IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 31, 2020, accepted February 12, 2020, date of publication February 24, 2020, date of current version March 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976124

Petri Net Based Data-Flow Error Detection and Correction Strategy for Business Processes

CONG LIU[1], QINGTIAN ZENG[2], HUA DUAN[2], LEI WANG[1], JIE TAN[3], CHONGGUANG REN[1], AND WANGYANG YU[4]

[1]School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China
[2]Department of Computer Science and Technology, Shandong University of Science and Technology, Qingdao 266590, China
[3]State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450000, China
[4]School of Computer Science, Shaanxi Normal University, Xi'an 710062, China

Corresponding authors: Qingtian Zeng (qtzeng@sdust.edu.cn) and Hua Duan (hduan@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 Scholar 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 under Grant 2015TDJH102.

ABSTRACT Conceptual modeling, which includes both control-flow and data-flow modeling, has posed great challenges for conventional business process management systems. To support systematic data-flow modeling and analysis, in this paper, we propose a novel and effective Petri net-based approach. We first introduce a new type of Petri net, called WFIO-net, the firing rule of which is formally defined by extending classical Petri net with read and write semantics, to model both the control-flow and data-flow information. Then, we discuss about three possible data-flow errors over a WFIO-net, i.e., the missing, redundant and conflicting data errors. To detect such data-flow errors, we develop a polynomial complexity algorithm based on the so-called Activity Data Incidence Matrix (ADIM) of a WFIO-net. Following by that, we propose three effective correction strategies to resolve the detected data-flow anomalies. Finally, we present a property loan approval business process case study for our approach, and the study results demonstrate that the proposed detection and correction approaches are indeed very effective and can be applied to real-life settings.

INDEX TERMS Petri nets, data-flow error detection, correction strategy, business process modeling, WFIO-net.

I. INTRODUCTION

With the growing of event data from large information systems, Business Process Management (BPM) has received more and more attention from researchers and practitioners from various domains. A business process is made up of a well-defined collection of activities (referred to as tasks), and also their execution orders. Modeling and verification of business processes (i.e., workflows) have been studied for many years, and it is well known that the control-flow is the backbone of a business process [1]. In fact, to capture useful information in a process in a more comprehensive way, many other characteristics of a process like time (e.g., constraints on activity execution duration), resources (e.g., roles), and data-flows (e.g., decisions) are becoming more and more important for today's business process analytics.

Up to now, the topic of business process structural correctness verification has been extensively studied and various approaches have been proposed in the past years [1]–[5]. In the meantime, to cope with the analysis requirements of a process, such as the time and resource as mentioned above, some more advanced techniques have been also studied. For example, time management [6]–[10] and resources management [11]–[13] in a business process. The former study often considers the issues of activity execution duration, deadline constraints, and the temporal consistency of a process, and the latter one focuses on analyzing resource usages of activities in a process, as the activities always need to access system resources (e.g., facilities) and social resources (e.g., employees) during their executions.

Similar to time and resource, data is another important factor for business process modeling and analysis. For instance, routing choices of a process are typically dependent on certain data elements. Since a constructed data-flow model could be erroneous, various methods on correctness verification of

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

data-flows have been investigated in these years [14]–[21]. However, to the best of our knowledge, all the approaches just focus on detecting data-flow errors, and how to correct the detected anomalies has not been fully studied yet.

In this paper, we introduce a novel approach, targeting to systematic modeling, and error detection and correction of data-flow in business processes. More accurately, our research goal is to develop a methodology for detecting data-flow errors and correcting those errors with effective strategies by extending our previous work [25]. Here, we summarize the main contributions of this paper as follows:

- We introduce a Petri net-based approach (WFIO-net) for formally modeling business processes, from both the control-flow and data-flow aspects.
- We present a comprehensive formal definitions for three possible data-flow errors in our approach.
- To detect the data-flow errors, we introduce a novel and effective algorithm, the complexity of which is in polynomial time. Moreover, to resolve the detected data-flow errors, we also develop a set of correction strategies.
- We present a property loan approval business process case study for our approach. Our results demonstrate that the proposed approach is very practical and can be applied to real-life settings.

The remainder of this paper is organized as follows. Section II discusses about the related work. Section III introduces a systematic data-flow modeling approach based on WFIO-net. Section IV presents the formal definitions of three different data-flow errors and their correction strategies. Section V gives a property loan approval business process to validate the applicability of the propose approach while Section VI draws the conclusions of the work.

II. RELATED WORK

This section summarizes the work related to control-flow and data-flow verification of business processes.

A. CONTROL-FLOW AND DATA-FLOW MODELING AND VERIFICATION

As one of the most dominant models, WF-net [1], which is a special type of Petri net, has been widely applied for process-oriented modeling and analysis. For a WF-net, a correctness criterion called soundness is defined by van der Aalst in the work [1]. Note that sometimes a WF-net is unable to represent a correct business process, since errors such as deadlocks and livelocks may occur in the process model. For some subclasses of WF-nets, for example the free choice net, soundness can be detected in polynomial time. However, most of current verification techniques always require an exhaustive exploration of the state space of a net, and thus they often suffer low efficiency [2], [5], [40], [41], in terms of detection performance. To handle this limitation, [41]–[44] propose a guard-driven reachability graph approach that can improve the analysis performance.

The concept of data-flow correctness verification of business process is first introduced in the work [14]. Generally, there are three main types of data-flow problems, i.e., missing data, redundant data and inconsistent data. Currently, there are only a few approaches can be used to detect and correct these errors. For example, approaches based on Dual Workflow Nets (DWF-net) [15] try to extend a control-flow model by adding data-flow elements. However, a DWF-net is always very complex even for a very small business process. Meanwhile, there is no explicit indication on how to create this type of model.

To obtain control-flow information from data-flow, Sun and Zhao present an approach in [17]. They investigate the possibility of incorporating formal analytics into business process designs, to alleviate the intensive intellectual challenges faced by business analysis. This analytics can construct a business process using relevant activity information and their associated (i.e., input and output) data. Moreover, in [18], Trcka et al. introduce the notion of Workflow Data (WFD) nets. Basically, a WFD is a formal business process model with data which can be read, written and destroyed. In [19], a data-flow matrix and a set of relations between data elements and activities are first defined. Then, some basic types of data-flow errors are conceptualized on the basis of UML activity diagrams. To extend the applicability of the approaches based on activity diagram, a number of rules are defined for the transformation from activity diagrams to Petri nets by Wohed *et al.* [39]. Their work is further generalized by Sundari et al. in [20]. More recently, a comprehensive survey on data-flow modeling and verification of business processes is reported by Dolean and Petrusel [21]. They conclude that: (1) data is essential in a business process and its execution requires data; (2) the control-flow cannot be executed without data; and (3) there is no modeling approach focusing on how data state changes during process execution.

In fact, the problem of how to find out the described data errors in our Petri net-based formal model is quite similar as the classical model checking problems, which are on the basis of state-based model checkers such as TAPAAL 2.0 [22]. However, traditional model checking tools (like TAPAAL) normally have the following three limitations in the scope of checking data-flow errors: (1) the timed-arc Petri nets do not support data elements explicitly. Therefore, we will have to extend the tool or to transform the data perspective constraints to timed-arc Petri net constructs; (2) the analysis of TAPAAL tool is based on generating reachability of a model. Although an efficient algorithm has been proposed, the performance is still an issue because the search space will be huge. In contrast, as we will present later, our proposed approach uses a structure based on Petri nets (i.e., the incidence matrix) to do the computation, and obviously our approach will be much clearer and simpler; and (3) TAPAAL-like tools provide full time factors support, and thus they are extremely suitable for time performance analysis. However, the data-flow error detection problem does not require a model to contain any time factors, as the temporal checking is based on the precedence relationship among transitions. In such scenarios,

the detection process will be over-complicated when using a TAPAAL tool.

B. SUMMARY

Based on above literature review, we can see that there has been a long history of studies on business process correctness verification. Although an abundance of formal techniques have been proposed to detect control-flow errors, there is very little support for systematic data-flow modeling and verification. Moreover, existing work on data-flow has following problems: (1) most of existing data-flow analysis approaches only provide methods to check data-flow errors, but not the formal models with operation semantics; and (2) how to correct the detected data-flow anomalies is totally ignored.

To address these issues, as we will present in the following that we propose an approach on defining and detecting data-flow anomalies in a formal way. There is no doubt that effective anomaly resolution methods will be very helpful on constructing correct workflow models. Therefore, we also introduce various strategies to support the correction of detected data-flow errors. Moreover, we also demonstrate the effectiveness and applicability of the proposed approach by a property loan approval business process case.

III. PETRI NET BASED MODELING OF CONTROL-FLOW AND DATA-FLOW

Our work is based on Petri net, or WF-net to be more precise. Here, we assume readers are familiar with the basic concepts of Petri nets [23]–[26]. Some basic terminologies and notations are given in the following for self-completeness of the paper.

A. PETRI NET AND WORKFLOW NET

A 3-tuple $N = (P, T, F)$ is called a net if: (1) P is a finite set of places and T is a finite set of transitions such that: $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$; and (2) $F \subseteq (P \times T) \cup (T \times P)$. Here, we give the formal definition of a Petri net following the work [16], [23], [24] and [2], [27]–[38].

Definition 1: A Petri net is a 4-tuple $\Sigma = (P, T, F, M_0)$, where (1) (P, T, F) is a net; and (2) $M_0 : p \rightarrow Z^+$ is the initial marking of Σ, where $M(p)$ represents the number of tokens in place p and Z^+ is a non-negative integer set.

For any $x \in P \cup T$, the set ${}^\bullet x = \{y | (y, x) \in F\}$ is named its pre-set (input), and $x^\bullet = \{y | (x, y) \in F\}$ is named its post-set (output). To describe the semantic of a Petri net, we use markings. A marking is a multi-set of places, indicating how many tokens each place contains. An initial marking is denoted by M_0. A transition $t \in T$ is enabled under marking M, iff $\forall p \in {}^\bullet t : M(p) > 0$, denoted as $M[t\rangle$. If $M[t\rangle$ holds, t may fire, and this will bring in a new marking M', such that (1) $M'(p) = M(p) - 1$, if $p \in {}^\bullet t \setminus t^\bullet$, or (2) $M'(p) = M(p) + 1$ if $p \in t^\bullet \setminus {}^\bullet t$, or (3) otherwise $M'(p) = M(p)$.

A Petri net can model a business process, and we call such a model as a WF-net. The definition of a WF-net is given as below, following the work [1].

Definition 2: A Petri net $\Sigma = (P, T, F, M_0)$ is a WF-net if: (1) there is one source place $p_s \in P$ such that ${}^\bullet p_s = \emptyset$; (2) there is one sink place $p_e \in P$ such that $p_e^\bullet = \emptyset$; (3) for any $x \in P \cup T$ is on a path from p_s to p_e; and (4) for any $p \in P$, $M_0(p) = 1$ if $p = p_s$, and otherwise $M_0(p) = 0$.

It should be noticed that a path in a Petri net is defined as the sequence of nodes (i.e., places and transitions) linked by directed arcs. WF-net is capable of modelling different control-flow constructs of a business process. In the following, we synonymously use transition and activity as the WF-net is specifically used to model business processes in this paper.

B. CONTROL-FLOW NET MODEL

The control-flow aspect of a business process is modelled by a WF-net model.

Definition 3: [25] A Control-flow Net Model is a 3-tuple $\Sigma_C = (P_L, T, F_L)$, where (1) P_L is the logic place set; (2) $T = T_A \cup T_L$, where T_A is the activity set of the business process and T_L is the logic transition set; and (3) $F_L \subseteq (P_L \times T) \cup (T \times P_L)$ is the control-flow set.

Essentially , a control-flow net model is a standard WF-net. For example , for the control-flow net model in Fig. 1 , we have that (1) $P_L = \{p_i | 1 \leq i \leq 12\} \cup \{p_e\} \cup \{p_s\}$; (2) $T = \{t_{Aj} | 1 \leq j \leq 9\}$; and (3) $F_L = \{(p_s, t_{A1}), (t_{A1}, p_1), (p_1, t_{A2}), (t_{A2}, p_2), (p_2, t_1), (t_1, p_3), (t_1, p_4), (t_1, p_5), (p_3, t_{A3}), (p_4, t_{A4}), (p_5, t_{A5}), (t_{A3}, p_6), (t_{A4}, p_7), (t_{A5}, p_8), (p_6, t_2), (p_7, t_2), (p_8, t_2), (t_2, p_9), (p_9, t_{A6}), (t_{A6}, p_{10}), (p_{10}, t_{A7}), (t_{A7}, p_{11}), (p_{11}, t_{A8}), (t_{A8}, p_{12}), (p_{12}, t_{A9}), (t_{A9}, p_e)\}$.

C. DATA-FLOW NET MODEL

The control-flow net model can be used to model business process logic, however, it does not support data-flow elements of a business process. To remedy this problem, we propose a data-flow net model as below.

Definition 4: [25] A *Data-flow Net Model* is a 5-tuple $\Sigma_D = (P_D, T_A, F_D, F_R, F_A)$, where (1) P_D represents the data element set; (2) T_A is the activity set of the business process; and (3) $F_D \subseteq (P_D \times T_A) \cup (T_A \times P_D)$ is the data-flow set, where $F_R \subseteq (P_D \times T_A)$ is the read flow and $F_W \subseteq (T_A \times P_D)$ is the write flow.

For the transitions in Fig. 1, a responsible data-flow net model is demonstrated in Fig. 2. It can be formalized as: (1) $P_D = \{p_{di} | 1 \leq i \leq 10\}$; (2) $T_A = \{t_{Aj} | 1 \leq j \leq 9\}$; and (3) F_D=$\{(t_{A1}, p_{d1}), (t_{A1}, p_{d2}), (t_{A1}, p_{d3}), (p_{d1}, t_{A2}), (p_{d2}, t_{A2}), (p_{d3}, t_{A2}), (t_{A2}, p_{d4}), (t_{A3}, p_{d5}), (t_{A4}, p_{d5}), (t_{A5}, p_{d5}), (p_{d5}, t_{A6}), (t_{A6}, p_{d6}), (p_{d6}, t_{A7}), (p_{d7}, t_{A7}), (t_{A7}, p_{d8}), (p_{d8}, t_{A8}), (t_{A8}, p_{d9}), (p_{d9}, t_{A9}), (t_{A9}, p_{d10})\}$.

D. WFIO-NET MODEL

To combine the modeling of the control-flow and data-flow elements of a business process, we introduce a new kind of workflow net, called WFIO-net, by extending each transition with its corresponding input and output data sets. We give its detailed definition as following:

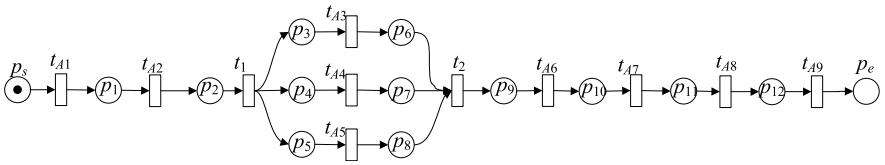


FIGURE 1. An Example of the Control-flow Net Model.

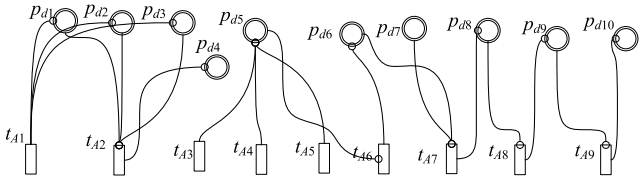


FIGURE 2. An Example of the Data-flow Net Model.

Definition 5: [25] Σ_{IO}=(P, T, F, I, O, M_0) is a WFIO-net if:

- $P = P_D \cup P_L$ and $P_D \cap P_L = \emptyset$, where R_D is the data place set and R_L is the logic place set;
- $T = T_A \cup T_L$ and $T_A \cap T_L = \emptyset$, where T_A is the activity transition set and T_L is the logic transition set;
- $F = F_L \cup F_D$ where $F_L \subseteq (P_L \times T_L) \cup (T_L \times P_L)$ is the control (or logic) flow and $F_D \subseteq (P_D \times T_D) \cup (T_D \times P_D)$ is the data flow;
- $I : T_A \rightarrow P_D$. For any $t_a \in T_A$, $I(t_a)$ is the input data set of transition t_a;
- $O : T_A \rightarrow P_D$. For any $t_a \in P_A$, $O(t_a)$ is the output data set of transition t_a; and
- $\forall p \in P$, $M_0(p) = 1$ if $p = p_s$, otherwise $M_0(p) = 0$.

Based on above definition, we can see that (P_L, T_L, F_L, M_0) is a WF-net and it represents the control-flow structure of a business process. In such a case, compared to a WF-net, a WFIO-net will have three main differences: (1) a WFIO-net has two types of places, i.e., the data place set P_D and the logic place set P_L; (2) a WFIO-net has two kinds of flow relations, i.e., one kind is used to represent the traditional token flow F_L while the other is to represent the data flow F_D. From a graph angle, in a WFIO-net, a token flow arc is drawn by a direct arc that ends with arrow, and a data flow arc is represented with a direct arc that ends with small circle; and (3) the transition firing rule of a WFIO-net is different from that of a classical WF-net (more details see later of this subsection).

From above differences and according to *Definition 5*, we can see that a WFIO-net model is actually the union of a control-flow net model and a data-flow net model, by merging their transitions with same labels. Therefore, if we take the control-flow net model and the data-flow net model in Fig. 1 and Fig. 2 as inputs, then we can get a WFIO-net model as demonstrated in Fig. 3.

For more details, we denote the pre-set of a transition t in a WFIO-net as ${}^\bullet t = {}^\diamond t \cup {}^\circ t$. Here, ${}^\diamond t$ represents the logic pre-set of transition t and ${}^\circ t$ represents the data pre-set of transition t (or read place set). Similarly, the post-set of a transition t also involves two parts, i.e., $t^\bullet = t^\diamond \cup t^\circ$, in which $t^\diamond$ represents the logic post-set of transition t and t° represents the data post-set (or write place set) of transition t. For example, Fig. 4(a) shows a simple transition model of a WFIO-net. Its pre-set of t_a is ${}^\bullet t_a = \{p_1, p_{read}\}$, logic pre-set is ${}^\diamond t_a = \{p_1\}$ and read place is ${}^\circ t_a = \{p_{read}\}$. In the meantime, we have the post-set of t_a is $t_a^\bullet = \{p_2, p_{write}\}$, logic post-set is $t_a^\diamond = \{p_2\}$ and write place is $t_a^\circ = \{p_{write}\}$.

Regarding to the firing rule of a WFIO-net, a transition $t \in T$ is enabled under marking M, iff $\forall p \in {}^\bullet t : M(p) > 0$, denoted as $M[t\rangle$. If $M[t\rangle$ holds, t may fire, resulting in a new marking M', such that $M'(p) = M(p) - 1$ if $p \in {}^\diamond t \setminus t^\diamond$, $M'(p) = M(p)+1$ if $p \in t^\bullet \setminus {}^\bullet t$, and otherwise $M'(p) = M(p)$. Based on this, we have the following explanations for the WFIO-net model illustrated in Fig. 4: (1) p_{read} and p_{write} represent data places and p_1 and p_2 represent logic places; (2) t_a is enabled in Fig. 4 (a) as p_1 and p_{read} contains a token; and (3) Fig. 4(b) gives the state after firing t_a where the token in p_1 is removed to p_2 and p_{write} obtains one token without consuming the token in p_{ready}, i.e., the data elements are un-consumable.

IV. DATA-FLOW ERROR DETECTION AND CORRECTION STRATEGIES

In this section, we present the details of the data-flow errors of a business process as well as their detection and correction strategies.

A. TAXONOMY OF BASIC DATA-FLOW ERRORS

As discussed by Sun et al. in the work [10], missing data error, redundant data error and conflicting data error are the most fundamental data-flow anomalies in business processes. Other types of advanced data-flow errors have been given by Sadiq et al. in [7], and they can be represented by the most basic ones. Therefore, in this work, we restrict ourselves to the three basic data-flow errors. We give their details and formal definitions in a WFIO-net as following.

1) MISSING DATA ERROR

A missing data error occurs when a data element is accessed without been initialized. It is very similar as a variable in a program that the variable is used without definition or initialization. For the missing data error in a WFIO-net, we give its formal definition as below.

Definition 6: Let $\Sigma_{IO} = (P, T, F, I, O, M_0)$ be a WFIO-net, for any $p_d \in P_D$, p_d is defined as a missing data element if: (1) ${}^\circ p_d = \emptyset$ and (2) $p_d^\circ \neq \emptyset$.

In above *Definition 6*, the first condition indicates that the data element p_d is not initialized, and the second one means this data element is used by at least one

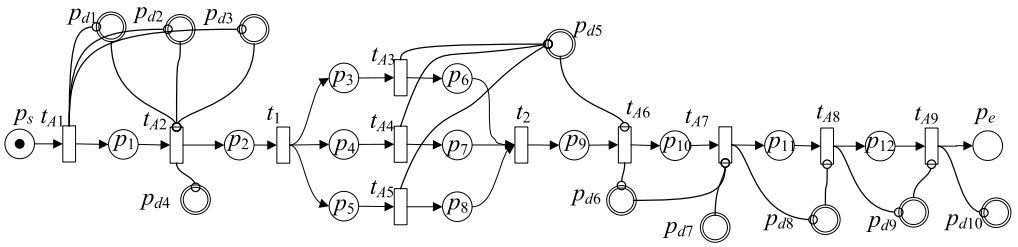


FIGURE 3. An Example of the WFIO-net Model.

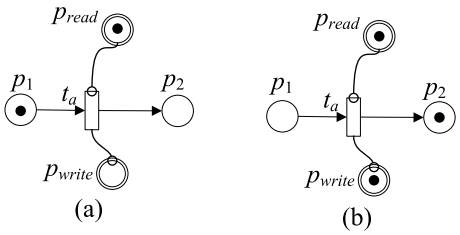


FIGURE 4. WFIO-net of a Transition (a) Enabled State and (b) Fired State.

	p_{d1}	p_{d2}	p_{d3}	p_{d4}	p_{d5}	p_{d6}	p_{d7}	p_{d8}	p_{d9}	p_{d10}
t_{A1}	1	1	1	0	0	0	0	0	0	0
t_{A2}	-1	-1	-1	1	0	0	0	0	0	0
t_{A3}	0	0	0	0	1	0	0	0	0	0
t_{A4}	0	0	0	0	1	0	0	0	0	0
t_{A5}	0	0	0	0	1	0	0	0	0	0
t_{A6}	0	0	0	0	-1	1	0	0	0	0
t_{A7}	0	0	0	0	0	-1	-1	1	0	0
t_{A8}	0	0	0	0	0	0	0	-1	1	0
t_{A9}	0	0	0	0	0	0	0	0	-1	1

FIGURE 5. The *ADIM* for the WFIO-net in Fig. 3.

follow-up activity. Similar to the analysis in the work [10] and [7], missing data errors would lead to system exceptions and dangling. Therefore, they should be detected and corrected before system enactment.

2) REDUNDANT DATA ERROR

A redundant data error occurs when some data elements are produced but never been used by any other follow-up activities. It is a bit like that a program variable is defined and initialized but it has never been used. The formal definition of redundant data errors based on a WFIO-net is given as below.

Definition 7: Let $\Sigma_{IO} = (P, T, F, I, O, M_0)$ be a WFIO-net, $\forall p_d \in P_D$, p_d is defined as a redundant data element if: (1) $^\circ p_d \neq \emptyset$ and (2) $p_d^\circ = \emptyset$.

Similarly, the first condition in *Definition 7* indicates that the data element is initialized, and the second one means the data element is not used by any follow-up activities. Redundant data error may lead to system inefficiency, and therefore, it should be detected and corrected beforehand.

3) CONFLICTING DATA ERROR

A conflicting data error occurs when there exists multiple versions of the same data element in one process instance. For example, multiple activities attempt to initialize (or write) the same data element at the same time. The formal definition of conflicting data error based on a WFIO-net is given as follows.

Definition 8: Let $\Sigma_{IO} = (P, T, F, I, O, M_0)$ be a WFIO-net, $\forall p_d \in P_D$, p_d is defined as a conflicting data element if: (1) $^\circ p_d \neq \emptyset$; (2) $p_d^\circ \neq \emptyset$; and $|^\circ p_d| \geq 2$.

In *Definition 8*, the first and third condition mean that the data element is initialized by multiple activities, and the second means that the data element will be used. Conflicting data errors may cause uncertainty or even confusion, and therefore, they should be detected and corrected before a business process is enacted.

B. DATA-FLOW ERRORS CHECKING ALGORITHM

In this subsection, we present an approach to detect these previously defined data-flow errors.

According to the definitions in [23], any kind of Petri nets can be represented in the form of an incidence matrix. Existing incidence matrix only shows the control-flow information but no data-flow relation is included. To this end, we define the *Activity-Data Incidence Matrix*, *ADIM* for short, as following.

Definition 9: Given M be an $m \times n$ matrix, where m is the number of transitions and n is the number of data places. For any $[i, j] \in M$, $v([i, j])$ is the value of this position. $v([i, j]) = 1$ means transition i produces data element j as output, $v([i, j]) = -1$ means transition i takes data element j as input, and $v([i, j]) = 0$ means there is no production and consumption relations between the transition i and the data element j.

The *ADIM* of the WFIO-net in Fig. 4 is constructed as the matrix demonstrated in Fig. 5.

To detect the previously described three dataflow errors in a WFIO-net based on its *ADIM*, we propose an efficient method as presented in Algorithm 1. There, we check the possible errors on each data elements (line 2). For each element, we first calculate its statistic information over each transitions using the ADIM (lines 3-9), and then we identify the errors based on the Definitions 6-8 (lines 10-16). Obviously, the complexity of Algorithm 1 is determined by its two loops (line 2 and line 3), and thus the complexity of the algorithm is $O(|P_d| \times |T_a|)$, where $|P_d|$ is the number of data elements and $|T_a|$ is the number of activities.

Considering the WFIO-net in Section III as an example, we run Algorithm 1 by taking the *ADIM* in Fig. 4 as input,

Algorithm 1 Data-Flow Error Detection

Inputs: *Activity-Data Incidence Matrix M* of a WFIO-net.
Outputs: *MisSet*, *RedSet* and *ConSet*.

Initialization:
1: $MisSet \leftarrow \emptyset$, $RedSet \leftarrow \emptyset$, $ConSet \leftarrow \emptyset$, $RNum \leftarrow 0$ and $WNum \leftarrow 0$

Error Detection:
2: **for** $i \leftarrow 0..(|P_d| - 1)$ **do**
3: **for** $j \leftarrow 0..(|T_a| - 1)$ **do**
4: **if** $p_{ij} == 1$ **then**
5: $WNum{+}{+}$
6: **else if** $p_{ij} == -1$ **then**
7: $RNum{+}{+}$
8: **end if**
9: **end for**
10: **if** $WNum == 0$ && $RNum > 0$ **then**
11: $MisSet \leftarrow MisSet \cup \{p_{dj}\}$
12: **else if** $WNum > 0$ && $RNum == 0$ **then**
13: $RedSet \leftarrow RedSet \cup \{p_{dj}\}$
14: **else if** $WNum > 1$ && $RNum > 0$ **then**
15: $ConSet \leftarrow ConSet \cup \{p_{dj}\}$
16: **end if**
17: $RNum \leftarrow 0$ and $WNum \leftarrow 0$
18: **end for**

and we can obtain that $MisSet = \{p_{d7}\}$, $RedSet = \{p_{d4}, p_{d10}\}$ and $ConSet = \{p_{d5}\}$. More specifically, (1) data element p_{d7} is missing because it is accessed by activity t_{A7} without been initialized; (2) data elements p_{d4} and p_{d10} are redundant because they are initialized by activities t_{A7} and t_{A9} respectively, but never been used by any activity in the business process; and (3) data element p_{d5} is conflicting because it is initialized (or written) by different activities (t_{A3}, t_{A4} and t_{A5}), i.e., there may exist different versions of p_{d5} in a single business process instance.

C. DATE-FLOW ERROR CORRECTION STRATEGIES

In this subsection, we propose different strategies to correct the three types of data-flow errors in a WFIO-net. When performing data-flow error corrections, we give a priority to the control-flow, so as to avoid changing the business process logic. In the following, we present the details of our correction strategies.

1) CORRECTION STRATEGY 1

Missing Data Error Correction: Let $\Sigma_{IO} = (P, T, F, I, O, M_0)$ be a WFIO-net, and $p_d \in P_D$, p_d is a missing data element. The activity t_{AM} that generates the data element p_d should be added to the WFIO-net, satisfying $T = T \cup \{t_{AM}\}$ and $F = F \cup \{(t_{AM}, p_d)\}$.

The activity t_{AM} can be added to any location in the WFIO-net before the missing data element is used. To make it simpler and straightforward, we suggest adding it sequentially, before the time that the activity uses the missing data element. An example of this strategy is shown in Fig. 6. There, p_d is used by an activity t_a without be initialized, therefore, we add the activity t_{AM} to initialize p_d before t_a. In this way, the missing data error is resolved.

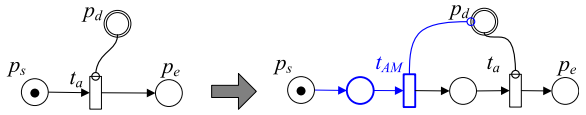


FIGURE 6. An Example of the Correction Strategy 1.

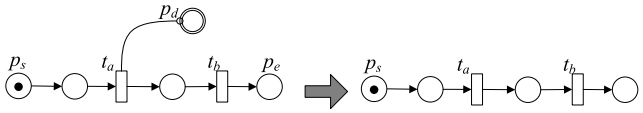


FIGURE 7. An Example of the Correction Strategy 2.

2) CORRECTION STRATEGY 2

Redundant Data Error Correction: Let $\Sigma_{IO} = (P, T, F, I, O, M_0)$ be a WFIO-net, and $p_d \in P_D$, p_d is a redundant data. The redundant data element p_d should be removed from the WFIO-net, satisfying that $P = P - \{p_d\}$ and $F = F - \{(t_a, p_d)\}$.

An example of this strategy is shown in Fig. 7. In the example, p_d is first produced by activity t_a but it is never been used by follow-up activities, therefore we just remove p_d from the WFIO-net, and consequently the redundant data error is corrected.

3) CORRECTION STRATEGY 3

Conflicting Data Error Correction: Let $\Sigma_{IO} = (P, T, F, I, O, M_0)$ be a WFIO-net, and $p_d \in P_D$, p_d is a conflicting. We duplicate the conflicting data element p_d according to their different producers, satisfying that $P = \{P - \{p_d\}\} \cup \{p_{d1}, p_{d2}\}$ and $F = \{F - \{(t_a, p_d), (t_b, p_d), (p_d, t_c)\}\} \cup \{(t_a, p_{d1}), (t_b, p_{d1}), (p_{d1}, t_c), (p_{d2}, t_c)\}$.

Fig. 8 shows an example of how to apply this strategy. There, p_d is initialized (or written) by activities t_a and t_b, therefore according to *Correction Strategy 3*, we duplicate the conflicting data element p_d according to their different producers. Therefore the conflicting data error is resolved.

Considering the WFIO-net in Fig. 3 as a running example. Based on Algorithm 1, we have known that the data-flow errors are $MisSet = \{p_{d7}\}$, $RedSet = \{p_{d4}, p_{d10}\}$, and $ConSet = \{p_{d5}\}$. Following the three proposed data-flow correction strategies, we can get a corrected WFIO-net model as illustrated in Fig. 9. More specifically, (1) data element p_{d7} is missing because it is accessed by activity t_{A7} without been initialized. According to *Correction Strategy 1*, we add activity t_{AM} to initialize p_{d7} before used by t_{A7}. Without changing the business logic, we add it sequentially before t_{A7}. Therefore the missing data error is resolved. (2) data elements p_{d4} and p_{d10} are redundant because they are initialized by activities t_{A7} and t_{A9} respectively, but without being used by

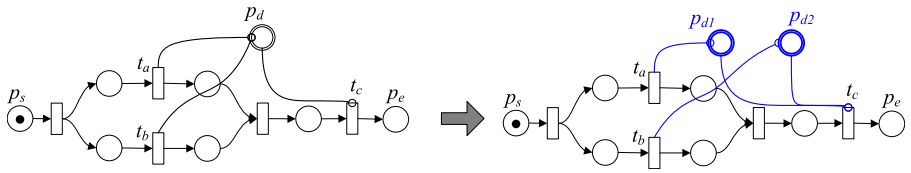


FIGURE 8. An Example of the Correction Strategy 3.

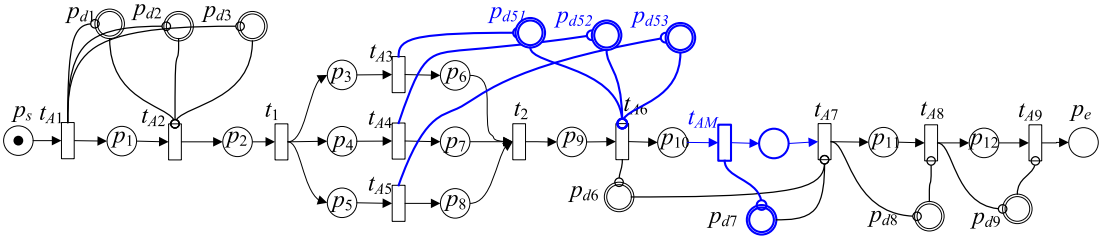


FIGURE 9. An Example WFIO-net Model after Correcting Data-flow Errors.

TABLE 1. Activity information.

Activity Name	Meaning
A_1	Receive an application
A_2	Verify the completeness of an application
A_3	Request missing information
A_4	Verify the employment status
A_5	Check the credit history
A_6	Verify the liquid asset
A_7	Determine the interest rate
A_8	Request appraisal information
A_9	Evaluate the loan application
A_{10}	Adjust the loan amount
A_{11}	Contact the applicant for agreement
A_{12}	Forward to a loan officer for signature
A_{13}	Forward to a manager for signature

TABLE 2. Data information.

Data Name	Meaning
D_1	Name of the applicant
D_2	Amount of the loan
D_3	Annual income
D_4	Completed application
D_5	Application summary
D_6	Verified employment status
D_7	Credit score
D_8	Account balance
D_9	Verified account balance
D_{10}	Qualified applicant name
D_{11}	Interest rate
D_{12}	Property address
D_{13}	Property of the current owner
D_{14}	Appraised value of property
D_{15}	Risk rate
D_{16}	Adjusted amount
D_{17}	Applicant agreement
D_{18}	Property insured
D_{19}	Applicant signature
D_{20}	Loan officer signature
D_{21}	Manager signature

any activity in the business process. According to *Correction Strategy 2*, we remove p_{d4} and p_{d10} from the WFIO-net. Therefore the redundant data error is resolved. (3) data element p_{d5} is conflicting because it is initialized by activities t_{A3}, t_{A4} and t_{A5}, i.e., there exists different versions of p_{d5} in one process instance. According to *Correction Strategy 3*, we duplicate the conflicting data element p_d according to their different producers as p_{d1}, p_{d2} and p_{d3}. Then, we correct the conflicting data errors.

V. CASE STUDY

In this section, a property loan approval business process case study, which is widely adopted in current literature such as [19], is used to validate the applicability of our proposed approach.

The whole business can be described in the following: (1) The process starts when a property loan application request is received; (2) then completeness of this application is verified; (3) if this application is not complete, the missing information is required; (4) to determine the qualifications of an applicant, the financial service company verifies the applicant's employment status, checks the applicant's credit history as well as the applicant's liquid assets at the same time; (5) if the applicant is judged to be qualified, the current interest rate is locked for a certain period.; (6) after this, the financial service company requests the appraisal information for the applicant; (7) then the loan application is evaluated. Based on the applicant's credit score, the loan amount and the appraised value of the property, the risk level associated with the loan are calculated; (8) if the risk is higher than the applicable threshold, the loan amount should be adjusted; (9) the financial service company then contacts the applicant to discuss the necessary adjustment and other options; (10) if the applicant agrees with everything, it is forwarded to a loan officer for signature; and (11) if the loan amount is over $500,000, the general manager's signature is required. The activity information and the corresponding data elements are described in Table 1 and 2 respectively.

The read/write relations between activities and data elements of the property loan approval business process are

TABLE 3. Dependency relation between activity and data elements.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}
D_1	w	r		r	r	r	r	r	r	r	r	r	r
D_2	w	r					r		r		r	r	r
D_3	w	r		r			r						
D_4		w	r										r
D_5	w												r
D_6				w					r				
D_7					w		r		r				
D_8	w	r				r			r				
D_9						w			r				
D_{10}				w	w	w			r		r	r	r
D_{11}							w		r	r		r	r
D_{12}	w	r						r					
D_{13}								w					
D_{14}							r	w	r	r			
D_{15}									w	r		r	r
D_{16}							r			w	r	r	r
D_{17}											w	r	r
D_{18}											r		
D_{19}											w	r	r
D_{20}												w	
D_{21}													w

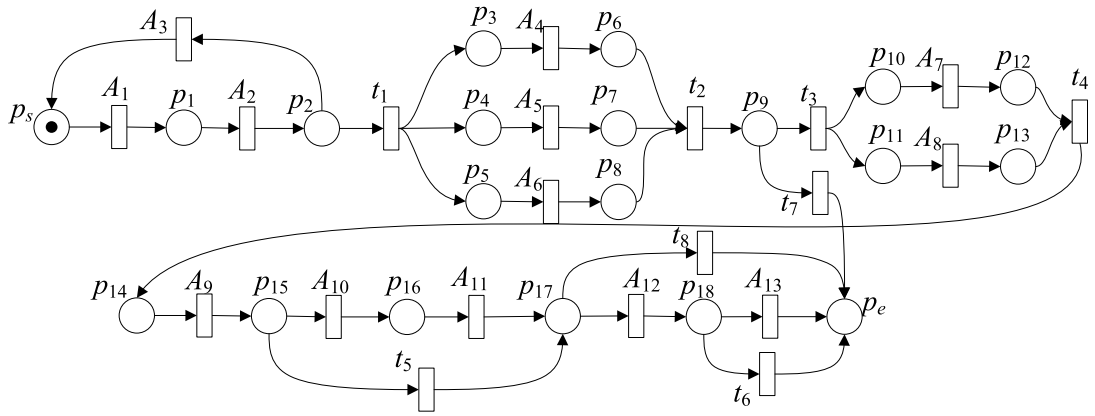


FIGURE 10. Control-flow Net Model of the Property Loan Approval Process.

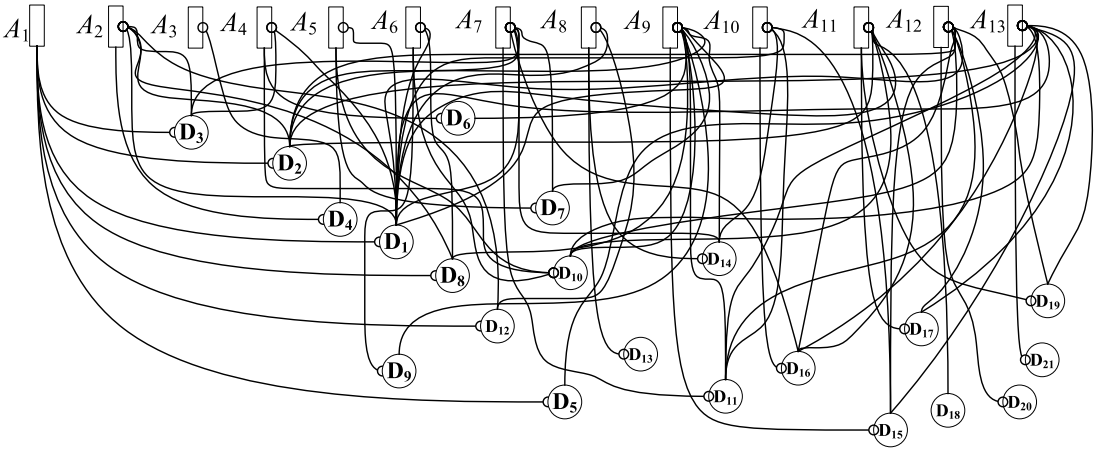


FIGURE 11. Data-flow Net Model of the Property Loan Approval Process.

illustrated in Table 3. There, "w" represents write operation and "r" stands for read operation. For example, for activity A_6, it needs to read data elements D_1 and D_8 before execution, and writes D_9 and D_{10} when finishes.

According to Definition 3, the control-flow net model of this property loan approval business case is shown in Fig. 10, where we have that (1) $P_L = \{p_i | 1 \leq i \leq 18\} \cup \{p_e\} \cup \{p_s\}$; (2) $T_A = \{t_{Aj} | 1 \leq j \leq 13\}$; and (3) $T_L = \{t_k | 1 \leq k \leq 8\}$.

According to Definition 4, the data-flow net model of this property loan approval business process is illustrated in Fig. 11, where we have that (1) $P_D = \{D_i | 1 \leq i \leq 21\}$; and (2) $T_A = \{t_{Aj} | 1 \leq j \leq 13\}$.

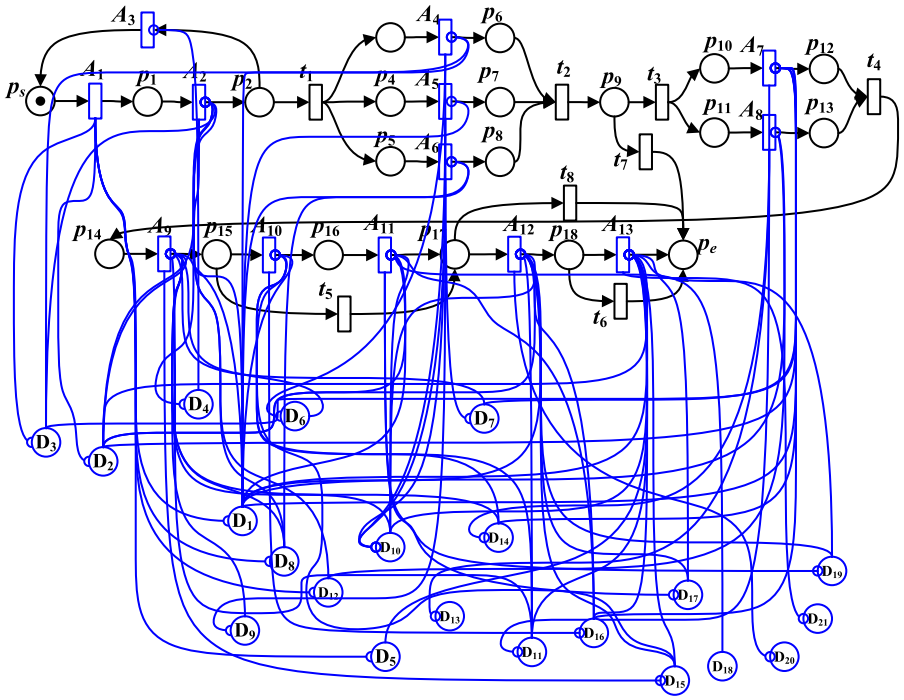


FIGURE 12. WFIO-net Model of the Property Loan Approval Process.

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}	D_{19}	D_{20}	D_{21}
A_1	1	1	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
A_2	-1	-1	-1	1	0	0	0	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0
A_3	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A_4	-1	0	-1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
A_5	-1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
A_6	-1	0	0	0	0	0	0	-1	1	1	0	0	0	0	0	0	0	0	0	0	0
A_7	-1	-1	-1	0	0	0	-1	0	0	0	1	0	0	-1	0	-1	0	0	0	0	0
A_8	-1	0	0	0	0	0	0	0	0	0	0	-1	1	1	0	0	0	0	0	0	0
A_9	-1	-1	0	0	0	-1	-1	-1	-1	-1	-1	0	0	-1	1	0	0	0	0	0	0
A_{10}	-1	0	0	0	0	0	0	0	0	0	-1	0	0	-1	-1	1	0	0	0	0	0
A_{11}	-1	-1	0	0	0	0	0	0	0	-1	0	0	0	0	0	-1	1	-1	1	0	0
A_{12}	-1	-1	0	0	0	0	0	0	0	-1	-1	0	0	0	-1	-1	-1	0	-1	1	0
A_{13}	-1	-1	0	-1	-1	0	0	0	0	-1	-1	0	0	0	-1	-1	-1	0	-1	0	1

FIGURE 13. *ADIM* of Property Loan Approval Process WFIO-net.

According to *Definition 5*, i.e., a WFIO-net model is the union of a control-flow net model and the data-flow net model by merging transitions with same labels, by taking the control-flow and data-flow nets in Fig. 10 and Fig. 11 as inputs, we have the WFIO-net in Fig. 12.

Moreover, the *ADIM* of the WFIO-net in Fig. 12 is constructed and shown in Fig. 13.

Taking the *ADIM* in Fig. 13 as input, based on Algorithm 1, we obtain that $MisSet = \{D_{18}\}$, $RedSet = \{D_{13}, D_{20}, D_{21}\}$ and $ConSet = \{D_{10}\}$. Namely, (1) data element D_{18} is missing because it is accessed by activity A_{11} without initialization; (2) data elements D_{13}, D_{20}, D_{21} are redundant because they are initialized by activities A_8, A_{12} and A_{13} respectively, but are not used by any activity; and (3) data element D_{10} is conflicting because it is initialized by different activities (A_4, A_5 and A_6), i.e., there exist different versions of D_{10} in one process instance.

Based on above analysis, applying the proposed data-flow correction strategies, we have that (1) data element D_{18} is missing because it is accessed by activity A_{11} without being initialized. According to *Correction Strategy 1*, we add activity $A_{Manual}(t_{AM})$ to initialize D_{18} before being used by A_{11}. Without changing the business logic, we add it sequentially before A_{11}. In this way the missing data error is resolved; (2) data elements D_{13}, D_{20} and D_{21} are redundant because they are initialized by activities A_8, A_{12} and A_{13} respectively, but without being used by any activity in the process. On the basis of *Correction Strategy 2*, we remove D_{13}, D_{20}, D_{21} from the WFIO-net. In such a case, the redundant data error is removed; and (3) data element D_{10} is conflicting as it is initialized by activities A_4, A_5 and A_6, i.e., there exists different versions of D_{10} in one process instance. Using *Correction Strategy 3*, we duplicate the conflicting data element D_{10} according to their different producers as D_{10A}, D_{10B}, and D_{10C}. Then, the conflicting data error is resolved. At this time point, the corrected WFIO-net model of the property loan approval business process is demonstrated in Fig. 14. There, the missing data error related correction is

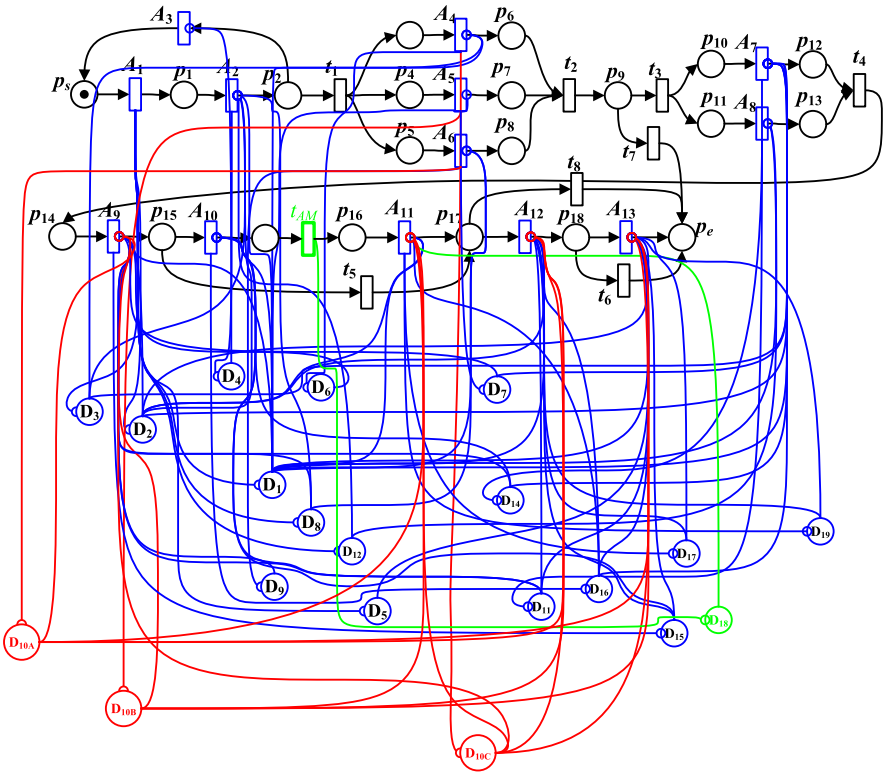


FIGURE 14. WFIO-net Model of Property Loan Approval Process after Correcting Data-flow Errors.

highlighted in light green color, and the conflicting data error related correction part is highlighted using red color. We can see that our approach is indeed very practical for real case studies.

VI. CONCLUSION

To formalize a systematic data-flow modeling, detection and corrections of business processes, we propose a Petri net-based approach in this paper. More precisely, we introduce a Petri net-based approach (WFIO-net) for formally modeling business processes, from both the control-flow and data-flow aspects. Moreover, we provide the formal definitions for three basic data-flow errors in our approach, and also develop an effective algorithm to detect these errors. To efficiently resolve the detected data-flow errors, we finally propose a set of correction strategies.

To the best of our knowledge, this is the first work towards formally modeling, detecting and correcting of data-flow anomalies in a business process. Our case study has shown that the proposed approach is actual very practical and thus can be applied to real-life settings. Our future work mainly lies in the following two aspects: (1) we will investigate the possibility of a more detailed taxonomy of data-flow errors. For instance, the missing data error could be divided into several sub-classes, such as absolute missing and conditional missing; and (2) we will explore the opportunities on incorporating access control permission factors in our WFIO-net so as to achieve a more accurate verification. An example is that writing operation to a data element will be not permitted when the element is being read/wrote by another activity.

REFERENCES

[1] W. M. P. Van Der Aalst, "The application of Petri nets to workflow management," *J. Circuits, Syst. Comput.*, vol. 8, no. 1, pp. 21–66, Nov. 2011.

[2] G. Liu, W. Reisig, C. Jiang, and M. Zhou, "A branching-process-based method to check soundness of workflow systems," *IEEE Access*, vol. 4, pp. 4104–4118, 2016.

[3] J. Desel and J. Esparza, *Free Choice Petri Nets* (Cambridge Tracts in Theoretical Computer Science), vol. 40. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[4] G. Liu and C. Jiang, "Co-NP-hardness of the soundness problem for asymmetric-choice workflow nets," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 8, pp. 1201–1204, Aug. 2015.

[5] K. M. van Hee, N. Sidorova, and M. Voorhoeve, "Generalised soundness of workflow nets is decidable," in *Applications and Theory of Petri Nets* (Lecture Notes in Computer Science), J. Cortadella and W. Reisig, Eds. vol. 3099. Berlin, Germany: Springer-Verlag, 2004, pp. 197–215.

[6] J. Li, Y. Fan, and M. Zhou, "Performance modeling and analysis of workflow," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 34, no. 2, pp. 229–242, Mar. 2004.

[7] J. Li, Y. Fan, and M. Zhou, "Timing constraint workflow nets for workflow analysis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 2, pp. 179–193, Mar. 2003.

[8] N. R. Adam, V. Atluri, and W. K. Huang, "Modeling and analysis of workflows using Petri nets," *J. Intell. Inf. Syst., Special Issue Workflow Process Manage.*, vol. 10, no. 2, pp. 131–158, 1998.

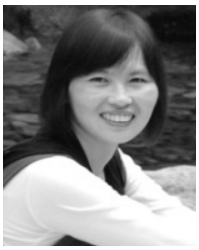
[9] J. Li, Y. Fan, and M. Zhou, "Approximate performance analysis of workflow model," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 2, Oct. 2003, pp. 1175–1180.

[10] S. Ling and H. Schmidt, "Time Petri nets for workflow modelling and analysis," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Nashville, TN, USA, Oct. 2000, pp. 3039–3044.

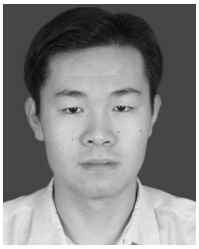
[11] H. Wang and Q. Zeng, "Modeling and analysis for workflow constrained by resources and nondetermined time: An approach based on Petri nets," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 802–817, Jul. 2008.
[12] Q. Zeng, H. Wang, D. Xu, H. Duan, and Y. Han, "Conflict detection and resolution for workflows constrained by resources and non-determined durations," *J. Syst. Softw.*, vol. 81, no. 9, pp. 1491–1504, Sep. 2008.
[13] H. Li, Y. Yang, and T. Y. Chen, "Resource constraints analysis of workflow specifications," *J. Syst. Softw.*, vol. 73, no. 2, pp. 271–285, Oct. 2004.
[14] S. Sadiq, M. Orlowska, W. Sadiq, and C. Foulger, "Data flow and validation in workflow modelling," in *Proc. 15th Australasian Database Conf.*, vol. 27, K.-D. Schewe and H. Williams, Eds. Darlinghurst, NSW, Australia: Australian Computer Society, 2004, pp. 207–214.
[15] S. Fan, W. Dou, and J. Chen, "Dual workflow nets: Mixed control/data-flow representation for workflow modeling and verification," in *Advances in Web and Network Technologies, and Information Management* (Lecture Notes in Computer Science), vol. 4537. Berlin, Germany: Springer, 2007, pp. 433–444.
[16] C. Liu, "Automatic discovery of behavioral models from software execution data," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1897–1908, Oct. 2018.
[17] S. X. Sun and J. L. Zhao, "Formal workflow design analytics using data flow modeling," *Decis. Support Syst.*, vol. 55, no. 1, pp. 270–283, Apr. 2013.
[18] N. Trcka, W. M. P. van der Aalst, and N. Sidorova, "Data-flow anti-patterns: Discovering data-flow errors in Workflows," in *Proc. 21st Int. Conf. Adv. Inf. Syst. Eng.*, 2009, pp. 425–439.
[19] S. X. Sun, J. L. Zhao, J. F. Nunamaker, and O. R. L. Sheng, "Formulating the data-flow perspective for business process management," *Inf. Syst. Res.*, vol. 17, no. 4, pp. 374–391, Dec. 2006.
[20] M. H. Sundari, A. K. Sen, and A. Bagchi, "Detecting data flow errors in workflows: A systematic graph traversal approach," in *Proc. 17th Workshop Inf. Technol. Syst.*, Montreal, QC, Canada, 2007, pp. 1–6.
[21] C. Dolean and R. Petrusel, "Data-flow modeling: A survey of issues and approaches," *Inf. Economica*, vol. 16, p. 117, Oct. 2012.
[22] A. David, L. Jacobsen, M. Jacobsen, K. Y. Jørgensen, M. H. Møller, and J. Srba, "TAPAAL 2.0: Integrated development environment for timed-arc Petri nets," in *Tools and Algorithms for the Construction and Analysis of Systems* (Lecture Notes in Computer Science), vol. 7214. Berlin, Germany: Springer, 2012.
[23] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
[24] W. Reisig, *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*. Heidelberg, Germany: Springer, 2013.
[25] C. Liu, Q. Zeng, and H. Duan, "Formulating the data-flow modeling and verification for workflow: A Petri net based approach," *Int. J. Sci. Eng. Appl.*, vol. 3, no. 4, pp. 107–112, Jul. 2014.
[26] Q. Zeng, C. Liu, J. Zou, F. Lu, and Q. Wu, "Invariant decomposition conditions for Petri nets based on the index of transitions," *Inf. Technol. J.*, vol. 11, no. 7, pp. 768–774, Jul. 2012.
[27] C. Liu, Q. Zeng, H. Duan, and F. Lu, "Petri net based behavior description of cross-organization workflow with synchronous interaction pattern," in *Process-Aware Systems* (Communications in Computer and Information Science), vol. 495. Berlin, Germany: Springer, 2015, pp. 1–10.
[28] 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.
[29] J. Cheng, C. Liu, M. Zhou, Q. Zeng, and A. Yla-Jaaski, "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.
[30] 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.
[31] 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.
[32] 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, Jan. 2015.
[33] 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.
[34] 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.
[35] 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.
[36] Q.-T. Zeng, F.-M. Lu, C. Liu, and D.-C. Meng, "Modeling and analysis for cross-organizational emergency response systems using Petri nets," *Chin. J. Comput.*, vol. 36, no. 11, pp. 2290–2302, Mar. 2014.
[37] 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.
[38] C. Liu, B. van Dongen, N. Assy, and W. M. P. van der Aalst, "Component behavior discovery from software execution data," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.
[39] P. Wohed, W. van der Aalst, M. Dumas, A. H. ter Hofstede, and N. Russell, "Pattern-based analysis of the control-flow perspective of UML activity diagrams," in *Proc. Int. Conf. Conceptual Modeling*. Berlin, Germany: Springer, 2005, pp. 63–78.
[40] F. Lu, Q. Zeng, Y. Bao, and H. Duan, "Hierarchy modeling and formal verification of emergency treatment processes," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 44, no. 2, pp. 220–234, Feb. 2014.
[41] F. Lu, Q. Zeng, M. Zhou, Y. Bao, and H. Duan, "Complex reachability trees and their application to deadlock detection for unbounded Petri nets," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 6, pp. 1164–1174, Jun. 2019.
[42] D. Xiang, G. Liu, C.-G. Yan, and C. Jiang, "A guard-driven analysis approach of workflow net with data," *IEEE Trans. Services Comput.*, to be published.
[43] Y. He, G. Liu, C. Yan, C. Jiang, and J. Wang, "Locating and controlling unsound transitions in workflow systems based on workflow net with data constraints," *IEEE Access*, vol. 6, pp. 62622–62637, 2018.
[44] D. Xiang, G. Liu, C. Yan, and C. Jiang, "Detecting data-flow errors based on Petri nets with data operations," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 251–260, Jan. 2018.

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 in computer science and information systems from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2019. He is currently a Professor with the Shandong University of Technology, Zibo, China. His current research interests include business process management, process mining, Petri nets, and big data.

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 are in the areas of 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 an Associate Professor with the Shandong University of Science and Technology. Her research interests include Petri nets, process mining, and machine learning.



LEI WANG received the B.S. degree from Ludong University (LDU), Yantai, China, in 2008 and the Ph.D. degree from the South China University of Technology (SCUT), Guangzhou, China, in 2013. He is currently a Lecturer with the Shandong University of Technology. His research interests include big data processing, image registration, and multimodal image fusion.

JIE TAN received the M.S. degree in computer science from the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China, in 2017, where she is currently pursuing the Ph.D. degree. Her current research interests include software architecture, technical debt, high-performance computing, and information security.

CHONGGUANG REN received the B.S. degree in computer science and technology from the Shandong University of Technology, Zibo, China, in 2005, the M.S. degree in software engineering from Tongji University, in 2009, and the Ph.D. degree in computer science and technology from the Nanjing University of Science and Technology, Nanjing, China, in 2013. He is currently an Associate Professor with the Shandong University of Technology. His current research interests include big data processing, cloud computing, and intelligent control.

WANGYANG YU received the Ph.D. degree from Tongji University, Shanghai, China, in 2014. He is currently an Associate Professor with Shaanxi Normal University, Xi'an, China. His research interests include the theory of Petri nets and formal methods in software engineering.