Business Process Verification using Formal Language Petri Net: An Approach

Pankaj Kasar
Assistant Professor
RCPatel Institute of Technology
Shirpur,Dist.Dhule(India)

Gaurav Paliwal
Assistant Professor
R C Patel Institute of Technology
Shirpur,Dist.Dhule(India)

ABSTRACT
In this paper, we propose an approach of verification of business processes shown in semiformal modeling notation BPMN, by transforming it into petri net a formal language. This paper covers the transformation of BPMN to petri net using model transformation language ATL available in Model Driven Engineering along with introduction of BPMN and petri net.

Keywords
Software System, BPMN, Petri net, Business Process, ATL and Formal semantics

1. INTRODUCTION
The verification and validation of present-day software systems is difficult job because, day by day, it is increasing tremendously in size. Along with the size, to model and analyse real-life applications, an increasing number of features and formalisms need to be developed and supported. Each system is represented in the form of model. There are many modeling notations available such as BPMN (Business Process Modeling Notation) [1], UML (Unified Modeling Language) [2], and BPEL (Business Process Execution Language) [3] etc. Among all only BPMN is an emerging standard for representing business processes and indirectly software systems. BPMN is visual process modeling notation which can be easily understood by business analysts. But BPMN lags behind formal semantics of systems. At the time of verification of particular system such modeling notations are not enough to give semantic correctness of system. Thus for verification and validation of systems, we proposed an approach. According to my approach BPMN model of system is converted into petri net [4] model. Once you are ready with petri net notation, you can apply it to various automated tools present. Thus verification of system is performed.

Therefore before deploying any software system into operation, it is required to verify that the behaviour of the final system is correct and safe means behaves as expected. Therefore to ensure the correctness of behaviour two things are needed, first, formal models should be enough powerful to faithfully describe all aspects of the system and have strong constructs to keep components and connectors belonged with each other's. Second, formal modeling notation, for analysis and verification of formal model's certain properties including correctness, reachability and complexity. Therefore to satisfy above both conditions we prefer petri net as formal modeling language.

2. BPMN
The term BPMN is made up of BPM+N. BPM is business process modeling which is used to understand, document and analyse the processes. N in BPMN is Notation nothing but language. Therefore BPMN is language to visualize, document, communicate, analyse, improve, simulate and execute the different business processes. The primary goal of the BPMN effort was to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. BPMN defines a Business Process Diagram (BPD), which is similar to a flowcharting technique used for designing various graphical models of business process operations. Thus a Business Process Model is a network of graphical elements, which are activities i.e. work and the flow controls that define their order of execution. All the elements present in BPMN are shown in Figure1. There are four basic categories of BPMN elements-

- Flow Objects
- Connecting Objects
- Swim lanes
- Artifacts

Flow objects consist of three core elements, Event, Activity and Gateway. An Event is something that happens in business process. Events are represented by circles with open centres so that you can show various types of events including Start, Intermediate and End events by marking particular symbol of event, e.g. Message, Timer, Error etc events. An Activity is work that going to perform and represented by rounded corner rectangle. Gateways are used to control the sequence flow. It controls divergence and convergence of flow of execution. Gateways are represented by diamond shape and internal portion of diamond shape decides the type of control flow by marking particular symbol of gateway, e.g. Fork, Join, Data based XOR etc. gateways. Connecting objects are used to connect the various flow objects. There are also three basic types - Sequence flow, Message flow and Association. Sequence flow is represented by solid line with solid arrowhead. Message flow is used to show communication between different participants involving in business process system and is represented by dashed line with open arrowhead. Similarly Association is used to associate Data, Text and Artifacts with flow objects and represented by dotted line with line arrowhead.

Swimlanes are used to categorize the activities based on functional capabilities and responsibilities. Two swimlane elements are Pool and Lanes. Pool is rectangle and is nothing but one graphical container which contains activities belonging to same functional capabilities Lane is nothing but either vertical or horizontal partition within the pool. Artifacts are used to represent more contexts about any

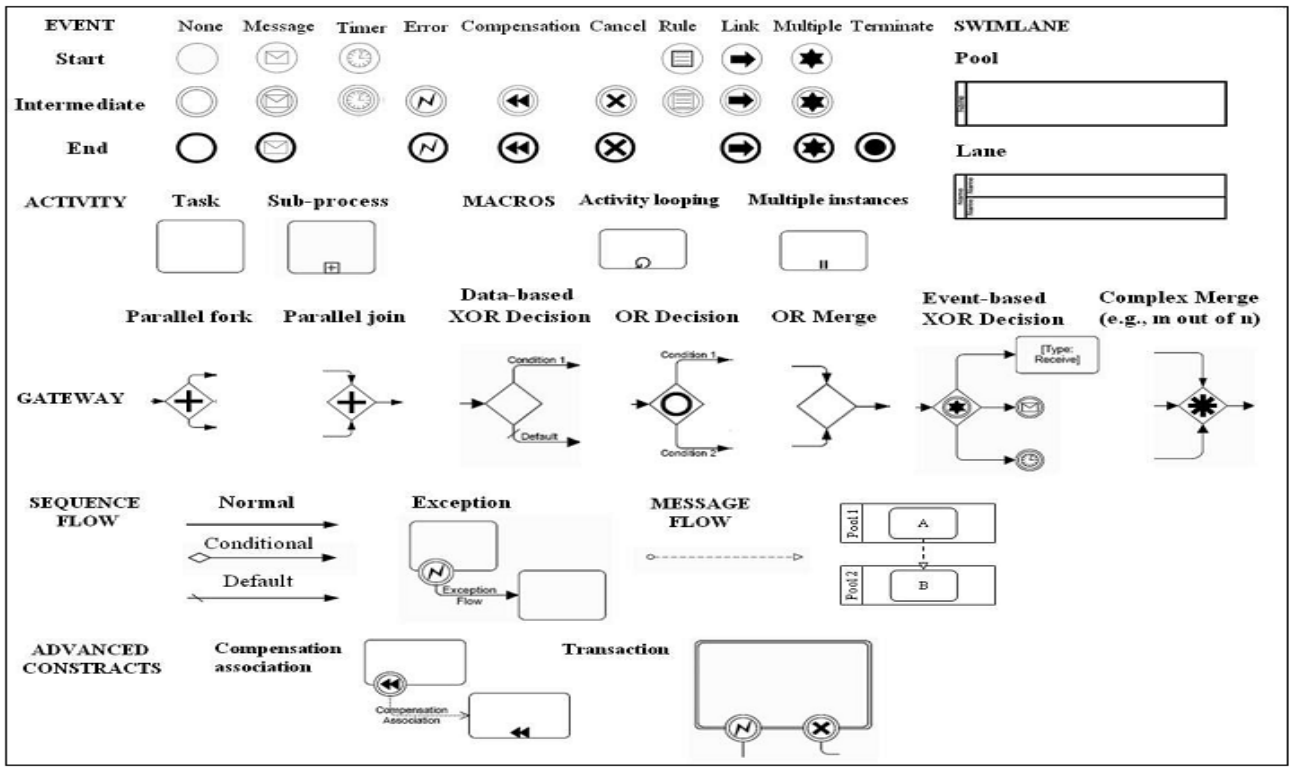

Fig 1: BPMN elements overview

modeling situation. Three types of BPMN artifacts available are Data object, Group and Annotation shown in Figure 1. Data object is used to show how data is required or produced by activities. Group is represented by dashed line rectangle with rounded corners and used for documentation and analysis purpose. Annotation is nothing but additional text about activities in business processes.

Thus along with all above basic elements BPMN can also have more advance constructs such as Compensation association and Transaction. Therefore various versions of BPMN have developed and each version has different number of elements.

3. PETRI NET OVERVIEW

Petri nets are a formal model of concurrent systems. Petri nets are used to model behavior of systems in terms of flow. The flow may be control flow or object flow or information flow. This specialty makes Petri nets a good candidate for formally designing the semantics of BPMN models, since BPMN is also flow-oriented notation. In addition, Petri nets have been studied from a theoretical point of view, and a number of tools that enable their automated analysis are invented. A Petri net is a directed graph and have two types of nodes: places and transitions. Places are represented as circles while transitions are represented as rectangles. Petri nets are bipartite graphs, means an arc may connect a place to a transition or vice-versa a transition to place, but no arc never connect a place to another place or a transition to another transition directly. A transition may have a number of immediately preceding places which are also called as its input places and a number of immediately succeeding places also called as its output places.

Places may contain Tokens which represent the thing(s) that flow through the system. At a given instance during the execution, each place may hold zero, one or multiple tokens.

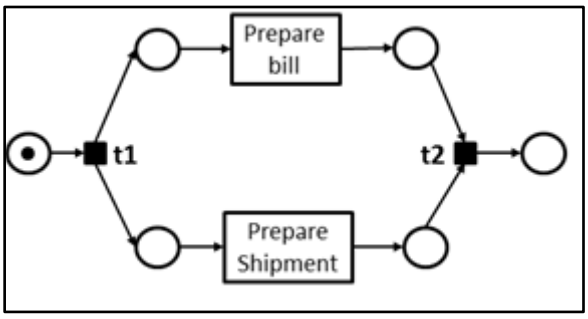

Fig 2(a): Workflow net in an initial marking.

Therefore a state of a Petri net is represented as a function that assigns a number of tokens to each place. Such a function is called a marking. For example, Figure 2(a) shows a marking of a Petri net where only one token is in the leftmost place and no token in any other place. The state of a Petri net changes when one of its transitions fires. A transition may fire only if at least one token should be in each of its input places and said to be enabled. For example, in Figure 2(a), the transition enabled since having only one input place with one token. When a transition fires, it removes one token from each of its input places and it adds that token to each of its output places. For example, Figure 2(b) shows the state obtained when transition t1 fires. The token in the leftmost place has been removed, and a token has been added to each of the output places of transition t1. In a given marking, there may be multiple enabled transitions simultaneously. In such situation, any one of these enabled transitions may fire at a time. For example, in Figure 2(b) there are two transitions enabled: “Prepare Bill" and “Prepare Shipment". Any one of these transitions may fire in the next execution step. Note that when there is name attached to a transition is long (e.g. “Prepare Bill") we place that label inside the rectangle representing this transition. Also, we will sometimes omit the name of a transition altogether. Transitions without names correspond to

"silent steps" which have no effect on the outside world, as opposed to a transition with name.

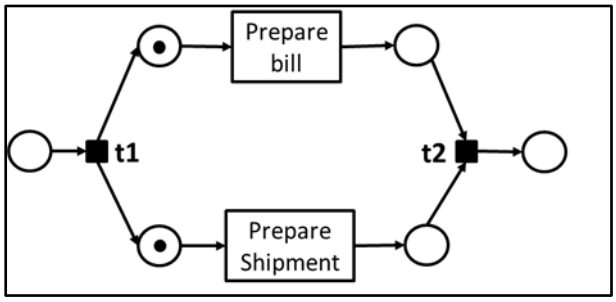

Fig 2 (b): Workflow net after transition t1 fires.

4. BPMN TO PETRI NET MODELTRANSFORMATION

In model-driven engineering, model transformation aims to provide a mean to specify the way to produce target models from a number of source models. For this purpose, it should enable developers to define the way source model elements must be matched and navigated in order to initialize the target model elements.

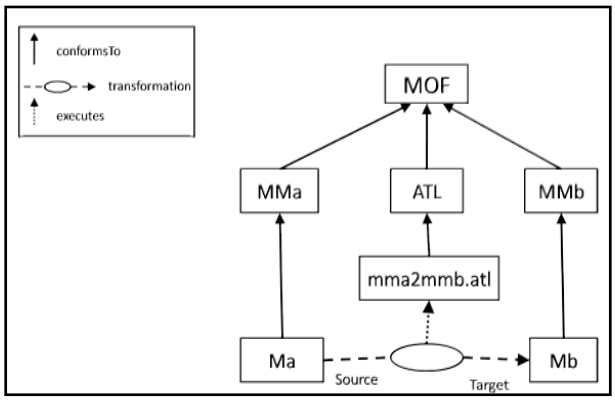

Fig 3(a): Overview of ATL transformation approach

There are many techniques available for such model transformation as a result many transformation languages are proposed such as ATL [5], QVT [6], and YATL [7]. Our approach uses ATL: a hybrid model transformation language that allows both declarative and imperative constructs to be used in transformation definitions

The schema in Figure 3(a) explains full model transformation process. A source model Ma, conforming to a metamodel MMa, is here transformed into a target model Mb that conforms to a metamodel MMb. The transformation rule defined in the MMa2MMb.atl is conforms to ATL. This ATL, along with the MMa and MMb metamodels, has to conform to meta metamodel such as MOF [8] or Ecore[9].

Using similar approach we proposed BPMN to petri net transformation in Figure 3(b) BPMN is source model which defined according to given BPMN metamodel. The transformation rules defined in *BPMN2petrinet.atl* file is conforms to ATL. As a result of successful transformation of BPMN into petri net according to defined ATL rules, we get the petri net model which must conforms to the petri net metamodel defined. The BPMN metamodel example is given in Figure 4 which covers all basic elements of BPMN. The core petri net metamodel example is given in Figure 5. All the

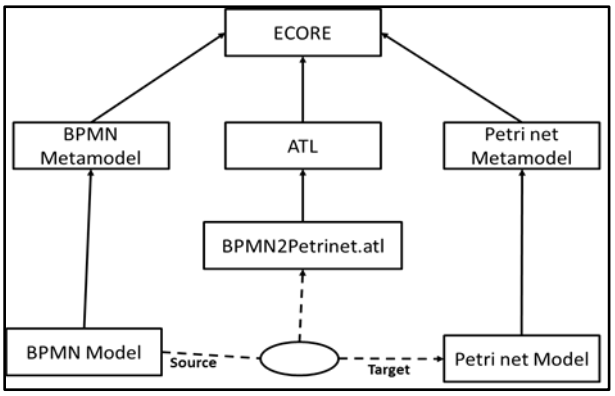

Fig 3(b):BPMN to Petri net using ATL

theoretical aspects about BPMN to petri net mapping which we have considered is described in [4]. BPMN elements are mapped to corresponding petri net elements without changing the semantics.

The transformation rules used for BPMN to petri net conversion are defined into *.atl* file. The ATL rules for BPMN to petri net conversion defined are based on the conceptual mapping done in [4]. One example of such ATL rule is as given below

```
rule Place {
          from
                    g:bpmn2!Event
          to
                    p:PetriNet!Place
          (
          location<-g.location,
          name<-g.name,
          net<-g.process,
          incomingArc<-g.incomingConnections,
          outgoingArc<-g.outgoingConnections
          )
}
```

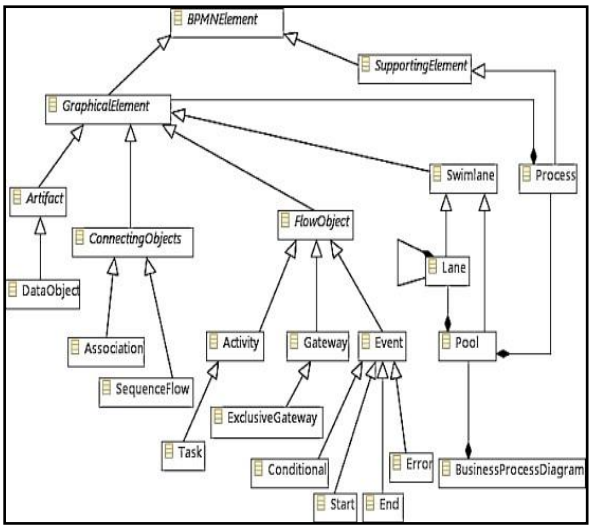

Fig 4:BPMN Metamodel Example

Thus the mapping of BPMN elements to petri net elements according to defined ATL transformation rules is performed. And petri net model formed is conforms to petri net metamodel defined.

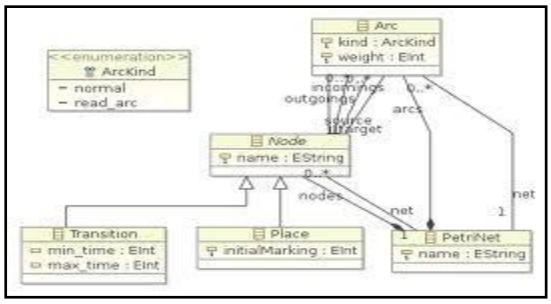

Fig 5: Petri net metamodel example

5. RELATED WORK

Samiraet. al. [10] describes Modeling Web Service Interactions Using the Coordination Language REO i.e. WS-BPEL to REO conversion. They propose an approach to derive the formal semantics of WS-BPEL processes compositionally using REO and constraint automata. They map each WS-BPEL process into a REO circuit and then construct the corresponding constraint automaton which shows the behaviour of the process. The constraint automaton can be used for analyzing the process behaviour. Their work covers the core part of the WS-BPEL language including basic and structured activities, correlation sets, variables, and links.

Behnazet. al.[11] developed convertor for BPMN, BPEL, and UML Sequence Diagrams into REO. In this they have given ECT plug-in for eclipse platform. All the transformation is done automatically. BPMN to REO conversion is done by using ATL rules. After such a transformation, refined and annotated REO process models can be visualized, verified and transformed into executable code.

Tscheschneret. al.[12] gives a direct approach of a transformation from EPC process model elements to BPMN. They tried to map every construct in EPC fully automated to BPMN.

LopezGrao et.al.[13] gives application software performance engineering by transforming UML activity diagrams to stochastic petri nets.

6. CONCLUSION

In this way we have described an approach to verify the business processes described in BPMN by transforming it into the petri net formal language. As we shown petri net has simple formal modeling language and has mathematical base.

7. REFERENCES

[1] Stephen A. white, "Introduction to BPMN", IBM Corporation.

[2] Thomas,R."Introductionto Unified Modeling Language", Technology of Object-Oriented Languages and Systems, 1997.TOOLS 25, Proceedings, p.354.

[3] Khalaf, R. "Business processes for Web Services: Principles and applications", IBM Systems Journal (Volume:45 , Issue: 2),2006, pp.425- 446

[4] Dijkman, R. "Semantics and Analysis of Business Process Models in BPMN", Information and software technology, 50(12), pp.1281-1294.

[5] Frederic Jouault, Freddy Allilaire, Jean Bezivin, Ivan Kurtev, and Patrick Valduriez."Atl:a qvt-like transformation language". In Peri L. Tarr and William R. Cook, editors, OOPSLA Companion, ACM, 2006 pp 719-720.

[6] The Object Management Group. Meta object facility (mof) 2.0 query/view/transformation. Specification Version 1.0, Object Management Group, April 2008.

[7] Octavian Patrascoiu. "Yatl: Yet another transformation language". InUniversity of Twente, the Nederlands, pp 83-90, 2004.

[8] OMG: Meta Object Facility (MOF) 2.0CoreSpecification, OMG Document formal 2006-01-01.(2006).

[9] Denivaldo Lopes, SlimaneHammoudi, Jean Bezivin , and Frederic Jouault, "Mapping Specification in MDA: From Theory to Practice".

[10] Samira.Tasharo,ZilouchianMoghaddam, R., M. Sirjani, and M. Vakilian,, "Modeling Web Service Interactions using the Coordination Language REO", in Proceedings of the 4th International Workshop on Web Services and Formal Methods, LNCS 4937(2007), pp. 108-123.

[11] B. Changizi, N. Kokash, and F. Arbab,"A unified toolset for business process model formalization," inProc. FESCA'10, 2010.

[12] WilliTscheschner, "Transformation from EPC to BPMN".

[13] Juan Pablo LopezGrao, JoseMerseguer, Javier Campos, "From UML Activity Diagrams to Stochastic Petri Nets:Application to Software Performance Engineering", WOSP 04 January 14-16, 2004.