Transformation of Business Process Model and Notation models onto Petri nets and their analysis

Mutarraf, Muhammad Umair; Barkaoui, Kamel ; Li, Zhiwu; Wu, Naiqi; Qu, Ting

Published in:
Advances in Mechanical Engineering

DOI (link to publication from Publisher):
10.1177/1687814018808170

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Advanced Production Planning, Scheduling, Logistics and Control - Research Article

Advances in Mechanical Engineering

Advances in Mechanical Engineering
2018, Vol. 10(12) 1–21

DOI: 10.1177/1687814018808170
journals.sagepub.com/home/ade
SAGE

Transformation of Business Process Model and Notation models onto Petri nets and their analysis

Umair Mutarraf[1], Kamel Barkaoui[2], Zhiwu Li[3,4], Naiqi Wu[3] and Ting Qu[5]

Abstract

Business Process Model and Notation is known as a widely used standard for business processes modeling. However, its main drawback is that it lacks formal semantics, leading to some undesirable properties, such as livelocks and deadlocks, such that it creates models with semantic errors. In order to formally verify them, we need to transform it onto a formal language, for example, Petri nets. The approach proposed in this article is an extension of previous approaches stated in the literature by adding probability to gateways and time to transitions. The first aim is to transform the Business Process Model and Notation process diagram onto Petri nets automatically using a developed software package. The developed software package is capable of transforming the XML file of a Business Process Model and Notation process diagram into "m" files of a Petri net. The "m" files of the Petri net are then coupled with the General Purpose Petri Net Simulator (GPenSIM) for analysis in MATLAB. The second aim is to manually transform the Business Process Model and Notation process diagram using mapping figures onto Petri nets and then analyze it using Timed Petri Net Analyzer tools. The advantage of transforming a Business Process Model and Notation diagram automatically is that we can add time to transitions and probability to gateways. Furthermore, the simulation time can be checked using MATLAB.

Keywords

Business Process Model and Notation, transformation of Business Process Model and Notation, petri net, formal verification

Date received: 27 February 2018; accepted: 17 September 2018

Handling Editor: Tatsushi Nishi

Introduction

The Business Process Model and Notation (BPMN) is a modeling tool for capturing business processes. Standard BPMN offers a power (ability) to figure out business procedures in a graphical representation.[1] It provides companies with the potentiality of interacting procedures in a typical mannerism and is widely used as a tool for business process modeling. In addition, for simplification and better understanding of collaborations and transactions between companies, graphical notations are used. This notation adopts elements from a number of previously proposed notations for business process modeling, including the XML Process

[1]Department of Energy Technology, Aalborg University, Aalborg, Denmark
[2]Department of Informatique (EPN 5), Conservatoire National des Arts et Métiers, Paris, France
[3]Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau
[4]School of Electro-Mechanical Engineering, Xidian University, Xi'an, China
[5]School of Electrical and Information Engineering, Jinan University (Zhuhai Campus), Zhuhai, China

Corresponding authors:
Zhiwu Li, School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China.
Email: zhwli@xidian.edu.cn

Naiqi Wu, Institute of Systems Engineering, Macau University of Science and Technology, Taipa 999078, Macau.
Email: nqwu@must.edu.mo

Definition Language (XPDL)[2] and the activity diagrams component of the Unified Modeling Language (UML).[3] A BPMN process model is composed of activity and control nodes and can be connected in a random way.

Several organizations (BPMI, OMG, OASIS, etc.) have explained a sequence of different standards for designing, executing, and monitoring business processes. These standards can be used either separately or in combined mode conditional upon the compatibility between them. With respect to notations, BPMN and UML are widely used in present. BPMN offers several advantages over other notations. For example, integrated definition (IDEF),[4,5] a family of modeling languages, does not have resource modeling and control capabilities and does not cover data objects that are a kind of resources for business process modeling. UML emphases on software models, whereas BPMN emphases on business processes.

However, BPMN shows even more semantic errors[6] as it combines graph-oriented features with other characteristics from many other resources, such as workflow (WF) patterns and business process execution language (BPEL).[7] These features include the message flow between processes and execution of sub-processes in multiple times. The wrong use of BPMN elements such as OR/XOR-join, AND-join, or an event that does not permit more than one outflowing arc, and so on, results in syntactical errors. The correctness of models, for instance, invalid flow or constructs can be found syntactically in lesser time using tools such as Intalio2 and BizAgi1. On the other hand, structural errors are usually found at the runtime as compared to syntactical errors that are found while designing using modeling tools. The structural errors such as wrong use of split and joins or incorrect combination of elements are hard to be identified at the runtime due to the absence of formal semantics of BPMN models. The ability to statically analyze a BPMN model has become a desirable feature for tools supporting process modeling in BPMN. It is found that semantic errors in a BPMN model that is occasionally built by a user can be detected using well-known verification techniques.

The efforts made to semantically analyze a BPMN model are hindered by uncertainties in it because of language complexity and specification of the standard. In order to formally verify a BPMN model, it is necessary to use a formal language, such as Petri nets. For example, Petri nets have found extensive applications to industrial systems such as deadlock analysis and solution,[8–10] supervisory control,[11–14] implementation[15] and scheduling,[16–19] and fault analysis.[20–23] As an important mathematical tool of discrete event systems,[24,25] Petri nets are found to be the most suitable language to do such analyses due to the availability of many existing analysis tools. However, a mapping between BPMN and Petri nets can be defined for semantic analysis in order to check the correctness of a BPMN model.

Moreover, the addition of firing times to the transitions and probability appended to the exclusive, inclusive gateway, and event-based gateway will introduce an additional feature for modeling business processes. Firing delay is the amount of elapse time between the enabling and firing of a transition. Adding firing time to a BPMN diagram can be done by adding annotations as they do not have any effect on the model. Upon transformation, annotations are discarded and time will be added to the transitions. Hence, the associated transitions are timed transitions. In BPMN, probability can be added to the sequence flows of gateways and upon transformation into Petri nets, the probabilities are added for all the outcoming paths. Moreover, the statistical information is required for adding the probability. This statistical information can be useful for gateways, such that the probability of all the outcoming paths can be determined. For example, in online shopping website, availability of statistical information will help determine the client opening a typical product to be prompted for one of the related products.

Contribution of this work

BPMN is a widely used standard for modeling business processes. The current version BPMN 2.0 is quite comprehensive. However, a major drawback of BPMN is the lack of verification techniques. In this article, we aim at solving the problems found in the existing works[26–33] by transforming BPMN onto Petri nets. Our approach integrates the ideas in previous works[30–44] to present a novel method of transforming BPMN elements (i.e. events, gateways, activities, pools, connecting objects, and lanes) onto equivalent Petri net elements. The major contributions of this article can be stated as follows:

- We propose a method for mapping a large number of subset elements in BPMN onto Petri nets.
- We present two methods for mapping: an automatic method and a manual method. For the automatic method, we develop a software package that can transform an XML file of BPMN models onto "m" files of Petri nets.
- We propose an addition of probability to the inclusive, exclusive, and event-based gateway. Moreover, time delays are added to the transitions.
- Experimental examples show that the proposed method can transform a large BPMN model onto an equivalent Petri net model for analysis.

In transforming a large number of subsets of BPMN elements onto Petri nets, the following subsets are included: events, activities, gateways, artifacts, swim

lanes, and connecting objects. Furthermore, the mapping can be done either manually or automatically. In the manual method, each element of a BPMN model is transformed into an equivalent Petri net element according to the mapping figures. The transformed Petri net is then analyzed in Timed Petri Net Analyzer (TINA). The automatic method fetches the XML file of a BPMN model and transforms it into an "m" file of Petri nets. The m file of Petri nets is fed into GPenSIM in MATLAB for analysis. In the transformation of the XML file of a BPMN model into an "m" file of Petri nets, time and probability are considered in this method. Finally, comparison of the number of elements in both the BPMN and its equivalent Petri net is provided. The time taken for simulating the obtained "m" file of the Petri net model in the GPenSIM tool is recorded to be low.

Article organization

The remaining of this article is organized as follows. Section "Literature review" introduces the background of both BPMN and Petri nets. The mapping of activities, events, gateways, and swim lanes onto Petri nets is discussed in section "Preliminary background." Section "Mapping BPMN elements onto Petri nets" describes the development of the BPMN framework and its basic modules and introduces tools used for analysis. The experimental studies are done using two examples to demonstrate the proposed method in section "Development of BPMN framework for analysis." Finally, related work is briefly discussed in section "Experimental studies" and section "Conclusion and future work" concludes this study and provides future research.

Literature review

In this section, we briefly review the languages other than BPMN which are transformed into Petri nets and the techniques used for analysis of business process models.

The execution and effective design of business processes are becoming progressively significant to present business organizations. The basic organization need is to constantly redesign its business processes on daily basis such that its operation practices are aligned with the change in business requirements, which has been acknowledged for decades.[45] It is vital that changes in business processes are understood well and then represented systematically such that their influences are clearly defined. In order to understand well, there is an utmost need to create business process models.[46] These models basically illustrate present or future organizational behavior in order to achieve several purposes, for instance, the monitoring and control of process execution. The fast-growing use of business process modeling necessitates that the activities should be carried out systematically in order to increase productivity, efficiency, and consistency.[47] During past decades, it was recommended to develop more systematic and flexible approaches to design business processes.[48,49] Levitt was the first who pointed out the significance of business processes in 1960s, and later on, researchers such as Harrington,[50] Davernport,[51] and Hammer[52] promoted its new perspective. The rapid increase in popularity of business processes has brought forth a promptly growing number of modeling techniques, methodologies, and tools for its support.[53] Hence, the selection of precise technique has become quite complex due to availability of large range of approaches. The first step before constructing any model is to identify its use and the purpose to select the right technique.

Kettinger et al.[54] present an overview of methods, tools, and modeling techniques utilized in Business Process Reengineering (BPR). Although the authors neither provide detailed description of tools nor the techniques, it has been the starting point of research. Phalp and Martin[55] differentiate between two benefits of business process models: one for restructuring business processes and the other for software development. In the work by Phalp,[56] the former purpose is described by arguing that pragmatic approaches are typically concerned with understanding and capturing business processes, whereas rigorous paradigms are mostly utilized for analysis of the process. Furthermore, for the analysis of business processes, it is mandatory to have proper sophisticated mechanism that includes both functional and dynamic aspects than qualitative analysis of static models. The user requires a model that can provide more interaction such as simulation to analyze business processes. Hence, approaches that are easily understandable and have diagrammatic notation such as BPMN, EPC, BPEL, and YAWL are selected in order to represent business processes.[57] The languages which are used other than BPMN for transformation onto Petri nets are EPC, BPEL, and YAWL. The first modeling technique besides BPMN is Event-driven Process Chains (EPCs). The main constraints in it are functions, events, and logical connectors.[58] Events are transformed onto Petri nets as places, whereas functions are modeled by transitions. In this setup, the main drawback is that the set of connectors are limited.

An XML-based language, that is, BPEL, is used to define business processes within web services. The foremost objective of BPEL is the standardization of format, such that companies can work and communicate with each other using web services. It includes activities, partner links, and variables. The formal semantics for BPEL has been proposed by several groups, among them the existing approaches are relied on finite state

machine,[59,60] abstract state machine,[61,62] and process algebras.[63] Most of the approaches do not provide provision to BPEL's most important features such as compensation, fault, and event handling. However, Petri nets provide much wider aspect especially for computer-aided verification purposes. In the work by Hinz et al.,[64] the authors considered Petri nets semantics for BPEL and established that the semantics is well suitable for computer-aided verification purposes. The study by Van Der Aalst and Ter Hofstede[65] developed a Yet Another Workflow Language (YAWL) based on WF patterns, where objects and tasks are used to model control flow aspects. The main advantage of YAWL is that it supports the following features: OR operations, AND operations, split and joint operations. For semantic analysis of a BPMN model, Dijkman et al.[26] focused on the control flow prospective only and did not include features such as artifacts, lanes, and pools. The study by Wong and Gibbons[27] used communicating sequential processes (CSP) as the formal language that can be semantically checked by a tool called FDR.[28] By this method, BPMN models are mapped onto CSP events, and processes and the relation between tasks are described via CSP events. The resulting CSP models are complex and their size is huge. Moreover, this mapping cannot preserve the structure of a BPMN model. In addition, no explanation is presented for how the errors of a BPMN model can be detected through CSP semantics.

Puhlmann and Weske[29] presented a tool to analyze business models statically. By their method, very few subsets of elements of BPMN are mapped onto calculus. The main feature of BPMN, that is, error handling, is ignored in it. Soundness can be checked by calculus expressions. Experimental studies show that this method cannot cope with large models (more than 10 nodes). Dijkman and Van Gorp[30] defined a subset of BPMN (Ver. 2.0) elements formally with regard to graph rewrite rules. They showed that execution rules formally defined in this method are simple because they can be stated graphically. Using graph rewriting tools, formal semantics can be directly used in execution of models which were created in the BPMN. The issue of their approach was that resource and data aspects are ignored, and the provided semantics is least suitable for verification of process correctness. Raedts et al.'s[31] study presented a BPMN model which is automatically transformed onto Petri nets and is analyzed using any of the following tools: Yasper, INA, LoLA, and Woflan. The Petri nets can be automatically transformed onto a process algebraic language (mCRL2). The limitations of this method include the following: (1) few elements (gateways) are formally defined in the transformation, that is, AND and XOR gateways and (2) the method ignores the use of time and probability to the transitions.

Ramadan et al.[32] proposed formal semantics for business process models by mapping BPMN onto Colored Petri Nets (CPN). The proposed mapping is then used for the validation of process models. Koniewski et al.[33] proposed the use of Petri nets and BPMN formalism for building the models of multimodal logistic chains. First, logistics operations related to business processes are defined and then are represented in BPMN. The model is transformed into several Petri net phases. After that, for every single net, a single phase of complicated multimodal logistic chain is defined. The proper union of these nets forms a simulation model. In the work by Wong and Gibbons,[41] BPMN semantics is defined using CSP and refinement procedure is defined for property checking. However, the work by Wong and Gibbons[41] does not display how the data are modeled and how to detect several sort of errors using CSP semantics. The study by El-Saber and Boronat[66] formulates a subset of BPMN in Maude but does not consider multiple instances and cancelation of sub-process. Furthermore, no tool was provided to automatically generate Maude description from BPMN models. Roa and colleagues[67,68] illustrate an approach focusing on anti-patterns in order to verify a block-structured collaborative business model defined in UP-ColBPIP language. The proposed approach covers numerous complex control flows but it has limitation in identifying constructs such as sub-processes, cancelation of instances, data object, and dynamic multiple instances. Furthermore, it is claimed that their approach can be utilized for BPMN language as well. In the work by Kheldoun et al.,[69] formal semantics of BPMN is proposed using recursive Petri nets and rewrite logic. In the obtained formalism, a large number of subsets of BPMN elements including multiple instantiation, cancelation of sub-processes, and exceptional behaviors are covered, by which Maude LTL model checker can be used to verify behavioral properties of BPMN processes.

Preliminary background

In the section, we will briefly discuss BPMN and Petri nets.

BPMN

BPMN offers a graphical representation for modeling business processes. It is somehow similar to the UML. It was first implemented in 2006 by Object Management Group, and its latest version is 2.0. It is a kind of languages with a huge pool of object types to denote features of business processes, including control flow, resources, data, and exceptions.

BPMN is generally designed for exhibiting business processes at the theoretical level. Several categories of

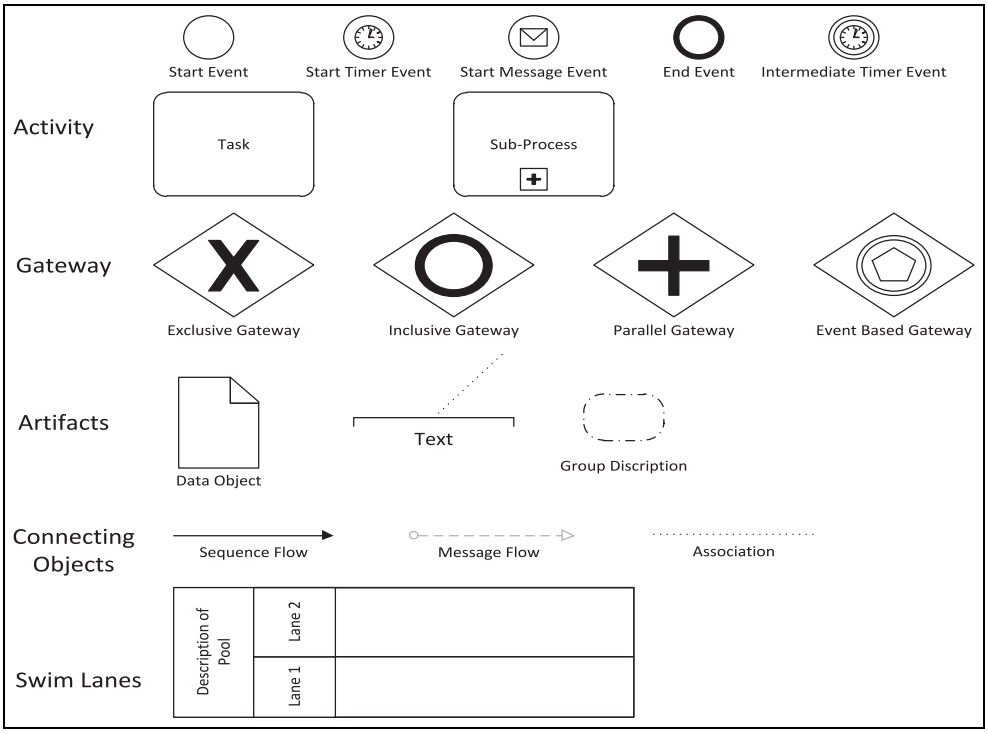

Figure 1. BPMN elements.

elements in BPMN exist. Flow objects, Swim lanes, connecting objects, and artifacts are among them. There are many types of elements in each category. Flow objects are used to represent the control flow features of a business process with three types of elements, that is, activities, gateways, and events. Moreover, control flow objects can be connected via sequence flows, and directed arcs are used to connect two objects for indicating the control flow.

The set of events in BPMN is shown in Figure 1 and is briefly introduced as follows. A start event presents the start of a process, while an end event specifies the ending of a process. An intermediate event indicates the occurrence of an event during a process. A message event is triggered when a message is received. A timer event is triggered when a particular date or time is reached. Finally, an error event is started when an error is detected during a process.

There are two types of activities: tasks and subprocesses. A task is modeled as an activity when a work is going to be performed within a process. A subprocess is composed of multiple tasks and there is a flow between these tasks. Gateways are used to model merging and diverging of sequence flows. An exclusive gateway indicates that only one outbound sequence is performed. Its purpose is to join several diagrams into one or divide one into several. A parallel gateway presents the outbound activities that are performed in parallel. More than two output flows are modeled via complex gateways.

BPMN elements are connected with each other via flows. Messages are exchanged between different pools via message flows, whereas control flow objects are linked through sequence flows. Artifacts are linked with each other via associations. Different members are disjointed from each other with a pool. An organization is usually represented by a pool. It can either be empty or have some processes. Lanes split activities in a pool from each other. Artifacts in a BPMN model are used to present new information to a user. Data objects are one of the artifacts which are used for displaying information. Basically, they do not affect the process but instead they provide extra information to the operator. Group does not have any effect on a process either and is used to categorize similar activities into a group. Finally, annotations deliver extra information to a user. BPMN is implemented in many software tools, among which are Oracle BPMN studio,[70] Intalio,[71] Signavio Process Editor,[72,73] and so on.

Petri nets

The old-fashioned control theory works for time-driven systems, that is, systems of synchronous and continuous

discrete variables, modeled by difference or differential equations. Due to an extension in the scope of control theory into the areas of robotics, manufacturing, communication and computer networks, there is an increasing demand for several models having the capability of describing systems that evolve in accordance with the abrupt occurrence, at possibly unknown irregular intervals of physical events. Such systems whose states are symbolic or logical, instead of numerical values that vary in reaction to events, are called discrete event systems and their corresponding models are called discrete event models.

As a mathematical and graphical tool for modeling DEDS (discrete event dynamics systems), Petri nets are suitable for modeling concurrent systems.[74] Petri nets have found their extensive applications to supervisory control of discrete event systems[75–78] and further in previous works.[79–81] This feature makes Petri nets a successful candidate for defining semantics of BPMN models in a formal way. Theoretically, Petri nets have become quite strong since numerous tools have been established for their analysis during the past several decades.

A Petri net is a bipartite-directed graph composed of places and transitions, where arcs are used to connect a transition to a place or vice versa, but it is not possible to connect the same type of nodes, that is, transition to transition, or place to place. Places are usually represented by circles and transitions are represented as rectangles. Tokens are placed in places and tokens represent the things that will flow through the system.

Definition 1. A Petri net is a 4-tuple $N = P, T, F, V$,[20–25,82–84] graphically represented by a bipartite graph, where

- P and T are finite sets of places and transitions, respectively.
- $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$.
- Flow relation $F \subseteq (P \times T) \cup (T \times P)$ is the set of arcs.
- V is a weight function: $V \in [F \to \mathbb{N}^+ = \{1, 2, \ldots\}]$.

Basic elements of a Petri net contain places, transitions, and arcs. Place p is said to be an input place of transition t if there is a directed arc from p to t. Place p is said to be an output place of transition t if there is a directed arc from t to p. ${}^\bullet t$ is used to denote the set of input places of transition t, whereas $t^\bullet$ denotes the set of output places of transition t. In a similar way, ${}^\bullet p$ denotes the set of input transitions of place p, whereas $p^\bullet$ denotes the set of output transitions of place p.

Definition 2. A marking of Petri net $N = P, T, F, V$ is a mapping from P to $\mathbb{N} = \mathbb{N}^+ \cup \{0\}$, where $M(p)$ represents the number of tokens in place p.

Definition 3. A marked Petri net is a tuple and is denoted as (N, M_o), where N represents the Petri net and M_o is the initial marking of N.

Definition 4. (WF nets). A Petri net $N = P, T, F, V$ is said to be a workflow (WF) net[85] if

- N includes two distinct places σ and ϕ such that σ is a source place while ϕ is a sink place, that is, ${}^\bullet\sigma = \emptyset$ and $\phi^\bullet = \emptyset$.
- If a transition t^* is added into it such that $\phi \in {}^\bullet t^*$ and $\sigma \in t^{*\bullet}$, then the resulting net is strongly connected.

In our case, we focus on an important consistency property called n-soundness. It means the proper termination of the n-cases execution and the lack of tasks or conditions which do not contribute to the processing of cases.

We denote the initial markings as $n \cdot \sigma$ and final markings as $n \cdot \phi$ where n represents n cases ($n \in \mathbb{N}$).

Definition 5. (Soundness). A WF net $(N, n \cdot \sigma)$ with N being a Petri net carrying n cases, M being a marking, and $[M\rangle$ being the marking set, is sound[83] if

$$\forall M \in [n \cdot \sigma\rangle, n \cdot \phi \in [M\rangle$$

- For all states Ms that are reachable from the initial marking with a firing sequence which leads from a state M to the final marking

$$\forall M \in [n \cdot \sigma\rangle, M(\phi) \geqslant n \Rightarrow M = n \cdot \phi$$

- The last state is the only reachable state starting from the initial state with n tokens in place ϕ

$$\forall t \in T, \exists M \in [n \cdot \sigma\rangle : M[t\rangle$$

- Finally, there should be no dead transitions in $(N, n \cdot \sigma)$.

Mapping BPMN elements onto Petri nets

BPMN and Petri nets are completely different standards. The former lacks proper semantics, whereas the latter includes proper semantics that is helpful in analyzing BPMN diagrams. A BPMN process has the following characteristics: (1) a start event has no incoming flow and a single outgoing flow, (2) an end event has no outgoing flow but can have multiple incoming flows,

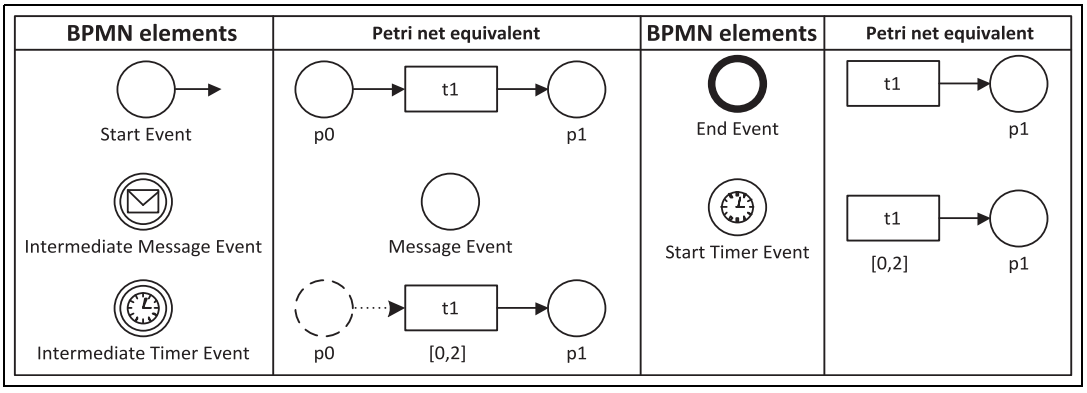

Figure 2. Mapping of events onto Petri nets.

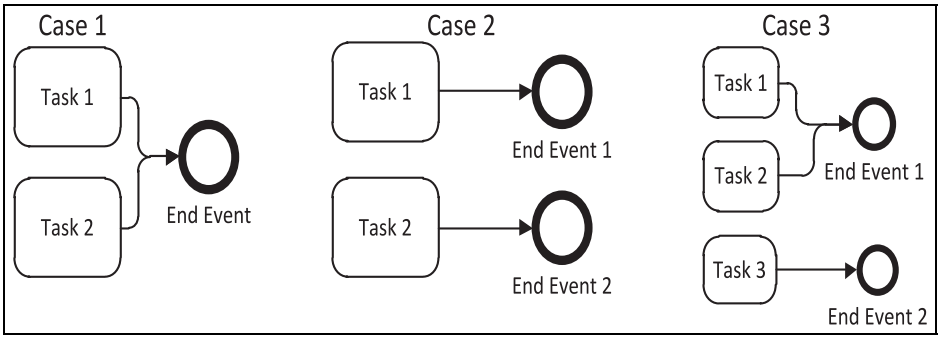

Figure 3. End event cases.

(3) events and activities have multiple input flows and multiple output flows, (4) a decision or fork has precisely single incoming flow and more than one outgoing flow, and (5) a merge or join gateway has more than one input and a single output.

Event mapping

Anything that occurs during a sequence of a business process is called an event. It can be referred to as a trigger or a result. They are centered on where they happen in a business process. An event can occur at the start of a process, middle, or at the end of a process. A token is generated when a start event is being triggered in a new process. A token is consumed when an end event signal appears and this token sinks under such a condition. In Figure 2, the mapping between a subset of events onto Petri net is given. A start event is mapped onto a Petri net containing two places and a transition. For an end event, we cannot simply just do the transformation as shown in the mapping in Figure 2. Three special cases of end events are shown in Figure 3: a) all sequence flows come to one end event; b) there are numerous end events and each of them has one incoming sequence flow; and c) the combination of Cases 1 and 2. If we simply use the transformation given in Figure 2 for an end event, we are not able to decide which case occurs. To solve this problem, we use a separate end event for each task. Hence, there is a single transition and single place for each sequence flow as in Figure 4.

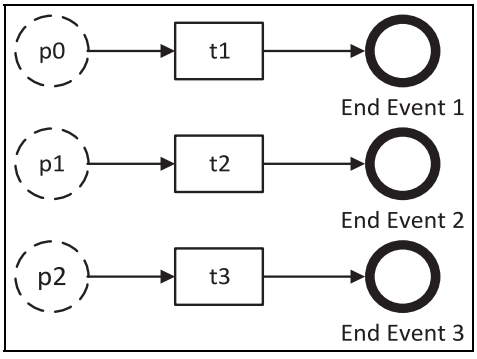

Figure 4. Solution to the end event problem.

If we simply use the transformation given in Figure 2 for an end event, we will not be able to decide which case occurs. To solve this problem, we use a separate end event for each task. Hence, there is a single transition and single place for each sequence flow as shown in Figure 2.

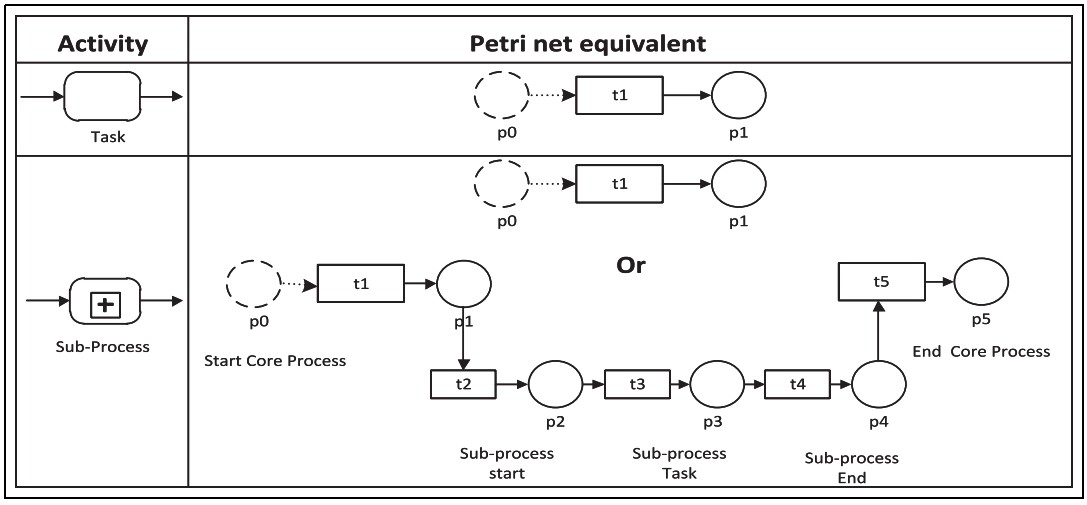

Figure 5. Mapping activities onto Petri nets.

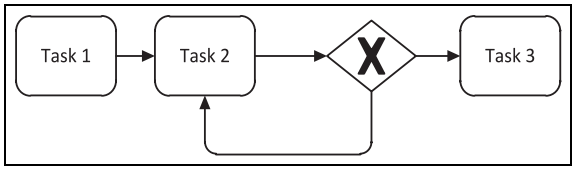

Figure 6. Case of task or gateway having cycles.

A timer event is mapped onto a Petri net using a transition and a place. Here, the transition should be associated with time, that is, the initial time and final time. At the initial time, the transition may or may not fire, but it is supposed to fire at the final time. An intermediate event is triggered when a message is received and it is transformed onto a Petri net using a single place.

Activity mapping

Tasks are found to be the most essential process elements, which show work in a process. In BPMN, a task indicates an atomic activity which is contained within a process. Sub-processes are used when the activity can be further decomposed. For our purpose, we map task and sub-process onto Petri nets as a single transition and a place as depicted in Figure 5.

For a task or a sub-process without any cycle, we can simply transform it onto a Petri net as shown in Figure 5. However, if there is a cycle in case of tasks as shown in Figure 6, we cannot simply do so.

A demonstration is given in Figure 7. If a task with a cycle is transformed simply using the method given in Figure 5, a deadlock occurs, that is, task $t1$ cannot be fired until there is a token in place $p1$ and $p5$.

Notice that Task $t1$ can fire if both $p1$ and $p5$ are filled with tokens. However, according to the BPMN diagram, Task 2 can be triggered if Task 1 or Gateway 2 is completed. To solve this problem, we just need to remove the place connected with Gateway 2 and connect transition $t4$ with $p1$. As a result, the obtained Petri net, depicted in Figure 8, shows how to transform a task with a cycle.

Gateway mapping

There are four types of gateways for mapping onto Petri nets: inclusive gateway, parallel gateway, data based exclusive (XOR) gateway, and event-based

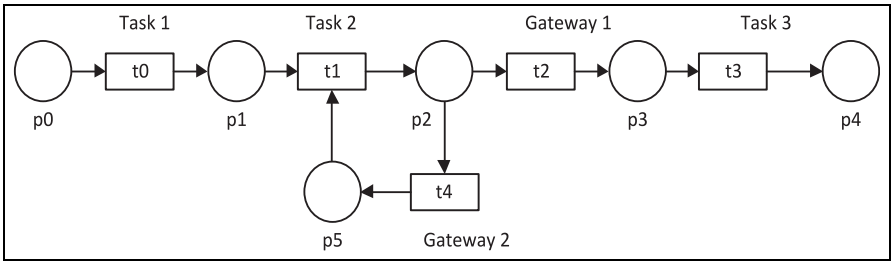

Figure 7. Transformation of a task or gateway having cycles.

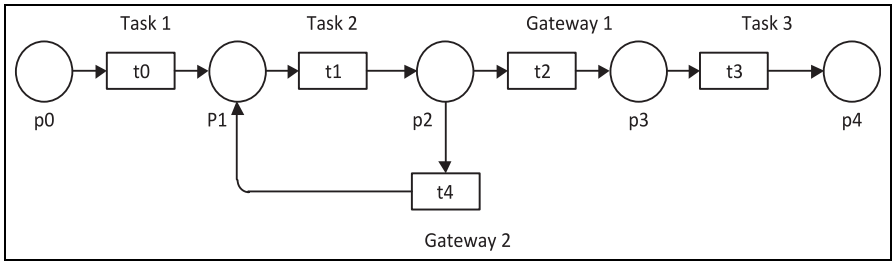

Figure 8. Recommended method to solve the cycle problem.

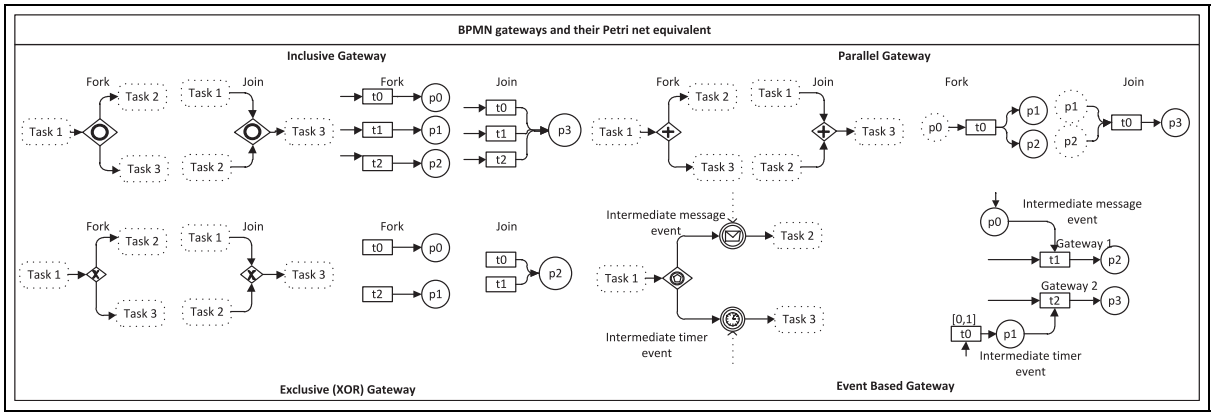

Figure 9. Mapping of gateways onto Petri nets.

gateway. Exclusive gateway is an alteration point of a business process. For instance, if there are many paths in a process, only one path can be taken among many paths. Inclusive gateway is also a partition point and it may trigger more than one outgoing paths but no substance that all outgoing conditions have satisfied or not. A parallel gateway is used when there is an execution of concurrent activities. In this situation, when the process arrives at fork of parallel gateway, tokens are split into multiple tokens and merged when it reaches at joining parallel gateway. Event-based gateway is somehow similar to exclusive gateway but the trigger of the gateway is triggered by third party, that is, message sent by the costumer or intermediate timer event. Mapping of gateways onto Petri net is shown in Figure 9. The probability can be added to exclusive, inclusive, and event-based gateways. The probability for exclusive and event-based gateways can be found using the property $P(A) = 1 - P(\bar{A})$, whereas in inclusive gateway, if there are two paths, each of them can be choosen or both can be choosen. In this case, the property $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ can be utilized.

Swim lanes

The frequently used element in BPMN is the swim lane which includes pools and lanes. A BPMN pool characterizes a contributor who contributes in a process. It is a rectangular package that can comprise flow object tasks and sub-processes. A lane is also a rectangular package, a sub-partition in a pool, where activities of a process are categorized and organized. The main functionality of a swim lane is to assembe a set of elements into a single unit. It does not have any impact on the WF. Hence, to transform a swim lane onto a Petri net, we simply convert it into different layers as depicted in Figure 10. Our developed software package is not capable to transform it into Petri nets as GPenSIM is not a graphical tool, but in the manual transformation, we can transform it as separate layers for each lane as depicted in Figure 10.

Development of BPMN framework for analysis

Based on the above transformation, we can discuss how to analyze a BPMN model. In the process of doing it, we present a framework. A block diagram as shown in Figure 11 is used to illustrate the framework. A BPMN diagram is created in Signavio Process Editor.[72,73] Then, there are two ways for transformation, that is, automatic and manual ways. By the former, an XML file from Signavio Process Editor is downloaded and inserted into a developed software package, where the

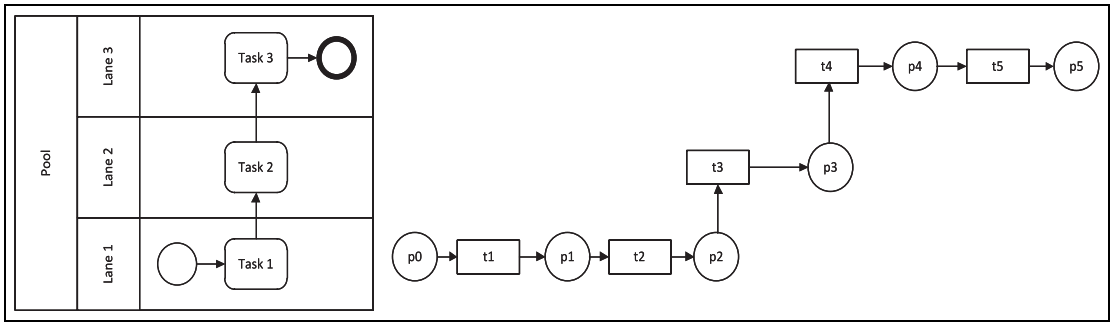

Figure 10. Mapping of swim lane onto Petri nets.

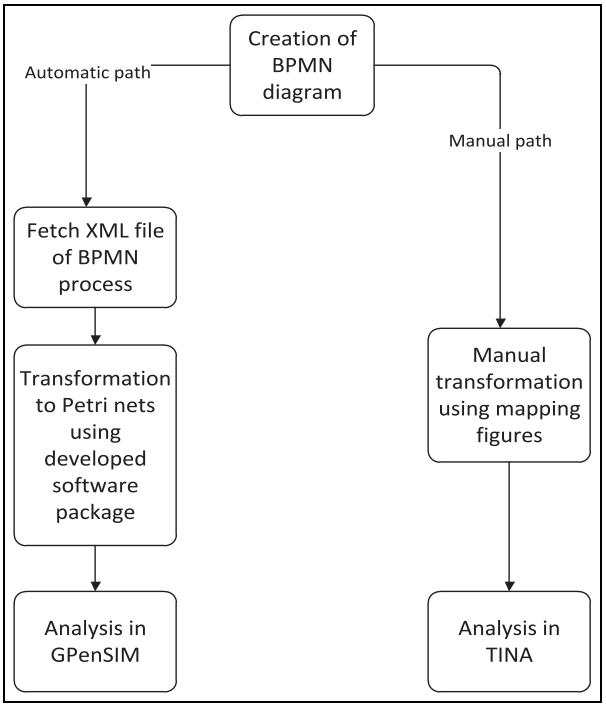

Figure 11. Task execution diagram.

BPMN process diagram is transformed onto Petri nets by creating "m" files. Afterward, these "m" files are used in MATLAB for analysis. By the manual way, the transformation from a BPMN model onto Petri nets is done manually using the mapping figures and then its analysis is done using TINA. First, the manual method can be used for a small-sized BPMN model having less complexity, while the automatic method can be implemented for large complex BPMN models without requiring any human effort involved in mapping. Second, the manual method can also be used to compare the results obtained by GPenSIM because GPenSIM is not a graphical-oriented tool. Figure 11 shows the assembly that is employed.

Step-by-step procedure of analysis using GPenSIM (automatic path) is as follows:

- Input: A BPMN diagram.
- Output: Analysis result obtained in GPenSIM.
- Create a BPMN diagram according to user's specification and requirements using a tool which can export the process diagram onto XML.
- Fetch the XML file using our developed tool.
- Select the probability function.
- Develop a tool process XML file and transform it into a Petri net.
- Output Petri net that describes the specification requirement of BPMN.
- Analyze the transformed Petri net in the GPenSIM tool.

End

Step-by-step procedure of analysis using TINA (manual path) is as follows:

- Input: A BPMN diagram.
- Output: Analysis result obtained in TINA tool.
- Create a BPMN diagram according to user's specification and requirements.
- Assign names to the tasks.
- Assign start and end event names to the BPMN diagram.
- Select the gateways according to the requirements of BPMN diagram.
- Use mapping figures to transform start and end events.
- Feed tokens to places and time to transitions using TINA.
- Obtain the analysis results of the Petri net from TINA.

End

Construction and format of BPMN models

To implement the proposed method, we need to decide which tools can be used to create a static BPMN diagram. The main concerns are that it should be an open source, have the option to export the BPMN diagrams

Algorithm 1: Transformation from BPMN into Petri net elements

- Input: XML file of BPMN diagram.
- Output: A Petri net model.
- $e_j = [e_s, e_e, t_s, g_e, g_i, g_p, g_{eb}, s_f, a_s]$
- // e_s = start event, e_e = end event, t_s = tasks, g_e = exclusive gateway, g_i = Inclusive gateway, g_p = parallel gateway, g_{eb} = event based gateway, s_f = sequence flows, a_s = annotations, δ = process end, $mapp_p$ = transformation rules (discussed in section "Mapping BPMN elements onto Petri nets").
- Convert XML file onto BPMN model.
- $B_{j,k}$ = Array of events, j = number of events, k = occurrence of each event.
- while (e_j! = end).
- {
- $m = e_j$
- $k = 1$
- while ($m \leqslant \delta$)
- {
- $B_{(j,k)} \leftarrow m$ (m = List of BPMN elements)
- $k++$
- $m++$
- }
- e_j++
- $j++$
- }
- $C_{j,k} = B_{j,k}$
- $C_{j,k} \leftarrow mapp_p$
- for ($a = 1$ to j)
- {
- for ($b = 1$ to k)
- {
- $B_{a,b} \leftarrow mapp_p$ (Transformation of BPMN elements into Petri nets)
- }
- }

END

into XML format, and be easy to use. It is known that Signavio Process Editor is one of the most popular tools for BPMN models and it is also an open source tool. Hence, we select Signavio as the tool for the creation of BPMN diagrams. We choose the XML format due to its usability, compatibility, and relaxed dealing out in Java program. Moreover, it is extendable and a widely used standard.

Transformation of BPMN models into Petri nets

This module consists of a developed Java Program whose input is an XML file that is exported from Signavio. This module outputs a Petri net model after the BPMN diagram is transformed. The elements, such as start events, end events, gateways, tasks, and sequence flows, can be inferred by this program. As done in the third section, that is, by mapping BPMN elements onto a Petri net, the mapping is explained clearly, and using those mapping rules, we develop a software package. In this module, a static BPMN model is converted into a dynamic Petri net model. The following algorithm shows the proposed approach.

Analysis

Several tools exist for Petri net modeling such as SPNP, INA, CPN Tools, GPenSIM, and Renew Tool. We choose GPenSIM for analysis since it has more functionalities (i.e. mathematical formulas) as it is embedded in the MATLAB environment, which makes the simulation to be faster; the only restriction is that it does not have a graphical view. The integration with MATLAB can also harness diverse toolboxes available in the MATLAB environment by combining GPenSIM with the Control System Toolbox. The advantages of GPenSIM over other toolboxes are as follows: we can add probability to the gateways, time can be easily added to transitions, and siphons and invariants can also be found easily.

Analysis under GPenSIM. In this section, we present a step-by-step procedure of how our developed software package interacts with GPenSIM software tool.[86,87]

Step 1: Defining a Petri net graph. In the first step, elements are defined in a Petri net definition file (PDF). In the PDF, a name of a Petri net is defined and then elements, that is, places and transitions, are identified. Moreover, the way in which these elements are connected is also defined. It is the static part. The weights on arcs are part of this file. The step-by-step procedure of the PDF file is as follows:

1. Mention all elements as a function: function [PN_name, set_of_places, set_of_trans, set_of_arcs] … = model_def (global_info)
2. Assign a name to a Petri net: PN_name = "Name of a Petri net";
3. Identification of places by their names: set_of_places = {"Place-1 "Place-2," "Place-3,"…};
4. Identification of transitions by their names: set_of_trans = {"Transition-1,"…};
5. Finally, elements connections are defined with the weights on arcs: set_of_arcs = {"Place-1," "Transition-1," 1, …"Place-2," "Transition-1," 2, …"Transition-1," "Place-3," 1, …};

It starts from defining functions and moving down with the set of places, the set of transitions, and the set of arcs. While defining the set of arcs, we should also decide the weight of an arc, as given above in the first

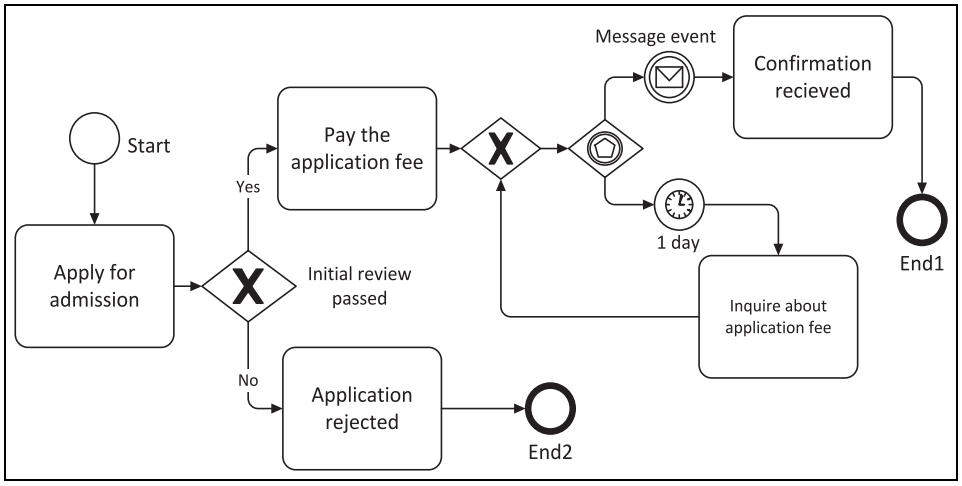

Figure 12. BPMN model of Example 1.

place and transition with weight 1 and in the second place and transition with weight 2.

Step 2: Simulation file. The second step includes the dynamics of a Petri net. In this step, we need to inscribe the main simulation file (MSF). At first, we load the static Petri net model, move on further with the dynamics of the Petri net, and decide the number of tokens in places and the firing time of transitions. The steps are as follows:

1. Load the Static Petri net Graph: png = petrinet-graph ("model_def");
2. Set the initial marking on places: dynamic_info.initial_markings = {"Place-1," 2, "Place-2," 4, …};
3. Assign firing time to transitions: dynamic_info.firing_times = {"Transition-1," 20, …};

Step 3: Simulation results

In this step, the function *gpensim* is used to simulate a Petri net model.
Sim_Results = *gpnsim* (png, dynamic_info);

Step 4: View simulation results

Simulation results can be viewed in this step;
print_statespace (Sim_Results).

Step 5: Plot the results. Finally, the function *plotp* is used to plot the results. The number of tokens in places is plotted on *y*-axis and time on *x*-axis.

plotp (Sim_Results, {"Place-1," "Place-2,"…});

Analysis under TINA tool. TINA[88,89] is a tool for analyzing Petri nets with the possibility of adding priorities, time, and stopwatches. The purpose of choosing TINA is its graphical view of places, transitions, arcs, and tokens, which is lacked in GPenSIM. It can help in building reachability analysis, structural analysis, and stepper simulator.

Table 1. Comparison of different Petri net tools used.

Properties	Tools	
	GPenSIM	TINA
Liveness	✓	✓
Safe	×	✓
Invariants	✓	✓
Siphon	✓	✓
Colored Petri Nets	✓	×
Reachability states	✓	✓
Graphical view	×	✓
Probability to the gateways	✓	×
Timed Petri nets	✓	✓

TINA: Timed Petri Net Analyzer.

Comparison of tools

There are two tools which we use in analyzing BPMN models. They are General Petri Net Simulator (GPenSIM) and TINA. Table 1 shows that GPesnSIM lack the graphical view of Petri nets, whereas TINA lacks addition of probability to transitions.

Experimental studies

Example 1

The BPMN model of a student applying to a university is shown in Figure 12. Upon the submission of documents to the university, an initial review process for the

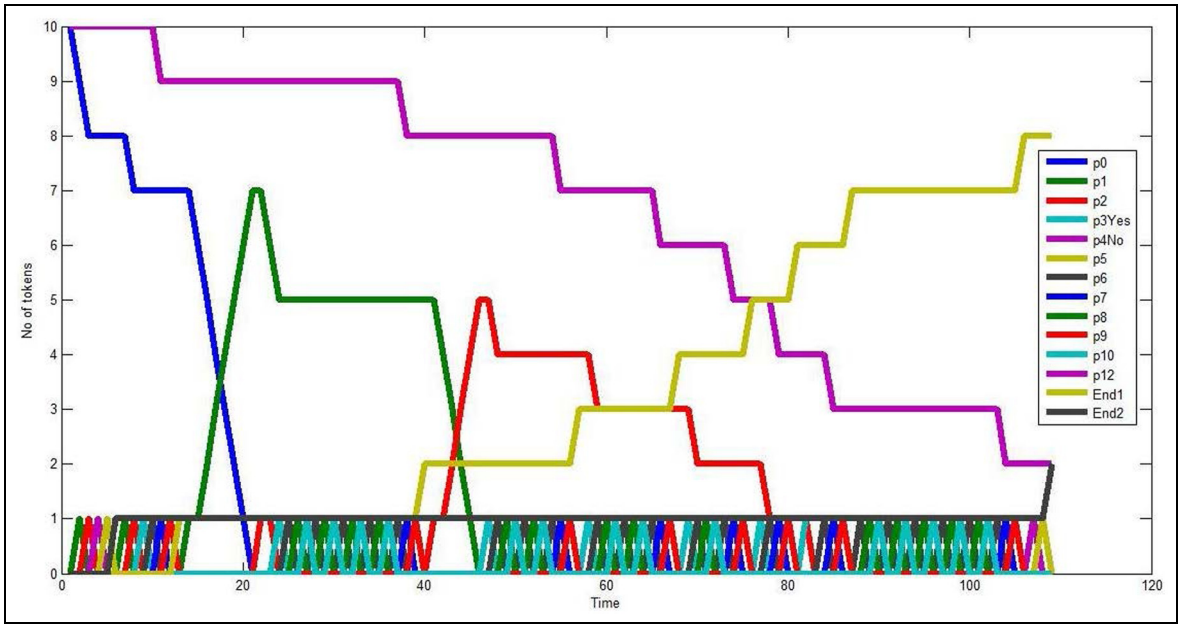

Figure 13. Simulation result of Example 1 in GPenSIM.

candidate's admission starts. If he fails to qualify, a rejection email is sent by the university, otherwise the student is prompted to pay the application fee. The student then pays the application fee and a message is generated acknowledging that the payment is done successfully. On the other hand, if the applicant is unable to receive a message, he waits for a duration of 1 day, till he receives an acknowledgment of the received payment.

Analysis for Example 1 using GPenSIM. GPenSIM, a tool integrated with MATLAB environment, is used for the analysis of Petri net models. Figure 13 illustrates the simulation result of a student who is applying to the university for an admission. The x-axis shows the firing times, whereas y-axis depicts the number of tokens in places. Initially, the tokens are set to be 10 in initial place $p0$ showing the number of applicants and 10 tokens are added to place $p11$ in order to trigger message events. The probabilities for the gateway is as follows: Initial_review_passed_yes = 0.8 and Initial_review_passed_no = 0.2. As the probability is set with the gateways, the initial tokens will divide accordingly in the end places. At the end, there are eight tokens in place *End*1 and two tokens in place *End*2, which shows that 80% of the students who applied for admission usually pass the initial review, whereas 20% students fail to pass initial review. Moreover, two tokens in place $p11$ indicate that the message event is triggered eight times. The average simulation time is noted as 0.587 s.

Analysis under TINA tool. The transformed Petri net model of Example 1 (applying for an admission) is shown in Figure 14 and the verification results are illustrated in Table 2. The result displays that the transformed Petri net model of Example 1 is neither live nor safe. In fact, it is an ordinary Petri net which consists of one place invariant and no siphons in it. Initially, tokens are set to be 10 in place $p0$. With the use of stepper simulator in TINA, we can randomly simulate the transformed Petri net. The benefit of using TINA tool over GPenSIM is its ability to represent transitions, places, and arcs graphically. Moreover, stepper simulator displays token movement in places and indicates the enabled transitions.

Example 2

The BPMN model of a patient having an accident being admitted to the hospital is created in Signavio Process Editor. As shown in Figure 15, the patient is brought to outpatient department (OPD) and a token is assigned to the patient. The patient is then shifted to emergency because of his critical condition. After that, he is referred to surgical specialist and at the same time his tests are conducted, and the lab reports are transferred to medical specialist (M–S), cardiac surgeon (C–S), or both which is represented as B–T because the patient was priorly suffering from cardiac disease. Based on consensus between doctors, he is transferred to operation theater. After the successful operation and treatment, the patient is discharged from the hospital.

Analysis for Example 2 using GPenSIM. The simulation result of an accidental patient after transforming it to Petri nets using our developed software package is

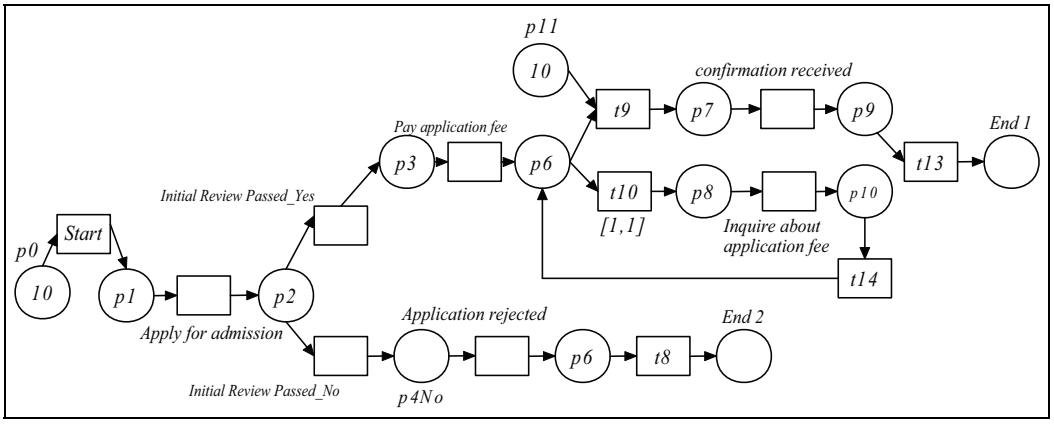

Figure 14. Transformed Petri net model of Example 1.

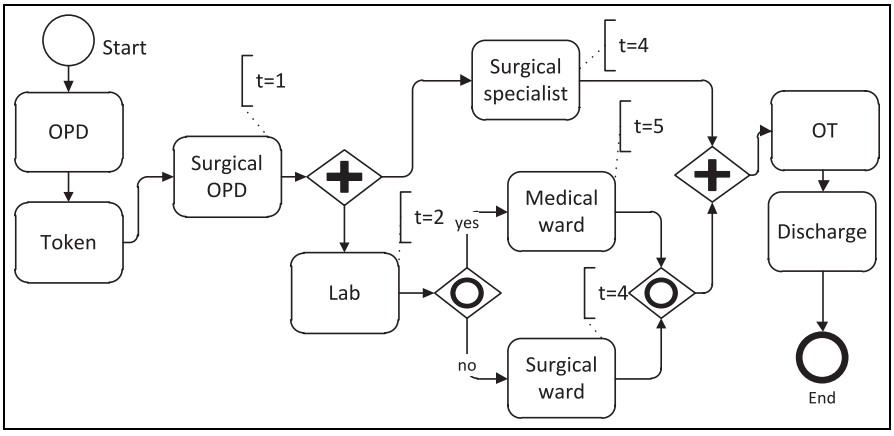

Figure 15. BPMN model of Example 2.

Table 2. Analysis of Example 1 in TINA.

	Tool
Properties	TINA
Liveness	×
Safe	×
Ordinary	✓
Pure	✓
Invariants	✓
Siphon	✓
Free choice	✓
Bounded	✓

TINA: Timed Petri Net Analyzer.

depicted in Figure 16. Initial tokens are set to be 10 in place $p0$. The probability is set with the inclusive gateway and the initial tokens will divide accordingly, whereas there is no need to involve probability with parallel gateway as all tasks need to be performed in parallel. The probability of the tasks M–S, C–S, and B_T is set to be 0.5, 0.0, and 0.5, respectively. The same number of tokens in the initial place and final place shows that net is sound.

Analysis under TINA tool. The BPMN model depicted in Figure 15 is transformed manually into Petri net using mapping figures depicted in section "Preliminary background." The transformed Petri net model of a BPMN model shown in Figure 15 is depicted in Figure 17 and verification results of the transformed Petri net model are illustrated in Table 3 which shows that the transformed net is pure and is ordinary, which neither has any place or transition invariants nor any siphon. Furthermore, it is bounded and a free-choice Petri net. As TINA lacks addition of probability to the gateways, we can only add firing delays to the transitions in TINA. Tokens can be moved between the initial and final time of transition once the initial time of a transition is enabled.

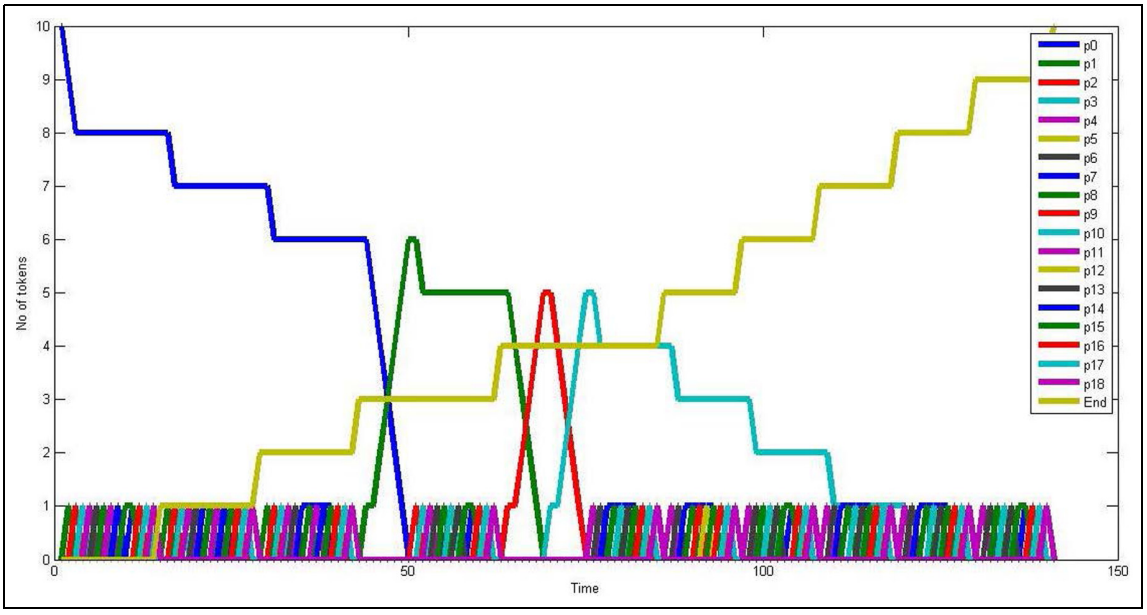

Figure 16. Simulation result of Example 2 in GPenSIM.

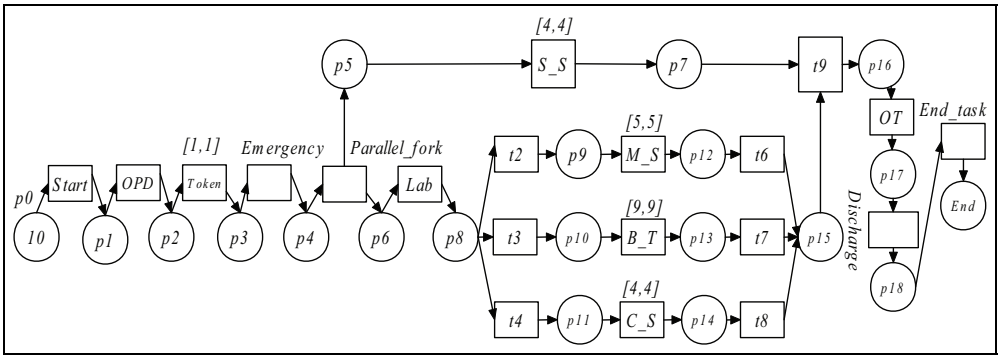

Figure 17. Transformed Petri net model of Example 2.

Table 3. Analysis of Example 2 in TINA.

	Tool
Properties	TINA
Liveness	×
Safe	×
Ordinary	✓
Pure	✓
Invariants	×
Siphon	×
Free choice	✓
Bounded	✓

TINA: Timed Petri Net Analyzer.

S-soundness

Soundness is a notion of correctness of WF nets. The need of S-soundness arises for the exclusive and event-based gateways. It states that the number of tokens in source place is equal to the sum of tokens in the sink places.

Definition 6. (S-WF net). A Petri net $N = <P, T, F, V>$ is an S-WF net if

- There is a source place σ and a set of sink places $\phi = \{\phi_1, \phi_2, \ldots\}$, that is, ${}^\bullet\sigma = \phi$ and $\phi_i^\bullet = \phi$ where $i = \{1, 2, \ldots\}$.
- If a transition t^* is added into it such that $\{\phi = \{\phi_1, \phi_2, \ldots\} \subseteq {}^\bullet t^*$ and $\sigma \in t^{*\bullet}$, then the resulting net is strongly connected.

Definition 7. (S-soundness). A S-WF net $(N, n.\sigma)$ with N being a Petri net carrying n cases, M being a marking, and $[M\rangle$ being the marking set, is sound if

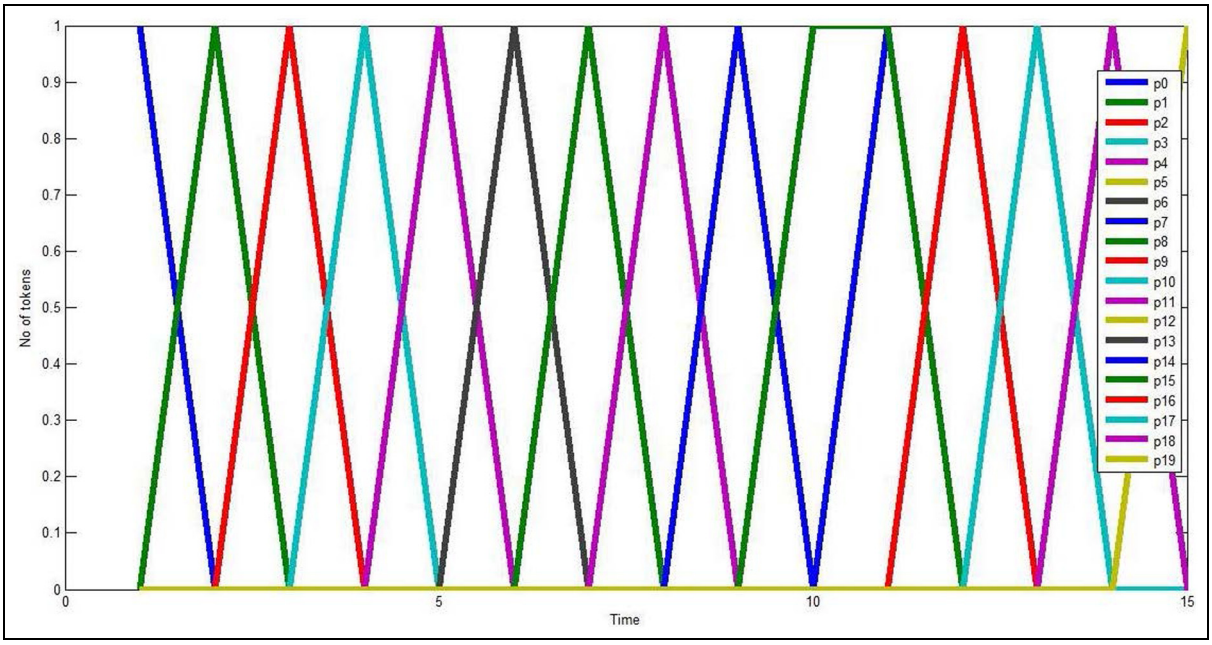

Figure 18. Example of S-soundness.

- The option to complete (for every case, it is always possible to reach the state marked as ϕ);
- Proper completion (for every case, if ϕ is marked, all other places should be empty);
- No dead transition;
- The number of tokens in the source place should be equal to the sum of tokens in the sink places.

Formally

- $\forall M \in [n \cdot \sigma\rangle, n \cdot \phi \in [M\rangle$
- $\forall M \in [n \cdot \sigma\rangle, M(\phi) \geqslant n \Rightarrow M = n \cdot \phi$
- $\forall t \in T_n \exists M \in [n \cdot \sigma\rangle : M[t\rangle$
- $\sigma = \sum_{k=1} \phi i$

The soundness property relates to the dynamic behavior of WF nets. As from Definition 7, the first and the foremost requirement is that it should always be possible to reach final state, that is, tokens in the sink places, starting from the initial state. From Figure 18, it is clear that there is a single token in place $p0$ at the start, and after firing transitions, the token is at a sink place which fulfills our first requirement. The second requirement is that when the token is in the sink place, all the places should be empty. Figure 18 depicts that when the token is in the place $p19$, rest of the places are empty. The first two requirements are known to be as proper termination. The third requirement states that there should be no dead transitions in a process and the last requirement states that number of tokens in the source place is equal to the sum of tokens in sink places. Moreover, it represents that either of the gateway paths is chosen the number of tokens in sink and source places are similar which shows the absence of deadlocks.

Remaining token problem

A process model is accomplished when at least one of the end tasks has been performed at least once and for that process instance there is no other enabled task. In terms of Petri nets, for a single process, if there is any token in a sink place, the rest of places should be empty. If there is still a token remaining in rest of the places except the source and sink ones, the process may not be completed properly. In Figure 19, if Task 3 is enabled and fired, the process will be completed but a token will be left in between Task 1 and Parallel join at place $p4$. In this scenario, the process needs to be corrected in order to properly remove this remaining token. The analysis results in Figure 20 depict that the number of tokens in the sink and source places is not equal, which shows a deadlock in the system. Now, if the token is not terminated, there might be two cases, that is, either it will be a deadlock or a livelock. For the manual method, the places will be back-tracked to determine the token location in the Petri net model, whereas in the automatic method, GPenSIM itself will determine the location of the unterminated token in the model, which indicates the defected part of the model.

Comparison of number of elements in both standards

The simulation experiments are carried out with the help of using randomly generated BPMN processes. In

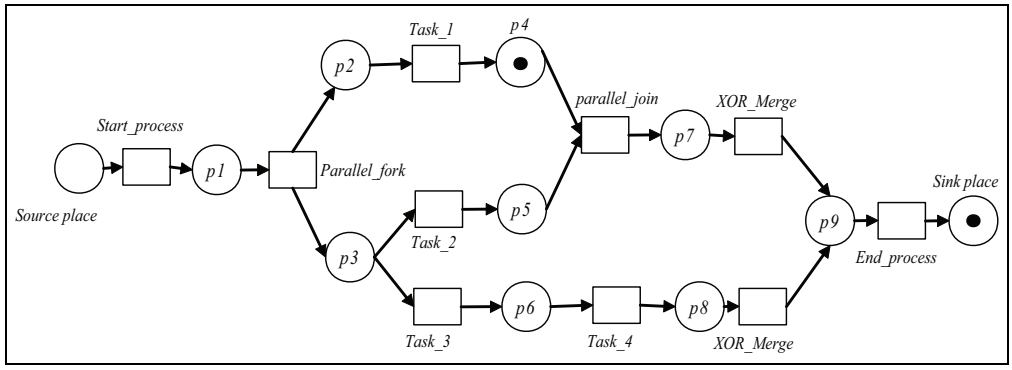

Figure 19. Petri net model with improper completion.

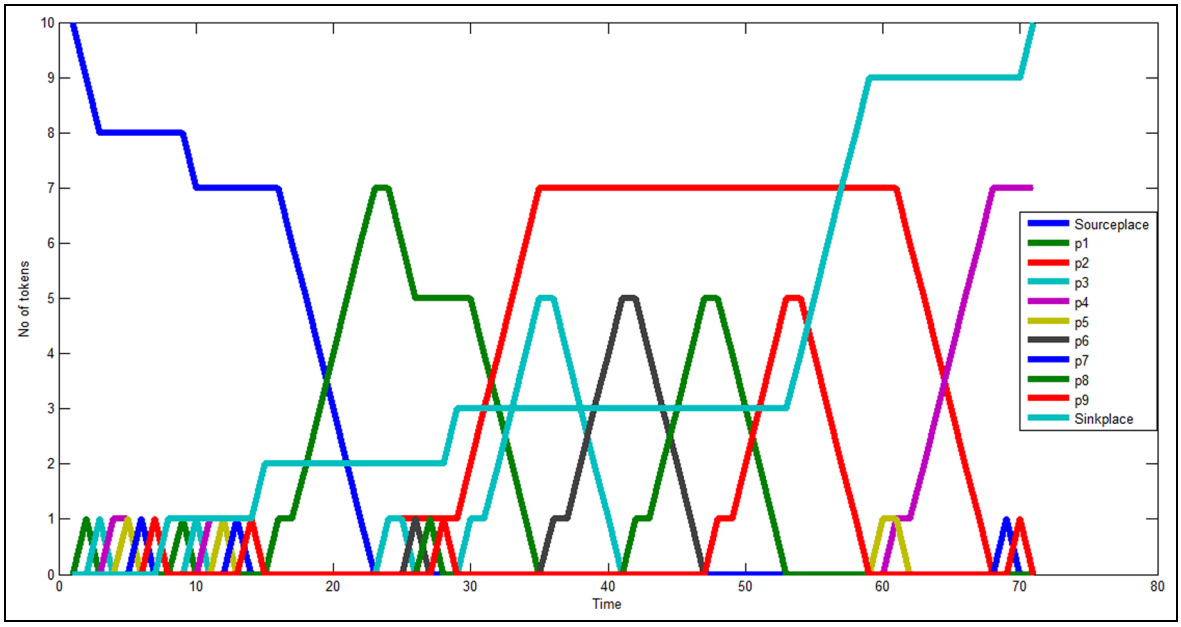

Figure 20. Analysis of Petri net model with improper completion in GPenSIM.

Table 4. Comparison of number of elements in BPMN and petri net standards.

S. no.	BPMN model			Transformed Petri net model		
	Tasks	Events	Gateways	Places	Transitions	Time (s)
1	5	5	2	14	13	0.587
2	9	3	4	20	20	1.388
3	46	10	9	83	74	2.184
4	4	3	1	10	9	0.27
5	8	3	1	14	13	0.301
6	5	3	3	14	12	0.298
7	5	2	2	13	13	0.286

BPMN: Business Process Model and Notation.

Table 4, based on the simulation, we present the comparison of the number of elements in these two models. Using a Petri net model, we can perform dynamic analysis in contrast to static analysis using BPMN. The simulation time using GPenSIM in MATLAB is shown in Table 4.

Table 5. Comparison of transformation time with the work by Dijkman et al.[26]

S. no.	BPMN model			Comparison of transformation time	
	Tasks	Events	Gateways	Time (s)	Time (s)[26]
1	11	2	11	1:086	1:234
2	4	2	2	0:483	0:703
3	3	2	4	0:587	0:734
4	7	4	8	0:971	1:297
5	2	2	4	0:493	0:797
6	5	2	2	0:285	0:641

BPMN: Business Process Model and Notation.

The tasks, events, and gateways of the BPMN model in Table 5 are taken from the work by Dijkman et al.[26] and the transformation time is compared with our proposed approach which includes transformation and analysis time. It can be seen that there is a significant reduction in time using the proposed approach compared with the work by Dijkman et al.[26]

Conclusion and future work

The BPMN standard lacks formal semantics, which limits its applications since one cannot examine the correctness of BPMN models from the perspective of syntax. We choose Petri nets as a dynamic model for verification due to their simulation capabilities, including mathematical functions and some other toolboxes in MATLAB. The contribution of this article is the extension of subset of BPMN elements that are transformed onto Petri nets. Furthermore, the addition of time with the tasks and the probability with gateways add additional features to business process modeling. Using the proposed mapping rules, a software package is developed that transforms BPMN elements onto Petri nets. Simulation results show that using this technique, the transformation time is subsequently reduced. The manual method further helps us view Petri net elements graphically, as well, and provides some other analysis properties such as safeness which GPenSIM lacks. Our future research is to make transformation of all BPMN constructs on the basis of the formalism proposed by Kheldoun et al.[69] and extended with time and probability. Moreover, we will study the deadlock problem[90] in a business process model and use this model to the wireless sensor networks.[91,92]

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the National Natural Science Foundation of China, under grant no. 61873342, and the Science and Technology Development Fund, MSAR, under grant no. 122/2017/A3.

ORCID iD

Zhiwu Li https://orcid.org/0000-0003-1547-5503

References

1. BPMN specification—business process model notation. *BPMN.org*, 2017. http://www.bpmn.org (accessed 3 April 2017).
2. Shapiro R. A technical comparison of XPDL, BPML and BPEL4WS. *Cape Visions* 2002; 37: 38–39.
3. Eriksson HE and Penker M. *Business modeling with UML: Business Patterns at Work*. 1st ed. John Wiley & Sons, Inc., New York, USA: 1998.
4. Yu H and Wu D. Enterprise modeling based on IDEF and UML. In: *2015 4th international conference on advanced information technology and sensor application (AITS)*, Harbin, China, 21–23 August 2015, pp.59–62. New York: IEEE.
5. Dennis AR, Hayes GS, Daniels RM, et al. Re-engineering business process modeling. In: *Proceedings of the 27th Hawaii international conference on system sciences*, Maui, HI, 4–7 January 1994. IEEE Comput. Soc. Press.
6. Mendling J, Lassen KB and Zdun U. On the transformation of control flow between block-oriented and graph-oriented process modeling languages. *Int J Business Process Integr Manage* 2008; 3: 96–108.
7. Ouyang C, Verbeek E, Van Der Aalst WM, et al. Formal semantics and analysis of control flow in WS-BPEL. *Sci Comp Program* 2007; 67: 162–198.
8. Chen YF, Li ZW, Barkaoui K, et al. Compact supervisory control of discrete event systems by Petri nets with data inhibitor arcs. *IEEE T Syst Man Cyber* 2017; 47: 364–379.
9. Uzam M, Li ZW, Gelen G, et al. A divide-and-conquer-method for the synthesis of liveness enforcing supervisors

for flexible manufacturing systems. *J Intel Manufact* 2016; 27: 1111–1129.
10. Li ZW, Liu GY, Hanisch MH, et al. Deadlock prevention based on structure reuse of Petri net supervisors for flexible manufacturing systems. *IEEE T Syst Man Cy A* 2010; 42: 178–191.
11. Ma ZY, Li ZW and Giua A. Design of optimal Petri net controllers for disjunctive generalized mutual exclusion constraints. *IEEE Trans Automat Contr* 2015; 60: 1774–1785.
12. Ye JH, Li ZW and Giua A. Decentralized supervision of Petri nets with a coordinator. *IEEE T Syst Man Cyber* 2015; 45: 955–966.
13. Ma Z, Li Z and Giua A. Characterization of admissible marking sets in Petri nets with conflicts and synchronizations. *IEEE Trans Automat Contr* 2017; 62: 1329–1341.
14. Ma Z, Tong Y, Li Z, et al. Basis marking representation of Petri net reachability spaces and its application to the reachability problem. *IEEE Trans Automat Contr* 2017; 62: 1078–1093.
15. Zhang JF, Khalgui M, Li ZW, et al. Reconfigurable coordination of distributed discrete event control systems. *IEEE Trans Contr Syst Tech* 2015; 23: 323–330.
16. Wu NQ and Zhou MC. Schedulability analysis and optimal scheduling of dual-arm cluster tools with residency time constraint and activity time variation. *IEEE Trans Automat Sci Eng* 2012; 9: 203–209.
17. Wu NQ and Zhou MC. Modeling, analysis and control of dual-arm cluster tools with residency time constraint and activity time variation based on Petri nets. *IEEE Trans Automat Sci Eng* 2012; 9: 446–454.
18. Wang X, Khemaissia I, Khalgui M, et al. Dynamic low-power reconfiguration of real-time systems with periodic and probabilistic tasks. *IEEE Trans Automat Sci Eng* 2015; 12: 258–271.
19. Zhang SW, Wu NQ, Li ZW, et al. Petri net-based approach to short-term scheduling of crude oil operations with less tank requirement. *Informat Sci* 2017; 417: 247–261.
20. Zhu GH, Li ZW, Wu NQ, et al. Fault identification of discrete event systems modeled by Petri nets with unobservable transitions. *IEEE Trans Syst Man Cyber Syst*. Epub ahead of print November 2017. DOI: 10.1109/TSMC.2017.2762823
21. Liu GY, Li P, Li ZW, et al. Robust deadlock control for automated manufacturing systems with unreliable resources based on Petri net reachability graphs. *IEEE Trans Syst Man Cyber Syst*. Epub ahead of print April 2018. DOI: 10.1109/TSMC.2018.2815618
22. Zhu GH, Li ZW and Wu NQ. Model-based fault identification of discrete event systems using partially observed Petri nets. *Automatica* 2018; 96: 201–212.
23. Cong XY, Fanti MP, Mangini AM, et al. On-line verification of current-state opacity by Petri nets and integer linear programming. *Automatica* 2018; 94: 205–213.
24. Zhang HM, Feng L, Wu NQ, et al. Integration of learning-based testing and supervisory control for requirements conformance of black-box reactive systems. *IEEE Trans Automat Sci Eng* 2018; 15: 2–15.
25. Zhang H, Feng L and Li Z. A learning-based synthesis approach to the supremal nonblocking supervisor of discrete-event systems. *IEEE Trans Automat Contr* 2018; 63: 3345–3360. DOI: 10.1109/TAC.2018.2793662
26. Dijkman RM, Dumas M and Ouyang C. Semantics and analysis of business process models in BPMN. *Informat Software Tech* 2008; 50: 1281–1294.
27. Wong PY and Gibbons J. A process semantics for BPMN. In: *Proceedings of the international conference on formal engineering methods*, London, 2–4 June 2008, pp.355–374. Berlin; Heidelberg: Springer.
28. Gibson-Robinson T, Armstrong P, Boulgakov A, et al. W. FDR3 –a modern refinement checker for CSP. In: *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Berlin, Heidelberg, pp. 187–201. Springer.
29. Puhlmann F and Weske M. Investigations on soundness regarding lazy activities. In: Dustdar S, Fiadeiro JL and Sheth A (eds) *International conference on business process management*. Berlin; Heidelberg: Springer, 2006, pp.145–160.
30. Dijkman R and Van Gorp P. BPMN 2.0 execution semantics formalized as graph rewrite rules. In: Mendling J, Weidlich M and Weske M (eds) *Proceedings of the international workshop on business process modeling notation*. Berlin; Heidelberg: Springer, 2010, pp.16–30.
31. Raedts I, Petkovic M, Usenko YS, et al. Transformation of BPMN Models for behaviour analysis. In: *Modelling, simulation, verification and validation of enterprise information systems*, Funchal, 12–13 June 2007, pp.126–137. INSTICC.
32. Ramadan M, Elmongui HG and Hassan R. BPMN formalisation using coloured Petri nets. In: *Proceedings of the 2nd GSTF annual international conference on software engineering & applications (SEA 2011)*, Singapore, 12–13 December 2011. Singapore GSTF.
33. Koniewski R, Dzielinski A and Amborski K. Use of Petri nets and business processes management notation in modelling and simulation of multimodal logistics chains. In: *Proceedings of the 20th European conference on modeling and simulation*, Institute of Control and Industrial Electronics, Warsaw University of Technology, Warsaw, 28–31 May 2006, pp.28–31. European Council for Modelling and Simulation.
34. Meghzili S, Chaoui A, Strecker M, et al. Transformation and validation of BPMN models to Petri nets models using GROOVE. In: *Proceedings of international conference on advanced aspects of software engineering (ICAASE)*, Constantine, Algeria, 29–30 October, pp.22–29. IEEE.
35. Dijkman R, Dumas M and Ouyang C. Formal semantics and analysis of BPMN process models using Petri nets. *Technical report, Queensland University of Technology, Brisbane, QLD, Australia*, 2007.
36. Groefsema H and Bucur D. A survey of formal business process verification: from soundness to variability. In: *Proceedings of the 3rd international symposium on business modeling and software design*, Vienna, 2–4 July 2013, pp.198–203. SciTePress.
37. Kherbouche OM, Ahmad A and Basson H. Using model checking to control the structural errors in BPMN models. In: *2013 IEEE seventh international conference on*

research challenges in information science (RCIS), Paris, 29–31 May 2013, pp.1–12. New York: IEEE.
38. Roy S, Sajeev ASM, Bihary S, et al. An empirical study of error patterns in industrial business process models. *IEEE Trans Serv Comp*, 2014; 7: 140–153.
39. Awad A, Decker G and Weske M. Efficient compliance checking using BPMN-Q and temporal logic. *BPM*, 2008; 5240: 326–341.
40. Van Gorp P and Dijkman R. A visual token-based formalization of BPMN 2.00 based on in-place transformations. *Informat Software Tech* 2013; 55: 365–394.
41. Wong PYH and Gibbons J. Formalisations and applications of BPMN. *Sci Comp Program* 2011; 76: 633–650.
42. Börger E and Sörensen O. BPMN core modeling concepts: inheritance-based execution semantics. In: Embley DW and Thalheim *B* (eds) *Handbook of conceptual modeling*. London: Springer, 2007, pp.287–332.
43. Wong PYH and Gibbons J. A process semantics for BPMN. In: *ICFEM 2008*, Kitakyushu, Japan, 27–31 October 2008, pp.355–374. New York: ACM.
44. Favre C, Völzer H and Müller P. Diagnostic information for control-flow analysis of workflow graphs. In: Chechik M and Raskin JF (eds) *Tools and algorithms for the construction and analysis of systems*. Berlin: Springer, 2016, pp.463–479.
45. Ackoff RL. *Scientific method: optimizing applied research decisions*, 1962. New York: Wiley.
46. Lindsay A, Downs D and Lunn K. Business processes-Attempts to find a definition. *Informat Software Tech* 2003; 45: 1015–1019.
47. Aguilar-Saven RS. Business process modelling: review and frame- work. *Int J Product Econ* 2004; 90: 129–149.
48. Hepp M, Leymann F, Domingue J, et al. Semantic business process management: a vision towards using semantic web services for business process management. In: *Proceedings of the E-business engineering (ICEBE)*, Beijing, China, 12–18 October 2005, pp.535–540. New York: IEEE.
49. Lin Y and Strasunskas D. Ontology-based semantic annotation of process templates for reuse. *Proc CAISE* 2005; 5: 593–604.
50. Harrington HJ. *Business process improvement: the breakthrough strategy for total quality, productivity and competitiveness*. New York, NY: McGraw-Hill, 1991.
51. Davenport TH. *Process innovation: reengineering work through information technology*. Boston, MA: Harvard Business Press, 1993.
52. Hammer M. Reengineering work: don't automate, obliterate. *Harvard Business Rev* 1990; 68: 104–112.
53. Hammer M and Champy J. *Reengineering the corporation: manifesto for business revolution*. Grand Rapids, MI: Zondervan, 2009.
54. Kettinger WJ, Teng JT and Guha S. Business process change: a study of methodologies, techniques, and tools. *MIS Quarterly* 1997; 21: 55–80.
55. Phalp KT and Martin S. Quantitative analysis of static models of processes. *J Syst Software* 2000; 52: 105–112.
56. Phalp KT. The CAP framework for business process modeling. *Informat Software Tech* 1998; 40: 731–744.
57. Lohmann N, Verbeek E and Dijkman R. Petri net transformations for business processes–a survey. In: Jensen K and Van Der Aalst W (eds) *Transactions on Petri nets and other models of concurrency (ToPNoC II)*, vol. 5460. Berlin: Springer, 2009, pp.46–63.
58. Gottschalk F, Van Der Aalst WM and Jansen-Vullers M. Merging event-driven process chains. In: Zahor T (ed.) *Proceedings on the move to meaningful internet systems: OTM*. Berlin: Springer, 2008, pp.418–426.
59. Fisteus JA, Fernández LS and Kloos CD. Formal verification of BPEL4WS business collaborations. In: *International conference on electronic commerce and web technologies*, Zaragoza, 31 August–3 September 2004, pp.76–85. Berlin: Springer.
60. Fu X, Bultan T and Su J. Analysis of interacting BPEL web services. In: *Proceedings of the 13th international conference on World Wide Web*, New York, 17–22 May 2004, pp.621–630. New York: ACM.
61. Fahland D and Reisig W. ASM-based Semantics for BPEL: the negative control flow. In: *Proceedings of the 12th international workshop on abstract state machines*, Paris, 8–11 March 2005, pp.131–152. New York: ACM.
62. Farahbod R, Glässer U and Vajihollahi M. Specification and validation of the business process execution language for web services. In: *International workshop on abstract state machines*, Wittenberg, 24–28 May 2004, pp.78–94. Berlin: Springer.
63. Ferrara A. Web services: a process algebra approach. In: *Proceedings of the 2nd international conference on service oriented computing*, New York, 15–19 November 2004, pp.242–251. New York: ACM.
64. Hinz S, Schmidt K and Stahl C. Transforming BPEL to Petri nets. In: *International conference on business process management*, Nancy, 5–8 September 2005, pp.220–235. Berlin: Springer.
65. Van Der Aalst WM and Ter Hofstede AH. YAWL: Yet Another Workflow Language. *Informat Syst* 2005; 30: 245–275.
66. El-Saber N and Boronat A. BPMN formalization and verification using Maude. In: *Proceedings of the workshop on behaviour modelling-foundations and applications*, York, 22 July 2014. New York: ACM.
67. Roa J, Chiotti O and Villarreal P. Specification of behavioral anti-patterns for the verification of block-structured collaborative business processes. *Informat Software Tech*, 2016; 75: 148–170.
68. Villarreal PD, Lazarte I, Roa J, et al. A modeling approach for collaborative business processes based on the UP-ColBPIP language. In: *International conference on business process management*, Ulm, 7 September 2009, pp.318–329. Berlin: Springer.
69. Kheldoun A, Barkaoui K and Ioualalen M. Formal verification of complex business processes based on high-level Petri nets. *Informat Sci* 2017; 385: 39–54.
70. Das M, Deb M and Wilkins M. *Oracle business process management suite 11g handbook*. New York: McGraw-Hill, 2012.
71. Bhandari R, Suman U and Ramani AK. Web service composition through BPEL using Intaglio. In: *Proceedings of the computational intelligence and information technology*, Pune, India, 7–8 November 2011, pp.873–876. Berlin: Springer.

72. Chinosi M and Trombetta A. BPMN: an introduction to the standard. *Comp Standard Interfaces* 2012; 34: 124–134.
73. Signavio. Signavio offers collaborative process decision management, 2017, https://www.signavio.com/ (accessed April 2017).
74. Peterson JL. Petri nets. *ACM Comp Surv (CSUR)* 1977; 9: 223–252.
75. Cong X, Fanti M, Mangini A, et al. Decentralized diagnosis by Petri nets and integer linear programming. *IEEE T Syst Man Cyber* 2018; 48: 1689–1700.
76. Chen YF, Li ZW and Barkaoui K. New Petri net structure and its application to optimal supervisory control: interval inhibitor arcs. *IEEE T Syst Man Cyber* 2014; 44: 1384–1400.
77. Chen YF, Li ZW, Barkaoui K, et al. On the enforcement of a class of nonlinear constraints on Petri nets. *Automatica* 2015; 55: 116–124.
78. Bai LP, Wu NQ, Li ZW, et al. Optimal one-wafer cyclic scheduling and buffer space configuration for single-arm multicluster tools with linear topology. *IEEE T Syst Man Cyber* 46: 1456–1467.
79. Tong Y, Li Z and Giua A. On the equivalence of observation structures for Petri net generators. *IEEE Trans Automat Contr* 2016; 61: 2448–2462.
80. Tong Y, Li ZW, Seatzu C, et al. Verification of state-based opacity using Petri nets. *IEEE Trans Automat Contr* 2017; 62: 2823–2837.
81. Wang X, Li Z and Wonham WM. Dynamic multiple-period reconfiguration of real-time scheduling based on timed DES supervisory control. *IEEE Trans Ind Informat*, 2016; 12: 101–111.
82. Barkaoui K and Petrucci L. Structural analysis of workflow nets with shared resources. In: *Proceedings of the workflow management: net-based concepts, models, techniques and tools (WFM98, Volume 98/7 of Computing Science Reports)*, Lisbon, 22 June 1998, pp.82–95. Eindhoven: Eindhoven University of Technology.
83. Barkaoui K, Ben Ayed R and Sbai Z. Uniform verification of workflow soundness. *Trans Inst Measure Contr J* 2011; 33: 133–148.
84. Barkaoui K and Pradat-Peyre JF. On liveness and controlled siphons in Petri nets. In: Billington J (ed.) *Application and theory of Petri nets 1996*. Berlin: Springer, 1996, pp.57–72.
85. Van Der Aalst WM. Verification of workflow nets. In: Azema P and Balbo G (eds) *International conference on application and theory of Petri nets*. Berlin; Heidelberg: Springer, 1997, pp.407–426.
86. GPenSIM: a tool for mathematical modeling simulation of discrete-event systems. *Davidrajuh.net*, 2017. http://www.davidrajuh.net/gpensim (accessed April 2017).
87. Davidrajuh R. Developing a Petri nets based real-time control simulator. *Int J Simulat Syst Sci Tech* 2012; 12: 28–36.
88. The TINA toolbox Home Page—TIme Petri Net Analyzer by LAAS/CNRS. *projects.laas.fr*, 2017. http://projects.laas.fr/tina (accessed April 2017).
89. Gardey G, Lime D and Magnin M. Romeo: a tool for analyzing time Petri nets. In: *Proceedings of the international conference on computer aided verification*, Edinburgh, 6–10 July 2005, pp.418–423. Berlin: Springer.
90. Gu C, Li ZW, Wu NQ, et al. Improved multi-step look-ahead control policies for automated manufacturing systems. *IEEE Access*, https://doi.org/10.1109/ACCESS.2018.2872572 (accessed 21 September 2018).
91. Grichi H, Mosbahi O, Khalgui M, et al. RWiN: New methodology for the development of reconfigurable WSN. *IEEE Transactions on Automation Science and Engineering* 2017; 14: 109–125.
92. Grichi H, Mosbahi O, Khalgui M, et al. New power-oriented methodology for dynamic resizing and mobility of reconfigurable wireless sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 2018; 48: 1120–1130.