Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 8 (2): 58-74, 2014
ISSN 1819-4311 / DOI: 10.3923/jse.2014.58.74

Analyzing BPMN with Extended Object Petri Net

[1]Ruiqiang Yu, [1]Zhiqiu Huang, [2]Lin Wang and [2]Hongjie Zhang
[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2]Yantai HaiYi Software Co. Ltd., Yantai, China

Corresponding Author: Ruiqiang Yu, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

ABSTRACT

Business Process Modeling Notation (BPMN) is the most influential graphical modeling notation in service composition aspect and has been widely used in modeling the Web service composition system, however BPMN lacks of formal semantics and can not be verified formally and automatically. Extended object Petri net (EOPN for short) is presented in order to model and verify BPMN process formally. Guard, flow valve and time constraint are introduced into EOPN. Because state class method for analyzing time Petri net is destitute of global temporal constraints, timestamp state class method is developed and the corresponding analysis approach is also presented. The enabled conditions of transition are listed and the firing condition and rules of transition of EOPN are discussed in detail. Mapping rules from BPMN to EOPN are depicted in detail. An example is provided for illustrating the feasibility of mapping BPMN process diagram to EOPN model and verifying its rightness formally.

Key words: BPMN, formal method, object net, timing constraints, state class

INTRODUCTION

Web service has become the prominent paradigm for electronic business and interoperable applications across heterogeneous systems based on Service Oriented Architecture. Although many works focus on formal checking WS-BPEL (Hinz *et al*., 2005), it's too late in this coding stage. Formal checking and verification should be carried out in business process modeling stage.

BPMN OMG (2006, 2008) is developed for process modeling. Because BPMN lacks formal semantics, there may be some semantic errors (for example deadlock, conflict) in BPMN diagram. A formal method is expected for checking and verifying BPMN diagram so that the errors mentioned above could be detected before being corrected.

Petri net is widely applied in system modeling and protocol verification (Valk and Girault, 2003; Van der Aalst and Ter Hofstede, 2000; Zhehui, 2006; Yi, 2005). Van der Aalst and Ter Hofstede (2000) concentrates on mapping workflow to Petri net. Different from workflow, BPMN can model business process cross-organization (Jian and Yanbo, 2006). Dijkman's work focuses on mapping BPMN model to Petri net for semantic analysis (Dijkman *et al*., 2008). But there are some flaws in that research:

- Compared with UML and XPDL, the advantage of BPMN is that business process can cross companies or organizations, but it is not reflected in their study

- Message flow and Sequence flow are two different objects, but the mapping rules don't address the difference
- The mapping rules for some BPMN object should be corrected and refined, such as gateway, trigger, etc.,
- The mapping model which is presented in that study (Dijkman *et al.*, 2008) is a little coarseness, non-functional properties depicted in BPMN association aren't addressed in its mapping rules, including execution duration, execution price

In fact, each component process in BPMN model has its own time constraints. The time which is spent on each component process must be treated seriously for the cost and safety concern. In some scenarios, peoples often expect that BPMN process satisfies some global temporal constraints. Especially more and more mobile-business application are deployed on the internet and more and more portable devices, such as smart phone and iPad, involve in the business process. Based on the cost and safety considerations, the time interval which is spent on BPMN process must be treated as a determinant factor. The work in this study aims at advancing the current state of the art in technologies for BPMN modeling and verifying by addressing the temporal constraint (an important non-functional property) with Petri net.

Object Petri net (OPN for short) is proposed by Valk (1998, 2004). The core of OPN is Petri nets as token object. OPN can model the message flow between different participants, but it also neglects non-functional properties. To eliminate the flaws mentioned above, this study augments OPN to extended object Petri net (EOPN for short).

This study is devoted to modeling BPMN diagram with EOPN and verifying its correctness. In this study, a preferable method is proposed. Compared with the existing works, the following contributions are presented:

- Extended object Petri net can describe the time-related properties of process accurately
- A refined enumerative approach, which is named timestamp state class method, is presented to analyze the temporal constraints
- A holistic mapping from BPMN to EOPN is given and some flaws which exists in Dijkman's research (Dijkman *et al.*, 2008) are eliminated
- A real-life case is used to validate the timestamp state class approach

The rest of the article is structured as follows. To start off, the formal definition of EPON is presented. Besides, a timestamp state class approach about temporal constraints analysis is proposed. Furthermore, the mapping rules from BPMN to EOPN are depicted. One more thing is that an example is provided for illustrating the feasibility of this approach. Finally, some concluding remarks and further research are outlined.

EXTENDED OBJECT PETRI NET

After proposing object Petri net in Valk (1998), depicted that the idea of object Petri nets is using the Nets-within-Nets paradigm in Valk (2004). Some researchers also contribute a lot to OPN theory. Lakos (2001) used object Petri net in object oriented modeling (Lakos, 2001). Kohler and Rolke (2004) discussed the properties of object Petri net deeply and minutely. But none of them paid attention to the time constraints when they discussed object Petri net. In this study, Extend object Petri net augments OPN with followings:

- Global guard and flow valve are introduced so that the firing rules of transitions can be controlled further
- Non-functional properties are considered. For the limitation of space, only time constraint is considered

Various petri net is developed to figure temporal constraints, such as timed Petri nets (Ramchandani, 1974) and timing constraint Petri net (Yu *et al*., 2009). In EOPN, the temporal constraints of transitions are described with the method which is used in Merlin's time Petri net (Merlin, 1974).

Definition 1: A base net of EOPN is defined as $\Sigma = (P, T, Guard, F, V, K, I, M_0)$ where:

- P and T are the finite sets of places and transitions respectively
- $P \cap T = \Phi \wedge P \cup T \neq \Phi$
- $F = P \times T \cup T \times P$ is a set of arcs (flow relation)
- $K : P \rightarrow N$ is capacity function. The default value is 1 unless addressed specially
- M_0 is the initial marking of $\sum$
- Guard is a bool variable for monitoring the system action. Its initial value is set to true and the value can be modified or not when each transition fired
- V is flow valve, $V: F \rightarrow \{0, 1\}$. The value of V (f) depends on the flow relation of f
- $I: T \rightarrow R^+ \times (R^+ \cup \infty)$ is called firing time interval. Where R^+ is the set of non-negative real numbers. I is a domain and can be denoted as a domain $[\alpha, \beta]$ and $0 \leq \alpha \leq \beta$ holds

For a certain transition t_i and its firing interval $I(t_i) = [\alpha_i, \beta_i]$:

- α_i is called the earliest firing time (EFT for short)
- β_i is called the latest firing time (LFT for short)

If the firing domain $[\alpha_i, \beta_i]$ is not defined, the corresponding transition is a classical Petri net transition and the domain should be $[0, \infty)$. During the net operation, the time intervals in the firing domain will be changed. To avoid confusion, $[\alpha_t^s, \beta_t^s]$ is used to denote the firing interval of t_i in initial marking M_0 and $[\alpha_t, \beta_t]$ for other marking

Definition 2: A transition t is called being enabled by marking M if and only if all of the followings hold:

- $\forall p \in {}^\bullet t$, M (p)>0. That is every pre-place of transition i must have a token
- $V(p_I \rightarrow t) = 1 \wedge V(t \rightarrow p_o) = 1 \wedge Guard = true$, where $\forall p_I \in {}^\bullet t \wedge \forall p_O \in t^\bullet$. i.e., for the transition t, there is at least one flow arc f (p→t) are enabled ($v_{f(p \rightarrow t)} = 1$) and at least one flow arc f (t→p) are enabled ($v_{f(t \rightarrow p)} = 1$) and the guard of net should be true

Definition 3: Let τ be the time when t is enabled by marking M and θ is the relative time elapse from the time τ to the time when transition t is fired, then transition t is firable if and only if the following conditions are met:

- t is enabled
- θ is not smaller than the EFT of transition t and not greater than the smallest LFT of all the transitions enabled by marking M:

$$EFT(t) \le \theta \le \min(LFT(t_k)) \quad (1)$$

where, k ranges over the set of transitions which are enabled by M.

Definition 4: An EOPN system is a tuple OS = (SN, ON, σ, R_0, M_0) where:

- $SN = (\hat{P}, \hat{T}, \hat{Guard}, \hat{F}, \hat{V}, \hat{K}, \hat{I})$ is the system net. $\hat{P} = \hat{P}_{ob} \cup \hat{P}_{bt}$. $\hat{P}_{ob}$ is the object token places and $\hat{P}_{bt}$ is the black-token places
- $\sigma \subseteq \hat{T} \times T$ is the interaction between system net transition and object net transition
- R_0: $P \rightarrow N \cup Bag(ON)$ specifies the initial token distribution
- M_0 is the initial marking of object net
- ON can be either base net or nested SN. For simplicity, only base net is considered in this study

The transition firing rules of OPN is addressed by Valk and Girault (2003). The time constraints on transitions of EOPN will change the firing rules of transition, so these rules should be renewed.

For EOPN, let $SN = (\hat{P}, \hat{T}, \hat{Guard}, \hat{F}, \hat{V}, \hat{K}, \hat{I})$ be the system net and ON = (P, T, Guard, F, V, K, I) be object net and pair (R, M) be a marking, then the renewed firing rules of EOPN are described in the following definition.

Definition 5: (Interaction). Transition pair $(\hat{t}, t) \in \rho$ can be fired in marking (R, M) if the followings are all met:

- $\hat{t}$ is enabled by marking R of system net
- $\hat{t}$ is firable in marking R of system net
- t is enabled by marking M of object net
- t is firable in marking M of object net

The two transitions in $(\hat{t}, t)$ must be fired simultaneously. Either $\hat{t}$ or t would prohibit the action of interaction if $\hat{t}$ or t is not friable. When interaction executed, the system action is denoted as $(R, M)[\hat{t}, t\rangle(R', M')$. That means $R(\hat{t}\rangle R')$ for system net and $M(t\rangle M')$ or object net

Definition 6: (Transport) If $\hat{t} \in \hat{T} \wedge \hat{t} \notin dom(\rho) = \{\hat{t}_1 \mid \exists t: (\hat{t}_1, t) \in \rho\}$, transition pair $(\hat{t}, \tau)$ is named transport in marking (R, M) if and only if the followings are all hold:

- $\hat{t}$ is enabled by marking R of system net
- $\hat{t}$ is firable in marking R of system net
- τ is the empty action of object net

In transport, the firing of $\hat{t}$ will change the marking of system net. Whereas the marking of object net don't change because no transition is fired. Then, the system action is denoted as $(R, M)[\hat{t}, \tau\rangle(R, M)$.

Definition 7: (Autonomous action). If $t \in T \wedge t \notin range(\rho) := \{t_1 \mid \exists t: (t, t) \in \rho\}$, transition pair (τ, t) is named autonomous action in marking (R, M) if and only if the followings are all hold:

- t is enabled by marking M of object net
- t is firable in marking M of object net
- τ is the empty action of system net

In autonomous action, the firing of t will change the marking of object net, whereas the marking of system net don't change because no transition is fired. Then the system action is denoted as $(R, M)[\tau, t>(R, M')$.

TIMESTAMP STATE CLASS

The state class approach, which is amply depicted by Berthomieu, is well accepted in time Petri net analysis (Berthomieu and Menasche, 1983; Berthomieu and Diaz, 1991). Strong state class is proposed by Berthomieu and Vernadat (2003). Both state class and strong state class are all concerned with local temporal constraints of the transition relative to the moment when the transition is enabled. They cannot be used to analyze the time info of state under global clock. The time span between any couple state classes in same trace cannot be calculated. In order to analyze time constraints of Petri net transition under global clock, timestamp state class is presented here. The timestamp state class memorizes the elapsed time since the beginning of Petri net's operation. The time span between any state-class-pair can also be calculated with this approach.

Definition 8: Timestamp state class (TSC for short) is denoted as C = (M, Q, TS), where:

- M is the marking of Petri net
- Q is a time domain. It is identical to the Q of Berthomieu's strong state class. For any transition enabled in marking M, a vector Q (t) is presented. Q (t) stands for the time elapsed since that transition was last enabled. Let θ represent the time elapsed since current marking is entered, then $\forall t \in En(M), EFT_t \leq Q(t) + \theta \leq LFT_t$ is held
- TS is the timestamp of state class. It means the temporal elapse since system begin running. That is: TS is the time span from the initial state class to current state class. The timestamp of initial sate class is 0

EOPN is a state transition system. Let C_0 be the initial state class, if transition t is fired at time θ and reach new state class C', the action can be denoted by:

$$C_0 \xrightarrow{t@\theta} C'$$

TSC can be calculated by following steps:
Start from the initial state class $C_0 = (M_0, Q_0, TS_0)$:

- M_0 is the initial marking of net
- For any transition t which is enabled in marking M_0, $Q_0(t) = 0$
- $TS_0 = [0, 0]$ is the initial timestamp state class. It is the origin of global clock

If a transition t_i is firable in marking M, the following should hold:

- t_i is enabled according to definition 2
- Let θ represent the time elapse in marking M and γ denote the accumulated time elapsed since the transition t_i was last enabled, then the following hold:

$$\begin{cases} 0 \le \theta \\ \theta + \gamma \ge EFT(t_i) \\ \theta + \gamma \le \min(LFT(t)), \forall t \in En(M) \end{cases} \quad (2)$$

where En (M) denotes all the transitions those are enabled in M

If the firing of t_i results in:

$$C \xrightarrow{t_i@\theta_i} C'$$

The state class C' = (M', Q', TS') can be calculated by:

- Calculate marking M' with M' = M-W (p, t_i)+W (t_i, p), where W is the function of token consumption. For simplicity, let us suppose W = 1 in this study
- Clock domain Q' can be calculated as follows:
  - Calculate θ with Eq. 2
  - For each transition t which is enabled at M and still enabled at M'' = M'-W (p, t_i), $\gamma_t' = \gamma_t+\theta$ and $\gamma_t' \le \min(LFT(t_j))$, $j \ne i$, where t_j covers the transitions that can be fired in M'
  - For those transitions which are newly enabled at M', γ' = 0
  - Eliminate transitions disabled in M'
- For the new TSC, the timestamp TS' = TS+θ. This means that the global timestamp of marking M is TS. After that, temporal interval θ is elapsed, the transition t_i is fired and leads to a new marking M'. TS' is the global timestamp of M'
- Repeat the above steps and get the whole timestamp state class space

For two clock domain $TS_1 = [a_1, b_1]$ and $TS_2 = [a_2, b_2]$, if $a_1 \le a_2 \wedge b_1 \le b_2 \wedge (b_2-b_1 > a_2-a_1)$, then $TS_1+TS_2 = [a_1+a_2, b_1+b_2]$ and $TS_2-TS_1 = [a_2-a_1, b_2-b_1]$ are all clock domain.

Theorem 1: If a timestamp state class $C_k = (M_k, Q_k, TS_k)$ can be reached from $C_0 = (M_0, Q_0, TS_0)$, the global timestamp for any timestamp state class which is in the state class space trace from C_0 to C_k must be less than TS_k.

Proof: The fact that C_k can be reached from C_0 means that there must be a transition sequence meeting:

$$C_0 \xrightarrow{t_0@\theta_0} C_1 \xrightarrow{t_1@\theta_1} C_2 \cdots C_{K-1} \xrightarrow{t_{k-1}@\theta_{k-1}} C_k$$

where t_i is the fired transition and θ is the time delayed in C_i. For n = 0, TS_0 = [0, 0] i.e., the timestamp of initial state is the base point, theorem is correct. Suppose that theorem is still correct for n≤k-1, then for:

$$C_0 \xrightarrow{t_0@\theta_0} C_1 \xrightarrow{t_1@\theta_1} C_2 \cdots \xrightarrow{t_{k-2}@\theta_{k-2}} C_{K-1}$$

the timestamp must be within the TS_{k-1} of C_{k-1}. Let $Ts_{k-1} = [\alpha, \beta]$, then the total elapsed time φ from the initial state class to C_{k-1}, based on global clock and must meet $\alpha \le \varphi \le \beta$. For C_{k-1}, transition t_{k-1} is fired and C_k is reached. Suppose t_{k-1} is fired after time θ since state class C_{k-1} is entered, then $TS'_k = TS_{k-1}+\theta$. Because EFT $(t_{k-1}) \le Q_{k-1}(t_{k-1})+\theta \le \min(LFT(t_i)) \mid t_i \in En(M_{k-1})$ and TS_k is the result for any θ, so $TS'_k \in TS_k$. Theorem is proved.

For a time-transition sequence $t_0\theta_0t_1\theta_1...t_n\theta_n$, its state is infinite because time is continuous and for bounded Petri net, its state class is finite, then the number of timestamp state class is also finite.

Theorem 2: If timestamp state class C' = (M', Q', TS') can be reached from timestamp state class C = (M, Q, TS), then the time span between them is [↓TS'-↓TS,↑TS'-↑TS], where ↓ and ↑ are the lower and upper bound of TS.

Proof: It can be known from the computing process of TSC that, for any two timestamp state classes in the same trace, there must be a feasible transition sequence:

$$C \xrightarrow{t_1@\theta_1} C_1 \xrightarrow{t_2@\theta_2} C_2 \cdots C_n \xrightarrow{t'@\theta'} C'$$

Because TS1 = TS+θ1 = [↓TS+↓θ_1,↑TS+↑θ_1], then θ_1 = TS_1-TS = [↓TS_1-↓TS,↑TS_1-↑TS]. The temporal interval from Ts_1 to TS_2 is θ2 = TS_2-TS_1 = [↓TS_2-↓TS_1,↑TS_2-↑TS]. The rest may be deduced by analogy, so θ' = TS'-TS_n = [↓TS'-↓TS_n,↑TS'-↑TS_n]. The total time elapse from state class C' to C is θ = $\theta_1+\theta_2+\theta_3+...+\theta'$ [↓TS_1-↓TS+↓TS_2-↓TS_1+ ... ↓TS'-↓TS_n,↑TS_1-↑TS+↑TS_2-↑TS_1+ ... ↑TS'-↑TS_n] = [↓TS↓-'TS,↑TS'-↑TS]. So, the theorem is held.

Definition 9: For the two timestamp state classes C = (M, Q, TS) and C' = (M', Q', TS'), they are identical if and only if M = M'∧Q = Q'∧TS = TS' are all held.

MAPPING BPMN DIAGRAM TO EOPN

BPMN V1.0 was published in 2004 and V1.1 was issued in 2008. It is a graph-oriented modeling notation and is easy to use. Yu and Han discussed its use in Web service composition process (Jian and Yanbo, 2006). However, rigorous mathematic semantics are lacked in BPMN. The quality and correctness of BPMN process heavily depends on the skill of its designer. In this section the rules mapping BPMN diagram to EOPN for formal verification are presented.

Task, event and trigger: A task can be mapped into one transition, one pre-set place and one post-set place (Fig. 1a). The name of the task can be the subscript of transition, e.g., t_1. If the task is assigned with time constraint, then static firing interval should be assigned, otherwise the transition is a classic transition and the static firing domain could be treated as [0, ∞). The mapped place of ∀p∈·t∪t· is drawn in dashed borders. This means that such place can fit with other place. For example, if one task is sequential-connect with another task, then its post-set place can be fused with the pre-set place of another task. Fig. 1b shows the fitting of places.

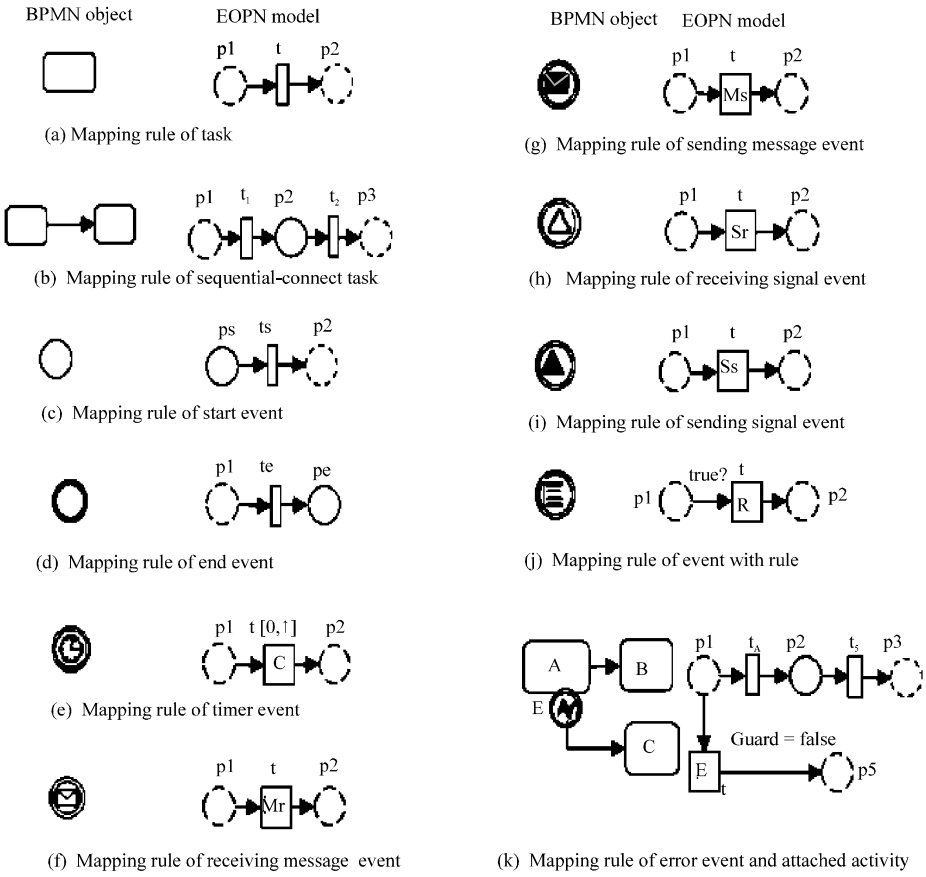


Fig. 1(a-k): Rules of mapping task and event into EOPN model, (a)Mapping rule of task, (b) Mapping rule of sequential-connect task, (c) Mapping rule of start event, (d) Mapping rule of end event, (e) Mapping rule of timer event, (f) Mapping rule of receiving message event, (g) Mapping rule of sending message event, (h) Mapping rule of receiving signal event, (i) Mapping rule of sending signal event, (j) Mapping rule of event with rule and (k) Mapping rule of error event and attached activity

The rules of mapping start, end and intermediate event into Petri net is familiar with the mapping rule of task (Fig. 1c, d). The mapped transitions are denoted as t_s, t_e and t_i, respectively. Because start event has no incoming flow and end event has no outgoing flow, so the pre-set place of t_s and the post-set place of t_e is drawn in solid borders, the post-set place of t_s and the pre-set place of t_e is still drawn in dashed borders. Intermediate event is always combined with certain trigger.

Event is often coupled with certain trigger to stand for the condition of start and end. For the limitation of letters, only the combinations of intermediate event with triggers are demonstrated and the mapping rules are provided. The rule of event coupled trigger is depicted by OMG (2006, 2008).

When coupled with start event or intermediate event, message means sending or receiving a message. When coupled with end event, it only means sending a message. Transition Ms (Fig. 1g) and Mr (Fig. 1f) mean sending and receiving a message, respectively.

The mapping rule for Signal is similar to the rule of message. The unique difference is that, for message, there are specified sender and receiver and broadcast is used in signal. Ss stands for sending a signal (Fig. 1i) and Sr represents receiving a signal (Fig. 1h).

Timer can only couple with start event or intermediate event. It means invoking the outgoing flow after a specific temporal interval. So, time constraint must be addressed for its mapping transition (Fig. 1e). Because the timer represents an unambiguous time delay since task acted, then the static EFT and LFT is decided.

Rule is used to wake the outgoing flow of event if the rule expression is true. In this study, the event with rule is mapped into a transition and the rule expression is mapped to a flow valve for flow arc F: $(t \rightarrow p_o)$, $p_o \in t^{\bullet}$ (Fig. 1j).

Error can only be coupled with intermediate event and end event. When coupling with end event, it means throwing an exception. Coupling error with intermediate event stands for catching a certain exception. After the firing of error, current process halts immediately and activates the outgoing flow of event. When mapping the event with error trigger, a mechanism is wanted to stop Petri net from operating once exception happens. This means that when the transition corresponding to the error event is fired, all transitions are no longer enabled. The mapping rule of error which is presented by Dijkman *et al.* (2008) is too complex. The guard defined in EOPN is for this use. The transition corresponding to the error event is marked with E. In the mapping rules, an assignment expression, Guard = false, is coupled with the outgoing flow of the E transition (Fig. 1k). It can be seen from Fig. 1 that there is a choice relation between error event and its attached activity.

Gateway: There are two types of exclusive decision gateway: Data-based and event-based. The action of data-based exclusive decisions gateway is based on the boolean expression which is contained in its outgoing flow. Only one outgoing flow can be activated, similar to the switch clause in Java or C++. Its mapping rule can be seen in Fig. 2 (a). The event-based exclusive gateway activates its outgoing flow based on the type of event (Fig. 2b). In the mapping rules of exclusive decisions, gateways are mapped into a place and the transitions of its post-set are in conflict relation.

Exclusive gateway can also be used as a merge for alternative sequence flow. The mapping rule of exclusive merge gateway which is presented by Dijkman *et al.* (2008) can be seen in Fig. 2h: Transitions T_A and T_B are in contact relation in place p_3, where p_3 is mapped from exclusive merge gateway. It can be known from the definition of exclusive gateway that task A and task B cannot be met simultaneously, i.e., only one of them can be activated. In the mapping rule presented in Fig. 2h, if the capacity of place p_3 is greater than 1, the transitions T_A and T_B can be fired both, so the mapping presented by Dijkman *et al.*, (2008) is insufficiency. In order to represent the relation that the incoming flow of exclusive gateways is exclusive, a guard place which always has a token is introduced (p_4 in Fig. 2b).

The renewed mapping rule of exclusive merge gateway can be seen in Fig. 2c: If T_A can be fired, then T_B has no possibility to be fired and vice versa. So, this rule can express the exclusive relation of action A and action B appropriately.

Inclusive decisions gateways can activate one or more outgoing flow. This can not be depicted by 1-safe Petri net. The capacity of place that mapped from Inclusive gateways must be greater than 1. Just as in Fig. 2f, $K(p_1) \geq 1 \wedge K(p_2) = 1 \wedge K(p_3) = 1$, K is capacity function. It can be seen in Fig. 2f that the flow arc between p_1 and its post-set are assigned rule valve. If the value = 0, the flow relation is inhibited.

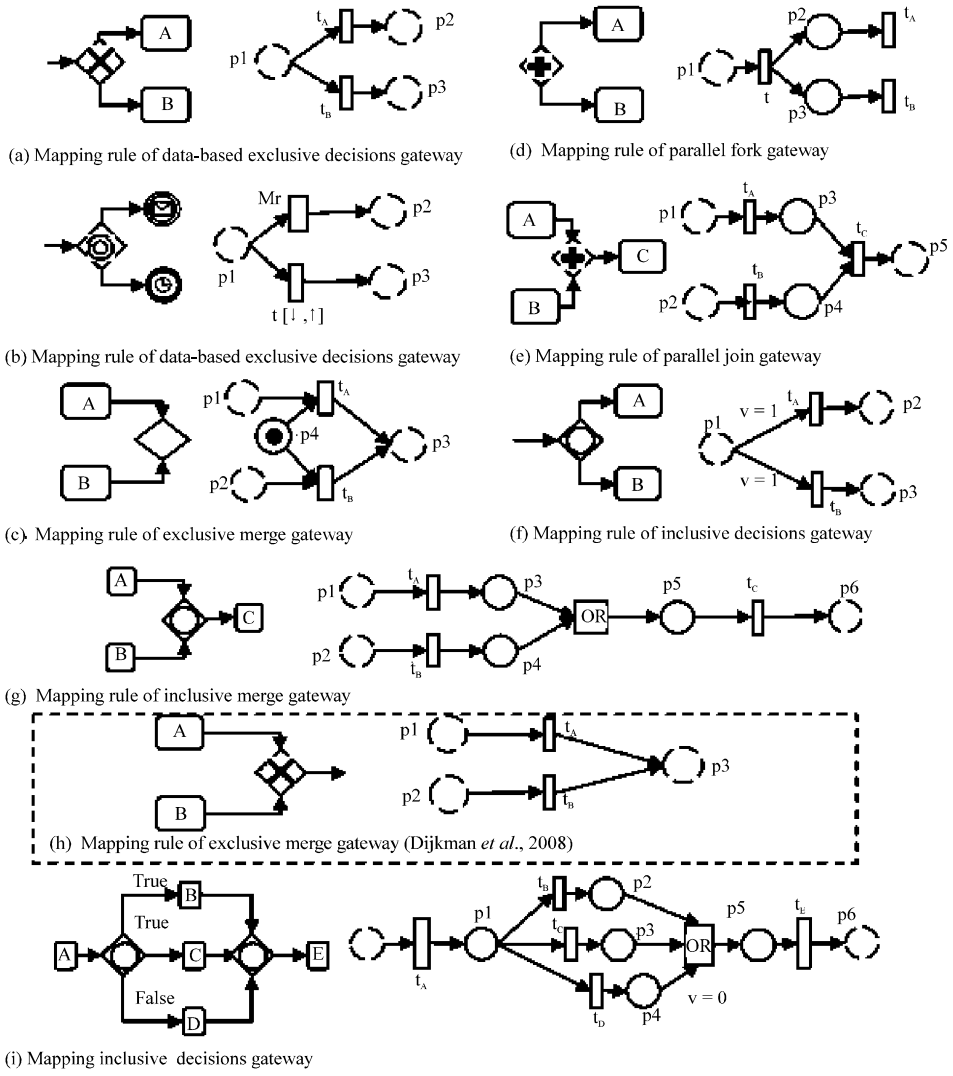


Fig. 2(a-i): Rules of mapping BPMN gateway into EOPN model, (a) Mapping rule of data-based exclusive decisions gateway, (b) Mapping rule of data-based exclusive decisions gateway, (c) Mapping rule of exclusive merge gateway, (d) Mapping rule of parallel fork gateway, (e) Mapping rule of parallele join gateway, (f) Mapping rule od inclusive decision gateway, (g) Mapping rule of inclusive merge gateway, (h) Mapping rule of exclusive merge gateway (Dijkman *et al.*, 2008) and (i) Mapping rule of inclusive decisions gateway

Sometimes inclusive merging gateway is used to synchronize incoming flow. It does not wait for those tokens produced by its all upstream, but only requires the tokens which are produced by the actually executed upstream (OMG, 2006, 2008). The classic Petri net model cannot express such semantic. This problem was also discussed by Dijkman *et al.* (2008) but no solution was provided.

Let t_1, t_2 be transitions of Petri net, M is a marking, if M $[t_1>M'\wedge\neg M[t_2>\wedge M'[t_2>$ is consistent, then t_1 and t_2 are in sequence relation.

Theorem 3: Let N be an EOPN net, p and t is a place and transition, respectively and $p \in {}^{\bullet}t$. Let T_{sub} be the transition set that is sequential to t, p_{pre} and P_{post} be the pre-set and post-set of T_{sub}, respectively. If $V(p \rightarrow t) = 0$, then for the flow relation $F_{sub} = (P_{pre} \times T_{sub}) \cup (T_{sub} \times P_{post})$, the V (f) can be treat as 0 for all $f \in F_{sub}$.

Proof: Let M be a marking of EOPN, transition t can be fired at M, and the relation is sequential among t, t_1, t_2, ..., t_n. i.e., there is a transition sequence $\eta = t, t_1, t_2, ..., t_n$ satisfying:

$$M \xrightarrow{t} M' \xrightarrow{t_1} M_1 \xrightarrow{t_2} \cdots M_{n-1} \xrightarrow{t_n} M_n$$

It can be learned from the condition in definition 2 that if the valve of a flow arc is set to 0, the corresponding transition t cannot be enabled and then not be fired, then none of the subsequent transition of t in the transition sequence η can be fired. So, the valves correlating with these transitions in η can be set to 0.

In order to map inclusive merging gateway, a new type transition, which is named OR-Merging transition and marked with t_{OR}, is introduced (Fig. 2g). Its enabled condition should be revised with:

- $V(p_i \rightarrow t_{OR}) = 0 \wedge Guard = true$ for all $p_i \in {}^{\bullet}t_{OR}$, or
- $M(p) > 0 \wedge gV(p_i \rightarrow t_{OR}) = 1 \wedge Fuard = true$ for all $p_i \in {}^{\bullet}t_{OR}$

It can be seen from Fig. 2 (i): Inclusive decisions gateway activates its outgoing flow B and C, but D cannot be fired. Only after B and C are all finished can inclusive merging gateway wake activity E. Transition t_D can't be fired in current running, i.e., $V(p_1 \rightarrow t_d) = 0$. According to theorem 3 one has $V(p4 \rightarrow t_{OR})$. So, if there are tokens in places p_2 and p_3 and Guard = 1, t_{OR} is enabled and fired if time constraint is met.

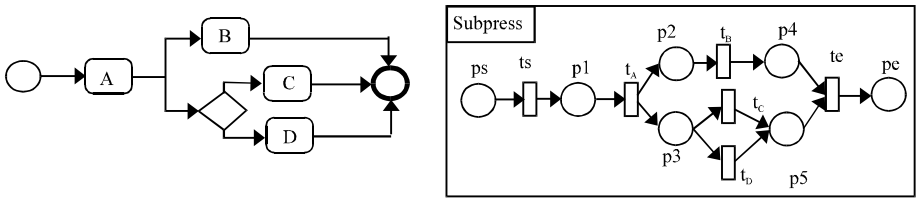
Parallel fork gateway is mapped into a transition, the outgoing flow can be mapped into several transitions which are concurrent (Fig. 2d). Different from inclusive merging gateway, the parallel join gateway is used to synchronize all of its incoming flows. Its mapping rule can be seen in Fig. 2e: Parallel join gateway is mapped to a transition, its pre-set is the post-set of transitions which are mapped from the incoming flows of parallel join gateway.

Subprocess was viewed as a standalone process by Dijkman *et al.* (2008). Dijkman and his team depicted the mapping of calling a subprocess via a Subprocess Invocation activity (SI). Two places drawn in dashed borders capture, respectively, the incoming and outgoing flows of activity SI. Their method would be too cumbersome when one attempts to depict a process which is composed of a caller process and several called subprocesses. In this study, object net is used to denote subprocesses just as is shown in Fig. 3a.

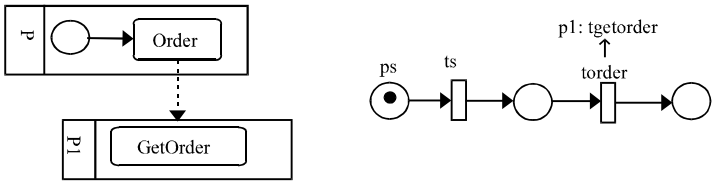
Swim lanes: Pool can be mapped into object net of EOPN according to the notion of net-within-net. The pool of sponsoring process will be mapped to System Net and the other participant process pool will be modeled by Object Net (Fig. 3b).

Connecting objects: A sequence flow links two objects in a process diagram and denotes a control flow relation. Sequence flow can be mapped into flow relation of EOPN. Message flows are used to capture the interaction between processes.

Pool is mapped to object net and the sponsoring process is mapped to system net. So, the message flow between different pools can be modeled by interaction.



(a) Mapping BPMN subprocess into EOPN object net



(b) Mapping BPMN swim lanes and connecting objects into EOPN model

Fig. 3(a-b): Rules of mapping subprocess and message flow into EOPN model, (a) Maping BPMN subprocess into EOPN object net and (b) Mapping BPMN swim laned and connecting object into EOPN model

As is seen in Fig. 3 (b): The process in pool p is the sponsor of the whole business process, so it is mapped into system net. Its start event is mapped to t_s. There is a token in the pre-set place, this object token is an object net mapped from process P_1. So, the token in p_s is an object token and is the instance of object net P_1. The task order of P is connected with task GetOrder of P_1 visa message flow. So, this message flow is mapped to a transition pair <P: Order, P_1: GetOrder>. It is an interaction of the EOPN model. ↑ denotes the message direction.

Example: Modeling and verifying BPMN diagram: The BPMN diagram in Fig. 4a depicts such a scenario: A customer makes an order to the producer. The producer receives the order and checks his reserve. If there is no enough reserve, he has to produce more to fulfill the demand of the customer. Otherwise if there is enough reserve the producer would pack and deliver those goods. There are errors in the BPMN diagram in Fig. 4a although, the diagram seems fluent and thorough.

The business process of producer can be mapped into an object net of EOPN (Fig. 4b) according to the rules mentioned above. The process of customer is mapped to system net (Fig. 4c). The token in P_1 of SN is an object token and is an instance of producer. The meaning of transitions and their time constraints are listed in Table 1.

There are five interactions in the EOPN model. The corresponding relationship between interaction and message flow is illustrated in Table 2.

According to the transition firing rules of EOPN, the state space of EOPN model can be taken just as in Fig. 5 and the state class info can be seen in Table 3.

Let SN and ON denote the system net and object net of Fig. 4, respectively. For a given marking (R, M), let θ_1 be the time when current state class is entered, θ_2 be the moment when the firable transition is fired, then the time delay θ between θ_1 and θ_2 can be denoted with $\theta = \theta_2 - \theta_1$. When the net begins to run, the initial timestamp state class $C_0 = ((R_0, M_0), Q_0, TS_0)$ where R_0 = (p1) and M_0 = (p1), for C_0, $Q_0 = 0$; $TS_0 = 0$. That means the global timestamp at initial timestamp state class is 0.

Table 1: Description of transitions in example

Transition	Explanation	Time (days)
Description of transitions in producer object net		
ts	When receiving the order message, producer process begins to run	
$t_{CheckOrder}$	Producer verifies whether the order is valid or not	[3, 5]
$t_{Invalid}$	The order is invalid and send message to the customer	
t_{e1}	Process is terminated with error	
t_{Valid}	The order is valid and send message to the customer	
t_{GetPay}	Receive message that customer paid and take an account of store	
$t_{Producing}$	The repertory is not enough, production scheduling	[5, 10]
$t_{Collecting}$	The repertory is enough and to be gathered	[1, 2]
t_{and}	Product is ready	
$t_{Deliver}$	Producer delivers goods and sends message to the customer	[3, 5]
t_e	Process is terminated successfully	
Description of transitions in customer system net		
ts	Process begin to start	
t_{Order}	Send order message to producer	[1, 2]
$t_{Invalid}$	The order is refused by the producer	
t_{e1}	Process terminate with error	
t_{Valid}	The order is accepted the producer	
t_{Pay}	Pay to the producer	[2, 4]
t_{Arival}	Get the goods from producer	
t_e	Process terminate successfully	

Table 2: Mapping BPMN message flow into EOPN interactions for example

Message flow	Interaction
Order	$<t_{Order}, ts>$
Pay	$<t_{Pay}, t_{GetPay}>$
Valid	$<t_{Valid}, t_{Valid}>$
Deliver	$<t_{Arrival}, t_{Deliver}>$
Not valid	$<t_{Invalid}, t_{Invalid}>$

Table 3: Timestamp state class computing result of example

State class	Marking of state class	Timestamp of state class
C_0	$R_0 = (p_1); M_0 = (p_1)$	0
C_1	$R_1 = (p_2); M_1 = (p_1)$	0
C_2	$R_2 = (p_3); M_2 = (p_2)$	[1, 2]
C_3	$R_3 = (p_3); M_3 = (p_3)$	[4, 7]
C_4	$R_4 = (p_4); M_4 = (p_4)$	[4, 7]
C_5	$R_5 = (p_5); M_5 = (p_5)$	[4, 7]
C_6	$R_6 = (p_6); M_6 = (p_4)$	[4, 7]
C_7	$R_7 = (p_7); M_7 = (p_6)$	[6, 11]
C_8	$R_8 = (p_7); M_8 = (p_7)$	[11, 21]
C_9	$R_9 = (p_7); M_9 = (p_8)$	[7, 13]
C_{10}	$R_{10} = (p_6); M_{10} = (p_{12})$	[4, 7]
C_{11}	$R_{11} = (p_4); M_{11} = (p_{12})$	[4, 7]

Because place p1 in SN has object token, the enabled condition and firing condition are all met according to definition 2 and definition 3, so transition t_s of SN can be fired. Then, $C_1 = ((R_1, M1), Q_1, TS_1)$ where:

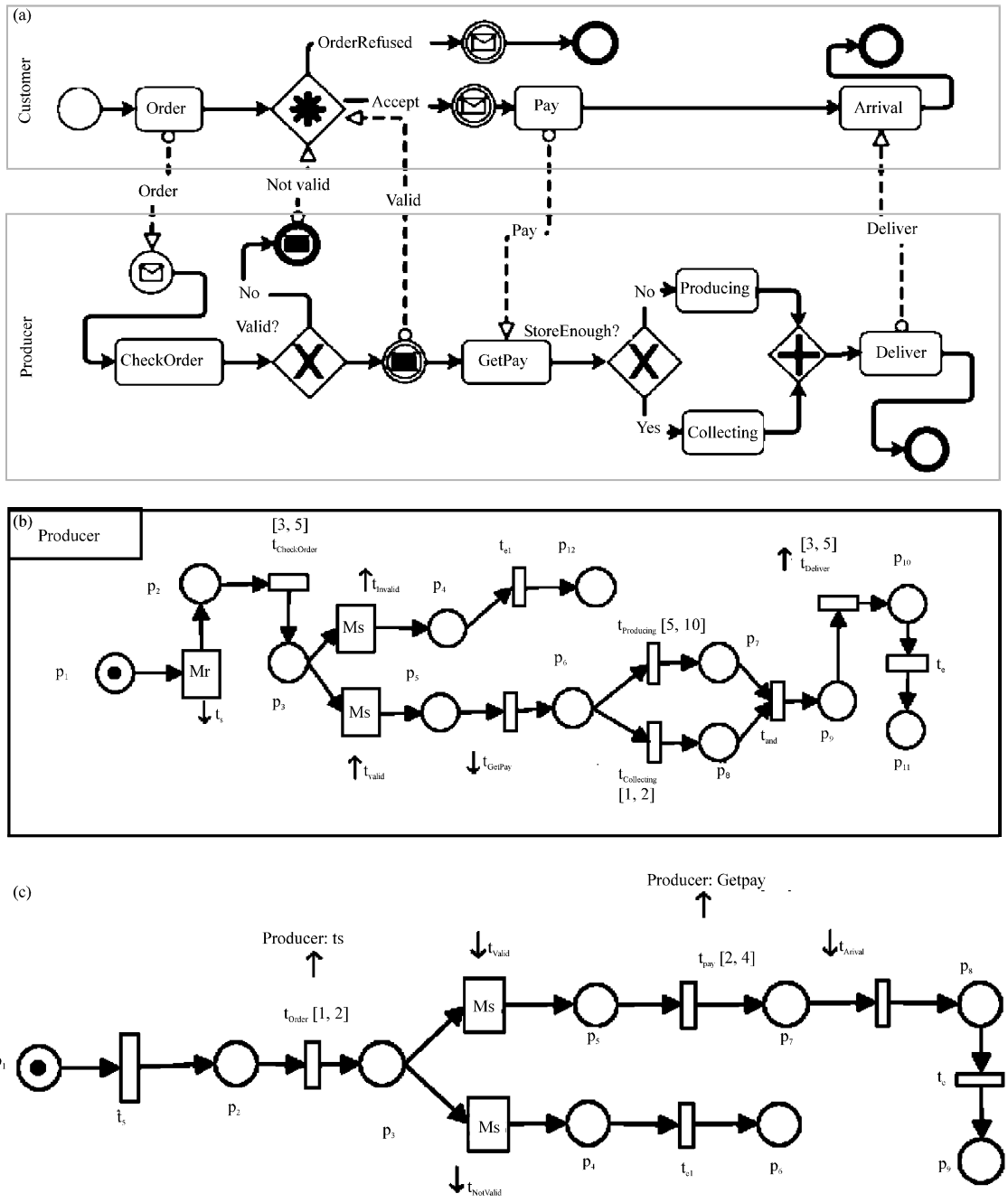


Fig. 4(a-c): An example of BPMN flow and its EOPN model, (a) BPMN diagram of example, (b) Mapping the swim lane of producer into EOPN object net and (c) Mapping the swim lane of customer into EOPN system net

- $R_1 = (p^2) \wedge M_1 = (p1)$
- $Q_1 = 0$. Because there is no time constraint on transition SN: ts, it can be fired immediately without time delay, that is $\theta = 0$
- $TS_1 = TS_0 + \theta = 0$

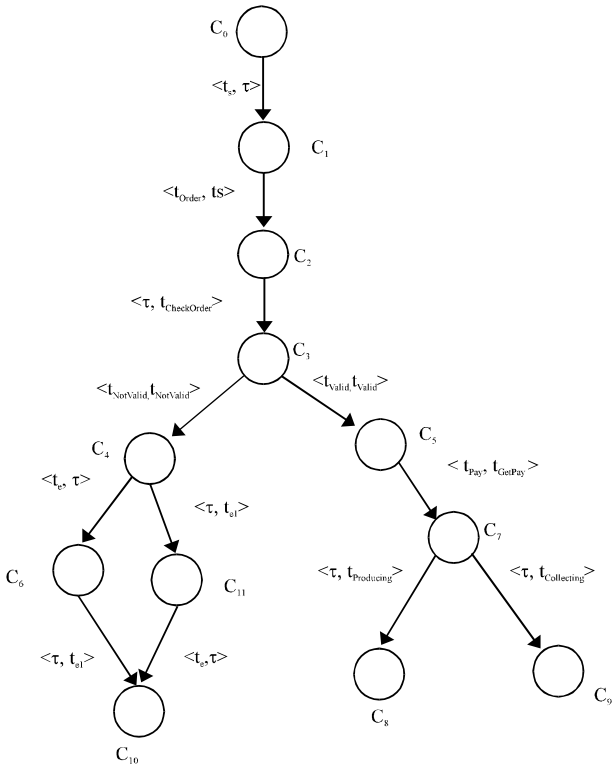


Fig. 5: Timestamp state class space of EOPN model of example

When there is a token in SN: p_2 , interaction $<t_{Order}, t_s>$ is enabled and can be fired in temporal interval [1, 2]. After transition pair $<t_{Order}, t_s>$ is fired, the places SN: p3 and ON: p2 get a token, respectively. For $C_2 = ((R_2, M_2), Q_2, TS_2)$:

- R_2 = (p3) and M_2 = (p2)
- Q_2 (t_{Order}) = [1, 2]. Although, enabled, transition SN: t_{Order} has to delay [1, 2] before it can be fired, then θ = [1, 2] . In C_1,Q (t_{Order}) = 0, then Q_2 (t_{Order}) = [1, 2]
- $TS_2 = TS_1+\theta$ = [1, 2]

Consequently, because the enabled transitions t_{Valid} and $t_{Invalid}$ all belong to interaction, they can't be fired. Object net transition ON: $t_{CheckOrder}$ can be fired in time domain [3, 5] and its firing results in a new timestamp state class C_3. If the order is valid, interaction $<t_{valid}, t_{valid}>$ can be activated. In the same way transition pair $<t_{pay}, t_{Getpay}>$ can be fired after time delay [2, 4] and state class C_7 would be reached. For the producer, if his store is enough, his following work is to collect goods for delivery, otherwise more goods should be produced in order to meet the customer. The relation between $t_{Producting}$ and $t_{Collecting}$ in Fig. 4c is conflict: Either place ON: p_7 or place ON: p_8 can get token. For the transition ON: t_{and}, condition M (p_7)>0/\M (p_8)>0 is required for firing, so ON: t_{and} can never be fired. So, there is a deadlock in the EOPN model.

The reason for the deadlock is that a parallel join gateway is used after the outgoing flow of exclusive decision gateway SotreEnough. This error is implicit and can not be corrected automatically. Without verification, the fault would occur after the process operated for 11-21 days. It is too late. By mapping BPMN diagram to EOPN and verifying the model, the drawback could be identified easily.

CONCLUSION AND FURTHER WORK

BPMN is a standard for business processes modeling especially at the level of business analysis and high-level systems design. It is widely accepted by mainstream IT companies. Graphical design tools based on BPMN have been developed. But for the lack of formal semantics in BPMN metal model, there is no way to verify the model formally and automatically. The research of Dijkman *et al.* (2008) suffers some drawbacks in mapping BPMN to Petri net. In order to model and verify BPMN in formal way, EOPN is developed. The analysis approach, which is named timestamp state class, is also presented. The rules mapping BPMN object to EOPN are depicted in detail. At last an example is presented and BPMN diagram is modeled with EOPN so one can formally verify its correctness.

It should be pointed that only a subset of BPMN metal objects is considered in this study. The mapping rules for exception handling, compensation, looping, etc., are not presented. Further research should be preceded.

REFERENCES

Berthomieu, B. and M. Menasche, 1983. An enumerative approach for analyzing time Petri nets. Proceedings of the IFIP 9th World Computer Congress, September 19-23, 1983, Paris, France, pp: 41-46.

Berthomieu, B. and M. Diaz, 1991. Modeling and verification of time dependent systems using time petri nets. IEEE Trans. Software Eng., 17: 259-273.

Berthomieu, B. and F. Vernadat, 2003. State Class Constructions for Branching Analysis of Time Petri Nets. In: Tools and Algorithms for the Construction and Analysis of Systems, Garavel, H. and J. Hatcliff (Eds.). Springer, Heidelberg, Germany, pp: 442-457.

Dijkman, R.M., M. Dumas and C. Ouyang, 2008. Semantics and analysis of business process models in BPMN. Inform. Software Technol., 50: 1281-1294.

Hinz, S., K. Schmidt and C. Stahl, 2005. Transforming BPEL to Petri Nets. In: Business Process Management, Van der Aalst, W.M. P., B. Benatallah, F. Casati and F. Curbera (Eds.). Springer, Heidelberg, Germany, pp: 220-235.

Jian, Y. and H. Yanbo, 2006. Service Oriented Computing: Theory and Application. Tsinghua University Press, Beijing, China.

Kohler, M. and H. Rolke, 2004. Properties of Object Petri Nets. In: Applications and Theory of Petri nets 2004, Cortadella, J. and W. Reisig (Eds.). Springer, Heidelberg, Germany, pp: 278-297.

Lakos, C., 2001. Object Oriented Modeling with Object Petri Nets. In: Concurrent Object-Oriented Programming and Petri nets, Agha, G.A., F. De Cindio and G. Rozenberg, Springer, Heidelberg, Germany, pp: 1-37.

Merlin, P., 1974. A study of the recoverability of computer system. Ph.D. Thesis, Department of the Computer Sciences University, California, USA.

OMG, 2006. Business process modeling notation. Version 1.0, OMG Final Adopted Specification, Object Management Group.

OMG, 2008. Business process modeling notation, version 1.1. Object Management Group.

Ramchandani, C., 1974. Analysis of Asynchronous Concurrent Systems by Timed Petri Nets. MIT, Cambridge, UK.

Valk, R., 1998. Petri Nets as Token Objects. In: Application and Theory of Petri nets 1998, Desel, J. and M. Silva (Eds.). Springer, Heidelberg, Germany, pp: 1-24.

Valk, R. and C. Girault, 2003. Petri nets for Systems Engineering: A Guide to Modeling, Verification and Applications. Springer-Verlag, Berlin, Germany.

Valk, R., 2004. Object Petri Nets. In: Lectures on Concurrency and Petri nets, Desel, J., W. Reisig and G. Rozenberg (Eds.). Springer, Heidelberg, Germany, pp: 819-848.

Van der Aalst, W.M.P. and A.H.M. Ter Hofstede, 2000. Verification of workflow task structures: A petri-net-based approach. Inform. Syst., 25: 43-69.

Yi, Y.C., 2005. Petri nets Theory and Application. Publishing House of Electronic Industry, Beijing, China.

Yu, R., Z. Huang and L. Wang, 2009. Modeling and analyzing project performance with timing constraint Petri net. Proceedings of the International Conference on Computer Engineering and Technology, January 22-24, 2009, Singapore, pp: 243-246.

Zhehui, W., 2006. Petri Net Introduction. China Machine Process, Beijing, China.