IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 17, 2020, accepted March 24, 2020, date of publication March 30, 2020, date of current version April 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984057

Top-Down Process Mining From Multi-Source Running Logs Based on Refinement of Petri Nets

QINGTIAN ZENG[1], HUA DUAN[1], AND CONG LIU[2]
[1]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2]School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China

Corresponding authors: Hua Duan (huaduan59@163.com) and Cong Liu (liucongchina@sdust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61902222, in part by the Science and Technology Development Fund of Shandong Province under Grant ZR2017MF027, in part by the Taishan Scholars Program of Shandong Province under Grant ts2090936 and Grant tsqn201909109, in part by the Humanities and Social Science Research Project of the Ministry of Education under Grant 18YJAZH017, in part by the Science and Technology Support Plan of Youth Innovation Team of Shandong Higher School under Grant 2019KJN024, and in part by the Shandong University of Science and Technology Research Fund.

ABSTRACT Today's information systems of enterprises are incredibly complex and typically composed of a large number of participants. Running logs are a valuable source of information about the actual execution of the distributed information systems. In this paper, a top-down process mining approach is proposed to construct the structural model for a complex workflow from its multi-source and heterogeneous logs collected from its distributed environment. The discovered top-level process model is represented by an extended Petri net with abstract transitions while the obtained bottom-level process models are represented using classical Petri nets. The Petri net refinement operation is used to integrate these models (both top-level and bottom-level ones) to an integrated one for the whole complex workflow. A multi-modal transportation business process is used as a typical case to display the proposed approach. By evaluating the discovered process model in terms of different quality metrics, we argue that the proposed approach is readily applicable for real-life business scenario.

INDEX TERMS Workflow models, multi-source running log, distributed process mining, petri nets, refinement operation.

I. INTRODUCTION

Workflow Management Systems (WfMSs) support the execution of business processes [1] as they require the definition of processes, automate the enactment of process steps and their execution is guided by business rules and execution logic, and finally they record the execution steps of a business process. In particular, workflow logs [2], [3]–[7], contain the execution information for all instances of activities of. They depict when and which actor performed which task, which contains very valuable information of the actual execution of business processes (as opposed of merely specified or desired descriptions of business processes). Thus they could be a valuable resource for business process improvement, reorganizations, and re-engineering. *Process mining* (also referred to as workflow mining) is a subfield of data mining concerned with method(s) of distilling a structured process description from a set of real executions [3]. Its goal is to analyze a running log to construct a workflow (or process) model that best describes all its recorded instances.

Today's information systems are incredibly complex and typically composed of a large number of applications or components. Applications typically support fragments [8] of a process and as a result, the information required for process mining is scattered over different enterprise information systems. Therefore, the step to collect the event log used as input for process mining is far from trivial [2]. Even within a single product, events may be logged at several different parts of the system. Consider for example an ERP (Enterprise Resource Planning) system like SAP (System Applications and Products) [9], there are dozens of logs relevant for process mining and these logs are always kept by different partners or organizations. One approach is to use a data warehouse to extract the information from these distributed logs [10], and then mining the process model directly from the centralized warehouse with integrated log data. The other way is to

The associate editor coordinating the review of this manuscript and approving it for publication was Shouguang Wang.

conduct a distributed mining technique, i.e. mining process models of different organizations separately, and then integrate them to obtain the whole one.

In contrast to the existing work [11]–[13], we explore the distributed process mining from heterogeneous logs which have the following characteristics: (1) The workflow logs used for process mining are distributed on different servers; (2) The workflow logs are recorded on different servers with different log structures; (3) The workflow logs are kept by their own organization or partner, and they are not accessible to others for security; and (4) The workflow logs of single organization can only reflect part of the business processes of the whole workflow and its interactions with other organizations. Therefore, this paper adopts the later idea. Towards this issue, we have introduced a bottom-to-top process mining approach in [2]. This work first separately obtain the process models of each organization, and then integrate these models using four coordination patterns to obtain the integrated process model. This work assumed that those distributed servers are the same, i.e. they are functionally equal with each other. However, servers are not always in the same status for some real-life applications. Consider for example, a service out-sourcing scenario will usually involve one main workflow describing the businesses of the whole enterprise and several out-sourcing sub-processes provided by other enterprises. Therefore, its corresponding running logs are distributed over one main server and several local servers. To cope with this problem, a top-down process mining approach is proposed in our work.

The rest of this paper is organized as follows. Section II presents a brief review of the related work. Section III defines some related preliminaries. Section IV presents the framework of the top-down process mining. Section V introduces the detailed mining approaches to obtain the integrated model. Section VI introduces a multi-modal transportation process as a typical case to illustrate our top-down process mining approaches. Finally, Section VII concludes the paper.

II. RELATED WORK

In this section, we mainly discuss two related research areas, i.e. process mining and Petri net refinement.

A. PROCESS MINING

Process mining is used to discover, monitor and improve real processes by extracting knowledge from event logs [14]. Many works that address on process mining techniques, which take event logs to produce a process model without using any priori information, have been published, in particular those on control-flow discovery, such as [11]–[13], [15], [16], and the existing process mining approaches and future directions are surveyed in [17].

A large number of techniques have been developed to solve process mining tasks in last decade. As an often cited example, $\alpha-$Algorithm [11] first defined four kinds of log-based ordering relations, based on which the ordering relations among activities are obtained. And then a workflow net was derived from these activity dependency relations. Following this work, some improvements [12], [13] on the $\alpha-$Algorithm were introduced to promote its performance. To further its application for less-structured event log and overcome the "*spaghetti-like*" models which contain all details without any hierarchies, Christian and Wil [18] proposed the fuzzy mining approach. In this approach, activities and their relations are clustered and abstracted according to their importance to demonstrate different hierarchies or levels. However, the fuzzy miner does not have any semantic significance with respect to the domain, therefore it may suffer the risk of aggravating some irrelevant activities together to a cluster. Towards this limitation, Bose *et al.* [19] proposed hierarchical discovery approaches using a set of interrelated plug-ins in ProM to deal with fine-grained event log and less structures process models. Different from the traditional fuzzy miner, the hierarchies are obtained through the automated discovery of pattern abstractions [20]. It is proved that the discovered patterns always have its specific domain semantics. To be able to analyze incomplete and noisy event logs with various guarantees, a set of inductive process mining techniques [21] on the basis on process trees are well-developed, which can well guarantee the discovered model to be sound.

In our previous work, a process mining approach was presented to obtain the structural model with timing constraints for a workflow from its timed running logs in [22]. By constructing its reachability graph, we found the running schemas of a workflow with timing constraints on each activity. In [8], we calculated the minimum execution time of a workflow and how to fragment it to achieve a high server usage according to the workflow model mined from its corresponding running logs. In [2], we conducted the application of process mining for workflow integration, where four coordination patterns between different organizations are defined. Process mining approaches are used to discover the workflow model of each organization and corresponding coordination patterns, based on which the process integration is conducted.

B. PETRI NET REFINEMENT

Generally speaking, Petri net refinement technique is used as a top-down approach for supporting hierarchical modeling and properties analysis of complex systems [23]. Zuberek *et. al.* [24] formalized the concepts of hierarchies of refinements in Petri nets and demonstrated some simple applications in traditional flexible manufacturing systems. On the work of [24], Huang and Mak [25] performed a further study on the structural and dynamical properties preservation of refinement in system design. More recently, Jiao *et. al.* [26] considered two kinds of refinement transformations, and proved that regularity can be preserved automatically for a kind of pure and ordinary connected nets. The two refinement transformations can be used to construct large and complex net models in Petri-net-based system design and verification.

In the area of business process management, van der Aalst [27] first applied the refinement technique to model hierarchical workflow nets. In [28], refinement operation of workflow nets was applied to model and analyze an integrated workflow. They proved that step-by-step refinement of transitions could realize hierarchical modeling of workflow and workflow integration. In [29], a series of concepts were defined for formalizing the refinement of workflow net. And then the net languages of the refined net can be obtained, which is proved in a lower complexity. In [30], Ding *et. al.* first introduced the Petri net refinement on the basis of a type of k-well-behaved Petri net and then the property relationships among sub-, original, and refined Petri nets from the perspective of system synthesis and net language preservation via stepwise refinement was studied in depth.

C. SUMMARY OF THE RELATED WORK

Based on the aforementioned summaries, we conclude that existing work on process mining [11]–[13] suffers from the following two limitations: (1) existing mining techniques only suit a centralized mining demand, i.e. they only take in centralized log set; and (2) only homogeneous-structured running logs are used to conduct the mining process. As the inner data of one organization is not accessible to other organizations for security sake, the former approach is not applicable in real-life use. Moreover, the running logs stored by different organizations are usually heterogeneous, which is not easy to integrate them for centralized mining. In contrast to the existing work we explore the distributed process mining from a set of multi-source heterogeneous running logs. Different from traditional idea that only takes the Petri net refinement operation [23]–[26] as a means to model and analyse structural and dynamic properties of complex systems, our scope is to use it as a technique to integrate the top-level process model and its corresponding bottom-level ones.

The main contributions of our work include: (1) The top-down process mining architecture is first presented to handle process mining in a distributed case; (2) The top-level process mining algorithm is proposed whose result is represented as an extension of Petri nets; (3) The bottom-level process mining algorithm is proposed and its result is formalized with traditional Petri net model; and (4) Petri net refinement operation is used to refine the abstract transitions in the top-level process model with their corresponding bottom-level models to obtain the integrated process model.

III. PRELIMINARY

As Petri nets [31] are capable of combining the graphical representation of workflows and a formal foundation, so they have been widely used to model, analyze and verify workflows [32]–[44]. Some of the essential terminology and notations regarding the Petri net used in this paper are presented as follows.

$N = (P, T; F)$ is named as a net if (1) $P \cap T = \phi, P \cup T \neq \phi$; (2) $F \subseteq (P \times T) \cup (T \times P)$; and (3) $Dom(F) \cup Cod(F) = P \cup T$. $\forall x \in P \cup T$, the set ${}^{\bullet}x = \{y | y \in P \cup T \ and \ (y, x) \in F\}$ is the preset of x and $x^{\bullet} = \{y | y \in P \cup T \ and \ (x, y) \in F\}$ is the postset of x. A *Petri net* is a 4-tuple $\Sigma = (P, T; F, M_0)$, where $N = (P, T; F)$ is a net, and $M_0 : P \rightarrow Z^+$ (Z^+ is the non-negative integer set) is the initial marking of Σ.

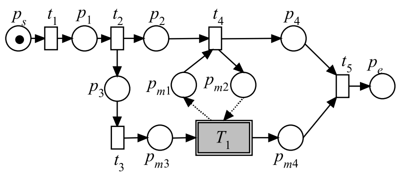


FIGURE 1. An example top-level process model.

Definition 1 (Top-Level Process Model): A Petri net $\Sigma_{TPM} = (P, T; F, M_0)$ is a top-level process model for a workflow if (1) $P = P_L \cup P_M$ where P_L represents the logic places and the P_M represents the message places exchanged between different organizations or partners; (2) $T = T_A \cup T_P$, $T_A \cap T_P = \emptyset$, where T_A represents the activities of a process, and T_P represents the abstract procedures in Σ_{TPM}; (3) $p_s \in P_L$ is the start place of Σ_{TPM} where ${}^{\bullet}p_s = \emptyset$, and $p_e \in P_L$ is the end place of Σ_{TPM} where $p_e^{\bullet} = \emptyset$; and (4) $\forall p \in P$, $M_0(p) = 1$ if $p = p_e$, $M_0(p) = 0$ otherwise.

The top-level process model Σ_{TPM} is a kind of Petri nets extended with abstract transitions, i.e. there are two kinds of transitions, one kind to represent the normal activities, and the other to represent the abstract procedures. To differ from the normal transitions, an abstract transition is represented by a double rectangle. For example, a top-level Petri net is shown in Fig. 1, in which T_1 is an abstract transition.

An abstract transition in the top-level model is just like a *black-box*, and its semantics and contents are not clear for this level, so the structure of each abstract transition should be refined. The operation to refine the content of an abstract transition is called Petri net refinement [23]–[26]. In this paper, an abstract transition will be refined by a bottom-level process model.

Definition 2 (Bottom-Level Process Model): A Petri net $\Sigma_{BPM} = (P, T; F, M_0)$ is a bottom-level process model for a workflow if (1) $P = P_L \cup P_M$ where P_L is the logic places and the P_M is the message places exchanged between different organizations or partners; (2) T represents the activities of a process; and (3) For any $p \in P$, $M_0(p) = 0$.

A bottom-level model Σ_{BPM} is different from a top-level model as it does not contain any abstract transition. Therefore, its firing rule is same as that of a standard Petri net, i.e., $\forall t \in T, \forall M \in R(M_0)$, t is enabled under M iff $\forall p \in^{\bullet} t, M(p) \geq 1$.

Definition 3 (Refinement Operation): Let $\Sigma_{TPM} = (P, T; F, M_0)$ be a top-level process model and $\Sigma_{BPM} = (P_1, T_1; F_1, M_{01})$ be a bottom-level process model. Given t ($t \in T$) is an abstract transition, t can be replaced by $\Sigma_{BPM} = (P_1, T_1; F_1, M_{01})$ if ${}^{\bullet}t = \{p | p \in P_1 \ and \ {}^{\bullet}p = \emptyset\}$ and $t^{\bullet} = \{p | p \in P_1 \ and \ p^{\bullet} = \emptyset\}$. The top-level model after refinement is $\Sigma'_{TPM} = (P', T'; F', M'_0)$, where (1) $P' = P \cup P_1$; (2) $T' = (T - \{t\}) \cup T_1$; (3) $F' = ((P' \times T') \cup (T' \times P')) \cap (F \cup F_1)$; and (4) For any $p' \in P'$, $M'_0(p') = M_0(p')$ if $p' \in P$, and $M'_0(p') = M_{01}(p')$ otherwise.

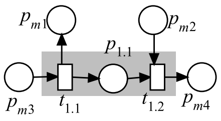


FIGURE 2. An example bottom-level process model.

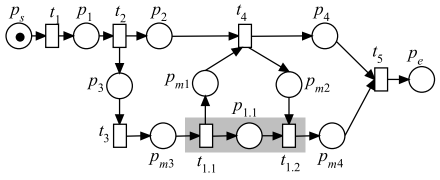


FIGURE 3. An example model after refinement.

TABLE 1. Symbols and meanings.

Symbol	Meaning
$^\bullet t$	the preset places of t
$t^\bullet$	the postset places of t
$^\circ t$	the input place set of t
t°	the output place set of t
$^\diamond t$	the read place set of t
$t^\diamond$	the write place set of t

According to Definition 3, the refinement operation aims to refine an abstract transition by a bottom-level model. The structure of a bottom-level model will replace the abstract transition and other parts in the original top-level model keep invariant. For example, a bottom-level process model is shown in Fig. 2, and we use this bottom-level process model to refine the abstract transition T_1. The top-level model after refinement is shown in Fig. 3. Obviously, the top-level model becomes a standard one after refinement.

Definition 4 (Input/Output Place and Read/Write Place): Let $\Sigma_{TPM} = (P, T; F, M_0)$ be a top-level process model, $t \in T$ be an abstract transition, and a bottom-level model $\Sigma_1 = (P_1, T_1; F_1, M_{01})$ be the refinement model of t. We have (1) The set $\{p|p \in \cup^\bullet(t'), t'$ *is a start transition of* $\Sigma_1\}$, is the input places of t, denoted by $^\circ t$; (2) The set $\{p|p \in \cup(t')^\bullet,$ t' *is an end transition of* $\Sigma_1\}$, is the output places of t, denoted by t°; (3) The set $^\bullet t -^\circ t$, denoted by $^\diamond t$, is named as the read places of t; and (4) The set $t^\bullet - t^\circ$, denoted by $t^\diamond$, is named as the write places of t.

Table 1 compares these six notations for each abstract transition. To differ the read (write) places of an abstract transition from its input (output) places, the arcs between the read (write) places and the abstract transition are drawn in broken lines. For example, in Fig. 1, T_1 is an abstract transition, and p_{m3} (p_{m4}) is the input (output) place and p_{m2} (p_{m1}) is the read (write) places of T_1. Both arcs from T_1 to p_{m1} and that from p_{m2} to T_1 are drawn in broken lines.

The firing rules of the top-level process model are defined as follows: (1) $\forall t \in T_A, \forall M \in R(M_0)$, t is enabled under M iff $\forall p \in^\bullet t, M(p) \geq 1$; and (2) $\forall t \in T_P, \forall M \in R(M_0)$,

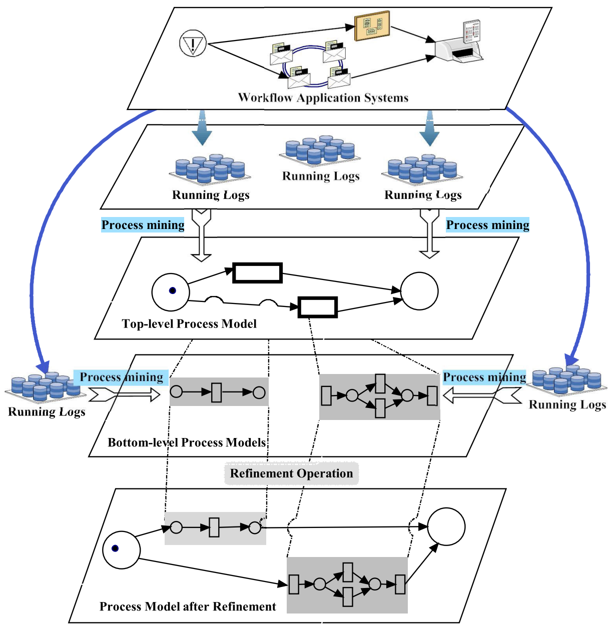


FIGURE 4. Framework for top-down process mining.

t is enabled under M iff $\forall p \in^\circ t, M(p) \geq 1$. These rules are different from that of a standard Petri net. For example, in Fig. 1, T_1 can be fired if p_{m3} has at least one token even if p_{m2} has no any token. Other properties about the top-level model such as reachability, boundedness, and etc. can be defined same as that of a standard one.

IV. FRAMEWORK FOR TOP-DOWN PROCESS MINING

In this section, the framework for top-down process mining is first proposed, and then formal definitions of the multi-source running logs are defined.

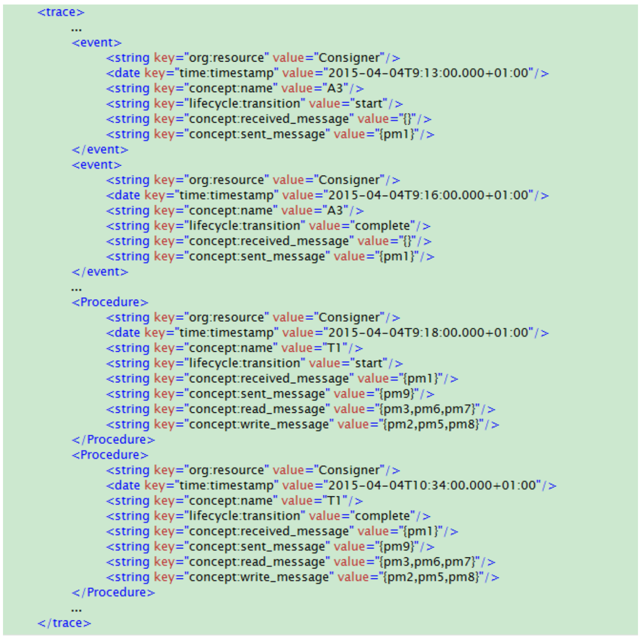
A. FRAMEWORK FOR TOP-DOWN PROCESS MINING

A framework for top-down process mining based on Petri net refinement operation is illustrated in Fig. 4, which includes four main steps:

Recording Running Logs: While a workflow system runs on several distributed servers, each server can record the running logs for each activity and store them into a log database. Such running logs collected from multi-source servers are used for our top-down process mining. An example of running logs will be presented in the following subsection.

Process Mining From Top-Level Workflow Running Logs: Using the collected running logs, our top-level process mining algorithm aims to discover the top-level process model of the workflow system. The mining results can be represented in the formalized form of Petri nets extended with abstract transitions. To protect security, the detailed contents of the abstract transition cannot be obtained in this step.

Process Mining From Bottom-Level Workflow Running Logs: Using the collected running logs, our bottom-level process mining algorithm aims to discover the detailed model for



```
<trace>
    ...
    <event>
        <string key="org:resource" value="Consigner"/>
        <date key="time:timestamp" value="2015-04-04T9:13:00.000+01:00"/>
        <string key="concept:name" value="A3"/>
        <string key="lifecycle:transition" value="start"/>
        <string key="concept:received_message" value="{}"/>
        <string key="concept:sent_message" value="{pm1}"/>
    </event>
    <event>
        <string key="org:resource" value="Consigner"/>
        <date key="time:timestamp" value="2015-04-04T9:16:00.000+01:00"/>
        <string key="concept:name" value="A3"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:received_message" value="{}"/>
        <string key="concept:sent_message" value="{pm1}"/>
    </event>
    ...
    <Procedure>
        <string key="org:resource" value="Consigner"/>
        <date key="time:timestamp" value="2015-04-04T9:18:00.000+01:00"/>
        <string key="concept:name" value="T1"/>
        <string key="lifecycle:transition" value="start"/>
        <string key="concept:received_message" value="{pm1}"/>
        <string key="concept:sent_message" value="{pm9}"/>
        <string key="concept:read_message" value="{pm3,pm6,pm7}"/>
        <string key="concept:write_message" value="{pm2,pm5,pm8}"/>
    </Procedure>
    <Procedure>
        <string key="org:resource" value="Consigner"/>
        <date key="time:timestamp" value="2015-04-04T10:34:00.000+01:00"/>
        <string key="concept:name" value="T1"/>
        <string key="lifecycle:transition" value="complete"/>
        <string key="concept:received_message" value="{pm1}"/>
        <string key="concept:sent_message" value="{pm9}"/>
        <string key="concept:read_message" value="{pm3,pm6,pm7}"/>
        <string key="concept:write_message" value="{pm2,pm5,pm8}"/>
    </Procedure>
    ...
</trace>
```

FIGURE 5. **An introduction example of the running logs.**

each abstract procedure. The obtained bottom-level model is shown in the standard form of Petri nets without any abstract transitions.

Model Integration Based on Petri Net Refinement Operation: After obtaining both the top-level process model and the bottom-level process models from the distributed running logs, Petri net refinement operation is applied to refine the abstract transitions with its corresponding bottom-level models to obtain the integrated model of the whole workflow system.

B. A MULTI-SOURCE RUNNING LOG EXAMPLE

During the execution of workflow systems, the information of each activity and abstract procedure is recorded. For example, Fig. 5 presents a screenshot segment of the running logs in the *XES* format, which is a standard format developed by the *IEEE Task Force* [45] for logging events. Its source event log data is available at [46]. The following explanations are given for the running logs.

(1) There are two running logs in the segment which records the information about one activity A_3 and one abstract procedure PA_1 (recorded as T_1 in the event log); (2) The running log of one activity records the activity ID, case ID, operator, the start time, the end time, input messages and output messages of this activity. For example, the operator of A_3 is the *Consigner*, the start time of activity A_3 is [09:13 April 04], and the end time is [09:16 April 04]. The input message record of A_3 is empty, which means that the execution of A_3 does not need any message from other partners, and its output message is p_{m1}; and (3) There are some differences between the running log of one activity and that of an abstract procedure. Obviously, the log of an abstract procedure also records the procedure ID, case ID, start time, and end time, input messages and output messages. In addition, the messages read and writen during its execution are also recorded. For example, the messages read of PA_1 are p_{m3}, p_{m6} and p_{m7}, and its write messages are p_{m2}, p_{m5} and p_{m8}, which means that during its execution PA_1 receives messages p_{m3}, p_{m6} and p_{m7} from other partners and sends messages p_{m2}, p_{m5} and p_{m8} to others.

C. FORMAL DEFINITIONS OF THE MULTI-SOURCE RUNNING LOGS

In this sub-section, we present the formal definitions of the multi-source running logs.

Definition 5 (Running Log of an Activity): A running log of an activity is a 7-tuple, $ARLog = (A_i, t_s, t_e, Operator, CaseID, InputMessage, OutputMessage)$, where (1) A_i is the name (ID) of the activity; (2) t_s is the start running time of activity A_i; (3) t_e is the end running time of activity A_i, and $t_e \geq t_s$; (4) *Operator* is the operator ID of A_i; (5) *CaseID* indicates the case which A_i runs in; (6) *InputMessage* is the input message set to execute A_i; and (7) *OutputMessage* is the output message set when finishing A_i.

For example, the formalized form of the first log of activity A_3 in Fig. 5 as (A_3, [09:13 April 04], [09:16 April 04], *Consigner*, $Case_{1122}$, Ø, $\{p_{m1}\}$). In the following discussions, we use $A_i.t_s$ and $A_i.t_e$ to represent the start and end time of activity A_i respectively, i.e., $A_3.t_s$ =[09:13 April 04] and $A_3.t_e$ =[09:16 April 04].

Definition 6 (Running Log of an Abstract Procedure): A running log of an abstract procedure is a 9-tuple, $PRLog = (PA_i, t_s, t_e, Operator, CaseID, InputMessage, OutputMessage, ReadMessage, WriteMessage)$, where (1) PA_i is the name (ID) of an abstract procedure; (2) t_s is the start running time of the abstract procedure PA_i; (3) t_e is the end running time of the abstract procedure PA_i, and $t_e \geq t_s$; (4) *Operator* is the operator ID of PA_i; (5) *CaseID* indicates the case which PA_i runs in; (6) *InputMessage* is the input message set to execute PA_i; (7) *OutputMessage* is the output message set when finishing PA_i; (8) *ReadMessage* is the read message set during the execution of PA_i; and (9) *WriteMessage* is the write message set during the execution of PA_i.

For example, the formalized form of the abstract procedure PA_1 in Fig. 5 is (PA_1, [09:18 April 04], [10:34 April 04], *Consigner*, $Case_{1122}$, $\{p_{m1}\}$, $\{p_{m3}, p_{m6}, p_{m7}\}$, $\{p_{m2}, p_{m5}, p_{m8}\}$, $\{p_{m9}\}$).

In the following, both the activity and abstract procedure are called by a joint name as the assignment, which is formalized as $ASLog = (AS_i, t_s, t_e, Operator, CaseID, RequiredMessage, SentMessage)$. It is worth noting that (1) for an activity, the *RequiredMessage* and the *SentMessage* are same as its *InputMessage* and *OutputMessage*; and (2) for an abstract procedure, we have $RequiredMessage = InputMessage \cup ReadMessage$ and $SentMessage = OutputMessage \cup WriteMessage$. For the rest of this paper, we use the term *assignment* synonymously with *activity* and *abstract procedure*.

Definition 7 (Case): A case of a set of running logs of assignments, i.e., $RCase = \{ASLog|\}$ *ASLog is a running log of an assignment*.

Definition 8 (Logs): A log is a set of cases, i.e., $RLogs = \{RCase|RCase \text{ is a running case}\}$.

Taking the running logs of *logtop.XES* in [46] as an example, case $RCase_{1122} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}, A_{12}, PA_1, PA_2, PA_3\}$ and the running logs $RLogs = \{Case_{1122}, Case_{1123}, Case_{1124}, Case_{1125}\}$.

Definition 9 (Activity Set): Let *RLogs* be the running logs of a workflow system, (1) $\forall RCase_i \in RLogs$, $ActivitySet(RCase_i) = \{A_j|\forall A_j \in RCase_i, A_j = (A_j, t_s, t_e, Operator, CaseID, InputMessage, OutputMessage)\}$ is the activity set of $RCase_i$; and (2) $ActivitySet(RLogs) = \bigcup_{RCase_i \in RLogs} ActivitySet(RCase_i)$ is the activity set of *RLogs*.

Definition 10 (Abstract Procedure Set): Let *RLogs* be the running logs of a workflow system, we have (1) $\forall RCase_i \in RLogs$, $ProcedureSet(RCase_i) = \{PA_j|\forall PA_j \in RCase_i, PA_j = (PA_j, t_s, t_e, Operator, CaseID, InputMessage, OutputMessage, ReadMessage, WriteMessage)\}$ is the abstract procedure set of $RCase_i$; and (2) $ProcedureSet(RLogs) = \bigcup_{RCase_i \in RLogs} ProcedureSet(RCase_i)$ is the abstract procedure set of *RLogs*.

Following Definitions 9-10, we can also define $AssignmentSet(RCase_i)$ as the assignment set of $RCase_i$ and $AssignmentSet(RLogs)$ of *RLogs* in the same way. Obviously, we have: (1) $AssignmentSet(RCase_i) = ActivitySet(RCase_i) \cup ProcedureSet(RCase_i)$ where $\forall RCase_i \in RLogs$, $1 \leq i \leq |RLogs|$; and (2) $AssignmentSet(RLogs) = ActivitySet(RLogs) \cup ProcedureSet(RLogs)$. Considering for example, we have $AssignmentSet(RCase_{1122}) = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}, A_{12}, PA_1, PA_2, PA_3\}$, $ActivitySet(RCase_1) = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}, A_{12}\}$ and $ProcedureSet(RCase_{1122}) = \{PA_1, PA_2, PA_3\}$.

Definition 11 (Pre-Assignments and Post-Assignments): Let *RLogs* be the running logs of a workflow system, $\forall AS_i, AS_j \in AssignmentSet(RLogs)$, AS_j is one of the post-assignments of AS_i (or AS_i is one of the pre-assignments of AS_j), denoted by $AS_i \preceq AS_j$, if $AS_i.t_e \leq AS_j.t_s$ holds in all cases of *RLogs*.

Definition 12 Direct Pre-Assignments and Direct Post-Assignments: Let *RLogs* be the running logs of a workflow system, $\forall AS_i, AS_j \in AssignmentSet(RLogs)$, AS_j is one of the direct post-assignments of AS_i (or AS_i is one of the direct pre-assignments of AS_j), denoted by $AS_i \prec AS_j$, if $AS_i \preceq AS_j$ and there is no assignment $AS_k \in AssignmentSet(RLogs)$ such that $AS_i \preceq AS_k$ and $AS_k \preceq AS_j$.

The direct pre-assignments (or post-assignments) set of AS_i is denoted by $AS_i.PreSet$ (or $AS_i.PostSet$). In the $Case_{1122}$ in [46], the post-assignments of A_3 are A_4 and PA_1, and A_4 is the direct post-activity of A_3, denoted as $A_3.PostSet = \{A_4\}$. The pre-assignments of PA_1 are A_3 and A_4, and A_4 is its direct post-activity, denoted as $PA_1.PreSet = \{A_4\}$.

In this paper, we assume that the multi-source running logs collected from those distributed servers are complete, i.e., the logs contain sufficient information to derive the model.

V. TOP-DOWN PROCESS MINING FROM MULTI-SOURCE RUNNING LOGS

In this section, top-down process mining approaches are first presented and then Petri net refinement operation is used to obtain the integrated model.

A. TOP-LEVEL PROCESS MINING

It is known that α−algorithm [11] is a classical algorithm for process mining. Unfortunately, α−algorithm cannot be applied directly to mine the process model for a top-level process model with abstract procedures and messages. Here we first present our mining approach to discover the top-level process model. Our approach is mainly composed of two functional components presented in Algorithms 1-2. Algorithm 1 obtains assignment dependency relations, and Algorithm 2 takes these relations as inputs to construct the final top-level process model. Before rendering our mining algorithms, we first define a function $PostSet(AS_i, RCase)$ to calculate the direct post-assignments of assignment AS_i in a specific running case *RCase*.

Function 1: To obtain the $AS_i.PostSet$ of AS_i in running case *RCase*.

```
Begin:
1: For each $AS_i \in AssignmentSet(RCase)$ Do
      $AS_i.PostSet \leftarrow \emptyset$;
      For each $AS_j \in AssignmentSet(RCase)$ Do
        If $AS_i.t_e < AS_j.t_s$ then
           $AS_i.PostSet \leftarrow AS_i.PostSet \cup \{AS_j\}$;
        End if
      End do
   End do
2: For each $AS_j \in AS_i.PostSet$ Do
      For each $AS_k \in AS_j.PostSet$ Do
        $AS_i.PostSet \leftarrow AS_i.PostSet - \{AS_k\}$;
      End do
   End do
3: return $AS_i.PostSet$.
End
```

The complexity of Function 1 mainly lies in its first step whose complexity is $O(|RCase|^2)$. Therefore, Function 1 has its $O(|RCase|^2)$ complexity.

Table 2 shows part of an example running logs of the top-level process model that involves two running cases, $Case_1$ and $Case_2$. Based on the running log, required messages and sent messages of each assignment can be obtained directly. Next, we propose Algorithm 1 to obtain the dependency relations between different assignments.

Theorem 1: The complexity of Algorithm 1 is $O(|Rlog| * |RCase|^3)$, where $|Rlog|$ is the number of running cases in the running logs and $|RCase|$ is the number of assignments in a running case.

TABLE 2. Part of the example running logs of the top-level process model.

Case	Assignment	Operator	Start Time	End Time	RequiredMessage	SentMessage
1	t_1	Operator1	$19:01\ May\ 04$	$19:03\ May\ 04$	$\emptyset$	$\emptyset$
1	t_2	Operator1	$19:08\ May\ 04$	$19:12\ May\ 04$	$\emptyset$	$\emptyset$
1	t_3	Operator1	$19:13\ May\ 04$	$19:16\ May\ 04$	$\emptyset$	$\{p_{m3}\}$
1	T_1	Operator2	$19:17\ May\ 04$	$19:30\ May\ 04$	$\{p_{m2}, p_{m3}\}$	$\{p_{m1}, p_{m4}\}$
1	t_4	Operator1	$19:20\ May\ 04$	$19:25\ May\ 04$	$\{p_{m1}\}$	$\{p_{m2}\}$
1	t_5	Operator1	$19:30\ May\ 04$	$19:50\ May\ 04$	$\{p_{m4}\}$	$\emptyset$
2	t_1	Operator1	$19:31\ May\ 04$	$19:33\ May\ 04$	$\emptyset$	$\emptyset$
2	t_2	Operator1	$19:38\ May\ 04$	$19:42\ May\ 04$	$\emptyset$	$\emptyset$
2	t_3	Operator1	$19:43\ May\ 04$	$19:46\ May\ 04$	$\emptyset$	$\{p_{m3}\}$
2	T_1	Operator2	$19:47\ May\ 04$	$20:20\ May\ 04$	$\{p_{m2}, p_{m3}\}$	$\{p_{m1}, p_{m4}\}$
2	t_4	Operator1	$20:00\ May\ 04$	$20:05\ May\ 04$	$\{p_{m1}\}$	$\{p_{m2}\}$
2	t_5	Operator1	$20:40\ May\ 04$	$20:50\ May\ 04$	$\{p_{m4}\}$	$\emptyset$

Algorithm 1 To Obtain the $AS_i.PreSet$ and $AS_i.PostSet$ of Each Assignment AS_i in the Running Logs

```
Input: $RLogs$
Output: $(AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage, AS_i.SentMessage)$.
1: For each $AS_i \in AssignmentSet(RCase)$ Do
       $(1.1) AS_i.PreSet \leftarrow \varnothing$;
       $(1.2) AS_i.PostSet \leftarrow \varnothing$;
   End do
2: For each $AS_i \in AssignmentSet(RCase)$ Do
       $AS_i.PostSet \leftarrow AS_i.PostSet \cup \bigcap_{1 \le j \le |RLog|} PostSet(AS_i, RCase_j)$;
3: For each $AS_i \in AssignmentSet(RCase)$ Do
       For each $AS_j \in AssignmentSet(RCase)$ Do
           $AS_i.PreSet \leftarrow AS_i.PreSet \cup \{AS_j | AS_i \in AS_j.PostSet\}$;
       End do
   End do
4: return $(AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage, AS_i.SentMessage)$.
```

Proof: The complexity of the function ***PostSet*** is $O(|RCase|^2)$, thereby complexity of Step 2 is $O(|Rlog| * |RCase|^3)$. Because the complexity of Algorithm 1 is mainly determined by its second step, the complexity of Algorithm 1 is $O(|Rlog| * |RCase|^3)$.

Take the example running logs in Table 2 as an example. By executing Algorithm 1, the *Pre-Set*, *Post-set*, *ReceivedMessage* and *SentMessage* are shown in Table 3. Based on the assignment dependency relations in Table 3, we present Algorithm 2 to construct the top-level process model.

Theorem 2: The complexity of Algorithm 2 is $O(|RCase|^2)$ where $|RCase|$ is the number of assignments in a running case.

Proof: The complexity of Algorithm 2 is mainly determined by Step 3 whose complexity is $O(|RCase|^2)$. Therefore, Algorithm 2 has a $O(|RCase|^2)$ complexity where $|RCase|$ is the number of assignments.

As the input of Algorithm 2 contains both abstract procedures and normal activities, thereby its mining result is a Petri net extended with abstract transitions, i.e. a top-level process model as defined in Definition 1. Take the assignment dependency relations in Table 3 as an example. By executing Algorithm 2, the model mined for the top-level example process is shown in Fig. 1, satisfying: (1) there are 5 activities and 1 abstract procedure that are represented by transition t_i ($i = 1, 2, \cdots, 5$) and abstract transition T_1 respectively; and (2) the detailed process of the abstract procedure cannot be obtained at this stage. Therefore, we need to mine its corresponding bottom-level models from its running logs to refine it.

TABLE 3. Pre-set and post-set of each assignment in the top-level example process model.

Assign.	Pre-Set	Post-Set	RequiredM.	SentM.
t_1	$\emptyset$	$\{t_2\}$	$\emptyset$	$\emptyset$
t_2	$\{t_1\}$	$\{t_3\}$	$\emptyset$	$\emptyset$
t_3	$\{t_2\}$	$\{T_1\}$	$\emptyset$	$\{p_{m3}\}$
T_1	$\{t_3\}$	$\{t_5\}$	$\{p_{m2}, p_{m3}\}$	$\{p_{m1}, p_{m4}\}$
t_4	$\{t_2\}$	$\{t_5\}$	$\{p_{m1}\}$	$\{p_{m2}\}$
t_5	$\{t_4, T_1\}$	$\emptyset$	$\{p_{m4}\}$	$\emptyset$

B. BOTTOM-LEVEL PROCESS MINING

To refine abstract procedures in a top-level process model, bottom-level process models are needed. Then Algorithm 3 is proposed to mine the bottom-level model from its corresponding running logs.

Theorem 3: The complexity of Algorithm 3 is $O(|RCase|^2)$ where $|RCase|$ is the number of activities in a running case.

Proof: The complexity of Algorithm 3 is mainly determined by Step 2 whose complexity is $O(|RCase|^2)$. Thus, the complexity of Algorithm 3 is $O(|RCase|^2)$ where $|RCase|$ is the number of activities.

As the input of Algorithm 3 contains only normal activities, thereby its mining result is a traditional Petri net, i.e. a bottom-level process model as defined in Definition 2. Take the running logs in Table 4 as an example. By executing Algorithm 1, the *Pre-Set*, *Post-set*, *ReceivedMessage* and *SentMessage* of each assignment are shown in Table 5. Then, by executing Algorithm 3, we can obtain the bottom-level process model as shown in Fig. 2. Then, how to integrate the

Algorithm 2 To Obtain the Top-Level Process Model Σ_{TPM}

```
Input: $\{(AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage,$
    $AS_i.SentMessage) | 1 \le i \le |RCase|, RCase \in RLog\}$.
Output: $\Sigma_{TPM} = (P, T; F, M_0)$.
1: $P \leftarrow \emptyset, P_L \leftarrow \emptyset, P_M \leftarrow \emptyset, T \leftarrow \emptyset, T_A \leftarrow \emptyset, T_{PA} \leftarrow \emptyset,$
   $F \leftarrow \emptyset$, and $M_0 \leftarrow \emptyset$.
2: For each $AS_i \in AssignmentSet(RCase)$ Do
     If $AS_i \in ActivitySet(RCase)$ then
       $T_A \leftarrow T_A \cup \{AS_i\}$;
     else if $AS_i \in ProcedureSet(RCase)$
       $T_{PA} \leftarrow T_{PA} \cup \{AS_i\}$;
     End if
   End do
   $T \leftarrow T_A \cup T_{PA}$;
3: For each $AS_i, AS_j \in T$ Do
     If $AS_j \in AS_i.PostSet$ then
       (3.1) $P_L \leftarrow P_L \cup \{p_{ij}\}$;
       (3.2) $F \leftarrow F \cup \{(AS_i, p_{ij}), (p_{ij}, AS_j)\}$;
     End if
   End do
4: For each $AS_i \in T$ Do
     If $AS_i.ReceivedMessage \neq \emptyset$ then
       For each $mi \in AS_i.ReceivedMessage$ Do
         (4.1) $P_M \leftarrow P_M \cup \{p_{mi}\}$;
         (4.2) $F \leftarrow F \cup \{(p_{mi}, AS_i)\}$;
       End do
     End if
   End do
5: For each $AS_i \in T$ Do
     If $AS_i.SentMessage \neq \emptyset$ then
       For each $mi \in AS_i.SentMessage$ Do
         (5.1) $P_M \leftarrow P_M \cup \{p_{mi}\}$;
         (5.2) $F \leftarrow F \cup \{(AS_i, p_{mi})\}$;
       End do
     End if
   End do
6: For each $AS_i \in T$ Do
     If $AS_i.PreSet == \emptyset$ then
       (6.1) $P_L \leftarrow P_L \cup \{p_s\}$;
       (6.2) $F \leftarrow F \cup \{(p_s, AS_i)\}$;
     Else if $AS_i.PreSet == \emptyset$ then
       (6.3) $P_L \leftarrow P_L \cup \{p_e\}$;
       (6.4) $F \leftarrow F \cup \{(AS_i, p_e)\}$;
     End if
   End do
7: For each $p \in P$ Do
     If $p == p_s$ then
       $M_0(p) \leftarrow 1$;
     Else
       $M_0(p) \leftarrow 0$;
     End if
   End do
8: return $\Sigma_{TPM} = (P, T; F, M_0)$.
```

Algorithm 3 To Obtain the Bottom-Level Process Model Σ_{BPM}

```
Input: $\{(AS_i, AS_i.PreSet, AS_i.PostSet, AS_i.ReceivedMessage,$
    $AS_i.SentMessage) | 1 \le i \le |RCase|, RCase \in RLog\}$.
Output: $\Sigma_{BPM} = (P, T; F, M_0)$.
1: $P \leftarrow \emptyset$, $P_L \leftarrow \emptyset$, $P_M \leftarrow \emptyset$, $T \leftarrow$
   $AssignmentSet(RCase)$, $F \leftarrow \emptyset$,and $M_0 \leftarrow \emptyset$.
2: For each $AS_i, AS_j \in T$ Do
     If $AS_j \in AS_i.PostSet$ then
       (2.1) $P_L \leftarrow P_L \cup \{p_{ij}\}$;
       (2.2) $F \leftarrow F \cup \{(AS_i, p_{ij}), (p_{ij}, AS_j)\}$;
     End if
   End do
3: For each $AS_i \in T$ Do
     If $AS_i.ReceivedMessage \neq \emptyset$ then
       For each $mi \in AS_i.ReceivedMessage$ Do
         (3.1) $P_M \leftarrow P_M \cup \{p_{mi}\}$;
         (3.2) $F \leftarrow F \cup \{(p_{mi}, AS_i)\}$;
       End do
     End if
   End do
4: For each $AS_i \in T$ Do
     If $AS_i.SentMessage \neq \emptyset$ then
       For each $mi \in AS_i.SentMessage$ Do
         (4.1) $P_M \leftarrow P_M \cup \{p_{mi}\}$;
         (4.2) $F \leftarrow F \cup \{(AS_i, p_{mi})\}$;
       End do
     End if
   End do
5: $P \leftarrow P_L \cup P_M$.
6: return $\Sigma_{BPM} = (P, T; F, M_0)$.
```

bottom-level process models with the top-level process model will be discussed in the following.

C. PETRI NET REFINEMENT FOR PROCESS INTEGRATION

Process mining technology is used to separately discover the top-level and bottom-level models. Then how to integrate them to obtain the integrated model is our main concern. Petri net refinement operation as defined in Definition 3 is used. With the refinement operation, one abstract transition in the top-level process model can be refined by its corresponding bottom-level model. Next, we present Algorithm 4 to conduct the refinement operation.

Theorem 4: The complexity of Algorithm 4 is $O(|T_{PA}|^2)$ where$|T_{PA}|$ is the number of abstract procedures in a running case.

Proof: The complexity of Algorithm 4 is mainly determined by Step 3 whose complexity is $O(|T_{PA}|^2)$. Therefore, the complexity of Algorithm 4 is $O(|T_{PA}|^2)$ where $|T_{PA}|$ is the number of abstract procedures.

TABLE 4. Part of the running logs of the example abstract procedure.

Case	Assignment	Operator	Start Time	End Time	RequiredMessage	SentMessage
1	$t_{1.1}$	Operator2	$19:18\ May04$	$19:26\ May04$	$\{p_{m3}\}$	$\{p_{m1}\}$
1	$t_{1.2}$	Operator2	$19:27\ May04$	$19:30\ May04$	$\{p_{m2}\}$	$\{p_{m4}\}$
2	$t_{1.1}$	Operator2	$19:48\ May04$	$19:56\ May04$	$\{p_{m3}\}$	$\{p_{m1}\}$
2	$t_{1.2}$	Operator2	$19:12\ May04$	$19:18\ May04$	$\{p_{m2}\}$	$\{p_{m4}\}$

Algorithm 4 To Refine a Top-Level Process Model Σ_{TPM} Using a Set of Bottom-Level Process Models Σ_{BPMi}

```
Input: $\Sigma_{TPM} = (P, T; F, M_0)$ and $\Theta = \{\Sigma_{BPMi} = (P_i, T_i; F_i, M_{0i}) | 1 \le i \le |T_P|\}$.
Output: $\Sigma'_{TPM} = (P', T'; F', M'_0)$.
1: $P' \leftarrow P, T' \leftarrow T, F' \leftarrow F$,and $M'_0 \leftarrow M_0$.
2: For each $\Sigma_{BPMi} \in \Theta$ Do
      (2.1) $In(\Sigma_{BPMi}) \leftarrow \emptyset$; /*$In(\Sigma_{BPMi})$ is the input place set of $\Sigma_{BPMi}$*/
      (2.2) $Out(\Sigma_{BPMi}) \leftarrow \emptyset$; /*$Out(\Sigma_{BPMi})$ is the output place set of $\Sigma_{BPMi}$*/
      For each $p_i \in P$ Do
         If $^\bullet p_i == \emptyset$ then
            $In(\Sigma_{BPMi}) \leftarrow In(\Sigma_{BPMi}) \cup p_i$;
         Else if $p_i^\bullet == \emptyset$ then
            $Out(\Sigma_{BPMi}) \leftarrow Out(\Sigma_{BPMi}) \cup p_i$;
         End if
      End do
3: For each $t \in T_{PA}$ Do
      For each $\Sigma_{BPMi} \in \Theta$ Do
         If $(^\bullet t == In(\Sigma_{BPMi})) \wedge (t^\bullet == Out(\Sigma_{BPMi}))$ then
            (3.1) $P' \leftarrow P' \cup P_i$;
            (3.2) $T' \leftarrow (T' - t) \cup T_i$;
            (3.3) $F' \leftarrow (F' \cup F_i) \cap ((P' \times T') \cup (T' \times P'))$;
         End if
      End do
   End do
4: return $\Sigma'_{TPM} = (P', T'; F', M'_0)$.
```

As all abstract transitions in the top-level process model is refined with its corresponding bottom-level process models, therefore no abstract transition is involved in the refined process model, i.e. it is a traditional Petri net. By executing Algorithm 4, the integrated model is shown in Fig. 3.

VI. RUNNING CASE AND EXPERIMENTAL VERIFICATION

In this section, a multi-modal transportation business process is used as a typical case to illustrate our top-down process mining approaches.

A. A MULTI-MODAL TRANSPORTATION BUSINESS PROCESS CASE

For security and privacy sake, the inner data of a partner will be stored in its own database and will not be accessed by others. To satisfy the requirements of security, a two-level system architecture is required.

To realize this two-level architecture, four distributed database servers will be used to record the system running logs. (1) The top-level architecture is an abstraction of the whole multi-modal transportation business process. The running logs of this level will be stored in a single database server; (2) The transportation preparation procedure consists of transportation planning, goods preparation with the sender and the payment processes with the carrier and the shipper. The running logs of this procedure will be stored in a database server owned by the consigner; (3) The carrier transportation procedure includes activities such as booking acceptance, goods loading, issue waybill and payment. The running logs of this procedure will be stored into the database server kept by the carrier; and (4) The shipper transportation procedure includes activities such as booking acceptance, shipper inventory, terminal receipt, payment, delivery, goods arrival and the interaction with the wharfinger. The running logs of this procedure will be store in a database server within the shipper.

TABLE 5. Pre-set and post-set of each assignment in the bottom-level example process model.

Assign.	Pre-Set	Post-Set	RequiredM.	SentM.
t_1	$\emptyset$	$\{t_2\}$	$\{p_{m3}\}$	$\{p_{m1}\}$
t_2	$\{t_1\}$	$\emptyset$	$\{p_{m2}\}$	$\{p_{m4}\}$

TABLE 6. Dataset and objectives of experiments.

Experiment	Dataset	Objective
1	logtop.xml in [46]	To verify Algorithms 1-2
2	logbottom1-3.xml in [46]	To verify Algorithm 3
3	———————————-	To verify Algorithm 4

B. EXPERIMENTAL VERIFICATION

In this subsection, the mining methods will be validated using the following experiments whose dataset [46] and objectives are concluded in Table 6.

Experiment 1: Table 7 shows part of the running logs of the top-level architecture that involves one running case, $Case_1$. According to Table 7, required messages and sent messages of each assignment can be obtained directly. Taking these running logs as input and execute Algorithm 1, the *Pre-Set*, *Post-set*, *ReceivedMessage* and *SentMessage* of each assignment are shown in Table 8. By executing Algorithm 2 which takes Table 8 as input, the top-level process model of multi-modal

TABLE 7. Part of the running logs of the top-level process model.

Case	Assignment	Operator	Start Time	End Time	RequiredMessage	SentMessage
1122	A_1	Sender	$09:01\ April04$	$09:03April04$	$\emptyset$	$\emptyset$
1122	A_2	Consigner	$09:08\ April04$	$09:12\ April04$	$\emptyset$	$\emptyset$
1122	A_3	Consigner	$09:13\ April04$	$09:16\ April04$	$\emptyset$	$\{p_{m1}\}$
1122	A_4	Sender	$09:17\ April04$	$09:20\ April04$	$\emptyset$	$\emptyset$
1122	T_1	Consigner	$09:18\ April04$	$10:34\ April04$	$\{p_{m1}, p_{m3}, p_{m6}, p_{m7}\}$	$\{p_{m2}, p_{m4}, p_{m5}, p_{m8}, p_{m9}\}$
1122	T_2	Carrier	$09:22\ April04$	$10:00\ April06$	$\{p_{m5}, p_{m8}, p_{m11}\}$	$\{p_{m7}, p_{m10}, p_{m12}\}$
1122	T_3	Shipper	$09:24\ April04$	$10:08\ April10$	$\{p_{m4}, p_{m12}, p_{m14}\}$	$\{p_{m6}, p_{m13}, p_{m15}\}$
1122	A_5	Sender	$09:30\ April04$	$09:50\ April04$	$\{p_{m2}\}$	$\{p_{m3}\}$
1122	A_6	Consigner	$12:48\ April04$	$12:56\ April04$	$\{p_{m9}, p_{m10}\}$	$\{p_{m11}\}$
1122	A_7	Consigner	$14:06\ April06$	$14:08\ April06$	$\{p_{m13}\}$	$\{p_{m14}\}$
1122	A_8	Consigner	$14:10\ April06$	$14:16\ April06$	$\emptyset$	$\emptyset$
1122	A_9	Sender	$14:20\ April06$	$14:25\ April06$	$\emptyset$	$\emptyset$
1122	A_{10}	Consigner	$15:10\ April06$	$15:20\ April06$	$\emptyset$	$\emptyset$
1122	A_{11}	Buyer	$15:40\ April06$	$15:46\ April06$	$\emptyset$	$\emptyset$
1122	A_{12}	Buyer	$10:10\ April10$	$10:20\ April10$	$\{p_{m15}\}$	$\emptyset$

TABLE 8. Pre-set and post-set of each assignment in the top-level process model.

Assign.	Meaning	Pre-Set	Post-Set	RequiredMessage	SentMessage
A_1	Apply Transportation	$\emptyset$	$\{A_2, A_4\}$	$\emptyset$	$\emptyset$
A_2	Accept Application	$\{A_1\}$	$\{A_3\}$	$\emptyset$	$\emptyset$
A_3	Generate Contract	$\{A_2\}$	$\{T_1, A_4\}$	$\emptyset$	$\{p_{m1}\}$
A_4	Assign Contract	$\{A_3\}$	$\{A_5\}$	$\emptyset$	$\emptyset$
T_1	Transportation preparation procedure	$\{A_3\}$	$\{T_2, T_3, A_6\}$	$\{p_{m1}, p_{m3}, p_{m6}, p_{m7}\}$	$\{p_{m2}, p_{m4}, p_{m5}, p_{m8}, p_{m9}\}$
T_2	Carrier transportation procedure	$\{T_1\}$	$\{A_6\}$	$\{p_{m5}, p_{m8}, p_{m11}\}$	$\{p_{m7}, p_{m10}, p_{m12}\}$
T_3	Shipper transportation procedure	$\{T_1\}$	$\{A_7, A_{12}\}$	$\{p_{m4}, p_{m12}, p_{m14}\}$	$\{p_{m6}, p_{m13}, p_{m15}\}$
A_5	Prepare Goods	$\{A_4\}$	$\{A_9\}$	$\{p_{m2}\}$	$\{p_{m3}\}$
A_6	Pay the Carrier	$\{T_1, T_2\}$	$\{A_7\}$	$\{p_{m9}, p_{m10}\}$	$\{p_{m11}\}$
A_7	Pay the Shipper	$\{T_3, A_6\}$	$\{A_8\}$	$\{p_{m13}\}$	$\{p_{m14}\}$
A_8	Generate Payment Receipt	$\{A_7\}$	$\{A_9\}$	$\emptyset$	$\emptyset$
A_9	Send pays the bill	$\{A_5, A_8\}$	$\{A_{10}\}$	$\emptyset$	$\emptyset$
A_{10}	Generate Delivery Order	$\{A_8, A_9\}$	$\{A_{11}\}$	$\emptyset$	$\emptyset$
A_{11}	Accept Release Form	$\{A_{10}\}$	$\{A_{12}\}$	$\emptyset$	$\emptyset$
A_{12}	Release Goods	$\{T_3, A_{11}\}$	$\emptyset$	$\{p_{m15}\}$	$\emptyset$

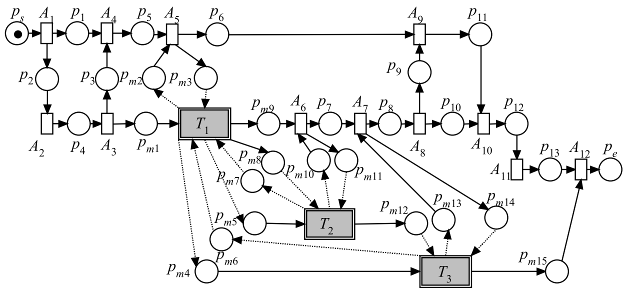


FIGURE 6. Top-level model of the multi-modal transportation business process.

transportation business process is shown in Fig. 6, and the meanings of message places are given in Table 9.

The result of our top-level mining is a top-level process model with abstract transitions that are represented by transition A_i $(i = 1, 2, \cdots, 12)$ and abstract transitions T_j $(j = 1, 2, 3)$. The detailed process about these three abstract procedures cannot be obtained at this stage. Therefore, we need to conduct *Experiment* 2, i.e. mine the bottom-level models of these three abstract procedures from their respective running logs.

TABLE 9. Meaning of each message.

Message	Meaning
p_{m1}	transportation contract
p_{m2}	packing notice
p_{m3}	goods preparation notice
p_{m4}	booking request to shipper
p_{m5}	booking request to carrier
p_{m6}	acceptance notice of shipper
p_{m7}	acceptance notice of carrier
p_{m8}	transportation notice to carrier
p_{m9}	payment notice of carrier
p_{m10}	bill form
p_{m11}	payment verification to carrier
p_{m12}	goods arrival notice to shipper
p_{m13}	payment notice of shipper
p_{m14}	payment verification to shipper
p_{m15}	goods arrival notice to buyer

Experiment 2: Part of the running logs of the transportation preparation procedure, the carrier transportation procedure and the shipper transportation procedure are shown in

TABLE 10. Part of the running logs of the transportation preparation procedure.

Case	Assignment	Operator	Start Time	End Time	RequiredMessage	SentMessage
1122	$A_{1.1}$	Consigner	$09:18\ April04$	$09:21\ April04$	$\{p_{m1}\}$	$\{p_{m4}, p_{m5}\}$
1122	$A_{1.2}$	Consigner	$09:27\ April04$	$09:29\ April04$	$\{p_{m6}, p_{m7}\}$	$\{p_{m2}\}$
1122	$A_{1.3}$	Consigner	$09:30\ April04$	$09:45\ April04$	$\{p_{m3}\}$	$\emptyset$
1122	$A_{1.4}$	Consigner	$09:51\ April04$	$10:02\ April04$	$\emptyset$	$\emptyset$
1122	$T_{1.5}$	Consigner	$10:03\ April04$	$10:05\ April04$	$\emptyset$	$\emptyset$
1122	$T_{1.6}$	Consigner	$10:06\ April04$	$10:10\ April04$	$\emptyset$	$\emptyset$
1122	$A_{1.7}$	Consigner	$10:07\ April04$	$10:30\ April04$	$\emptyset$	$\emptyset$
1122	$A_{1.8}$	Consigner	$10:32\ April04$	$10:34\ April04$	$\emptyset$	$\{p_{m8}, p_{m9}\}$

TABLE 11. Part of the running logs of the carrier transportation procedure.

Case	Assignment	Operator	Start Time	End Time	RequiredMessage	SentMessage
1122	$A_{2.1}$	Carrier	$09:22\ April04$	$09:24\ April04$	$\{p_{m5}\}$	$\{p_{m7}\}$
1122	$A_{2.2}$	Carrier	$10:36\ April04$	$12:30\ April04$	$\{p_{m8}\}$	$\emptyset$
1122	$A_{2.3}$	Carrier	$12:40\ April04$	$12:45\ April04$	$\emptyset$	$\{p_{m10}\}$
1122	$A_{2.4}$	Carrier	$13:00\ April04$	$10:00\ April06$	$\emptyset$	$\{p_{m11}, p_{m12}\}$

TABLE 12. Part of the running logs of the shipper transportation procedure.

Case	Assign.	Operator	Start Time	End Time	RequiredMessage	SentMessage
1122	$A_{3.1}$	Shipper	$09:24\ April04$	$09:26\ April04$	$\{p_{m4}\}$	$\{p_{m6}\}$
1122	$A_{3.2}$	Shipper	$09:30\ April04$	$14:00\ April04$	$\emptyset$	$\emptyset$
1122	$A_{3.3}$	Wharfinger	$10:05\ April06$	$12:30\ April06$	$\{p_{m12}\}$	$\emptyset$
1122	$A_{3.4}$	Shipper	$13:00\ April06$	$13:30\ April06$	$\emptyset$	$\emptyset$
1122	$T_{3.5}$	Wharfinger	$13:10\ April06$	$13:25\ April06$	$\emptyset$	$\emptyset$
1122	$T_{3.6}$	Wharfinger	$13:35\ April06$	$13:50\ April06$	$\emptyset$	$\emptyset$
1122	$A_{3.7}$	Shipper	$14:00\ April06$	$14:05\ April06$	$\emptyset$	$\{p_{m13}\}$
1122	$A_{3.8}$	Shipper	$14:10\ April06$	$10:00\ April10$	$\{p_{m14}\}$	$\emptyset$
1122	$A_{3.9}$	Shipper	$10:02\ April10$	$10:08\ April10$	$\emptyset$	$\{p_{m15}\}$

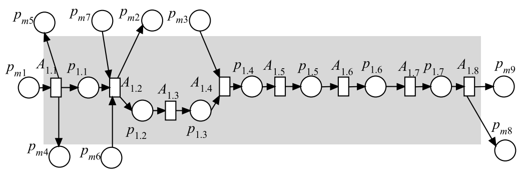


FIGURE 7. Process model of the transportation preparation procedure.

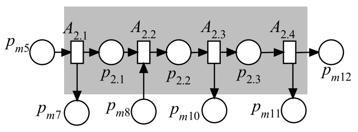


FIGURE 8. Process model of the carrier transportation procedure.

Tables 10-12. First, we consider Table 10, the *Pre-Set*, *Post-set*, *ReceivedMessage* and *SentMessage* of each assignment are shown in Table 13 with Algorithm 1. Then, by executing Algorithm 3 we can obtain the bottom-level process model for the transportation preparation procedure (T_1) as shown in Fig. 7 where eight activities represented as $A_{1.i}$ ($i = 1, 2, \cdots, 8$) are involved.

Finally, we consider the shipper transportation procedure whose running log segment is shown in Table 12. By executing Algorithm 1, the dependency relations between each

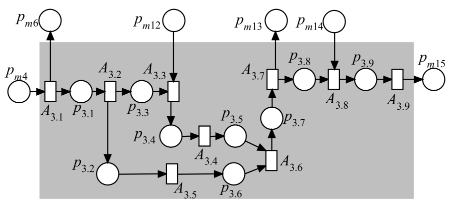


FIGURE 9. Process model of the shipper transportation procedure.

assignment are shown in Table 15. And then taking the these dependency relations as inputs to run Algorithm 3, we can obtain the bottom-level process model for the shipper transportation procedure (T_3) as shown in Fig. 9 where nine activities represented as $A_{3.i}$ ($i = 1, 2, \cdots, 9$) are involved.

Next, we consider the running logs in Table 11. By executing Algorithm 1, the dependency relations between each assignment are shown in Table 14. And then taking the these dependency relations as inputs to run Algorithm 3, we can obtain the bottom-level process model for the carrier transportation procedure (T_2) as shown in Fig. 8 where four activities represented as $A_{2.i}$ ($i = 1, 2, 3, 4$) are involved.

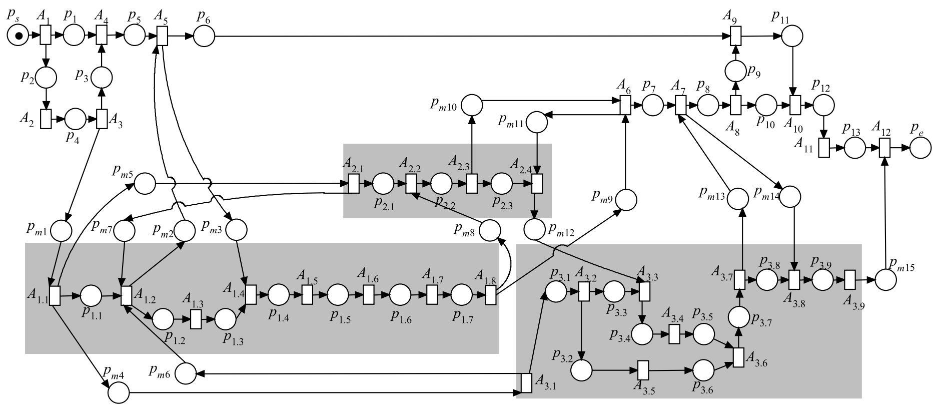


FIGURE 10. Refinement model for the multi-modal transportation business process.

Experiment 3: The bottom-level process models in Figs. 7-9 are correspond with the three abstract procedures in Fig. 6. Then the abstract transitions T_1, T_2 and T_2 can be refined by the models in Figs. 7-9 using Algorithm 4. The refined multi-modal transportation business process model is shown in Fig. 10. Because no abstract transition is involved in the refined model, we argue that the integration result of Algorithm 4 is a traditional Petri net.

C. EXPERIMENTAL RESULT ANALYSIS

According to integrated model shown in Fig. 10, the typical scenario of this multi-modal transportation business process is described as follows/ There are seven roles in the process including the sender, consigner, carrier, shipper, and buyer in the main transportation process. Besides, the consigner should have customer service declaration before delivering the goods to the carrier, especially for overseas business. And, the shipper should transport goods to wharfinger for long-term storage. Therefore, another two roles, customer service center and wharfinger, are also involved. This typical multi-modal transportation business process scenario includes the following steps: (1) To start a transportation, the sender should first apply for a transportation task from the consigner; (2) After accepting a transportation application from the sender, the consigner will generate a transportation contract; (3) Then the sender signs the contract together; (4) Receiving the contract assigned by the sender, the consigner will send a booking request to the carrier partner and the shipper partner, respectively; (5) If the booking requests are both accepted, the consigner will prepare containers and packing notice is sent to the sender to prepare the transportation goods; (6) When the goods arrive, the consigner pack the goods and make the transportation declaration. The customer service center will audit and then release the legitimate goods to load by the carrier; (7) The carrier loads the transportation goods and then issues its waybill to the consigner; (8) According to the waybill, the consigner will give the payment to the carrier; (9) After obtaining the payment, the goods will be delivered and be transferred to the shipper by the carrier; (10) After receiving the booking request from the consigner, the shipper partner will inform the wharfinger to tally the goods and prepare shipping; (11) When the shipper receives the goods transferred from the carrier, the wharfinger will inform the container entrance to prepare shipping; (12) The terminal receipt will be sent to the consigner to obtain the shipping payment; (13) After receiving the payment, the goods will be delivered by the shipper; (14) Next the sender will be informed to pay for the transportation goods after payment by the consigner to the carrier and shipper; (15) After payment, the consigner will generate a delivery order and send a release form to the buyer; and (16) When the goods arrival, the buyer will pick up them with the release form.

As there are lots of messages exchanged between different organizations or partners, the business logic is really complex. As a consequence, to directly construct the model for such a complicated cross-organizational business process is obviously a time-consuming and error-prone process. Fortunately, our top-down process mining approaches proposed such a method to discover the complex business process model from its system running logs.

D. QUALITY METRICS EVALUATION

Process mining algorithms typically aim to discover a process model from event log that describe the recorded behavior. In this sub-section, we measure the quality of the discovered process model with our proposed method. Usually, the quality of a process discovery approach is measured by the following three quality dimensions:

- *Fitness* quantifies the extent to which a discovered model can accurately reproduce the cases recorded in the log.
- *Precision* quantifies the fraction of the behavior allowed by the model which is not seen in the event log.
- *Generalization* assesses the extent to which the resulting model will be able to reproduce future behavior of the process.

TABLE 13. Pre-set and post-set of each assignment in the transportation preparation procedure.

Assign.	Meaning	Pre-set	Post-set	RequiredMessage	SentMessage
$A_{1.1}$	Request Booking	$\emptyset$	$\{A_{1.2}\}$	$\{p_{m1}\}$	$\{p_{m4}, p_{m4}\}$
$A_{1.2}$	Pack Notice	$\{A_{1.1}\}$	$\{A_{1.3}\}$	$\{p_{m6}, p_{m7}\}$	$\{p_{m2}\}$
$A_{1.3}$	Prepare Packing	$\{A_{1.2}\}$	$\{A_{1.4}\}$	$\{p_{m3}\}$	$\emptyset$
$A_{1.4}$	Pack Container	$\{A_{1.3}\}$	$\{A_{1.5}\}$	$\emptyset$	$\emptyset$
$A_{1.5}$	Make Declaration	$\{A_{1.4}\}$	$\{A_{1.6}\}$	$\emptyset$	$\emptyset$
$A_{1.6}$	Auditing	$\{A_{1.5}\}$	$\{A_{1.7}\}$	$\emptyset$	$\emptyset$
$A_{1.7}$	Release Goods	$\{A_{1.6}\}$	$\{A_{1.8}\}$	$\emptyset$	$\emptyset$
$A_{1.8}$	Forward Transportation	$\{A_{1.7}\}$	$\emptyset$	$\emptyset$	$\{p_{m8}, p_{m9}\}$

TABLE 14. Pre-set and post-set of each assignment in the carrier transportation procedure.

Assign.	Meaning	Pre-set	Post-set	RequiredMessage	SentMessage
$A_{2.1}$	Accept Booking	$\emptyset$	$\{A_{2.2}\}$	$\{p_{m5}\}$	$\{p_{m7}\}$
$A_{2.2}$	Load the Goods	$\{A_{2.1}\}$	$\{A_{2.3}\}$	$\{p_{m8}\}$	$\emptyset$
$A_{2.3}$	Issue Waybill	$\{A_{2.2}\}$	$\{A_{2.4}\}$	$\emptyset$	$\{p_{m10}\}$
$A_{2.4}$	Deliver Container	$\{A_{2.3}\}$	$\emptyset$	$\emptyset$	$\{p_{m11}, p_{m12}\}$

TABLE 15. Pre-set and post-set of each assignment in the shipper transportation procedure.

Assign.	Meaning	Pre-set	Post-set	RequiredMessage	SentMessage
$A_{3.1}$	Accept Booking	$\emptyset$		$\{p_{m4}\}$	$\{p_{m6}\}$
$A_{3.2}$	Invent Shipper	$\{A_{3.1}\}$	$\{A_{3.3}, A_{3.5}\}$	$\emptyset$	$\emptyset$
$A_{3.3}$	Receive Goods from Carrier	$\{A_{3.2}\}$	$\{A_{3.4}\}$	$\{p_{m12}\}$	$\emptyset$
$A_{3.4}$	Tally Goods	$\{A_{3.3}\}$	$\{A_{3.6}\}$	$\emptyset$	$\emptyset$
$A_{3.5}$	Container Entrance	$\{A_{3.2}\}$	$\{A_{3.6}\}$	$\emptyset$	$\emptyset$
$A_{3.6}$	Prepare Shipping	$\{A_{3.4}, A_{3.5}\}$	$\{A_{3.7}\}$	$\emptyset$	$\emptyset$
$A_{3.7}$	Terminal Receipt	$\{A_{3.6}\}$	$\{A_{3.8}\}$	$\emptyset$	$\{p_{m13}\}$
$A_{3.8}$	Deliver Shipping	$\{A_{3.7}\}$	$\{A_{3.9}\}$	$\{p_{m14}\}$	$\emptyset$
$A_{3.9}$	Goods Arrival	$\{A_{3.8}\}$	$\emptyset$	$\emptyset$	$\{p_{m15}\}$

Besides the quality metric evaluations of the diovered process model, we also compare the discovered process models using the proposed approach with some related works, i.e., *Alpha Miner* [11], *ILP (language-based region) Miner* [47], *Inductive Miner* [21]), in terms of different quality metrics. Some of the comparison results and explanations are demonstrated in the following. Generally speaking, our experiment is conducted based on the open source process mining toolkit, ProM, developed by the AIS group of TU/e. It contains the following consecutive steps:

Step 1: As these existing process discovery approaches cannot handle the distributed event logs, we first merge them into an integrated data set using the plug-in ''Merge Event Logs'' [48] by configuring the merge attribute as *Case Id*.

Step 2: By taking the merged event log as input, we apply *Alpha Miner*, *ILP Miner*, *Inductive Miner* to discover their respective process models.

Step 3: Using these discovered process models and the merged event log, we run *Replay a Log on Petri net for conformance Analysis* plugin [49] to measure the replay fitness. Similarly, the precision and generalization metrics are evaluated using *Measure Precision and Generalization* plugins. The evaluation results is shown in Table 16.

According to Table 16, the typical algorithms guaranteeing perfect replay fitness are *ILP miner* and our top-down miner.

TABLE 16. Quality comparison of process models using different miners.

Discovery Approach	Fitness	Precision	Generalization
AlphaMiner	0.93	0.98	0.75
ILPMiner	1.00	0.81	0.63
InductiveMiner	0.99	0.62	0.84
$Top-DownMiner$	1.00	0.92	0.82

However, the generalization of the *ILP miner* is relatively low. The precision of the *Alpha miner* is the highest while it tends to be less general. Moreover, while the Inductive miner can guarantee a high fitness and generalization its precision is poor. In a nutshell, our proposed mining approach allow for more traces to fitting and are more precise even though it is less general. This evaluation results also prove the conclusion that process discovery algorithms typically consider at most two out of these quality dimensions by [50].

VII. CONCLUSION

A complex enterprise information system is usually implemented on a distributed platform. The running logs of the workflow systems contain detailed information about the execution behaviors of activities. In this paper, we discuss how to discover the model for a complex workflow from multi-source heterogeneous logs collected from distributed servers.

By obtaining the top-level process model with abstract procedures and the bottom-level process models for each abstract procedures, Petri net refinement operation is used to integrate these process models to obtain the process model for the whole workflow system.

In this paper, we assume that the running logs of the workflow used for process mining are well-formed and without noise. However, a set of well-formed running logs is usually difficult to obtain. Noise may occur when, for example, a wrong activity is executed before or after another activity. Obviously, running logs with noise will definitely lead to improper mining result. Therefore, the detection approach of noise and infrequent behavior in the distributed running logs is badly needed. Meanwhile, the running logs also record the messages exchanged between different organizations. In fact, the execution of some activities in one organization usually need to access messages sent by other partner. As a consequence, the approach towards cross-organizational message consistency verification will also be highly desired in the future.

REFERENCES

[1] W. M. P. van der Aalst, "Business process management: A comprehensive survey," *ISRN Softw. Eng.*, vol. 2013, pp. 1–37, Feb. 2013.

[2] Q. Zeng, S. X. Sun, H. Duan, C. Liu, and H. Wang, "Cross-organizational collaborative workflow mining from a multi-source log," *Decis. Support Syst.*, vol. 54, no. 3, pp. 1280–1301, Feb. 2013.

[3] W. M. van der Aalst and B. F. van Dongen, "Discovering Petri nets from event logs," in *Transactions on Petri Nets and Other Models of Concurrency VII*. Heidelberg, Germany: Springer, 2013, pp. 372–422.

[4] C. Liu, H. Duan, Q. Zeng, M. Zhou, F. Lu, and J. Cheng, "Towards comprehensive support for privacy preservation cross-organization business process mining," *IEEE Trans. Services Comput.*, vol. 12, no. 4, pp. 639–653, Jul. 2019.

[5] C. Liu, J. Zhang, G. Li, S. Gao, and Q. Zeng, "A two-layered framework for the discovery of software behavior: A case study," *IEICE Trans. Inf. Syst.*, vol. E101.D, no. 8, pp. 2005–2014, Aug. 2018.

[6] C. Liu, S. Wang, S. Gao, F. Zhang, and J. Cheng, "User behavior discovery from low-level software execution log," *IEEJ Trans. Elect. Electron. Eng.*, vol. 13, no. 11, pp. 1624–1632, 2018.

[7] C. Liu, Y. Pei, Q. Zeng, and H. Duan, "LogRank: An approach to sample business process event log for efficient discovery," in *Proc. Int. Conf. Knowl. Sci., Eng., Manage.* Cham, Switzerland: Springer, 2018, pp. 1–11.

[8] S. X. Sun, Q. Zeng, and H. Wang, "Process-mining-based workflow model fragmentation for distributed execution," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 2, pp. 294–310, Mar. 2011.

[9] T. A. Curran and A. Ladd, *SAP R/3 Business Blueprint: Understanding Enterprise Supply Management*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.

[10] J. Eder, G. E. Olivotto, and W. Gruber, "A data warehouse for workflow logs," in *Proc. 1st Int. Conf. Eng. Deployment Cooperat. Inf. Syst. (EDCIS)*. London, UK: Springer-Verlag, 2002, pp. 1–15.

[11] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.

[12] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 145–180, Oct. 2007.

[13] L. Wen, J. Wang, W. M. P. van der Aalst, B. Huang, and J. Sun, "A novel approach for process mining based on event types," *J. Intell. Inf. Syst.*, vol. 32, no. 2, pp. 163–190, Apr. 2009.

[14] W. M. Van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Heidelberg, Germany: Springer, 2011.

[15] G. Greco, A. Guzzo, L. Pontieri, and D. Sacca, "Discovering expressive process models by clustering log traces," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 8, pp. 1010–1027, Aug. 2006.

[16] M. Prodel, V. Augusto, X. Xie, B. Jouaneton, and L. Lamarsalle, "Discovery of patient pathways from a national hospital database using process mining and integer linear programming," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 1409–1414.

[17] W. van der Aalst, "Process mining: Overview and opportunities," *ACM Trans. Manage. Inf. Syst. (TMIS)*, vol. 3, no. 2, p. 7, 2012.

[18] C. W. Günther and W. M. Van Der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics," in *Business Process Management*. Heidelberg, Germany: Springer, 2007, pp. 328–343.

[19] R. J. C. Bose, E. H. Verbeek, and W. M. van der Aalst, "Discovering hierarchical process models using ProM," in *IS Olympics: Information Systems in a Diverse World*. Heidelberg, Germany: Springer, 2012, pp. 33–48.

[20] R. J. C. Bose and W. M. van der Aalst, "Abstractions in process mining: A taxonomy of patterns," in *Business Process Management*. Heidelberg, Germany: Springer, 2009, pp. 159–175.

[21] S. J. Leemans, D. Fahland, and W. M. van der Aalst, "Discovering block-structured process models from event logs-a constructive approach," in *Applications and Theory of Petri Nets and Concurrency*. Heidelberg, Germany: Springer, 2013, pp. 311–329.

[22] H. Duan, Q. Zeng, H. Wang, S. X. Sun, and D. Xu, "Classification and evaluation of timed running schemas for workflow based on process mining," *J. Syst. Softw.*, vol. 82, no. 3, pp. 400–410, Mar. 2009.

[23] W. Brauer, R. Gold, and W. Vogler, "A survey of behaviour and equivalence preserving refinements of Petri nets," in *Proc. Adv. Petri Nets (APN)*. New York, NY, USA: Springer-Verlag, 1991, pp. 1–46.

[24] W. M. Zuberek, "Hierarchical analysis of manufacturing systems using Petri nets," in *Proc. SMC Conf. IEEE Int. Conf. Syst., Man Cybern., Evolving Syst., Humans, Organizations, Complex Interact.*, Nashville, TN, USA, Oct. 2000, pp. 3021–3026.

[25] H. Huang, T.-Y. Cheung, and W. Ming Mak, "Structure and behavior preservation by Petri-net-based refinements in system design," *Theor. Comput. Sci.*, vol. 328, no. 3, pp. 245–269, Dec. 2004.

[26] L. Jiao, "Refining and verifying regular Petri nets," *Int. J. Syst. Sci.*, vol. 39, no. 1, pp. 17–27, Jan. 2008.

[27] W. M. P. van der Aalst, "Workflow verification: Finding control-flow errors using Petri-net-based techniques," in *Business Process Management*. London, U.K.: Springer-Verlag, 2000, pp. 161–183.

[28] Z. Ding, Z. Zhang, C. Jiang, and M. Pan, "Formal model of workflow integration and its application in STISAG," in *Proc. 10th Int. Conf. Comput. Supported Cooperat. Work Design*, May 2006, pp. 1173–1178.

[29] Z. Ding, Y. Zhang, C. Jiang, and Z. Zhang, "Refinement of Petri nets in workflow integration," in *Proc. CSCWD* in Lecture Notes in Computer Science, vol. 4402, W. Shen, J. Luo, Z. Lin, J.-P. A. Barthès, and Q. Hao, Eds. Heidelberg, Germany: Springer, 2006, pp. 667–678.

[30] Z. Ding, C. Jiang, M. Zhou, and Y. Zhang, "Preserving languages and properties in stepwise refinement-based synthesis of Petri nets," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 38, no. 4, pp. 791–801, Jul. 2008.

[31] W. Reisig, *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*. Heidelberg, Germany: Springer, 2013.

[32] J. Zhou, J. Wang, and J. Wang, "A simulation engine for stochastic timed Petri nets and application to emergency healthcare systems," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 4, pp. 969–980, Jul. 2019.

[33] Q. Zeng, H. Wang, D. Xu, H. Duan, and Y. Han, "Conflict detection and resolution for workflows constrained by resources and non-determined durations," *J. Syst. Softw.*, vol. 81, no. 9, pp. 1491–1504, Sep. 2008.

[34] H. Duan, C. Liu, Q. Zeng, and M. Zhou, "Refinement-based hierarchical modeling and correctness verification of cross-organization collaborative emergency response processes," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Apr. 4, 2019, doi: 10.1109/TSMC.2018.2838053.

[35] Q. Zeng, F. Lu, C. Liu, H. Duan, and C. Zhou, "Modeling and verification for cross-department collaborative business processes using extended Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 2, pp. 349–362, Feb. 2015.

[36] Q. Zeng, C. Liu, and H. Duan, "Resource conflict detection and removal strategy for nondeterministic emergency response processes using Petri nets," *Enterprise Inf. Syst.*, vol. 10, no. 7, pp. 729–750, Sep. 2016.

[37] C. Liu, Q. Zeng, H. Duan, M. Zhou, F. Lu, and J. Cheng, "E-net modeling and analysis of emergency response processes constrained by resources and uncertain durations," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 1, pp. 84–96, Jan. 2015.

[38] J. Cheng, C. Liu, M. Zhou, Q. Zeng, and A. Ylä-Jääski, "Automatic composition of semantic Web services based on fuzzy predicate Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 680–689, Apr. 2015.

[39] C. Liu and F. Zhang, "Petri net based modeling and correctness verification of collaborative emergency response processes," *Cybern. Inf. Technol.*, vol. 16, no. 3, pp. 122–136, Sep. 2016.

[40] C. Liu, J. Cheng, Y. Wang, and S. Gao, "Time performance optimization and resource conflicts resolution for multiple project management," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 3, pp. 650–660, 2016.

[41] C. Liu, Q. Zeng, H. Duan, L. Wang, J. Tan, C. Ren, and W. Yu, "Petri net based data-flow error detection and correction strategy for business processes," *IEEE Access*, vol. 8, pp. 43265–43276, 2020.

[42] Q. Li, Y. Deng, C. Liu, Q. Zeng, and Y. Lu, "Modeling and analysis of subway fire emergency response: An empirical study," *Saf. Sci.*, vol. 84, pp. 171–180, Apr. 2016.

[43] W. Duo, X. Jiang, O. Karoui, X. Guo, D. You, S. Wang, and Y. Ruan, "A deadlock prevention policy for a class of multithreaded software," *IEEE Access*, vol. 8, pp. 16676–16688, 2020.

[44] L. Wang, Y. Du, and L. Qi, "Efficient deviation detection between a process model and event logs," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 6, pp. 1352–1364, Nov. 2019.

[45] W. van der Aalst *et al.*, "Process mining manifesto," in *Business Process Management Workshops*. Berlin, Germany: Springer, 2012, pp. 169–194.

[46] *XES Formate Running Logs of the Multi-Model Transportation Business Process*. Accessed: Jan. 11, 2020. [Online]. Available: http://u.163.com/Bvjl21Qq,accessNumber:dJvXEilN

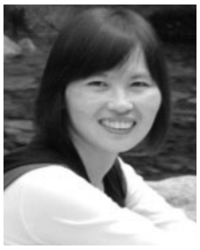
[47] J. M. E. M. van der Werf, B. F. van Dongen, C. A. J. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," in *Applications and Theory of Petri Nets*. Heidelberg, Germany: Springer, 2008, pp. 368–387.

[48] J. Claes and G. Poels, "Merging event logs for process mining: A rule based merging method and rule suggestion algorithm," *Expert Syst. Appl.*, vol. 41, no. 16, pp. 7291–7306, Nov. 2014.

[49] W. van der Aalst, A. Adriansyah, and B. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *WIREs Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 182–192, Mar. 2012.

[50] J. C. Buijs, B. F. van Dongen, and W. M. van der Aalst, "On the role of fitness, precision, generalization and simplicity in process discovery," in *On Move to Meaningful Internet Systems (OTM)*. Heidelberg, Germany: Springer, 2012, pp. 305–322.

QINGTIAN ZENG received the B.S. and M.S. degrees in computer science from the Shandong University of Science and Technology, Tai'an, China, in 1998 and 2001, respectively, and the Ph.D. degree in computer software and theory from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. He is currently a Professor with the Shandong University of Science and Technology, Qingdao, China. His research interests include petri nets, process mining, and knowledge management.



HUA DUAN received the B.S. and M.S. degrees in applied mathematics from the Shandong University of Science and Technology, Tai'an, China, in 1999 and 2002, respectively, and the Ph.D. degree in applied mathematics from Shanghai Jiaotong University, in 2008. She is currently a Professor with the Shandong University of Science and Technology. Her research interests include petri nets, business process management, and machine learning.

CONG LIU received the B.S. and M.S. degrees in computer software and theory from the Shandong University of Science and Technology, Qingdao, China, in 2013 and 2015, respectively, and the Ph.D. degree from the Section of Information Systems (IS), Department of Mathematics and Computer Science, Eindhoven University of Technology, in 2019. He is currently a Full Professor with the Shandong University of Technology. His research interests include business process mining, petri nets, and software process mining.