process. Taking $t_2$ in $W_1$ and $t_{11}$ in $W_2$ in Fig. 4 as examples, $t_2$ has a related variable named Start-time and $t_{11}$ has a related variable name End-time. Assuming that in the interaction represented by *token*=(1,$t_2$,$t_{11}$,2,1) in Table 1, $t_2$ started at 9:00 am and $t_{11}$ ended at 4:00 pm, then the IT of this interaction is *itoken*=(*token*, 1, {(Start-time,9:00 am)}, {(End-time, 4:00 pm)}).

## 5 Token-based Interaction Monitoring

In this section, details of applying the proposed token-based interaction monitoring framework in those small light-industry enterprises are illustrated. First, historical logs of local processes and messages of interactions among local processes are transformed into token-based system log, i.e., token log. Then, constructing algorithm for constructing a global interacting process as well as token-based data checking criteria from token logs is introduced. At last, how to monitor runtime token-carried-data of the global interaction process according to different data checking criteria is described.

### 5.1 Generating Token Log

Currently, in those small light-industry enterprises, the interactions among local processes are delivered through traditional message sending methods, such as phones, paper files, e-mails or MSNs. In order to apply the token-based monitoring framework, we need to construct a global interaction process from these messages. As analyzed in Section 3.2, we will employ token-log-based process discovery algorithm here. So, we have to generate usable token logs from the traditional message records, i.e., extract useful information from multiple sources [58]. Since each department already has standard business process supporting system, the token logs of the local processes already exist. The problem is to derive token log representing interactions. Fig. 5 shows the process for generating interaction tokens. There are many kinds of message records available for generating interaction tokens. No matter what types of the message records are, the retrieving sequences of Interaction Token data fields (i.e., CID, PT, CT, PEID, CEID, PTCID, PSV, and CSV) are the same:

1. Find CID, PT, CT and PTCID: PT and CT can be found directly from the message or indirectly with people, equipment or other items. CID is the instance id of CT and PTCID is the instance id of PT. They can be found through message date-time or a same reference used by two local processes, such as Order Number in FMC.
2. Find CEID: Search the pair of (CID, CT) in TL of process containing CT and there must be other tokens consumed together with this one by CT and they share a same CEID.
3. Find PEID: Search the pair of (CID, PT) in the TL of the process containing PT for getting PEID.
4. Generate PSV for PT and CSV for CT.

The process of generating interaction tokens from messages is not the key point of interorganizational business process reengineering. No more details of how to generate interaction tokens from different interaction messages will be presented. Here, we show detailed steps of retrieving interaction tokens without considering message source. The input is the interaction message record (*IMR*) which contains all messages (*mgs*) delivered among processes for interacting. The destination is to turn each *mg* in *IMR* into an interaction token *IT*. First, as said in Section 5.1, CID of the running instance that receive *mg* and PTCID of the running instance that send *mg* can be found in *mg* itself. The task sets *PPT* and *PCT* represent the set of potential producers and consumers of *IT* respectively. Referring to the example illustrated in Section 4.1, *PPT* and *PCT* can be directly derived from *mg* or indirectly with related information, such as person or resource. Then *PT* is the single item in $PPT \cap T_i$ and *CT* is the single item in $PCT \cap T_i$ where $i \in \{1,...,n\}$. If *PT* is a task in $W_i$, then $PEID=f_{peid}(token)$ where $token \in TL_i \wedge f_{cid}(token) == CID \wedge f_{pt}(token) == PT$. If *CT* is a task in $W_i$, then $CEID= f_{ceid}(token)$ where $token \in TL_i \wedge f_{cid}(token) == CID \wedge f_{ct}(token) == CT$. At last, generate *PSV* and *CSV* of *PT* and *CT* according to the context of *PT* and *CT* respectively in *mg*. So the interaction token *itoken* of the message *mg* is *itoken*= ((*CID*,*PT*,*CT*,*PEID*,*CEID*),PTCID,*PSV*,*CSV*). Upon tokens of all messages are generated, they can be put together to build an interaction token log $TL_{int}$.

In the Case Study section, a real-life example for illustrating how to retrieve interaction tokens from e-mail records in FMC processes will be presented.

### 5.2 Constructing Algorithm

In order to ensure the accuracy of local processes used in collaboration constructing, we apply mining algorithm to their token logs ($TL_1$,$TL_2$,…,$TL_n$) to derive exactly executing process models ($W_1$,$W_2$,…,$W_n$). At the same time, interaction structures represented by interaction places (*IPs*) are derived from interaction token log ($TL_{int}$).

Note that tasks are unique globally as illustrated in Section 4.2, i.e., $W_1 \cap W_2 \cap \ldots \cap W_n = \emptyset$, we do not need to mark the owner process of each task in interaction tokens. To conclude, there are three steps in mining IOWF: discovering local processes, mining interaction places and combining models. Next we illustrate details of each step.

**Discovering local processes.** It is to construct a process model representing the real running local business process from its token logs. There are designed models of these business processes but they may be very different from the actual running ones. As time goes on, local business process must have been updated a lot. It is essential to find out how it works exactly. This is the service that process discovery provides. We employ algorithm $\tau$ [5] to discover local process models. The $\tau$ algorithm takes a token log as input and returns a workflow net. For the index of token logs $i$ from 1 to $n$, each local process $W_i=(P_i,T_i,F_i)$ can be derived from $\tau(TL_i)$.

**Mining interaction places.** The interaction structure which is also a process model is denoted with $W_{int}=(P_{int},T_{int},F_{int})$. In algorithm $\tau$ [5], tokens are merged according to values of CID, PEID and CEID. However, in mining interaction places from interaction tokens, besides CID, PEID and CEID, PTCID should also be considered. So, we made some adjustments of algorithm $\tau$ here and denote the changed algorithm $\tau$ with $\tau^*$ for mining interaction places, i.e., MININGIP. Besides, in order to monitor the interactions among processes in the Internet context, we have to construct interaction places (*IPs*) for $W_{int}$. First, initialize an *IP* for place $p \in P_{int}$ with empty *PrIV*, *PoIV* and *PrS*, *PoS* for its *PrSC* and *PoSC* respectively (i.e., *PrSC*=(∅,∅) and *PoSC*=(∅,∅)). Then for each token $itoken \in TL_{int}$ merged into $p$, put all *SV* in *PSV* of *itoken* into *IP*'s *PrSC* and put all *SV* in *CSV* of *itoken* into *IP*'s *PoSC*. Eventually, algorithm $\tau^*$ is as following:

**Algorithm $\tau^*$.** MININGIP($TL_{int}$)
01 ▷ Gather interaction tokens of the same interaction place
02 Let $ITOKEN_{IP}$ be the set of ITs belonging to an IP,
03 Let $IP_{ITOKEN}$ be the set of $ITOKEN_{IP}$ for all IPs,
04 $IP_{ITOKEN} \leftarrow \emptyset$
05 **for** *itoken* **in** $TL_{int}$ **do**
06 **if** $IP_{ITOKEN} == \emptyset$ **then**
07 $ITOKEN_{IP} \leftarrow \{itoken\}$, $IP_{ITOKEN} \leftarrow IP_{ITOKEN} \cup \{ITOKEN_{IP}\}$
08 **else**
09 $ITOKEN_{IPsender} \leftarrow \emptyset$, $ITOKEN_{IPreceiver} \leftarrow \emptyset$
10 **for** $ITOKEN_{IP}$ **in** $IP_{ITOKEN}$ **do**
11 **if** ($\exists itoken' \in ITOKEN_{IP} \Rightarrow itoken'$.PT == *itoken*.PT $\wedge$
12 *itoken'*.PTCID $\neq$ *itoken*.PTCID) **then**
13 $ITOKEN_{IPsender} \leftarrow ITOKEN_{IP} \cup \{itoken\}$
14 $IP_{ITOKEN}$.remove($ITOKEN_{IP}$)
15 **if** ($\exists itoken' \in ITOKEN_{IP} \Rightarrow itoken'$.CT == *itoken*.CT $\wedge$
16 *itoken'*.CID $\neq$ *itoken*.CID) **then**
17 $ITOKEN_{IPreceiver} \leftarrow ITOKEN_{IP} \cup \{itoken\}$
18 $IP_{ITOKEN}$.remove($ITOKEN_{IP}$)
19 **if** $ITOKEN_{IPsender} \neq \emptyset \wedge ITOKEN_{IPreceiver} \neq \emptyset$ **then**
20 $IP_{ITOKEN} \leftarrow IP_{ITOKEN} \cup \{ITOKEN_{IPsender} \cup ITOKEN_{IPreceiver}\}$
21 **else if** $ITOKEN_{IPsender} \neq \emptyset$ **then**
22 $IP_{ITOKEN} \leftarrow IP_{ITOKEN} \cup \{ITOKEN_{IPsender}\}$
23 **else if** $ITOKEN_{IPreceiver} \neq \emptyset$ **then**
24 $IP_{ITOKEN} \leftarrow IP_{ITOKEN} \cup \{ITOKEN_{IPreceiver}\}$
25 **else**
26 $ITOKEN_{IP} \leftarrow \{itoken\}$, $IP_{ITOKEN} \leftarrow IP_{ITOKEN} \cup \{ITOKEN_{IP}\}$
27 **end for**
28 **end for**
29 ▷ Next is to generate *IP* from each $ITOKEN_{IP}$ in $IP_{ITOKEN}$
30 **for** $ITOKEN_{IP}$ **in** $IP_{ITOKEN}$ **do**
31 $PrTS \leftarrow \emptyset$, $PoTS \leftarrow \emptyset$, $PrSC \leftarrow \emptyset$, $PoSC \leftarrow \emptyset$
32 $IP \leftarrow (PrTS,PoTS,PrSC,PoSC)$
33 **for** *itoken* **in** $ITOKEN_{IP}$ **do**
34 $PrTS \leftarrow PrTS \cup \{itoken$.PT$\}$, $PoTS \leftarrow PoTS \cup \{itoken$.CT$\}$
35 ▷ Start of gathering history safe values
36 **for** *SV* **in** *PSV* of *itoken* **do**
37 *PrSC.PrIV*.add(*SV.IV*), *PrSC.PrS*.add(*SV.V*)
38 **for** *SV* **in** *CSV* of *itoken* **do**
39 *PoSC.PoIV*.add(*SV.IV*), *PoSC.PoS*.add(*SV.V*)
40 ▷ End of gathering history safe values
41 **end for**

**Combining models.** The last step is much easier than others. Find location of each task $t_{int} \in T_{int}$ in local processes $W_i$ and link the interaction place in *IPs* ($p_{int}$) which contains $t_{int}$ to the $t_{int}$ in $W_i$.

After constructing the global interaction process model containing all local ones and interactions among them, it is ready to monitor the interactions. Details of interaction monitoring based on the derived global interaction process model are presented in the next section.

### 5.3 Monitoring Strategy

The ability of monitoring business process together with analyzing, controlling and predicting is addressed as core features in business process intelligence (BPI) [59]. In BPI, process monitoring is defined with three key points: monitoring and analyzing running process instances, informing the users of unusual or undesired situations or states so that the users can view the monitoring results, and users can define critical situations or states of running processes themselves. These determine the technical problems that should be addressed in process monitoring. First, there should be a strategy to acquire values of business variables of running instances and an appropriate triggering strategy for data checking. Second, usual or desired values of business variables should be defined. Third, unusual or undesired values of variables should be detected accurately. At last, the monitoring strategy should be highly user-customizable. However, we need to declare that we do not monitor the whole process but concentrate on the interactions among processes, and we characterize the framework by protecting cross-business-interactions from outer tampering threatens conducted through Internet, because interactions are easier to be attacked than internal process components in the Internet context. In this section, we will illustrate how the proposed framework addresses these technical problems.

We would introduce the structure of the proposed monitoring strategy with the help of the workflow engine YAWL (YAWL is a BPM/Workflow engine described at

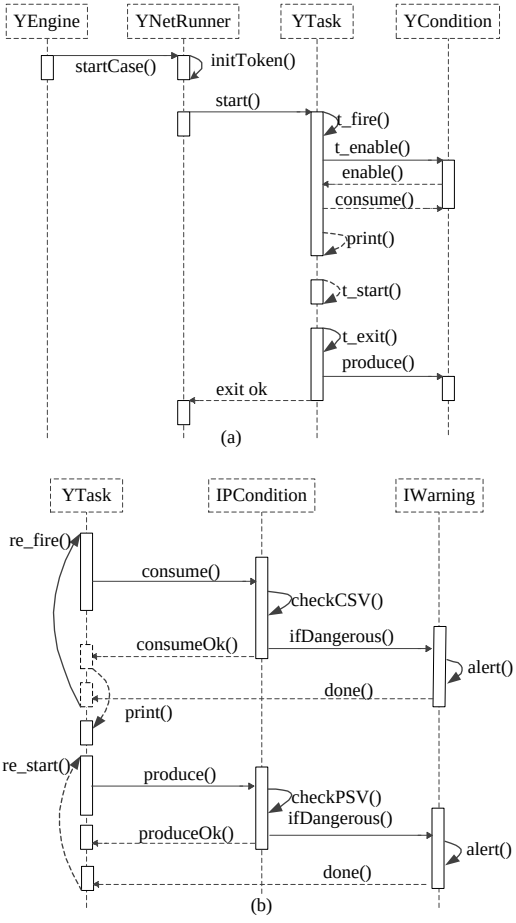



Fig. 6. Process of (a) Print Token Log in YAWL, (b) Monitor IP in YAWL

'http://www.yawlfoundation.org/'). In [5] and [8], the strategy for printing token log is also illustrated under the YAWL context, and Fig. 6a repeats the printing process. According to Petri nets theory, when a transition is fired, it will consume some tokens from its input places and when it ends, it will put some tokens into its output places. In Fig. 6a, *YEngine* is the object controlling all the processes driven by the workflow engine and the *YNetRunner* is the object managing one instance of an executed process. *YTask* and *YCondition* represent for transition and place in process model. After being started (*start*()) by the net runner, a task will be fired (*t_fire*()) and then consume (*consume*()) tokens from its input conditions. Based on Fig. 6a, we present the monitoring strategy with Fig. 6b. In Fig. 6b, the new object, *IWarning*, is defined to handle the data checking results. Monitoring details will be illustrated based on Fig. 6b as follows:

- **Acquire runtime variable-value pairs**: In the offline token-log-based conformance checking method [8], whether tokens can be put into places connecting their producing task and consuming task is used as a criterion to check the conformance between the token and the model. However, in the runtime process monitoring method, it is not to check the conformance between the tokens and places that checked, but to check the compliance between the data carried by the runtime tokens and those carried by historical tokens. So, values of business variables should be acquired by tokens firstly. In pre-checking, data retrieved from database or other data persistence media need to be checked to exclude tempering threatens from Internet. To check this data, we let *YTask* create an initialized token, called *fake token*, with all data fields set to null after retrieving data to be checked, and then set the token carried data with the retrieved data. In the next step, data carried by this fake token will be checked according to the token to be consumed (i.e., the one in *IPCondition*). In post-checking, results of the executed task need to be checked. Since there will be output tokens created by the *YTask*, we just let them carry the result data, and in the next step, check them based on historical ones in the output *IPCondition* of *YTask*. The method here for acquiring runtime values of variables to be checked overcomes the first part of the first technical hardness of runtime process monitoring discussed at the beginning of this section.

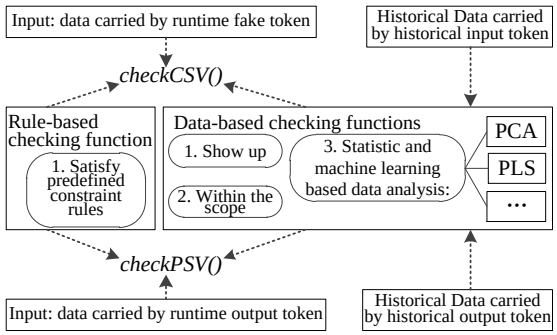



Fig. 7. Relation between *checkCSV()*, *checkPSV()* and concrete checking functions

- **Trigger data checking**: This is for overcoming the second part of the first technical hardness. As mentioned at the end of Section 3.3, we need to define data checking method for each task if we want to trigger data checking functions within the executions of tasks. Besides, each task has to maintain both historical inputs and outputs. This would make the implementation a bit more complex than trigging data checking functions through state changes of places. By this, inputs and outputs of all tasks that connect to one place would be maintained by this place and only two data checking functions are needed to be implemented, one is *checkCSV()* for pre-checking and another is *checkPSV()* for post-checking. Specifically, *checkCSV()* will be triggered by tokens being taken away from the place, i.e., *comsume()* function, and *checkPSV()* will be triggered by new tokens being put into the place, i.e., *produce()* function. Next, how to detect undesired or unusual values of variables in *checkCSV()* and *checkPSV()* is introduced.
- **Concrete data checking functions**: Recall what we said in Section 4.2, to monitor runtime interactions among local business processes is to check the compliance between runtime values and historical ones of business variables. With revised process mining algorithm $\tau^*$ illustrated in Section 5.2, we have gathered interaction tokens that carry historical

values of variables together in their corresponding interaction places, which means tokens are recorded by the places they have been put into. Specifically, historical input tokens of a task are recorded in its input places, and historical output tokens of this task are recorded in its output places. In this way, desired or usual values (i.e., *PrSC* and *PoSC*) of business variables are set (i.e., line 35 to line 40 in **Algorithm $\tau^*$**) and this overcomes the second technical hardness of process monitoring. Checking functions *checkCSV()* and *checkPSV()* are implemented for each interaction place. Each *checkCSV()* function takes the fake token created by a task before consuming a real one as input. Each *checkPSV()* function takes the output token produced by a task after completing as input. Both *checkCSV*() and *checkPSV*() will call concrete checking functions to check compliance between runtime data and data criteria. Proposing concrete data checking function is not a contribution of this paper, because there has been many research and achievements on data checking methods for process monitoring already. In the proposed framework, they could be reused. In Fig. 7, we present details of *checkCSV()* and *checkPSV()*. Although the mainly discussed checking criteria are derived from historical token-carried-data, we also introduce predefined-rule-based checking criteria to improve the completeness of data checking criteria. For data-based concrete functions, there are three kinds, namely *show up*, *within scope*, and *statistic and machine learning based* ones. *Show up* means to check whether the data value carried by runtime token had shown up in those carried by historical ones. This is useful in checking discrete variables. *Within scope* means to find whether the value of a continuous variable is within the scope consists of historical values. These two functions are simple. In [37], they are introduced to check variable values carried by messages in message-based process monitoring. As said in Section 3.3, the third data checking methods are widely used in offline business process monitoring. We can first learn a checking model from historical data recorded in interaction places and then apply the model in runtime data checking by binding the model to function *checkCSV()* or *checkPSV()*. You could learn more about the statistic and machine learning based data checking methods in [40] where a comparison study on different checking methods was presented.

- **Handle checking results**: As shown in Fig. 6b, there is a *ifDangerous()* function in *IPCondition* class. If the data checking result implies any risk to the business process, function *alert()* of *IWarning* class would be called by *ifDangerous()*. Otherwise, the running instance of the process would proceed successfully. Function *alert()* is used to inform the process users of the detected risks. As mentioned in Introduction section, this makes sure that the user or executor of the current task is informed of the risks as soon as the task is completed in post-checking. If the task is able to be rolled back, an automatic rolling back function can be bound to the *alert()* function as an action handler, which means the task can be rolled back automatically based on user reaction. Otherwise, the task would be reset to fired state and the produced tokens would be removed from its output places. Besides, users have to roll back the task manually. In pre-checking, if risk are detected, the task would be reset to initial state, i.e., unfired, and no input tokens would be consumed from its input places.

To summarize, the four core steps in the proposed monitoring framework: acquire runtime variable-value pairs to be checked, trigger data checking method, check data compliance between runtime and historical ones (i.e., defined as *Security Condition* in Section 4.1) with concrete data checking functions, and handle data checking results. In the third step, historical data are firstly prepared with the process mining technique in Section 5.2. However, after checking each new running instance, the historical data would be updated by adding data carried by tokens related to that instance. Note that concrete data checking functions are not contributions of this paper, and we just reuse those proposed in existing research in the third step. In the case study shown in the following section, the history data are not too many, and only the simple data checking functions are used, i.e., check whether runtime data exists in or within the scope of historical ones.

## 6 Case Study

In this section, we illustrate the proposed reengineering and monitoring framework in detail with FMC in Fig. 4. The entire interaction constructing and monitoring process contains three steps 1) constructing interaction token logs from e-mail records, 2) mining global interaction process model, and 3) simulating online process monitoring with designed running instances. Fig. 4 shows the integrated interaction process model of FMC after applying our reengineering strategy. There are mainly three sub-management processes, Ordering, Finance and Warehouse, and the interactions among them are recorded as e-mails in this section. Since the three local processes are already deployed online, their token logs are available and Fig. 8 shows pre-processed examples of their token logs of the same instances with task names abbreviated and sensitive information replaced.

### 6.1 Constructing Logs and Mining Interaction

The first column of token log in Fig. 8 is Order_Num which represents for Case ID in our framework and the process instances belonging to the same order in different logs are marked with the same Order_Num. Besides the basic fields defined in tokens, the executors of tasks and related files and participants are also recorded in the sample logs. Table 2 explains the content of each column in the sample logs. These logs are used directly for deriving actual running local processes with algorithm $\tau$. The main work is to generate interaction tokens from the emails sent among the executors in different processes.

(a) $W_1$: Ordering

Order_Num, Producer, PExecutor, PEID, P_Real_Time, P_Data, Consumer, CEID, C_Real_Time
1SZ, Neg, Zhang1, 1, 04/25-15:42:23, {Place:*;PC:p1, p2, p3}, OP, 2, 04/26-10:52:11,
1SZ, OP, Li1, 2, 04/28-11:53:38, {Files:F1, F2, F3}, DV, 3, 04/28-12:02:26,
1SZ, OP, Li1, 2, 04/28-11:53:38, {Files:F1, F2, F3}, CE, 4, 04/28/2015-14:34:53,
1SZ, DV, Zhu, 3, 04/30-09:12:41, {Place:*;Files:F2;PC:p2, p5}, OV, 5, 04/30-11:48:15,
1SZ, CE, Chen1, 4, 04/30-11:37:21, {Place:*;Files:F3;PC:p6, p7}, OV, 5, 04/30-11:48:15,
2WX, Neg, Zhang1, 1, 05/05-14:21:56, {Place:*;PC:p1, p2, p3}, OP, 2, 05/06-09:15:47,
1SZ, OV, Li2, 5, 05/06-10:45:06, {Place:*;Files:F2, F3, F4, PC:p1, p3}, RO, 6, 05/06-11:03:54,
1SZ, RO, Meng, 6, 05/06-11:24:31, {Files:F1, F2, F3, F4}, OP, 7, 05/06-11:35:13,
2WX, OP, Li1, 2, 05/07-11:41:26, {Files:F5, F6, F7}, DV, 3, 05/07-14:22:15,
2WX, OP, Li1, 2, 05/07-11:41:26, {Files:F5, F6, F7}, CE, 4, 05/07-14:38:41,
1SZ, OP, Li1, 7, 05/07-16:15:47, {Files:F1, F2, F3, F4}, DV, 8, 05/07-17:10:26,
1SZ, OP, Li1, 7, 05/07-16:15:47, {Files:F1, F2, F3, F4}, CE, 9, 05/07-17:13:52,
1SZ, DV, Zhu, 8, 05/08-10:21:14, {Place:*;Files:F2;PC:p2, p5}, OV, 10, 05/08-10:37:19,
1SZ, CE, Chen, 9, 05/08-09:52:46, {Place:*;Files:F3;PC:p6, p7}, OV, 10, 05/08-10:37:19,
2WX, DV, Zhu, 3, 05/09-16:52:27, {Place:*;Files:F6;PC:p2, p5}, OV, 5, 05/09-17:09:34,
2WX, CE, Chen1, 4, 05/09-14:22:50, {Place:*;Files:F7;PC:p6, p7}, OV, 5, 05/09-17:09:34,
1SZ, OV, Li2, 10, 05/10-09:33:54, {Place:*;Files:F2, F3, F4;PC:p1, p3}, SO, 11, 05/10-10:12:51,
2WX, OV, Li2, 5, 05/11-10:40:18, {Place:*;Files:F6, F7, F8, PC:p1, p3}, SO, 6, 05/11-11:01:25,
1SZ, SO, Ruan, 11, 05/12-10:08:42, {Files:F2, F9}, Pro, 12, 05/16-10:38:45,

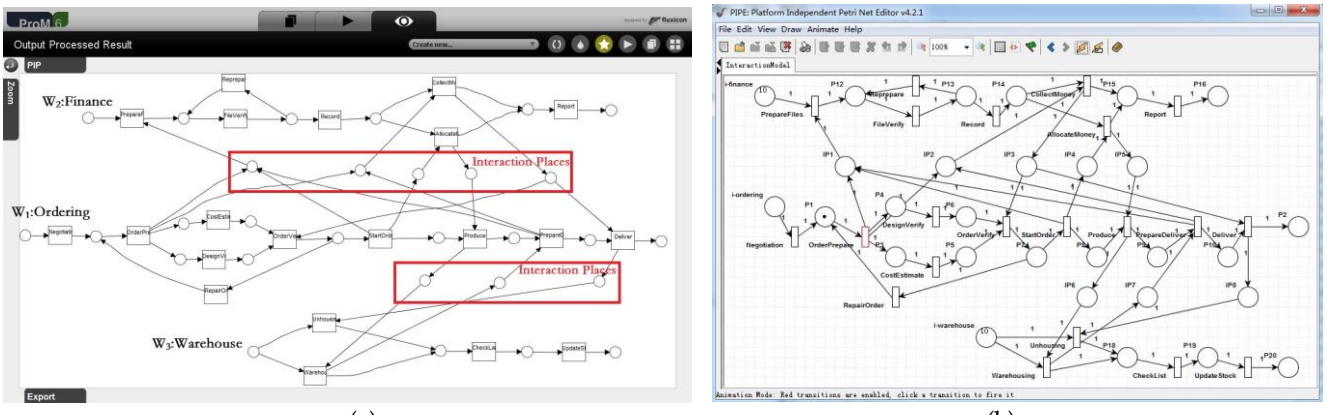
(b) E-mail Records:

1 Li1,Xie,04/28-14:33:15,1SZ,Deposit of Order,Collect Deposit,{F1},
2 Wang1,Li2,04/30-15:12:31,1SZ,Get Deposit,Return Invoice,{F4},
3 Ruan,Xie,05/12-10:11:03,1SZ,Money for Material,Allocate Money,{F9},
4 Wang1,Wei,05/15-09:47:32,1SZ,Approve,NONE,{F13},
5 Chen2,Xie,06/27-11:00:17,2WX,Apply Deliver,Collect All Money,{F6,F11},
6 Wang1,Chen2,06/29-14:10:46,2WX,Approve Deliver,NONE,{F12},
7 Chen2,Wang2,06/30-14:58:31,2WX,NONE,Deliver,{F6},
8 Wei,Wang2,07/20-13:57:09,1SZ,Store Products,Manage Warehouse,{F13},
9 Wang2,Chen2,07/21-18:43:29,1SZ,NONE,Prepare Deliver,{F13},

(c) Interaction Tokens:

1 1SZ,OP,Li1,2,04/28-14:33:15,{File:F1},PF,Xie,1,NONE,
2 1SZ,CM,Wang1,4,04/30-15:12:31,{File:F4},OV,Li2,5,05/09-17:09:34,
3 1SZ,SO,Ruan,11,05/12-10:11:03,{File:F9},PF,Xie,1,NONE,
4 1SZ,AM,Wang1,4,05/15-09:47:32,{File:F13},Pro,Wei,12,05/16-10:38:45,
5 2WX,PD,Chen2,8,06/27-11:00:17,{File:F6,F11},PF,Xie,1,NONE,
6 2WX,CM,Wang1,4,06/29-14:10:46,{File:F12},Del,Chen2,9,06/30-11:17:08,
7 2WX,Del,Chen2,9,06/30-14:58:31,{File:F6},Unh,Wang2,1,NONE,
8 1SZ,Pro,Wei,12,07/20-13:57:09,{File:F13},War,Wang2,1,NONE,
9 1SZ,War,Wang2,1,07/21-18:43:29,{File:F13},PD,Chen2,13,07/22-14:04:57,

Fig. 8. Example token logs of local processes in Fig. 4: (a) the log of $W_1$, (b) integrated information of interaction emails, and (c) interaction tokens derived from these emails




(a) (b)

Fig. 9. Screenshots of (a) mining result of the whole interaction model in ProM, (b) monitoring interaction process in the simulating tool implemented in PIPE.

TABLE 2
CONTENTS OF EACH COLUMN IN THE EXAMPLE TOKEN LOGS

| Column | Content |
|---|---|
| *Order_Num* | Serves as CID for marking process instance uniquely |
| *Producer/ Consumer* | The PT/CT of the token, the abbreviations of tasks are used in the example logs |
| *P_Executor/ C_Executor* | The human operators of PT/CT |
| *P_EID/C_EID* | EIDs of the producer /consumer of the token |
| *P_Real_Time/ C_Real_Time* | The real ending time of PT and the real firing time of CT |
| *P_Data* | Related firing values of the producer, such as files, participants and locations, etc. |

Fig. 8b and c show the integrated information of emails and the derived interaction token log. In the email records (see Fig. 8b), the sender, receiver, sent time and attached files are directly logged. The other information including the related order number (i.e., Case ID) is excerpted from the main body. In constructing the interaction tokens from the email items, the columns without underlines in Fig. 8c can be filled in directly while the column with underlines should be searched in the local processes' token logs. Taking line 5 in Fig. 8c as an example: first, find the token whose Order_Num and PExecutor are '2WX' and 'Chen2' respectively in the token log of $W_1$ (see Fig. 9a) and the results are tokens in the fourth and fifth last lines in Fig. 8a; next, filter result tokens with the conditions that P_Real_Time is before sent time (i.e., '06/27-11:00:17') and P_Data contains attached files (i.e., 'F6' and 'F11'), and after that only one token is left; at last, fill in the first underlined column of line 5 in Fig. 8c with the Producer 'PD' of the remained token. It is the same with filling in the other required information of the interaction tokens.

Upon deriving interaction tokens, the interaction structure can be discovered based on the algorithm shown in Section 5.2. Fig. 9a shows the derived global interaction process model from token logs in ProM (ProM is an open-source framework for process mining and described at 'http://www.processmining.org/'). The three local processes are marked with their name and the interaction places are circled with big rectangles. Along with the interaction structure, security conditions of each interaction place are also constructed and they would be used as data criteria for data checking in interaction monitoring in Section 6.2.

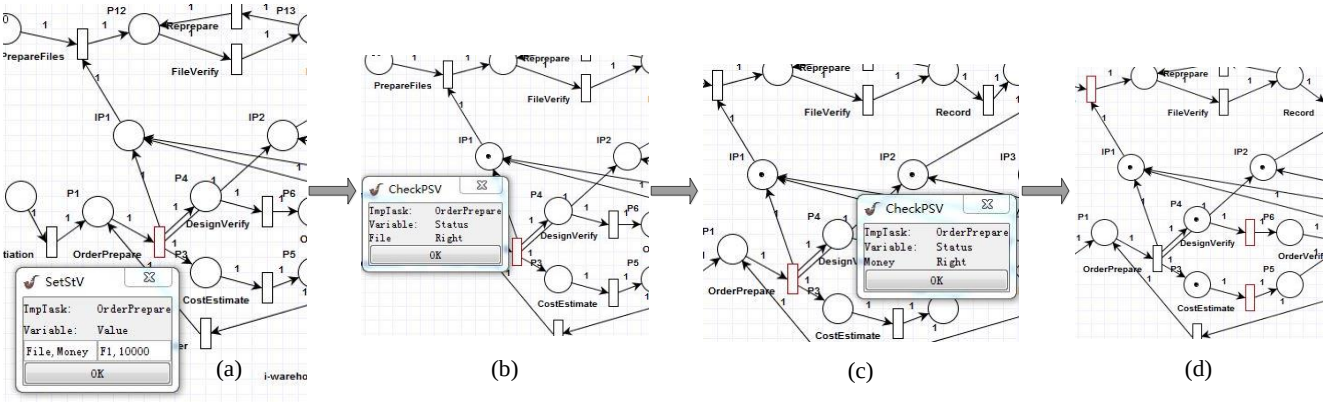



Fig. 10. Monitoring process of checking producer security values of task Order Prepare (a) before the start of task, input related execution values, (b) check the token put into the first interaction place, (c) check the token put into the second interaction place, and (d) the status after all tokens are produced

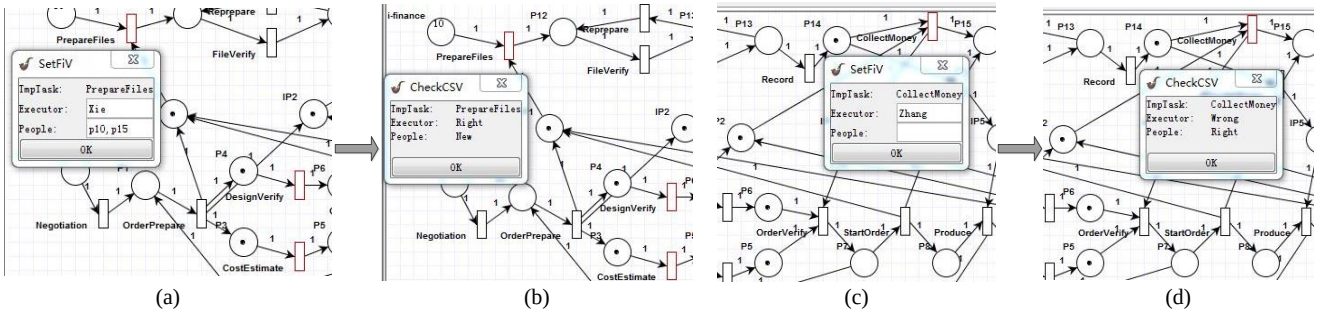



Fig. 11. Monitoring process of checking the consumer security values of task Prepare Files and Collect Money (a) set related variables before Prepare Files firing, (b) check CSV after firing and their status are right, and (c, d) the checking process on task Collect Money and the executor of the task is wrong

### 6.2 Simulating Monitoring

To simulate the process of monitoring the global interaction process model which is used in case study, we develop a monitoring plug-in based on the animation of process execution in PIPE (a tool for analyzing Petri nets from 'http://sourceforge.net/projects/pipe2/'). Fig. 9b shows the screenshot of deployed sample process model in PIPE under simulating perspective. In PIPE, places, tasks and tokens are represented by circles, rectangles and black dots respectively. The execution of a task is divided into two steps and they are 1) setting firing values, 2) setting starting values. The checking actions of monitoring are triggered after the task firing and starting separately. After firing values are set, the task will consume tokens from its input places and before the token is consumed, the firing values of the task are checked according to the history data stored in the interaction places. Fig. 10 and Fig. 11 illustrate the details of monitoring the interactions between 'Order Prepare' in $W_1$, 'Prepare Files' and 'Collect Money' in $W_2$.

The simulation of process monitoring related to these interactions starts from setting starting values of 'Order Prepare' (see Fig. 10a). Then the task is executed and upon the ending, four tokens will be produced and put into corresponding places. First, one token is put into IP1 and the related starting values of 'Order Prepare' recorded in IP1 are checked (see Fig. 10b). Then, another token is put into IP2 and IP2 related starting values of 'Order Prepare' are checked (see Fig. 10c). The values of variable 'File' and 'Money' would be checked according to those collected from historical interaction tokens shown in Fig. 8c. For example, historical values of 'File' contain 'F1', checking of 'File' with a value 'F1' is passed. After checks are passed, the status of the model related to output places of 'Order Prepare' is as shown in Fig. 10d.

Now, the task 'Prepare Files' in $W_2$ is ready to execute. After firing, the first thing is to set firing values of the task (see Fig. 11a). Then the task will consume a token from IP1 to start execution. But before the token is consumed, IP1 checks related firing values of 'Prepare Files' by those stored in IP1 (see Fig. 11b). After the checking is passed, 'Prepare Files' starts successfully. Another firing values checking related to these interactions happens when the 'Collect Money' in $W_2$ is fired after setting firing values (see Fig. 11c and d). However, the historical executor derived from Fig. 8c is 'Wang1' but not 'Zhang', so the checking of executor of 'Collect Money' is not passed. Then, the task will not be executed and the process needs to be rolled back for re-setting firing values of the 'Collect Money' task.

In the next section, the results of the evaluation experiments conducted on the proposed frameworks based on the interaction model used in case study will be discussed.

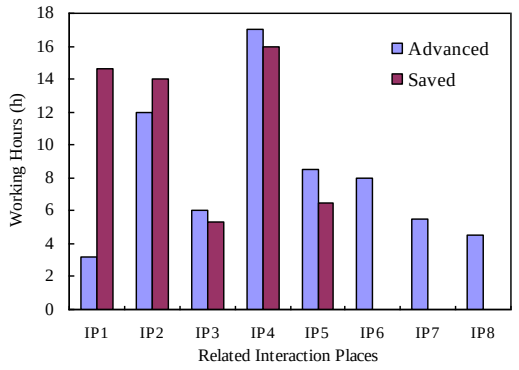



Fig. 12. Statistics of advanced hours of faults detecting and saved working hours by process monitoring in the ideal situation

## 7 Evaluation Results

This section presents the experiment results to answer the question that whether the monitoring framework helps to lower the risk of process fault and improve process recovery efficiency.

The significances of the process monitoring framework proposed in this paper are 1) ensuring process security, 2) improving process fault-handling efficiency. The key techniques for achieving these targets are 1) pre-checking related information of tasks before they start, 2) post-checking outputs upon current task ends rather than until the next one starts. If the needed information for starting one task is not correct, it will be figured out in pre-checking and the users need to reset required information to start the tasks. This will ensure the business security. A simple example is that the monitoring step shown in Fig. 11d figures out that the executor of 'Collect Money' has never existed in the collection of former ones so the task is not permitted to start. This may prevent an insider intending to conduct the task for illegal benefits. Besides, pre-checking will especially benefit process data security considering the outer tempering threatens through the Internet. If the execution results of task do not have high compliance with historical ones, it will be figured out in post-checking and the users will be informed with the checking result upon the end of the execution.

To evaluate the contribution made by our monitoring framework in improving process fault-handling efficiency, we calculate the average saved hours (*SH*) and the average advanced hours (*AH*). These two metrics shows the advantages of the proposed framework compared with a basic one that without pre and post data checking. The assumption of the comparison based on *SH* and *AH* is that the related faults would be detected by both monitoring frameworks. *SH* means hours saved in avoiding repeating an interaction task with pre-checking. If a fault occurs in firing a task but is not detected until the end of the task, then the task should be rolled back and executed again. However, if pre-checking is set to detect the fault, then time used for repeating the task would be saved. The time used for executing a task is calculated through the time it consumes a token and the time it produces a token. So, *SH* can be calculated with:

$$SH_{IP} = \frac{\sum_1^n token_{p\_real\_time} - token'_{c\_real\_time}}{n} \quad (1)$$

where $token \in IP$, $token' \in TL_i$ ($i \in \{1,\ldots,n\}$) satisfies $f_{cid}(token')==f_{cid}(token) \wedge f_{ct}(token')== f_{pt}(token)$, and *p_real_time* is the produced time of the token, *c_real_time* is the consumed time of the token. *AH* means hours advanced in detecting a possible execution fault by post-checking. A fault in executing a task would not be detected until the next task starts without post-checking. This means, the time when the fault is detected is advanced by the time span between the finishing of a risk task and the beginning of the next task. So, *AH* can be calculated with:

$$AH_{IP} = \frac{\sum_1^n token_{c\_real\_time} - token_{p\_real\_time}}{n} \quad (2)$$

where variables have the same meaning with those in (1). Fig. 12 shows the *AH*s and *SH*s of the interaction places in the model in Fig. 4. Since that the input tasks of IP6, IP7 and IP8 spend lots of time on completing to be expressed in the comparison, we do not mark them in the figure. Note that the precision and recall of detecting faults in the experiments for computing *SH* and *AH* in FMC process are both 100%, because precision and recall in detecting faults actually depend on the concrete data checking functions. Besides, for calculating *SH* and *AH* for each IP in evaluation experiments, test data should be manually designed for detecting all faults. So, all faults are detected and all detected faults are true ones. Comparing the *AH*s and *SH*s with the executing time cost by the whole process instance, it is found that the monitoring framework does help a lot in avoiding a quite long time delay if any fault is detected.

The evaluation results tell that the online monitoring framework based on history data does well in protecting the business process from outer data tempering threaten and avoiding considerable execution delay.

## 8 Conclusion And Future Work

In this paper we propose an online process interaction monitoring framework based on token carried data for multi-processes. The IOWF is employed for modeling the interactions among collaborative sub-processes. The token-log-based process discovery algorithm is used in constructing collaborative model. However, in those small light industrial companies, the interactions among departments were not recorded in system logs. Therefore, before applying the process discovery algorithm, corresponding token log of interactions is generated from records of traditional interacting ways. During the collaboration constructing process, the relative history interacting data are also discovered for the monitoring usage and this is one of the innovations of our framework. Another innovation is setting pre and post checking points in monitoring at when a token is going to be taken away from and just being put into a place. That means we trigger the checking action by detecting two kinds of

tokens' actions, i.e., being created and consumed. The evaluation results gathered from the conducted mining and monitoring experiments prove that 1) the mining strategy we choose for constructing global interaction model is feasible, 2) the checking points we set in the monitoring framework do well in protecting the collaboration as well as avoiding considerable execution delay if any fault is detected.

Since task executing related data processing in business process becomes a big challenge in the fast developed big data era, as future work, we would like to apply data-driven techniques to the analysis of task executing related variables in detail. Other possible future research directions are adapting our interaction mining framework for dealing with big data, implementation in current BPM systems, such as JBPM, and enhancing the implementation of the monitoring framework with better user experience.

## ACKNOWLEDGMENT

This work was supported by the National 863 Program (2015AA01A203), the National Natural Science Foundation, China (No.61572251), the Natural Science Foundation of Jiangsu Province (No.BK20131277), the Fundamental Research Funds for the Central Universities. Jidong Ge is the corresponding author.

## REFERENCES

[1] W. Yu, C. Yan, Z. Ding, C. Jiang and M. Zhou, "Modeling and Validating E-Commerce Business Process Based on Petri Nets", *IEEE T. Systems, Man, and Cybernetics: Systems*, vol. 44, no.3, pp. 327-341, 2014

[2] H. Mei, G. Huang, H. Zhao and W. Jiao, "A software architecture centric engineering approach for Internetware", Science in China Series F: Information Sciences, vol. 49, no. 6, pp. 702-730, 2006

[3] J. Lü, Y. Huang, C. Xu and X. Ma, "Managing Environment and Adaptation Risks for the Internetware Paradigm", Proc. Theories of Programming and Formal Methods, Vol. 8051, pp. 271-284, 2013

[4] W.M.P. van der Aalst, "Interorganizational Workflows: An Approach based on Message Sequence Charts and Petri Nets", Systems Analysis Modeling Simulation, vol. 34, no.3, pp.335-367, 1999

[5] C. Li, J. Ge, L. Huang, H. Hu, B. Wu, H. Yang, H. Hu and B. Luo, "Process mining with token carried data", Information Sciences, vol.328, no.20, pp. 558–576, January 2016.

[6] N. Herzberg, A. Meyer. "Improving Process Monitoring and Progress Prediction with Data State Transition Events". Proceedings of the 5th Central-European Workshop on Services and their Composition, pp.20-23, February, 2013.

[7] R. Conforti, M. de Leoni, M. La Rosa, W.M.P. van der Aalst, A.H.M. ter Hofstede. "A recommendation system for predicting risks across multiple business process instances". Decision Support Systems Volume 69, pp. 1-19, 2015.

[8] C. Li, J. Ge, H. Hu, H. Hu and B. Luo, "Method for Conformance Checking Based on Token Log", Journal of Software, vol. 26, no. 3, pp. 509-532, March 2015 (in Chinese)

[9] I. M. Lazarte, L. H. Thom, C. Iochpe, O. Chiotti and P. D. Villarreal,"A distributed repository for managing business process models in cross-organizational collaborations", Computers in Industry, vol. 64, no. 3, pp. 252-267, April 2013

[10] M. Montali, F. M. Maggi, F. Chesani, P. Mello and Wil M. P. van der Aalst, "Monitoring business constraints with the event calculus", ACM TIST, vol. 5, no.17, December 2013

[11] N. Desai, A. K. Chopra and M. P. Singh, "Amoeba: A methodology for modeling and evolving cross-organizational business processes", ACM Trans. Softw. Eng. Methodol, vol.19, no.2, October 2009

[12] G. Decker and M. Weske, "Interaction-centric modeling of process choreographies," Information Systems, vol. 36, no. 2, pp. 292–312, April 2011

[13] N. Tr˘cka, W. M. P. van der Aalst and N. Sidorova, "Data-flow antipatterns: Discovering data-flow errors in workflows," Proc. CAiSE'09, vol. 5565, pp. 425–439, June 2009

[14] D. Knuplesch, R. Pryss and M. Reichert, "Data-aware interaction in distributed and collaborative workflows: Modeling, semantics, correctness", CollaborateCom, pp. 223-232, December 2012

[15] D. Bianchini, c. cappiello, V. De Antonellis, B. Pernici, "Service Identification in Interorganizational Process Design", IEEE Transactions on Services Computing, vol. 7, no. 2, pp. 265-278, May 2013

[16] G. Huang, Y. Ma, X. Liu, Y. Luo, X. Lu and M. B. Blake, "Model-Based Automated Navigation and Composition of Complex Service Mashups", IEEE Transactions on Services Computing, vol. 8, no. 3, pp. 494-506, MAY/JUNE 2015

[17] M. B. Blake, M. F. Nowlan, "Knowledge Discovery in Services (KDS): Aggregating Software Services to Discover Enterprise Mashups", Transaction on Knowledge and Data Engineering, vol. 23, no. 6, pp. 889-901, JUNE 2011

[18] X. Liu, G. Huang, Q. Zhao, H. Mei and M. B. Blake, "iMashup: a mashup-based framework for service composition", SCIENCE CHINA: Information Sciences, vol. 57, no. 1, pp. 1-20,January 2014

[19] IEEE Task Force on Process Mining, "Process Mining Manifesto", Proc. Business Process Management Workshops, Springer-Verlag, Berlin, vol. 99, pp. 169-194, 2012

[20] W. Song, H. Jacobsen, C. Ye, X. Ma, "Process Discovery from Dependence-Complete Event Logs", IEEE Transactions on Services Computing, In Press, April 2015

[21] W. van der Aalst, T. Weijters and L. Maruster, "Workflow mining: Discovering process models from event logs", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, no. 9, pp.1128-1142, September 2004

[22] A. de Mederios, B. van Dongen, W. van der Aalst and T. Weijters, "Process mining: Extending the α‐algorithm to mine short loops", Tech. rep., Univ. of Technology, Eindhoven, 2004

[23] L. Wen, Wil M. P. van der Aalst, J. Wang and J. Sun, "Mining process models with non-free-choice constructs", Data Min. Knowl. Discov, vol. 15, no. 2, pp. 145-180, March 2007

[24] L., Wen, J., Wwang, W.M.P., van der Aalst, B., Huang, J., Sun, 2010. "Mining process models with prime invisible tasks". Data and Knowledge Engineering, 69(10), 999–1021.

[25] L. Wen, J. Wang, W. van der Aalst, B. Huang and J. Sun, "A novel approach for process mining based on event types", Journal of Intelligent Information Systems, Vol. 32, pp. 163-190, January 2009

[26] D. Wang, J. Ge, H. Hu, B. Luo and L. Huang, "Discovering process models from event multi-set", Expert Systems with Applications, Vol. 39, pp.11970-11978, November 2012

[27] J. Carmona. "Projection approaches to process mining using region-based techniques". Data Mining and Knowledge Discovery, 24(1), pp. 218-246, 2012.

[28] M.L. van Eck, J.C.A.A. Buijs, B.F. van Dongen. "Genetic Process Mining: Alignment-based Process Model Mutation". Lecture Notes in Business Information Processing, Volume 202, pp. 291-303, 2015.

[29] D. Deutch, T. Milo, N. Polyzotis and T. Yam, "Optimal top-k query evaluation for weighted business processes". PVLDB, vol. 3, no. 1, pp. 940–951, September 2010

[30] W.M.P. van der Aalst, A. Adriansyah and B. van Dongen, "Replaying History on Process Models for Conformance Checking and Performance Analysis", WIREs Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 182-192, March/April 2012

[31] L. Bentakouk, P. Poizat and F. Zaidi, "Checking the Behavioral Conformance of Web Services with Symbolic Testing and an SMT Solver", TAP 2011, vol. 6706, pp. 33-50, June 2011

[32] K. Bhattacharya, C. E. Gerede, R. Hull, R. Liu, J. Su. "Towards Formal Analysis of Artifact-Centric Business Process Models". Business Process Management, Volume 4714 of Lecture Notes in Computer Science, pp. 288-304, 2007.

[33] E.H.J. Nooijen, B.F. van Dongen, D. Fahland. "Automatic Discovery of Data-Centric and Artifact-Centric Processes". Business Process Management Workshops, Volume 132 of Lecture Notes in Business Information Processing, pp. 316-327, 2013.

[34] V. Popova, D. Fahland, M. Dumas. "Artifact Lifecycle Discovery". Int. J. Cooperative Inf. Syst. 24(1), March, 2015.

[35] X. Lu, M. Nagelkerke, D. van de Wiel, D. Fahland. "Discovering Interacting Artifacts from ERP Systems". IEEE Trans. Services Computing 8(6), pp. 861-873, 2015.

[36] D. Fahland, M. de Leoni, B. F. van Dongen, W.M.P. van der Aalst. "Behavioral Conformance of Artifact-Centric Process Models". in Proceedings of 14th International Conference on Business Information Systems, pp. 37-49, 2011.

[37] S. Halle and R. Villemaire, "Runtime monitoring of message-based workflows with data", Proc. International IEEE Enterprise Distributed Object Computing Conference, pp.63–72, September 2008

[38] L.T. Ly, S. R. Ma, D. Knuplesch and P. Dadam, "Monitoring Business Process Compliance Using Compliance Rule Graphs", Proc. On the Move to Meaningful Internet Systems (OTM 2011), LNCS, Vol. 7044, pp. 82-99, October 2011

[39] L. T. Ly, F. M. Maggi, M. Montali, S. R. Ma and Wil M.P. van der Aalst, "Compliance monitoring in business processes: Functionalities, application, and tool-support", Information Systems, Vol. 54, pp. 209-234, December 2015

[40] S. Yin, S. X. Ding, A. Haghani, H. Hao and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process", Journal of Process Control, vol. 22, no. 9, pp. 1567-1581, October 2012

[41] S. J. Qin, "Survey on data-driven industrial process monitoring and diagnosis", Annual Reviews in Control, vol. 36, no. 2, pp. 220-234, 2012

[42] S. J. Qin, "Data-driven fault detection and diagnosis for complex industrial processes", Proc. 7th IFAC Symposium on Fault Detection and Supervision and Safety of Technical Processes, Barcelona, Spain, 2009.

[43] A. A. Khan, J.R. Moyne and D.M. Tilbury, "Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares", Journal of Process Control, vol. 18, no. 10, pp. 961-974, December 2008

[44] B. M. Wise and N. B. Gallagher, "The process chemometrics approach to process monitoring and fault detection", Journal of Process Control, vol. 6, no. 6, pp. 329-348, December 1996

[45] L. Chiang, E. Russell and R. Braatz, "Fault Detection and Diagnosis in Industrial Systems", Springer-Verlag, London, pp. 71-84, 2001

[46] J. M. Lee, C. Yoo, I. B. Lee, "Statistical process monitoring with independent component analysis", Journal of Process Control, vol. 14, no. 5, pp.467-485, August 2004

[47] Wil M. P. van der Aalst, Kees M. van Hee, Jan Martijn E. M. van der Werf and Marc Verdonk, "Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor", IEEE Computer, vol. 43, no. 3, pp. 90-93, 2010

[48] F. L. Bezerra and J. Wainer, "Algorithms for anomaly detection of traces in logs of process aware information systems", Inf. Syst, vol. 38, no. 1, pp. 33-44, 2013

[49] N. Herzberg, M. Kunze and A. Rogge-Solti, "Towards Process Evaluation in Non-Automated Process Execution Environments", Proc. 4th Central-European Workshop on Services and their Composition (ZEUS,2012) pp. 97-103, February 2012

[50] S. Helal, "Sustaining Computer's Impact and Adapting to Change", IEEE Computer, vol.48, no. 1, pp. 7-9, JANUARY 2015

[51] Y. Wei, M. Brian Blake, "Adaptive Web Services Monitoring in Cloud Environments", International Journal of Web Portals, vol. 5, no. 1, pp. 15-27, January-March 2013

[52] J. Simmonds, Y. Gan, M. Chechik, S. Nejati, B. O'Farrell, E. Litani, J. Waterhouse."Runtime Monitoring of Web Service Conversations". IEEE Trans. Services Computing 2(3), pp. 223-244, 2009.

[53] Victor Chu, Raymond Wong, Simon Fong, Chi-Hung Chi. "Emerging Service Orchestration Discovery and Monitoring". IEEE Transactions on Services Computing, in press. DOI:http://doi.ieeecomputersociety.org/10.1109/TSC.2015.2511000.

[54] Q. Wang, J. Shao, F. Deng, Y. Liu, M. Li, J. Han, H. Mei. "An Online Monitoring Approach for Web Service Requirements". IEEE Trans. Services Computing 2(4), pp. 338-351, 2009.

[55] N. Herzberg, M. Kunze and M. Weske, "Monitoring Business Process Interaction", Proc. Service-Oriented Computing (ICSOC 2012 Workshops), Vol. 7759, pp. 228-240, 2012

[56] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management", The Journal of Circuits, Systems and Computers, vol. 8, no. 1, pp. 21-66, February 1998

[57] T. Murata, "Petri nets: Properties, analysis and applications", Proc. of the IEEE, vol. 77, no. 4, pp. 541-580, April 1989

[58] X. Liu, Y. Ma, G. Huang, J. Zhao, H. Mei and Y. Liu, "Data-Driven Composition for Service-Oriented Situational Web Applications", IEEE Transactions on Services Computing, vol. 8, no. 1, pp. 2-16, JANUARY/FEBRUARY 2015

[59] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal and M. Shan, "Business Process Intelligence", Computers in Industry, vol. 53, no. 3, pp. 321-343, April 2004