Composability Test of BOM based models using Petri Nets

Imran Mahmood[1], Rassul Ayani[1], Vladimir Vlassov[1], and Farshad Moradi[2]

[1]Royal Institute of Technology (KTH), Stockholm, Sweden
[2]Swedish Defense Research Agency (FOI), Stockholm, Sweden

Abstract. Reusability is a widely used concept which has recently received renewed attention to meet the challenge of reducing cost and time of simulation development. An approach to achieve effective reusability is through composition of predefined components which is promising but a daunting challenge in the research community. Base Object Model (BOM) is a component-based standard designed to support reusability and composability in distributed simulation community. BOM provides good model representation for component reuse however this framework lacks capability to express semantic and behavioral matching at the conceptual level. Furthermore there is a need for a technique to test the correctness of BOM-compositions in terms of structure and behavior. In this paper we discuss verification of BOM based model and test its suitability for the intended purpose and objectives. We suggest a technique through which the composed model can automatically be transformed into a single Petri Net (PN) model and thus can further be verified using different existing PN analysis tools. We further motivate our approach by suggesting a deadlock detection technique as an example, and provide a case study to clarify our approach.

Keywords: Verification, Model Based testing, Composability, Petri Nets

1 Introduction

Software reuse is the process of creating dynamic systems from existing components instead of building them from scratch. Reusability has recently gained renewed interest as an effort to minimize the cost and time associated with the development process. Composability is one of the effective means to achieve reusability. The concept of Composability was pioneered by Mike Petty in his theory of composability in Modeling and Simulation (M&S) community[6], according to which, "Composability is the capability to select and assemble simulation components in various combinations into simulation systems to satisfy specific user requirements". Composability is divided into syntactic and semantic levels. Syntactic composability means that the components can be connected to each other. Semantic composability refers to the fact that the coupling of components is considered meaningful, computationally valid and conforms to the intended objectives of the simulation. Semantic composability is broken down into two

sub-levels namely Static Semantic, which means that the components have same understanding of the concepts and the relations between them, and Dynamic Semantic that deals with the behavioral correctness of the composition[3]. There are some other levels of composability such as Pragmatic composability which we do not consider in this paper.

BOM (Base Object Model) represents an integrated framework that posses the ability to rapidly compose simulation components. It provides a foundation to define and characterize these components at a conceptual level. BOM encapsulates information needed to formally represent a simulation component. BOM is a SISO standard and encapsulates information needed to describe a simulation component. BOM was introduced as a conceptual modeling framework for HLA (High Level Architecture) which is an IEEE standard for distributed simulation. State-machine, which is an essential part of BOM provides means to formalize the change in the state of an entity with respect to its corresponding actions, thus in a way it depicts the abstract model of the behavior of the BOM towards each action[2]. However external techniques are needed for the composability verification of BOM based components.

PN is an effective graphical tool and mathematical formalism for modeling concurrent systems and their behaviors. PN has existed for many decades and has been used in modeling, analysis and verification of a large variety of systems [5]. A PN is an algebraic structure of 3-tuple: PN = (P, T,φ) where:

- P is a finite set of places P = $\{p_1, p_2, \ldots, p_n\}$
- T is a finite set of transitions T = $\{t_1, t_2, \ldots, t_n\}$
- φ is a set of arcs $\varphi \subseteq (P \times T) \cup (T \times P) \mid P \cap T = \phi$ and $P \cup T \neq \phi$

A major strength of PN is its support for analysis of many properties and problems associated with concurrent systems. Some of the important PN properties are briefly defined as follows:

Reachability A marking μ is said to be reachable from a marking μ_0 if there exists a sequence of firings that transforms μ_0 to μ. The set of all possible markings reachable from μ_0 in a net P(N, μ_0) is denoted by R(N, μ_0). The reachability property determines whether $\mu_x \in R(N, \mu_0)$. Reachability is a fundamental basis for studying the dynamic behavior of a system [4].

Liveness A PN P(N, μ_0) is said to be live if, no matter what marking has been reached from μ_0 it is still possible to make further progress by firing an enabled transition of the net. A live PN guarantees deadlock-free operation, no matter what firing sequence is chosen [4]

In this paper we employ the well-suited analytical strength of PN tools to test various system properties such as deadlock freedom. We propose a verification process and a framework to allow the assessment of a composed model by transforming it into a PN model and apply suitable analytical methods which are commonly being practiced by the PN community. As compared to our previous approach[3] in which we used Finite State Automata, using PN for verification proves to be more fruitful due to the broader range of verification tools and techniques that are available. The remainder of this paper is organized as follows: Section 2 contains the major contribution of this work covering the proposed

methodology for the verification. A case study is presented in section 3 to support our concept whereas section 4 concludes this paper.

2 Test and Verification of a composed model

In this section we discuss our proposed verification process and the framework in which this process has been implemented. Our approach for the verification of BOM based composed models is essentially based on the conversion of a composed model to PN model and applying different analytical techniques to verify the required system properties. We use the Platform Independent PN Editor (PIPE) as an underlying layer for PN analysis [1]. PIPE is an open source Java based PN graphical editor and analysis tools library. It not only provides a graphical preview and editing facility for PN models, but also supports visual simulation (commonly known as Token game animation) and analysis techniques. Our verification framework is integrated with PIPE environment to utilize these facilities. Following are the two main steps in our proposed verification process:

2.1 Transformation of BOM to Petri

In this step, BOM is transformed into PN model. In the preparatory phase a composed BOM model is parsed using our BOM parser and the objects of state-machines, and corresponding events of all the members of composition are collected. Then the composition is analyzed to check completeness of model in terms of structure of state-machines and the presence of associated events for each state to be able to exit. In case of no such association it is verified whether the particular state is a final state or not. The event pairs are also matched among the member components so that for each receiver component a corresponding sender of an event is present. This procedure is applied to ensure that the composition is structurally correct and suitable to proceed with. For more details see [3].

Finally the BOM model is automatically transformed into a PN model. A PN model is ideally represented using PNML (PN Markup Language) which is an XML-based standard interchange format for PN. We propose the basic transformation algorithm as given in figure 2a. The input for this procedure is a BOM object C of the composed model which is generated by the parser. In order to meet the requirements of PNML, we also assume default values such that tokens at each place are set to zero and arc weights are set to 1. As we are not considering Timed PN in this work so we also set the timed Boolean variable to false. These settings can be modified later for experimentation at the time of execution & analysis.

In this algorithm for each state-machine in the composed model (line 2) every state is traversed one by one (line 3) and its corresponding place is created[1] in

[1] Create functions are used to write XML code to a file according to the PNML notation.

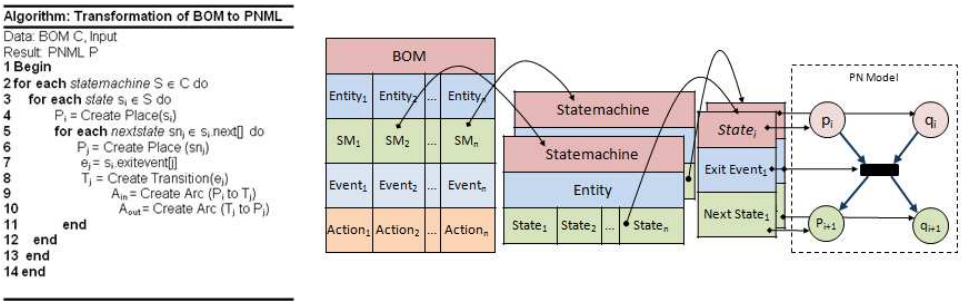

Fig. 1. a) Our Algorithm b) Transformation from BOM to PN

PNML format (line 4). Since a state may have multiple next states and their associated exit events so for each next state (line 5), a place is created (line 6) provided it does not already exist. Then a transition is created for the exit event associated with the next states (line 8). After that an input arc is created from the place to transition (line 9) and an output arc is created from the transition to the next place (line 10). Duplication is avoided in the entire process. This procedure can be graphically viewed in figure 2b, (assuming that the two state-machines have a common exit event) Also the graphical placing (X and Y positions) for each place, transition and arc is handled automatically in such a way that all the places of each member component are aligned in a vertical lane and their transitions are created between the lanes to show the interaction among different components in order to facilitate visibility. When the entire BOM model is traversed, a PNML file is generated that corresponds to the PNML standard and can be used in any PN simulation tool for execution.

2.2 Petri Net Analysis

In the analysis step, we perform Reachability analysis on the PNML model generated in the previous step with user given input parameters. To initiate this procedure we use PIPE library to construct a reachability tree and populate it with our PNML model generated in the previous step. We perform this operation programmatically to facilitate automation. After the initialization of the root marking, the tree recursively expands by constructing further markings until all the markings of the model have been created. If no more marking can be created, i.e., no further transition is enabled it reaches a dead marking. By detecting a dead marking in the tree, we assert that a deadlock is detected provided that it is not a final marking. We propose a minor change in the original definition of deadlock in a Dead (or L0-live) PN as some models may have terminal points e.g., when a Final State in a finite State-machine is reached. So we define a "Final Marking" as a successful termination point which is not considered as a dead marking even though no further transitions could be fired. Thus we propose a deadlock detection method in which we find at least one dead marking from

where no further transitions could be fired, provided it is not a final marking. We input the final marking as a parameter for reachability analysis.

3 Case Study

In this section we discuss a restaurant case study to test our verification approach. This is a simple composed model of two BOM components namely Customer and Waiter. In step 1, Restaurant BOM is parsed and the corresponding objects are generated and the model is transformed into PNML format using our proposed algorithm. Figure 2a represents the sequence diagram of the restaurant. Statemachines of customer and Waiter are presented in Figure 2b where as the generated PN is illustrated in 2c.

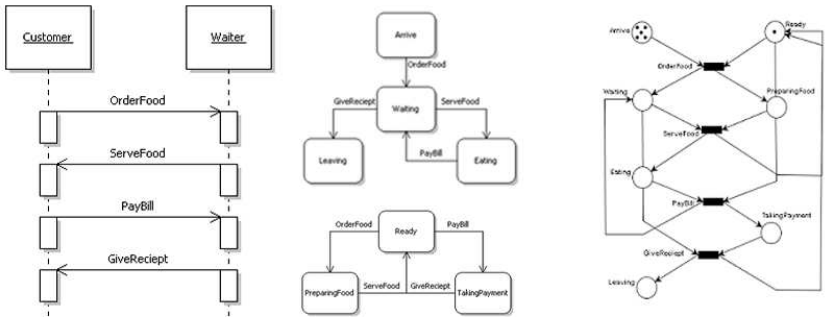

Fig. 2. a) Sequence Diagram b) Statemachine c)PN

When PN model is executed in PIPE simulator, a token is dispatched from the Arrive place representing the arrival of a customer who orders food. But it requires an available Waiter from Ready place to proceed. After Order Food transition is fired, one token will be produced to Waiting and the other will be produced to Preparing Food, place. In the same way, all the interactions between customer and waiter will continue, until the customers have reached Leaving whereas the waiter goes back to Ready. In step 2, the reachability analysis module is initiated and the final marking parameter [0,0,0,5,1,0,0] (i.e., all five customers leave the restaurant and the waiter is back to ready) is given as input to distinguish it from the dead markings. When the module completed the execution, it verified that the model is deadlock free as only one dead marking was detected in the tree which was exempted by the final marking parameter. The Reachability Tree can be viewed at: `http://web.it.kth.se/~imahmood/SimpleRestaurant.html`. In the tree each node (in blue) represents a marking and is denoted by a number and transitions are the arrows connecting next marking. From the tree it can be seen that node 50 (in green) is the only dead marking from where no further transition is possible. Thus we verify that although the model terminates it does not have a deadlock.

In a different experiment we introduced another component Kitchen in the composed model. Same procedure was applied however it was noted that few dead markings were detected because, there are some situations in which kitchen is in the cooking place and can be released only when waiter takes food to serve, whereas waiter is waiting for the kitchen to take more orders. This results in a potential deadlock which may or may not occur depending on the sequence of transitions fired. This potential deadlock is detected by our framework. The figures of the PN model and reachability tree of this experiment cannot be shown due to space limitations and can be viewed at: `http://web.it.kth.se/~imahmood/Restaurant.html` . We also encourage interested readers to view more case studies at our web site: `http://web.it.kth.se/~imahmood`

4 Summary and Conclusion

In this paper we discuss composability test of BOM based compositions using PN. We suggest an algorithm to transform a composed BOM model into a standard PN format known as PNML which can further be used with any PN analysis tool that conforms to this standard. We however suggest using PIPE tool in our framework and provide a technique to detect deadlocks in the composed model. Finally we explain the entire process using a Restaurant case study.

We are further interested to consider alternative techniques in the PN to extend the capability of our verification process, such as Timed PN and Colored PN to test and verify more realistic system models. We also are inclined to apply different state space reduction techniques to our framework in order to solve the state explosion problem and optimize the verification process.

References

1. Chung, E., Kimber, T., Kirby, B., Master, T., Worthington, M.: Platform independent petri net editor. Tech. rep., Imperial College, London, Project Report (2007)
2. Gustavson, P.: Guide for base object model (bom) use and implementation. Tech. Rep. SISO-STD-003-2006, Simulation Interoperability Standard Organizations (SISO), Orlando, FL USA (2006)
3. Mahmood, I., Ayani, R., Vlassov, V., Moradi, F.: Statemachine matching in bom based model composition. In: Proc. 13th IEEE/ACM Int. Symp. Distributed Simulation and Real Time Applications DS-RT '09. pp. 136–143 (2009)
4. Murata, T.: Petri nets: Properties, analysis and applications 77(4), 541–580 (1989)
5. Peterson, J.L.: Petri nets. Computing Surveys vol. Vol 9, no. No. 3 (September 1977)
6. Petty, M.D., Weisel, E.W.: A theory ofsimulation composability. Tech. rep., Virginia Modeling Analysis & Simulation Center, Old Dominion University, Norfolk, Virginia (2004)