

Received 4 July 2022, accepted 2 August 2022, date of publication 16 August 2022, date of current version 24 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3198969

## RESEARCH ARTICLE

# A Colored Petri Net Executable Modeling Approach for a Data Flow Well-Structured BPMN Process Model

FENGLAN HUANG<sup>1</sup>, FENG NI<sup>1</sup>, JIANG LIU<sup>1</sup>, FAN YANG<sup>2</sup>, AND JIAYI ZHU<sup>1</sup>

<sup>1</sup>Business School, University of Shanghai for Science and Technology, Shanghai 200093, China

<sup>2</sup>Shanghai Sunshine City Kindergarten, Shanghai 200540, China

Corresponding author: Feng Ni (nifeng@usst.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61403255 and Grant 11701370, in part by the Shanghai System Science Peak Discipline Construction Project, and in part by the Shanghai Innovation and Entrepreneurship Training Program for College Students.

**ABSTRACT** BPMN process models have been widely used in software designs. The BPMN process models are characterized by a static graph-oriented modeling language and a lack of analytical capabilities as well as dynamic behavior verification capabilities, which not only leads to inconsistencies in the semantics of the BPMN process models, but also leads to a lack of model error detection capabilities for the BPMN process models, which also hinders the correctness verification and error correction efforts of the models. In this study, we propose an executable modeling approach for CPN-based data flow well-structured BPMN (dw-BPMN) process models, and consider both control-flow and data-flow perspectives. First, we present a formal definition of the dw-BPMN process model, which is formally mapped into a CPN executable model in three steps: splitting, mapping and combining. Then, we discuss four types of data flow errors that can occur in the model: missing, lost, redundant, and inconsistent data error. To detect these four data flow errors, we propose a detection method based on the execution results of the CPN model. Subsequently, we propose correction strategies for these four data flow errors. Finally, a dw-BPMN process model of a robot's temperature detection system for COVID-19 prevention and control in a kindergarten was used as an example to verify the validity of the method.


**INDEX TERMS** BPMN, formal verification, CPN, data flow, model transformation.

## I. INTRODUCTION

BPMN is a modeling standard in the field of business processes that bridge the communication gap between software system designers and developers [1]. BPMN models include process or orchestration models, choreography models [2], and collaboration models [3]. BPMN process models are frequently used to model business processes within an organization. The BPMN process models focus on business orchestration among the participants while describing the overall control flow and data flow in the software process model.

BPMN has the following two characteristics: on the one hand, the BPMN models are characterized by a static

graph-oriented modeling language [4]; on the other hand, the BPMN models are characterized by a lack of analytical capabilities and dynamic behavior verification capabilities. This not only leads to inconsistencies in the semantics of the BPMN process models, but also leads to a lack of detection and analysis capability of the BPMN process models for control-flow errors and data-flow errors, and also hinders the correctness verification [5] and error-correction work of the models. For instance, a large BPMN model of a concurrent system with complex behavior makes it challenging for system process designers to confirm the presence of unwanted features. Model flaws include data flow flaws, such as missing, lost, and redundant data as well as control flow flaws, such as deadlocks and live locks. BPMN model errors can be found via model checking.

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

Therefore, executable modeling of the BPMN process models is an effective means of model validation [6]. However, the validation of the BPMN process models faces the following challenges. First, the BPMN process models use graphical and natural language descriptions and lack formal descriptions. Second, the BPMN process models cover elements of control and data flows, and the large number of message exchanges and data associations may lead to large model sizes and generate structural diversity.

In view of the above challenges, we propose an executable modeling approach for the CPN-based dw-BPMN process model. Because model correctness depends not only on their control flow but also on their data flow, we consider both the control flow and data flow perspectives. One of the crucial methods for assessing the dynamic behavior of models, which can model the control and data flows of business systems and be used to study and verify their correctness, is Petri net executable modeling [7], [8], [9], [10]. To address the problem of structural diversity of BPMN process models, we propose the concept of the dw-BPMN process model and then formally transform it into a CPN model in three steps: splitting, mapping, and combining. Finally, we examined and corrected the data flow errors. The main contributions of this study are summarized as follows.

- We propose the concept of a dw-BPMN process model that covers both control flow and data flow information, which not only eliminates the diversity of model construction structures, but also reduces the possibility of state-space explosion after mapping from a BPMN process model to a CPN model.
- We formally detected four possible types of data flow errors in the dw-BPMN process models: missing data, lost data, redundant data, and inconsistent data, and proposed correction strategies for these four types of data flow errors.
- We provide a dw-BPMN process model of a robot's temperature detection system for COVID-19 prevention and control in a kindergarten, which is used as an example to verify the validity of the method.

The remainder of this paper is organized as follows. Section II discusses related work. Section III presents relevant definitions of the BPMN, dw-BPMN, and CPN models. Section IV presents an executable modeling approach for the CPN-based dw-BPMN process model, in which the three rules of the transformation process, that is splitting, mapping, and combination rules, are described in detail. Section V describes the four possible types of data flow errors and detection method of the dw-BPMN process model, and the corresponding correction strategies. Section VI describes the executable modeling process using the dw-BPMN model of a kindergarten intelligent morning inspection robot as an example. Finally, Section VII concludes this paper.

## II. RELEVANT WORKS

This section summarizes research works related to executable modeling and validation of BPMN models.

### A. EXECUTABLE MODELING OF BPMN MODELS

The BPMN is a graphical workflow-based language. BPMN models lack explicit formal semantics as well as analysis and verification capabilities. Kheldoun *et al.* [11] proposed a formal mapping method for BPMN using recursive ECATNets. With this method, the modeler can further use the Maude LTL model checker to verify the behavioral properties of the BPMN model. Diaz *et al.* [12] proposed a GUI rule that enables semi-automated modeling of BPMN models based on GUI rules, which has the benefit of saving time by improving the efficiency of model modeling. However, the mapping rule relies on class diagrams of the BPMN model. Corradini and Fornari [13] proposed an executable modeling approach for BPMN models based on BProVe, which combines the LTL model checking technique of MAUDE and the MultiVeStA statistical model checking technique for modeling and verifying BPMN models. However, BProVe relies on the non-formal direct semantics of BPMN models. Zafar and Azam [14] proposed rules for converting the BPMN model into a SoaML model, enabling automatic generation of web service models from the BPMN model. The advantage of this is that the system developer simplifies the development of the ERP system, except that the analysis of the BPMN model was not mentioned in that paper. Duran *et al.* [15] proposed an executable specification of rewrite logic for BPMN with time and extended and modeled it with probabilities, further describing how to use rewrite logic for the stochastic analysis of BPMN timing properties. Houhou *et al.* [16] presented a direct formalization of the BPMN model based on first-order logic and provided a verification of the executable model framework, fbpmn, to verify the specific properties of BPMN models. Other researchers have used a purely mathematical approach to describe the formal execution semantics of BPMN models, such as process algebra [17], communication sequential process (CSP) [18], formal concept analysis, and Backus Naur Form (BNF) grammar [19]. However, the frameworks proposed by these studies lack validation of the dynamic behavior of the BPMN model.

In addition to the above executable methods, the Petri net is an important tool for verifying and evaluating the dynamic behavior of the model. Several scholars have been attracted to the performable modeling of BPMN models using Petri net [20]. Li and Ye [21] proposed a method to support the formal transformation of dynamically evolving BPMN models into Petri net models and maintain the consistency of model behavior. Dechsupa *et al.* [22] proposed a rule for converting BPMN elements into colored generalized random Petri net, as well as providing a stepwise refinement and verification method for each component of the CGSPN model. The advantage of this is that the designer can automate the construction of the CGSPN model, but it is time consuming in the simulation process. Meghzili *et al.* [23] investigated the conversion of BPMN orchestration models into workflow nets and then used petri net to formally define the BPMN 2.0-orchestration semantics and used the semantics to analyze the errors in the structure and control flow aspects of

the model, achieving a feasibility verification of the model conversion.

However, because of the limitations of Petri net, they are less suitable for modeling large-scale BPMN models; therefore, Maarouk et al. [24] investigated executable modeling of BPMN models using CPN and verified the rationality of this modeling approach.

The above studies considered, control flow as the backbone of the business process and considered only control errors and behavioral inconsistencies in the model. They only performed executable modeling and validation of the control-flow aspects of the BPMN model, and the data-flow aspects were not mentioned in the modeling process. Data flow analysis is becoming increasingly important in today's business process analysis to capture useful information and many other features of business processes in a more comprehensive way.

Based on the above issues, Dechsupa [25], [26] proposed an executable modeling approach for BPMN design models based on CPN, covering both the control flow and data flow of BPMN models. However, the consideration of BPMN model diversity and checking of data flow errors were not mentioned in their study. In general, the errors present in the model can be divided into two types: control flow errors and data flow errors, where control flow errors mainly include dead and live locks, and data flow errors mainly include missing, lost, redundant, and inconsistent data errors.

## B. SUMMARY

Based on the above literature review, we can understand that there is a long history of research on executable modeling methods for BPMN process models. Although a number of researchers have proposed formal methods for executable modeling and formal validation of BPMN process models, the following problems still exist in the process of executable modeling of BPMN process models: (1) while considering the problem of structural diversity of BPMN process models, the problem of the influence of data flow on the correctness of the models is ignored; (2) the lack of BPMN models themselves based on data flow errors was examined and corrected.

To address these issues, we propose a CPN-based executable modeling approach for dw-BPMN process models. Our proposed dw-BPMN process model addresses the problem of model structural diversity while considering data flow. Subsequently, we also detect data flow errors in the model based on the generated CPN model execution results and propose corresponding error correction strategies. Finally, the effectiveness of the proposed approach is verified through a case study.

## III. RELEVANT DEFINITION

### A. BPMN PROCESS MODEL

The BPMN model is a collection of BPMN elements comprising control flows, data flows, and associated activities

that generate specific operations to achieve specific business requirements. The core subset of BPMN elements was divided into six groups: event elements, task elements, gateway elements, connecting elements, artefacts, swim lanes, and pools [31].

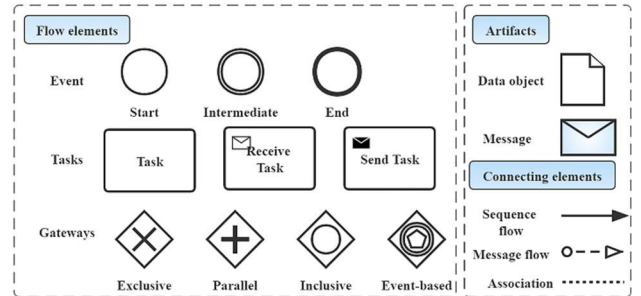


FIGURE 1. The core set of the BPMN process model elements.

Modelers use these elements to describe the BPMN process, the collaboration, and the orchestration models [15]. The BPMN process model describes the sequence of activities and data flows in an organization. The main elements of the model include flow elements, connecting elements, and artifacts, as shown in Fig. 1.

*Definition 1 (A BPMN Process Model):* The BPMN process model is a tuple.  $MBPMN = (N, A, E, ID, D, M, G, F, f)$ , where:

- 1)  $N$  is a finite set of nodes.
- 2)  $A$  is a set of tasks, and  $A \subseteq N$ . For  $\forall a_i \in A$  denotes  $a_i = (T_{Name}, T_{Oper}, T_{Marker})$ , where  $T_{Name}$  denotes the task name of  $a_i$  described as a string expression,  $T_{Oper}$  denotes the task operation of  $a_i$ , and  $T_{Marker}$  denotes the task marker of  $a_i$  consisting of a loop with a loop condition and multiple instances.
- 3)  $E$  denotes an event and  $E \subseteq N$ .  $E = \{E_s, E_e, E_i\}$ ,  $E_s$  denotes a start event,  $E_e$  denotes an end event,  $E_e \subseteq N$ , and  $E_i$  denotes an intermediate event,  $E_i \subseteq N$ .
- 4)  $ID$  indicates a set of variables and item definitions of the data structures used in the BPMN process.
- 5)  $D$  denotes the set of data, and  $D = \{D_i, D_o\}$ ,  $D_i$  denotes the input dataset of the task and  $D_o$  denotes the output data of the task.
- 6)  $M$  indicates a set of message associations for the task.
- 7)  $G$  denotes a set of gateways,  $G \subseteq N$ .
- 8)  $F$  denotes a set of sequence streams,  $F \subseteq N \times N$ .
- 9)  $f$  denotes a set of mapping functions, and  $f = \{f_{AT}, f_{GT}, f_{GD}, f_{GL}\}$ , where  $f_{AT}$  denotes the mapping function indicating the task type,  $f_{AT} : A \rightarrow \{task, service, send, receive, user, manual\}$ ;  $f_{GT}$  denotes the mapping function indicating the gateway type  $f_{GT} : G \rightarrow \{exclusive, inclusive, parallel\}$ ;  $f_{GD}$  denotes a mapping function indicating the direction of the gateway,  $f_{GD} : G \rightarrow \{divergent, convergent\}$ ;  $f_{GL}$  denotes a conditional statement label function

indicating the gateway,  $f_{GL} : (G \times N) \rightarrow \text{Guardlabel}$ , and  $\text{Type}(f_{GD}(g_i)) = \text{boolean} \forall g_i \in G | f_{GD}(g_i) \in \{\text{exclusive}, \text{inclusive}\} \wedge f_{GD}(g_i) \in \{\text{diverg} - \text{ent}\}$ .

### B. dw-BPMN PROCESS MODEL

Because a BPMN process model describes the sequence of activities and data flows in a business organization and has an asynchronous and concurrent character, it is suitable for formal modeling and verification using the CPN modeling language. To eliminate the diversity and complexity of the BPMN process model structure, we propose the concept of a dw-BPMN process model. This concept eliminates the complexity of the model structure while retaining the generality of the model.

The dw-BPMN process model has the following characteristics.

- 1)  $|E_s| = 1$ , there is one and only one start event.
- 2)  $\forall e_{sc} \in E_s, \text{in}(e_{sc}) = \emptyset \wedge |\text{out}(e_{sc})| = 1$ , the start event has and has only one sequence stream output and no input.
- 3)  $|E_e| = 1$ , there is only one end event.
- 4)  $\forall e_{ec} \in E_e, |\text{in}(e_{ec})| = 1 \wedge \text{out}(e_{ec}) = \emptyset$ , the end event has and has only one sequence stream input and no input.
- 5)  $\forall e_{ic} \in E_i, |\text{in}(e_{ic})| = 1 \wedge |\text{out}(e_{ic})| = 1$ , for any intermediate event, there is only one sequence stream input and output.
- 6)  $\forall x \in G_{PF} \cup G_{XD} \cup G_{ED}, |\text{in}(x)| = 1 \wedge |\text{out}(x)| > 1$ . Parallel forking gateways, exclusive data decision gateways, and exclusive event decision gateways have only one sequence stream input and greater than one sequence stream output.
- 7)  $\forall y \in G_{PJ} \cup G_{DM} \cup G_{EM}, |\text{in}(y)| > 1 \wedge |\text{out}(y)| = 1$ . Parallel aggregation gateways, exclusive data merging gateways, and exclusive event merging gateways have only one sequence stream output and more than one sequence stream input.
- 8)  $\forall a_i \in A, |D_i(a_i)| = 1 \wedge |D_o(a_i)| = 1 \wedge |\text{in}(a_i)| = 1 \wedge |\text{out}(a_i)| = 1$ . For any given task, there is only one data output and output, and there is only one input and output for the sequence stream.
- 9)  $\forall a_j \in A, (|\text{in}(M(a_j))| = 1 \wedge |\text{out}(M(a_j))| = 0) \vee (|\text{in}(M(a_j))| = 0 \wedge |\text{out}(M(a_j))| = 1)$ . For any task there is only one message input or output.
- 10)  $\forall G \rightarrow \{\text{exclusive}, \text{parallel}\}, |G_{PF}| = |G_{PJ}|, |G_{XD}| = |G_{DM}|, |G_{ED}| = |G_{EM}|$ . For any gateway, the number of parallel forking gateways must equal the number of parallel aggregation gateways, exclusive data decision gateways must equal the number of exclusive data merging gateways, and exclusive event decision gateways must equal the number of exclusive event merging gateways.
- 11) For any non-initial non-ending data object, there is at least one output and input; a non-initial data object is the input data that is not the first task or parallel first

set of tasks in the model, and a non-ending data object is the output data that is not the last task or last parallel set of tasks.

The above definitions of the BPMN process model allow for a well-defined structure of data and control flows and ensure that the model structure is standardized. Therefore, a process model that meets the above definitions is called the dw-BPMN process model.

*Definition 2 (A dw-BPMN Process Model):* A dw-BPMN process model is a tuple  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$ , where the model elements all conform to the above rules.

### C. CPN MODEL

CPN is one of the most widely used formal modeling languages [27] for the design and verification of concurrent systems. The core elements of CPN are the place, transition, arc, guard condition, arc inscription, and token.

*Definition 3 (A CPN Model):* A CPN model is a tuple  $M_{CPN} = (P, T, A_c, \Sigma, V, f_c)$ , where:

- 1)  $P$  denotes a finite set of system places.
- 2)  $T$  denotes a finite set of transitions,  $T \cap P = \emptyset, P \cup T \neq \emptyset$ .
- 3)  $A_c$  denotes a finite set of arcs,  $P \cap T = P \cap A = A \cap T = \emptyset$ , and  $A \subseteq ((P \times T) \cup (T \times P))$ .
- 4)  $\Sigma$  denotes a non-empty finite set of colours, called a colour set.
- 5)  $V$  indicates a limited type variable,  $\forall v \in V$  and  $\text{Type}(v) \subseteq \Sigma$ .
- 6)  $f_c$  denotes a set of expression functions, and  $f_c = \{f_{CC}, f_{GG}, f_{EA}, f_{II}\}$ . where  $f_{CC}$  denotes the color function and  $f_{CC} : (P \cup T) \rightarrow \Sigma_s, \Sigma_s$  is a finite subset of  $\Sigma$ .  $f_{GG} : T \rightarrow \text{expression}$ , denoting the guard function of  $T$  and satisfying  $\forall t \in T : [\text{Type}(f_{GG}(t)) = \text{Bool} \wedge (\text{Var}(f_{GG}(t))) \subseteq \Sigma]$  and  $\text{Bool} = \{\text{true}, \text{false}\}$ .  $f_{EA} : A_c \rightarrow \text{expression}$ , denoting the arc expression function and satisfying  $\forall a_c \in A_c : [\text{Type}(f_{EA}(a_c)) = f_{CC}(P_P(a_c))M_s \wedge \text{Type}(\text{Var}(f_{EA}(a_c))) \subseteq \Sigma]$ ;  $f_{II}$  denotes the initial identity of an  $P \rightarrow \text{expression}$  and satisfying  $\forall p \in P : [\text{Type}(f_{II}(p)) = f_{CC}(p)M_s \wedge \text{Var}(f_{II}(p)) = \emptyset]$ .

CPNTools is a suite of CPN editing, simulation, and analysis tool [28]. By analyzing the reach graphs and state space reports generated by the model, errors in the executable model can be detected, such as the presence of dead loops, deadlocks, and state space explosion [29], [30]. Therefore, this study used CPNTools for modeling.

### IV. METHODOLOGY

This section describes the process of transformation from the dw-BPMN process model to the CPN model, as shown in Fig. 2. This process consists of three steps: splitting, mapping, and combining. In the following section, we describe each of these three steps in detail.



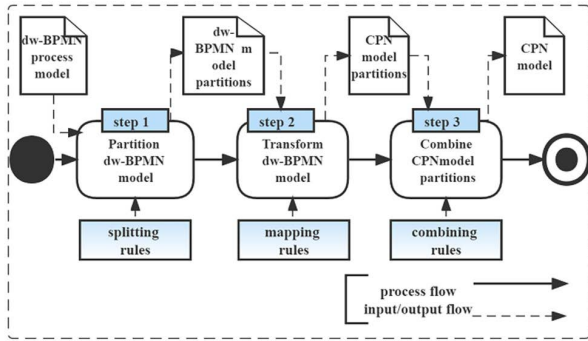


FIGURE 2. Transformation process of dw-BPMN process model to CPN model.

### A. SPLITTING RULES FOR *dw*-BPMN PROCESS MODEL

The large number of business organizational activities leads to the large size of the corresponding *dw*-BPMN process model, making it more difficult to map and test the model. The purpose of this step is to split larger process models, particularly those containing the direct nesting of gateways, into element groups. The complexity of the model was further reduced, thus facilitating the mapping exercise. Algorithm 1 illustrates the process of partitioning the *dw*-BPMN process model. We identify *dw*-BPMN partitions using gateway pairs and node analysis, so the number of divergent gateways must equal the number of convergent gateways. Our principles for partitioning the *dw*-BPMN model are as follows.

- 1) The start event is used as the initial node, and the element between it and the first divergent gateway as the first element group.
- 2) Use the end event as the end node and the element between it and the last convergent gateway as the last-element group.
- 3) If the model contains a gateway-nesting relationship, it is divided into inner and outer gateway pairs. That is, elements between inner gateway pairs contain that gateway pair as a child layer element group and are represented as a black box when the outer gateway pair is divided; elements between outer gateway pairs contain that gateway pair and a black box as a parent layer.
- 4) If the model does not contain gateway nesting relationships, it is divided directly by gateway pairs.

In Algorithm 1, lines 3 to 4 are used to detect whether the model is a BPMN process model. Lines 5 to 24 are the specific process of partitioning the model, which treats each node as a variable  $n$  and the partition marker name  $\{n, partition\_id\}$ . In lines 5 to 7, nodes are identified individually from node  $E_s$  to node  $E_e$ . The  $E_s$  start event is used as the initial node and the  $E_e$  end event is used as the end node. where the element list ( $EL$ ) denotes the list of elements used to collect and control the nodes, and  $n$  is a variable in the  $EL$  of the element list. Lines 8 to 12 indicate that if  $n$  is the first

### Algorithm 1 Splitting Rules for *dw*-BPMN Process Model

```

1 Require: BPMN is a process diagram.
2 Ensure: dw-BPMN process model.
3 If BPMN model is without pool then set  $Pl := 1$ 
4 End If
5 for  $EL := label(n) \rightarrow Es$ 
6 do  $EL = (Es; Ee)$ 
7    $n := choose\ the\ node\ from\ EL$ 
8   If  $n$  is a gateway
9     If  $n$  is the first divergent gateway  $N_g1$ 
10    then all elements from the initial node  $E_s$  to
11    node  $N_g1$  as the 1st element group
12    set  $partition\_id = 1$  (initial)
13     $label\_name = \{N_g1, 1\}$ 
14    End if
15    Else if  $n$  is the last convergent gateway  $N_gj$ 
16    set  $partition\_id = increase\ partition$ 
17    number (last)
18     $label\_name = \{N_gj, partition\_id\}$ 
19    End if
20    Else if  $n$  does not a pairwise of previous node
21    set  $partition\_id = increase\ partition$ 
22    number (child)
23     $label\_name = \{n, partition\_id\}$ 
24    Else
25    set  $partition\_id = increase\ partition$ 
26    number (parent)
27     $label\_name = \{n, partition\_id\}$ 
28    End if
29  End if

```

divergent gateway  $N_g1$ , then all the elements from the initial node  $E_s$  to node  $N_g1$  are used as the first element group and the element group name is  $\{N_g1, 1\}$ . Lines 13 to 16 indicate that if  $n$  is the last converged gateway  $N_gj$ , then the element between the end node  $E_e$  and node  $N_gj$  contains the end event element as the JTH model element group and the element group name is  $\{N_gj, partition\_id\}$ . Lines 17 to 23 indicate that if  $n$  is not a paired node of the previous node, it is divided into a new part called a substratum. If  $n$  is a paired node of the previous node, it forms a gateway pair with the previous gateway node and the element between them contains the gateway pair as a father hierarchy. Clearly, the complexity of Algorithm 1 is determined by one loop from rows 5 to 7, the complexity of the algorithm is  $O(|n|)$ , where  $|n|$  is the number of data elements. A graphical depiction of the splitting process is shown in Fig. 3.

### B. MAPPING RULES FOR *dw*-BPMN PROCESS MODEL TO CPN MODEL

The *dw*-BPMN process model considers control-flow elements and data-flow elements, and its core elements include events, gateways, sequence flows, data, messages, and tasks. On the other hand, the CPN model includes component units such as place, transition, color set, arc expression function,

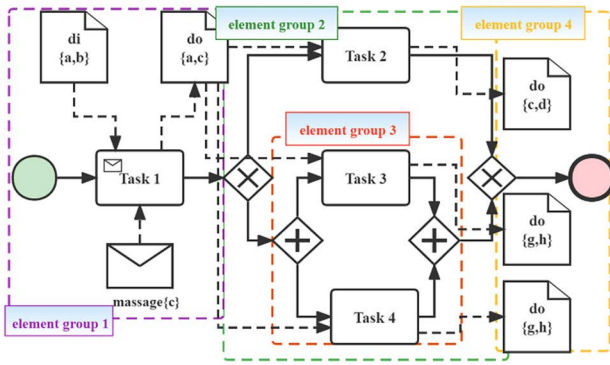


FIGURE 3. Splitting rules of dw-BPMN process model.

and guard function. The relationship between the dw-BPMN elements and CPN elements can be illustrated by the formal definition and structural description of the mapping. Therefore, we propose corresponding mapping rules. After the dw-BPMN process model is split, each dw-BPMN model element group can be converted into a CPN model block according to the mapping rules.

It is known that  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$  and  $M_{CPN} = (P, T, AC, \Sigma, V, f_c)$ , let  $\varphi$  be the mapping function from the dw-BPMN process model to the CPN model, namely  $\varphi : M_{dw-BPMN} \xrightarrow{\varphi} M_{CPN}$ . The mapping rules are as follows.

- 1)  $ID \xrightarrow{\varphi} V$ , The item definitions of the sets of variables and data structures declared using BPMNML in the dw-BPMN process model are mapped to the variables declared using CPNML in the CPN model. They are used to determine the type of place in the CPN model.
- 2)  $Es \xrightarrow{\varphi} E_{scpn}$ , and  $E_{scpn} = (p_s, p1, t_s, \Sigma1)$ . The start event of the dw-BPMN process model is mapped to the start event fragment of the CPN model, where the output set defined in the dw-BPMN start event attribute is used to create the CPN color set to determine the type of place. The transformation process is illustrated in Fig. 4.

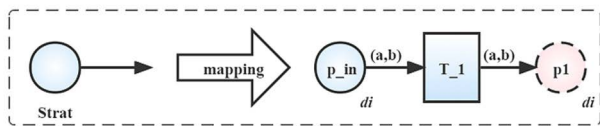


FIGURE 4. Mapping rule of Start event.

- 3)  $E_e \xrightarrow{\varphi} E_{ecpn}$ , and  $E_{ecpn} = (p_e, p2, t_e, \Sigma2)$ . The end event of the dw-BPMN process model is mapped to the end event fragment of the CPN model, where the input set defined in the dw-BPMN end event attribute is used to create the CPN color set to determine the type of places. The transformation process is illustrated in Fig. 5.

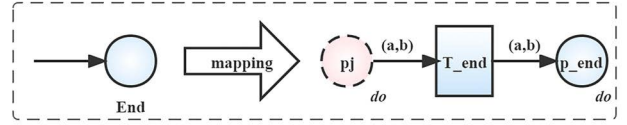


FIGURE 5. Mapping rule of end event.

- 4)  $A_{dw-BPMN} \xrightarrow{\varphi} A_{CPN}$ . The input and output sets of a BPMN task are defined by its properties. Both its input and output sets may contain data objects and messages; therefore, the tasks in the dw-BPMN process model can be divided into the following three categories for conversion, i.e., tasks containing only input and output data, tasks containing input and output data and input messages, and tasks containing input and output data and output messages. The three-class transition rule is shown in Fig.6. The input data is mapped to the color set before and after the ‘read’ transform and the color set before the ‘write’ transition, and the output data is mapped to the color set after the ‘write’ transition; the input of a message is mapped to the output of a new place, and the output of a message is mapped to the output of a new place.

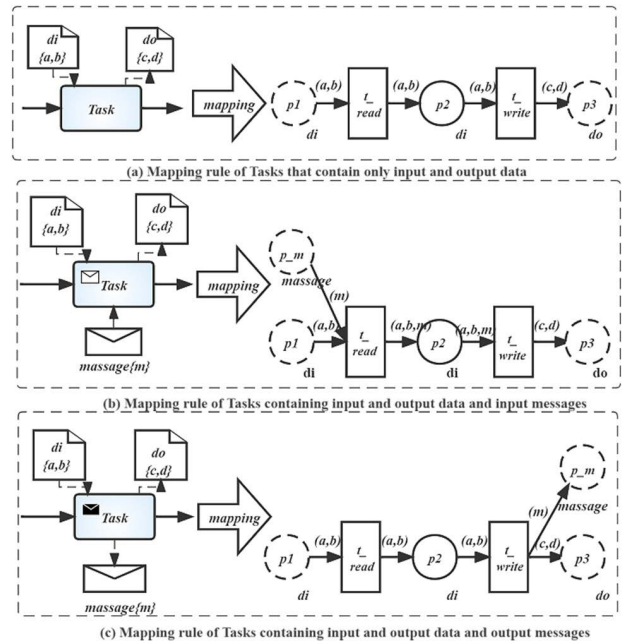
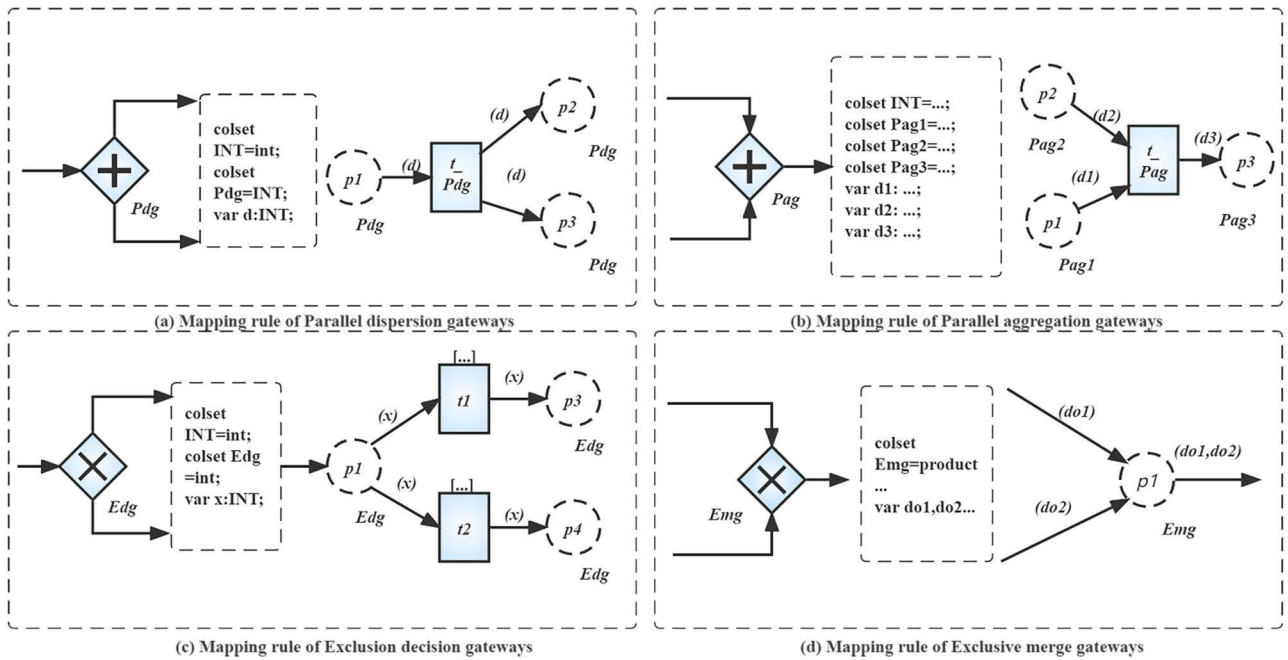


FIGURE 6. Mapping rules of Tasks. (a) Mapping rule of tasks containing only input and output data. (b) Mapping rule of tasks containing input and output data and input messages. (c) Mapping rule of tasks containing input and output data and output messages.

- 5)  $G_{dw-BPMN} \xrightarrow{\varphi} G_{CPN}$ . The gateway fragments in the dw-BPMN process model are mapped to the CPN model fragments. The gateways of the dw-BPMN process model can be divided into divergent and convergent. Divergent gateways are divided into parallel divergent gateways, exclusion data decision gateways,



**FIGURE 7. Mapping rules of gateway. (a) Mapping rule for parallel dispersion gateways. (b) Mapping rule for parallel aggregation gateways. (c) Mapping rule for exclusion decision gateways. (d) Mapping rule of Exclusive merge gateways.**

and exclusion event decision networks, which divided into parallel convergent gateways, exclusive data convergent gateways, and exclusive event convergent gateways. The mapping of data-based BPMN divergent gateways involves the mapping of BPMN gateway elements and guard functions on sequence flows to CPN model segments. In this case, the sequence flow of the divergent gateway is mapped to the CPN transition and the guard function is mapped to the guard function on the CPN transition. However, the convergent gateway does not need to map the guard function to the transition guard function in the sequence flow. The specific transformation rules are illustrated in Fig. 7.

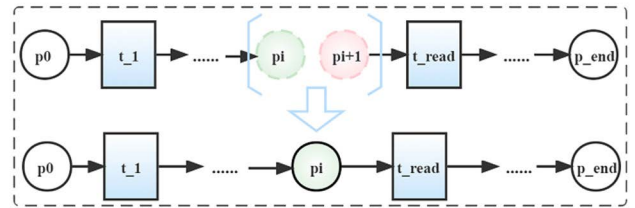
- 6)  $F \xrightarrow{\varphi} A$ . The sequence flow in the dw-BPMN process model is mapped to the arcs in the CPN model.

### C. COMBINING RULES FOR CPN ELEMENT GROUPS

The dw-BPMN process model forms a finite group of CPN elements after the splitting and mapping processed are completed.

The only way to create a complete CPN model is to structurally connect these model element groups, for which a combination rule is proposed. The transformation rules are illustrated in Fig. 8.

The formal definition of the combining rule is as follows. Let  $CPN1 = (P1, T1, Ac1, \Sigma1, V1, fc1)$  and  $CPN2 = (P2, T2, Ac, \Sigma2, V2, fc2)$  be two CPN element blocks. If  $p1 \in P1$  is the output depot of  $CPN1$ ,  $p2 \in P2$  is the input depot of  $CPN2$ ,  $p1$  and  $p2$  are the common places of  $CPN1$  and  $CPN2$ , then maintaining only one place of



**FIGURE 8. The rules for combining CPN element groups.**

$p1$  and  $p2$  makes the group of  $CPN1$  and  $CPN2$  elements connected to form a CPN model, namely  $M_{CPN} = (P0, T0, Ac0, \Sigma0, V0, fc0)$ .

The model satisfies:

- 1)  $P0 = (p1 \cup p2) \setminus \{p1\}$ ;
- 2)  $T0 = T1 \cup T2$ ;
- 3)  $Ac0 = Ac1 \cup Ac2$ ;
- 4)  $\Sigma0 = \Sigma1 \cup \Sigma2$ ;
- 5)  $V0 = V1 \cup V2$ ;
- 6)  $fc0 = \{fcc0, fgg0, fea0, fi0\}$ .

Where  $fcc0 = (fcc1 \cup fcc2) \setminus \{(t1, col) | col = fcc(p1)\}$ ;  
 $fgg0 = fgg1 \cup fgg2$ ;

$$\begin{aligned}
 fea0 &= (feaA1 \cup feaA2) \setminus \{(t1, p1), a_{ex} | a_{ex} \\
 &= fea(t1, p1) \wedge t1 \in T1\} \cup \{(t1, p1), a_{ex} | \\
 &\quad \times ((t1, p1) \in Ac1 \wedge t1 \in T1 \wedge a_{ex} \\
 &= fea((t1, p1)) * fea(p2, t2) \setminus (fea(t1, p1)) \wedge t2 \\
 &\quad \in T2 \wedge (p2, t2) \in Ac2\}; \\
 fi0 &= fi1 \cup fi2.
 \end{aligned}$$

The steps for combining model element groups are as follows.

- 1) First, all CPM model element groups were arranged in order of occurrence of the original dw-BPMN model element groups.
- 2) Then, the input place or output place shared among the element groups is found in order according to the sequence.
- 3) Finally, only one of the common pairs of output and output places is maintained, so that the element groups are connected to form a CPN model.

## V. DATA FLOW ERROR DISCOVERY MECHANISM BASED ON CPN MODEL

The validation of the correctness of the dw-BPMN process model consists of two aspects: on the one hand, the validation of the correctness of the model structure, i.e., the detection of model control flow errors; on the other hand, the validation of the correctness of the data aspect, i.e., the detection of model data flow errors. The main errors in the control flow include deadlock, live lock, and dead tasks in the model, and the main errors in the data flow include missing data, lost data, redundant data, and inconsistent data in the model. In this paper, we only analyze and discuss the detection methods and correction strategies for the four types of errors in data flow, and further derive the basis for correctness analysis.

### A. TAXONOMY OF DATA FLOW ERRORS

We classify the data flow errors of the dw-BPMN model as follows.

#### 1) MISSING DATA

If data  $d$  is the input data of a task, but there is no task output corresponding to the output data before the input data of that task, that is, data  $d$  is not initialized, the model generates missing data.

*Definition 4 (Missing Data):* Suppose  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$ ,  $\forall d \in D$ , If (1)  $|D_i(d)| = \emptyset$ ; (2)  $|D_o(d)| \neq \emptyset$ , then this dw-BPMN process model has an error due to missing data.

In Definition 4, the first condition indicates that there is no input data  $d$ , that is, it is not initialized, and the second condition indicates that data  $d$  will be used by at least one subsequent task. The presence of missing data errors in the model leads to business process system anomalies and suspension. Therefore, detection and correction of the BPMN model is necessary.

#### 2) LOST DATA

If data  $d$  is the output data of multiple tasks, but there is no input data  $d$  before data  $d$  is used as the output data of another task later, the model generates data loss.

*Definition 5 (Missing Data):* Suppose  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$  and  $\forall d \in D$ . If (1)  $|D_i(d)| \neq \emptyset$ ,

and  $|D_o(d)| \neq \emptyset$ ; (2)  $|D_i(d)| < |D_o(d)|$ . Thus, there is a lost-data error in the dw-BPMN process model.

In Definition 5, the first condition indicates the presence of at least one task output data  $d$  and at least one task input data  $d$ ; and the second condition indicates that the number of tasks with output data  $d$  is greater than the number of tasks with input data  $d$ , that is, the input of data  $d$  is missing from the model pass. The presence of missing data errors in the model can lead to interruptions in the process of data delivery. Therefore, the BPMN model must be corrected before the system can be built.

#### 3) REDUNDANT DATA

If data  $d$  is the output data of one task, but not all subsequent tasks in the execution path use data  $d$  as input data, the model will have data redundancy.

*Definition 6 (Redundant Data):* Suppose  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$  and  $\forall d \in D$ . If (1)  $|D_o(d)| \neq \emptyset$ ; (2)  $|D_i(d)| = \emptyset$ , then the dw-BPMN process model has the error of redundant data.

In Definition 6, the first condition indicates the presence of at least one task outputting data  $d$ ; the second condition indicates that none of the tasks uses data  $d$  as input data in subsequent tasks.

#### 4) INCONSISTENT DATA

The presence of redundant data errors in the model may lead to an inefficient and more costly system, thus, the prior detection and correction of this error is essential.

*Definition 7 (Inconsistent Data):* Suppose  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$  and  $\forall d \in D$ . If (1)  $|D_i(d)| = \emptyset$ ; (2)  $|D_o(d)| \neq \emptyset$ ; and (3)  $|D_i(d)| \geq 2$ , then the dw-BPMN process model has an error of inconsistent data.

In Definition 7, the first and third conditions indicate that data  $d$  is initialized by multiple tasks and the second condition indicates that data  $d$  is used by at least one task. Inconsistent data errors in the model may lead to uncertainty and confusion, resulting in system operation errors. Therefore, errors must be corrected before designing the business system.

### B. DATA FLOW ERRORS DETECTION METHOD

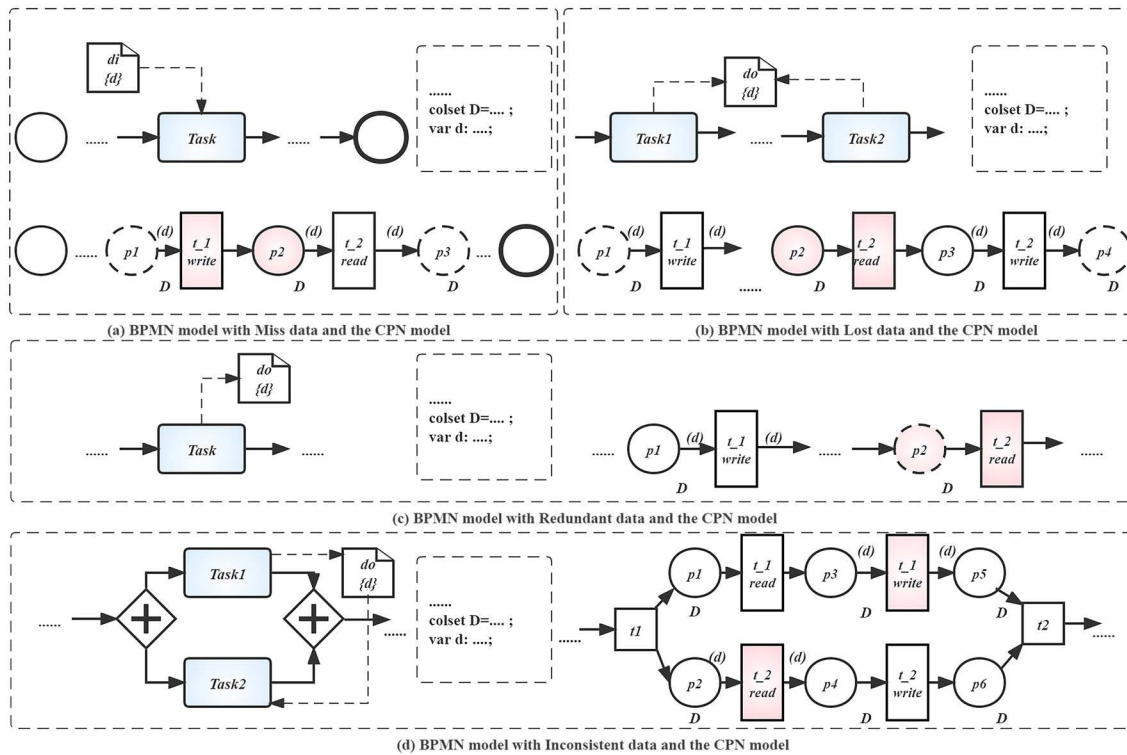
The above four types of errors generated by the data flow in the dw-BPMN process model can be stated as problems of the arc expression function or variations transfer in the CPN model. A detailed analysis is presented in Fig. 9.

The specific analysis is as follows.

For problem (a): the problem of missing data in the BPMN model is regulated by the absence of data variable  $d$  in all arc functions before the ' $t\_write$ ' transition in the corresponding CPN model. This problem can be detected by checking whether the arc functions before and after the ' $read$ ' transition in the CPN model generated by the mapping are empty.

For problem (b): The problem of lost data in the BPMN model is regulated by the absence of the data variable  $d$  in the previous arc function of a ' $t\_read$ ' transition in the corresponding CPN model. This problem can be detected





**FIGURE 9.** Data flow errors in the BPMN and CPN models. (a) BPMN model with missing data and the CPN model. (b) BPMN model with lost data and the CPN model. (c) BPMN model with redundant data and the CPN model. (d) BPMN model with inconsistent data and the CPN model.

by checking whether the previous arc function of the ‘read’ transition of the CPN model generated by the mapping is empty.

For problem (c): The problem of redundant data in the BPMN model is statutorily defined as the data variable  $d$  in the corresponding CPN model is not used in the transmission process, i.e., the data  $d$  no longer appears in the arc function after a certain ‘ $t\_write$ ’ transition in the CPN model. The CPN model generated by mapping can be verified for path failures caused by inconsistent variables.

For problem (d): The problem of inconsistent data in the BPMN model can be statistically defined as a function of the ‘ $t\_read\_2$ ’ transition and the ‘ $t\_write\_2$ ’ transition of the variable  $d$  in the corresponding CPN model, i.e., the same data The same data is being initialized and read, causing the most basic logical error. To detect this type of problem, we can check whether the CPN model generated by mapping has a path failure caused by the wrong variable order.

The correctness of the generated CPN executable model is further reflected by the correctness of the dw-BPMN process model, based on the detection methods of the possible data flow errors of the dw-BPMN process model described above.

### C. CORRECTION STRATEGIES FOR DATA FLOW ERRORS

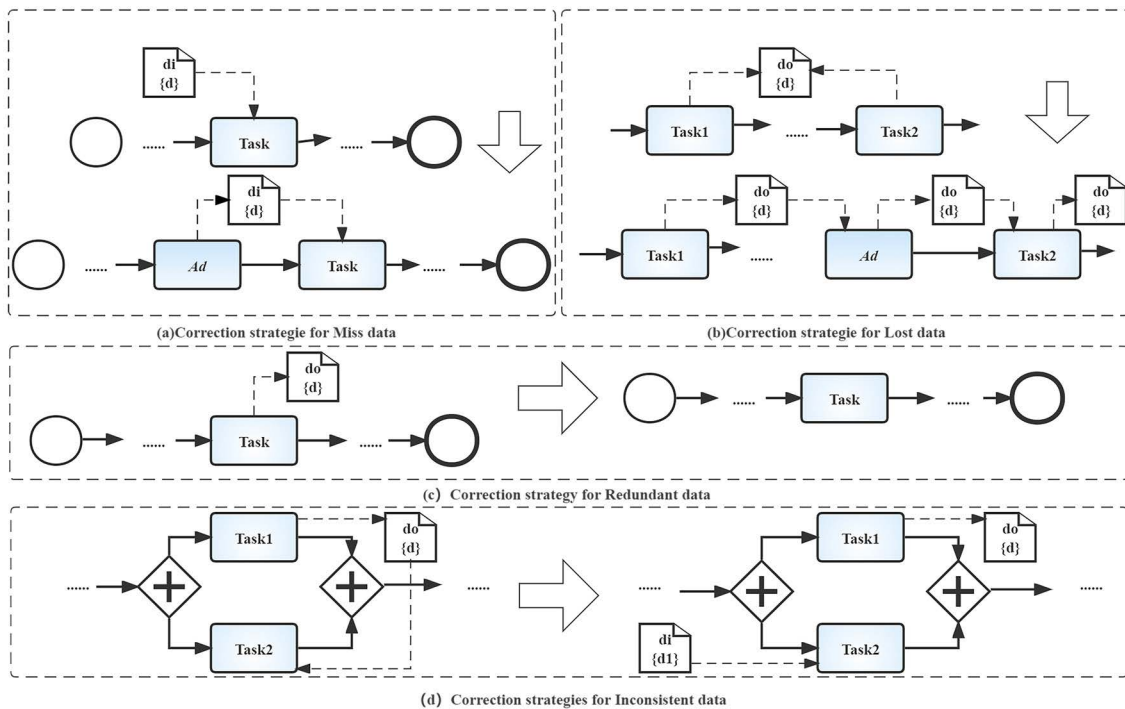
According to the above detection methods for data flow errors in the dw-BPMN process model, errors in the model can be

accurately detected, which facilitates the correction of the model. Next, we propose four correction strategies corresponding to the four types of data-flow errors. The specific strategies are as follows:

*Strategy 1:* Let  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$ ,  $\exists d \in D$ , and that data  $d$  is a missing data element, i.e., uninitialized. The BPMN task activity  $A_d$ , which generates data  $d$ , should be added to the model to satisfy  $A = A \cup \{A_d\}$  and  $F = F \cup \{(A_d, d)\}$ . As shown in Fig. 10(a), data  $d$  in the model is used by  $Task$ , but data  $d$  is uninitialized, that is, task output data  $d$  is missing. The problem of missing data in the model can be solved by adding a task  $A_d$ , before  $Task$ .

*Strategy 2:* Let  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$ ,  $\exists d \in D$ , and data  $d$  is a lost data element. The intermediate BPMN task with no input data  $d$  is active as  $A_d$  and the previous task outputs data  $d$ . Then, the data  $d$  is directly connected to the task  $A_d$  and satisfies  $F = F \cup \{(A_d, d)\}$ . As shown in Fig. 10(b), the data  $d$  in the model is the output of  $Task 1$  and the output of  $Task 2$ , but  $Task 2$  is lost the input of data  $d$ . To correct the error of lost data in the model, the input of data  $d$  should be added for  $Task 2$ .

*Strategy 3:* Let  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$ ,  $\exists d \in D$ , and data  $d$  is a redundant data element. The task of outputting redundant data is called  $A_d$ , and the redundant data element  $d$  should be removed from the model to satisfy  $D = D - \{d\}$  and  $F = F - \{(A_d, d)\}$ . As shown in strategy 3



**FIGURE 10.** Data flow errors in the BPMN and CPN models. (a) BPMN model with missing data and the CPN model. (b) BPMN model with lost data and the CPN model. (c) BPMN model with redundant data and the CPN model. (d) BPMN model with Inconsistent data and the CPN model.

in Fig. 10(c), the *Task* outputs data  $d$ , and it is not used as input data for any subsequent task. To correct the errors of redundant data in the model, data  $d$  can be deleted directly.

*Strategy 4:* Let  $M_{dw-BPMN} = (N, A, E, ID, D, M, G, F, f)$ ,  $\exists d \in D$ , and data  $d$  is an inconsistent data element. The required data elements can be added according to the different task activity requirements. As shown in Fig. 10(d), data  $d$  is both the output of *Task1* and the input of *Task2*, which is not logical; therefore, the appropriate data  $d1$  needs to be added for *Task2*.

## VI. CASE STUDY

Since the outbreak of COVID-19, the effective combination of intelligent robotics and medical technology has significantly increased the efficiency of the screening process in the fight against the epidemic. With the demand for smart kindergartens and the implementation of epidemic prevention and control guidance, an increasing number of kindergartens have chosen morning-check robots with intelligent temperature detection to conduct morning checks for students.

At the same time, the bases of today's competitive and highly dynamic marketplace, business process management strategies and BPMN process modeling have been widely used in large organizations to ensure continuous improvement of business processes and their adaptation to change. Currently, the kindergarten smart morning check robot is equipped with high-definition camera, high-definition display, infrared sensor, and bass speaker, thus having the

functions of face recognition, report display, body temperature measurement, and alarm alerts. The operational organization of the temperature detection function of the robot is illustrated in Fig. 11.

Next, we modeled and analyzed the dw-BPMN process model for body temperature detection of a kindergarten smart morning check robot using a CPN-based executable modeling approach, thus verifying the feasibility of our proposed approach. The dw-BPMN process model for body temperature detection is shown in Fig. 12; the names of the elements are described in Table 1, and the variable names are listed in Table 2.

**TABLE 1.** Descriptions of activity dw-BPMN process model elements.

elements	names
d1	Basic student information
d2	Student name
d3	Temperature report
d4	report1
d5	report2
d6	report3
Task1	Identity verification
Task2	Input temperature
Task3	Normal temperature report
Task4	Abnormal temperature report
Task5	Abnormal body temperature alert
message1	Information
message2	Temperature

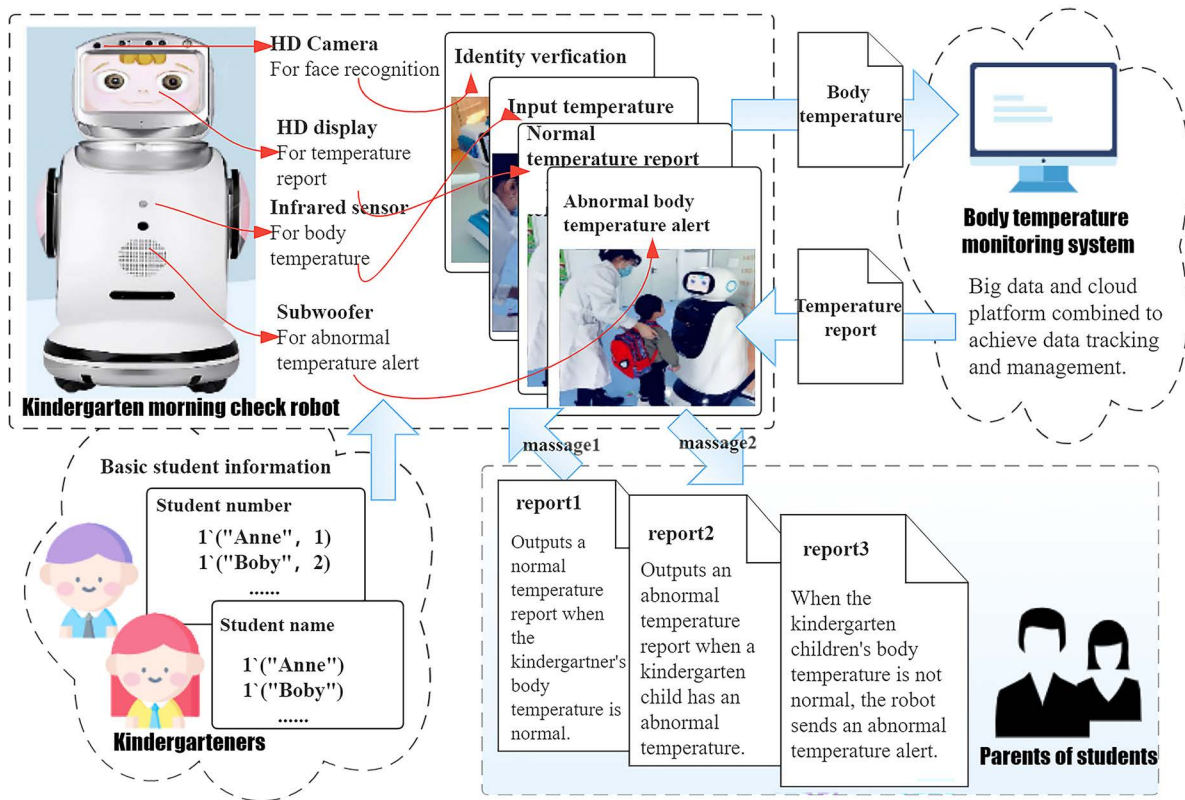


FIGURE 11. The organization of temperature detection system of intelligent morning check robot in kindergarten.

TABLE 2. Description of dw-BPMN process model data elements.

elements	names
n	Student name
p	Student number
r	Body temperature
t	test number
t	Body temperature

The first step of modeling starts by dividing the dw-BPMN process model into element groups according to the splitting rules. The splitting result is shown in Fig. 12, which divides the model into four element groups. The first element group is  $E_G1 = \{E_s, Task1, Task2, d1, d2, d3, mass - age1, message2, F1\}$ , where  $E_s$  is the start event, and  $F1$  is the connected element of the control and data streams in  $E_G1$ . The second element group is  $E_G2 = \{Task3, d3, d4, G2, F2\}$ , where  $F2$  is the connecting element for the control and data streams of  $E_G2$  and  $G2$  is the exclusion gateway group in  $E_G2$ . The third element group is  $E_G3 = \{Task4, Task5, d3, d5, d6, G3, F3\}$ , where  $F3$  is the connecting element for the control and data streams of  $E_G3$  and  $G3$  is the parallel gateway group in  $E_G3$ . The fourth element group is  $E_G4 = \{E_e, F4\}$ , where  $E_e$  is the end event, and  $F4$  is the connecting element of the control flow and data flow of  $E_G4$ .

In the second modeling step, the split element groups of the dw-BPMN process model were converted according to the mapping rules. The conversion process is shown in Fig. 13, and the above four element groups are mapped to four CPN element groups.

In the third step of modeling, the four dw-BPMN element groups were combined according to the combination rules. According to the combination rule, the shared place is combined with  $SP = \{(p7, p8), (p11, p20), (p18, p21), (p19, p20)\}$ , where  $SP$  is the shared location set, and each group of shared places is limited to keep one of them. And the combination result is shown in Fig. 14.

Following the above three-stage modeling approach, we converted the dw-BPMN process model of the temperature detection system into an executable CPN model. Next, the CPN model is verified for errors in data flow by checking whether the arc functions before and after the ‘ $t\_read$ ’ and ‘ $t\_write$ ’ transitions in the CPN model are empty and whether there are inconsistencies in the variables and logical errors in that order.

As shown in Fig. 14, the generated CPN model did not contain the above four types of errors. In order to further verify the data consistency and logical consistency of the model with the expected system, we cited the example of two kindergartener temperature detection in kindergarten. And after analyzing its operation path and results, we found

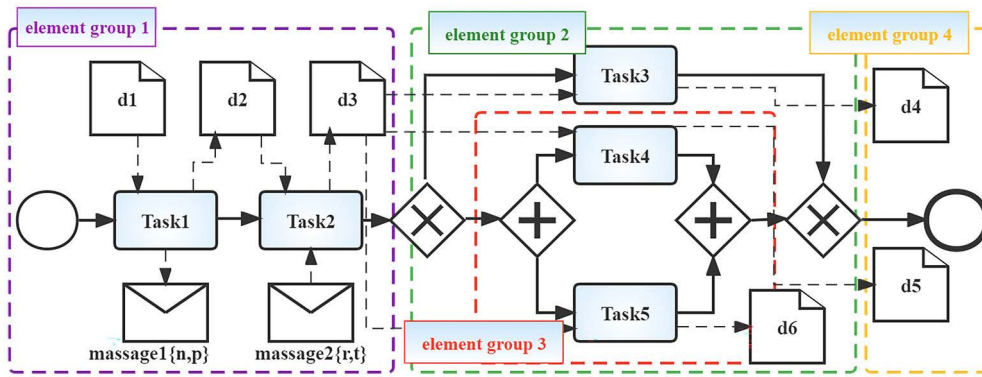


FIGURE 12. The dw-BPMN process model of temperature detection and Model Partition.

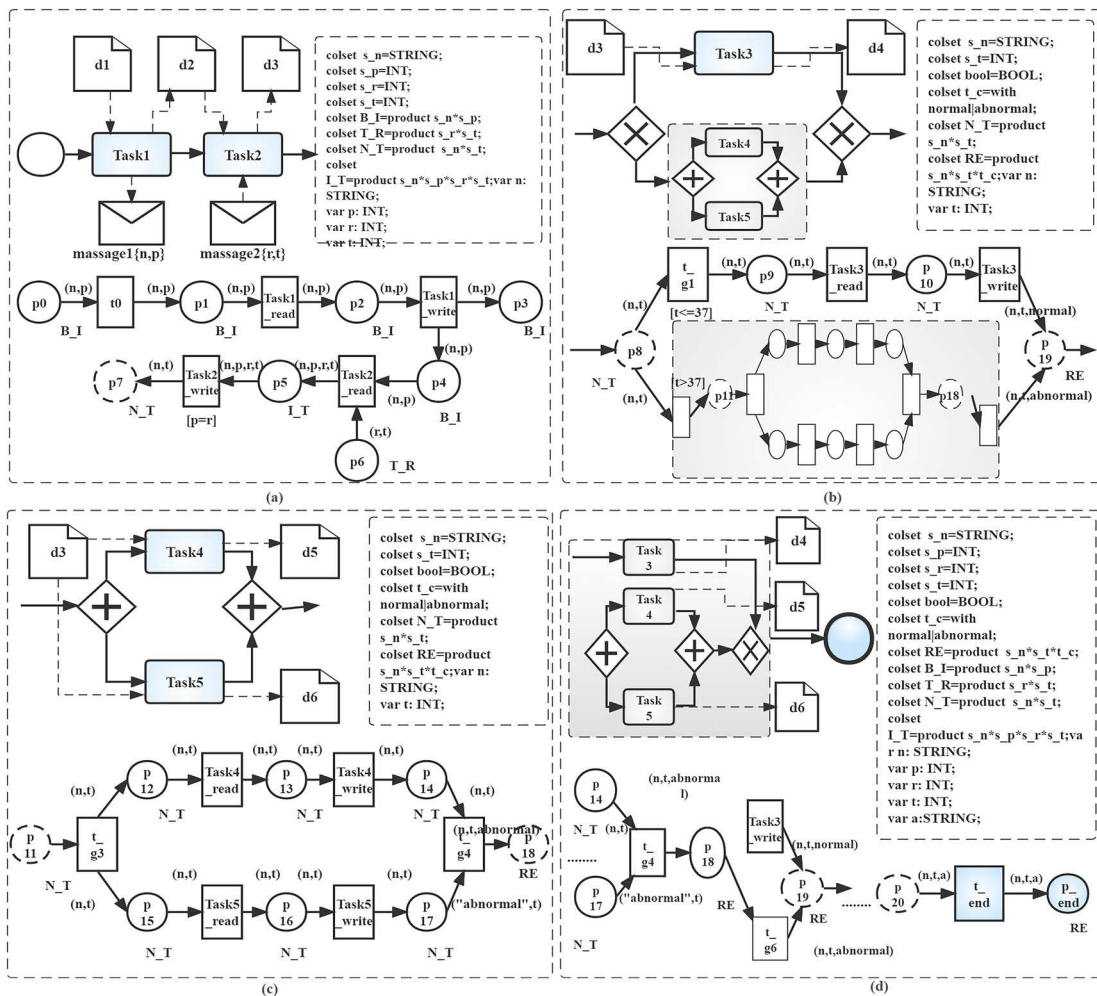


FIGURE 13. Transformation of the dw-BPMN process model to the CPN model. (a) Mapping of dw-BPMN model element group 1 to CPN model. (b) Mapping of dw-BPMN model element group 2 to CPN model. (c) Mapping of dw-BPMN model element group 3 to CPN model. (d) The mapping of dw-BPMN model element group 4 to CPN model.

that the model is consistent with the expectation, and then analyzed its generated state space report. We found that the

model does not have the problems of deadlock, live lock, and dead change, as shown in Table 3.





- [9] S. Rova, P. Meire, F. Müller, M. Simeoni, and F. Pranovi, "A Petri net modeling approach to explore the temporal dynamics of the provision of multiple ecosystem services," *Sci. Total Environ.*, vol. 655, pp. 1047–1061, Mar. 2019.
- [10] W. Zhijun, M. Haolin, and Y. Meng, "Reliability assessment model of IMA partition software using stochastic Petri nets," *IEEE Access*, vol. 9, pp. 25219–25232, 2021.
- [11] A. Kheldoun, K. Barkaoui, and M. Ioualalen, "Formal verification of complex business processes based on high-level Petri nets," *Inf. Sci.*, vols. 385–386, pp. 39–54, Apr. 2017.
- [12] E. Díaz, J. I. Panach, S. Rueda, and J. Vanderdonckt, "An empirical study of rules for mapping BPMN models to graphical user interfaces," *Multimedia Tools Appl.*, vol. 80, no. 7, pp. 9813–9848, Mar. 2021.
- [13] F. Corradini, F. Fornari, A. Polini, B. Re, F. Tiezzi, and A. Vandin, "A formal approach for the analysis of BPMN collaboration models," *J. Syst. Softw.*, vol. 180, Oct. 2021, Art. no. 111007.
- [14] I. Zafar, F. Azam, M. W. Anwar, B. Maqbool, W. H. Butt, and A. Nazir, "A novel framework to automatically generate executable web services from BPMN models," *IEEE Access*, vol. 7, pp. 93653–93677, 2019.
- [15] F. Durán, C. Rocha, and G. Salaün, "Stochastic analysis of BPMN with time in rewriting logic," *Sci. Comput. Program.*, vol. 168, pp. 1–17, Dec. 2018.
- [16] S. Houhou, S. Baair, P. Poizat, P. Quéinnec, and L. Kahloul, "A first-order logic verification framework for communication-parametric and time-aware BPMN collaborations," *Inf. Syst.*, vol. 104, Feb. 2022, Art. no. 101765.
- [17] M. Gudemann, P. Poizat, G. Salaun, and L. Ye, "VerChor: A framework for the design and verification of choreographies," *IEEE Trans. Services Comput.*, vol. 9, no. 4, pp. 647–660, Jul. 2016.
- [18] L. E. Mendoza, M. I. Capel, and M. A. Pérez, "Conceptual framework for business processes compositional verification," *Inf. Softw. Technol.*, vol. 54, no. 2, pp. 149–161, Feb. 2012.
- [19] F. U. Muram, H. Tran, and U. Zdun, "Supporting automated containment checking of software behavioural models using model transformations and model checking," *Sci. Comput. Program.*, vol. 174, pp. 38–71, Apr. 2019.
- [20] F. Corradini, A. Polini, and B. Re, "Inter-organizational business process verification in public administration," *Bus. Process Manage. J.*, vol. 21, no. 5, pp. 1040–1065, Sep. 2015.
- [21] Z. Li and Z. Ye, "A Petri nets evolution method that supports BPMN model changes," *Scientific Program.*, vol. 2021, pp. 1–16, Jun. 2021.
- [22] C. Dechsupa, W. Vatanawood, and A. Thongtak, "Stepwise verification for the BPMN with timed and stochastic process using a colored generalized stochastic Petri net," *IEEE Access*, vol. 10, pp. 42983–43002, 2022.
- [23] S. Meghzili, A. Chaoui, M. Strecker, and E. Kerkouche, "An approach for the transformation and verification of BPMN models to colored Petri nets models," *Int. J. Softw. Innov.*, vol. 8, no. 1, pp. 17–49, Jan. 2020.
- [24] T. M. Maarouk, E. Merah, S. Ghaoui, and N. Rahabi, "Formal semantics and transformation of BPMN models," *Int. J. Bus. Process Integr. Manage.*, vol. 9, no. 3, p. 158, 2019.
- [25] C. Dechsupa, W. Vatanawood, and A. Thongtak, "Transformation of the BPMN design model into a colored Petri net using the partitioning approach," *IEEE Access*, vol. 6, pp. 38421–38436, 2018.
- [26] C. Dechsupa, W. Vatanawood, and A. Thongtak, "Hierarchical verification for the BPMN design model using state space analysis," *IEEE Access*, vol. 7, pp. 16795–16815, 2019.
- [27] Z. Ma, G. Zhu, and Z. Li, "Marking estimation in Petri nets using hierarchical basis reachability graphs," *IEEE Trans. Autom. Control*, vol. 66, no. 2, pp. 810–817, Feb. 2021.
- [28] L. Han, K. Xing, X. Chen, and F. Xiong, "A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems," *J. Intell. Manuf.*, vol. 29, no. 5, pp. 1083–1096, Jun. 2018.
- [29] 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.
- [30] D. Xiang, G. Liu, and C. Yan, "Detecting data inconsistency based on the unfolding technique of Petri nets," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 2995–3005, Dec. 2017.



**FENGLAN HUANG** received the B.S. degree in mathematics and applied mathematics from the Shanghai Institute of Technology, China, in 2021. She is currently pursuing the master's degree in systems science with the University of Shanghai for Science and Technology, China. Her research interests include business process management, system architecture design, and Petri net.



**FENG NI** received the B.S. degree from the Harbin Institute of Technology, China, in 2005, and the M.S. and Ph.D. degrees from the Department of Control Science and Engineering, Huazhong University of Science and Technology, China, in 2008 and 2012, respectively. He is currently an Associate Professor at the University of Shanghai for Science and Technology, China. His research interests include business process management, system architecture design, discrete-event systems, and Petri net.



**JIANG LIU** received the B.S. degree in mathematics and applied mathematics from Nanjing University, China, in 2004, and the M.S. and Ph.D. degrees in applied mathematics from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China, in 2009. She was a Visiting Scholar at the School of Mathematics, University of Minnesota, USA, from 2019 to 2021. She is currently an Associate Professor at the University of Shanghai for Science and Technology, China. Her research interests include symbolic computation, nonlinear systems, and deep learning.



**FAN YANG** received the B.S. degree from East China Normal University, China, in 2010. She is currently an Early Childhood Second-Grade Teacher at Shanghai Sunshine City Kindergarten, Jinshan, Shanghai, China. Her main research interest includes early childhood health management.



**JIAYI ZHU** is currently pursuing the bachelor's degree with the University of Shanghai for Science and Technology, China. His research interests include business process management and system optimization.

...