Composability Verification of Real Time System Models using Colored Petri Nets

Imran Mahmood, Rassul Ayani, Vladimir Vlassov
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden
e-mail: imahmood@kth.se, ayani@kth.se,
vladv@kth.se

Farshad Moradi
Swedish Defense Research Agency (FOI)
Stockholm, Sweden
email: farshad.moradi@foi.se

Abstract — The discipline of component based modeling and simulation offers promising gains including reduction in development cost, time, and system complexity. It also promotes (re)use of modular components to build complex simulations. Many important issues in this area have been addressed, but composability verification is still considered a daunting challenge. In our observation most of the component based modeling frameworks possess weak built-in support for the composability verification, which is required to guarantee the correctness of the structural, behavioral and temporal aspects of the composition. In this paper we stage a practical approach to alleviate some of the challenges in composability verification and propose a process to verify composability of real-time system models. We emphasize on dynamic semantic level and present our approach using Colored Petri Nets and State Space analysis. We also present a Field Artillery model as an example of real-time system and explain how our approach verifies model composability.

Keywords- Composability; Verification; Real-time systems; discrete event; Colored petri nets; Field Artillery

I. INTRODUCTION

The Modeling and Simulation (M&S) community has been conducting research on methods and technologies to construct complex simulation systems by combining new or reusing existing simulation components. This paradigm of component-based modeling and simulation has gained growing impetus due to its promising gains including reduction in development cost, time, and the system complexity. It follows the principle of modularity which essentially helps to master the complexity of reality by decomposing it into parts [1] and by enabling the designer to (re)use appropriate parts for different purposes.

Composability is the capability to select and assemble components in various combinations (meaningfully) to satisfy specific user requirements [2]. It is an important quality characteristic of the M&S discipline, yet difficult to achieve [3], [4]. This is mainly due to the underlying intricacies of the individual components and substantive subtleties of the combined effect. Composability is a property of the models, as it essentially contends with the alignment of issues on the modeling level [5], where it is viewed as creation of complex models from a collection of modular components, which might themselves be the abstraction of subsystems. Composability essentially relies on a suitable component modeling framework that must provide accurate reasoning for the correctness and the ability to leverage certain component standard. One such standard that support composability is the Base Object Model (BOM) [6], which is a SISO (Simulation Interoperability Standards Organization) standard. Composability is further divided into different sublevels, as discussed in [5], [7] and [8].

In this paper, our focus is centered on the correctness of composability at the Dynamic Semantic level, which is a necessary condition for the credibility of overall composability. Dynamic Semantic Composability implies that the components are dynamically consistent, i.e., they have correct behavior, necessary to reach the desired goals and subsequently satisfy user requirements. In essence, a set of components can possibly fit together (syntactically), and their communication is meaningful and understood (semantically), but unless all components preserve essential behavior (dynamically), in order to reach the desired composition goals, the correctness of the composed model cannot be certified. We further elaborate that correctness of behavioral composability relies on three factors: firstly each component is at the right state while interacting with the others, and secondly the composition should satisfy required behavioral properties, as prescribed in the requirement specifications and thirdly the must fulfill the required time constraints (in case of real-time model components).

In M&S, verification is concerned with building the model right. It is typically defined as a process of determining whether the model has been implemented correctly [9] and whether it is consistent with its specifications [10]. In principle, verification is concerned with the accuracy of transforming the model's requirements into a conceptual model and the conceptual model into an executable model [10]. We postulate that model's requirements are identified by means of requirements specification which includes a set of verification goals, listed in terms of desired system behavior properties such as deadlock freedom, live-lock freedom, mutual exclusion and fairness. Systems where the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical time when these results are produced and within given time bounds, are known as Real-Time systems [11]. When models of such systems are composed, they may also require having certain time properties or constraints that should be satisfied for correct composability.

Various approaches have been suggested concerning the model verification of real-time systems. A formal model of hierarchical time system is presented in [12] and adjoined

978-0-7695-4994-1/13 $26.00 © 2013 IEEE
DOI 10.1109/UKSim.2013.49

with an 'Uppaal-tool' for formal verification. Another interesting approach uses DEVS for real-time system development, and transforms it to Timed Automata for verification using "Uppaal-tool" [13]. A rather different approach focuses on the formal validation of semantic composability of time based systems [14].

In this paper, we present composability verification of real time system models, where time constraints are defined in the requirement specification and their behavior is evaluated to guarantee response within the required time constraints. In our approach, we suggest to use BOM as a conceptual modeling standard, and Colored Petri Nets (CPN) as an executable modeling framework. We propose a formal Timed-CPN based component model, which is used for implementation and execution of BOM components (Note that BOM does not support time modalities, so we also propose a BOM extension to model Time behavior). After implementing BOM components (using our automatic transformation method) the generated CPN component models are composed and subjected to verification for the composability evaluation at dynamic semantic level. A verified composition of CPN component models asserts that its corresponding BOM conceptual model is correct, with respect to the given requirement specifications. We also present a Field Artillery model as an example to illustrate how our approach verifies a composed model.

The rest of the paper is organized as follows. Section 2 covers basic definitions and concepts used in this paper. Section 3 furnishes the details of our composability verification process. In Section 4 we discuss a case study of a Field Artillery model to explain our approach, whereas section 5 frames summary and conclusion.

II. Definations And Basic Concepts

In this section we briefly discuss some essential concepts that are used later in this paper.

A. Base Object Model

The SISO standard BOM is defined as, "a piece part of a conceptual model, comprised of a group of interrelated elements, which can be used as a building block in the development and extension of simulations and simulation environments" [6]. BOM includes the aspects of conceptual modeling that captures static descriptions of elements abstracted from the real system (simuland), described in terms of conceptual entities and events, and contains information on how these elements interact with each other in terms of Patterns of Interplay and state-machines. In this paper, we harness the capability of BOM as a conceptual modeling framework, because it provides a component standard as a basis of model specification; helps determine the appropriateness of the model or its parts for model reuse; and most importantly, it strongly supports composability.

B. Colored Petri Nets

In this paper, we incorporate Colored Petri Nets formalism (developed at the University of Aarhus), as an executable modeling framework, focusing on its Time extension in particular and propose to utilize its strength by implementing BOM based conceptual model into a Timed-CPN based executable model. CPN is a general purpose discrete event graphical language for constructing models of concurrent systems and analyzing their properties. TCPN is an extension to CPN, in which tokens can carry timestamps in addition to the token color, which implies that the marking of a place where the tokens carry timestamps become timed multi-sets. Also, the model has a global clock, representing model time. The distribution of tokens on the places, together with their timestamps and the value of the global clock, is called a timed marking. For detailed explanation of the concepts of Timed CPN, interested readers are recommended to consult [15], [16]. 'CPN Tools' is a modeling and execution environment based on CPN language, and is used for the editing, simulation, state space analysis, and performance analysis of CPN models. The most important features of CPN tool from our point of view are hierarchal CPN modeling and the generation and analysis of state spaces. Hierarchal CPN modeling offers modular development. CPN tools offer facility to construct hierarchal CPN models, by replacing an entire CPN model with a substitute transition that can be connected to a main model. In this paper, we utilize this feature, and propose a CPN based hierarchal "Component Model", for the automated implementation and execution of a BOM conceptual model (To understand our proposed component model the knowledge of CPN constructs and functionality is required [16], [17]).

C. State Space Analysis

State space analysis is one of the most prominent approach for conducting formal analysis and verification. The basic idea in this approach is to calculate all possible system states and represent them as vertices in a directed graph and represent the transitions of one state to another state by directed edges. In theory a state space is a directed graph where we have a node for each reachable marking (system state) and an arc for each transition. A constructed state space can answer a large set of analysis and verification questions concerning the behavior of the system such as absence of deadlocks, the possibility of being able to reach good state(s), and never reach bad state(s) and the guarantee of reaching goal state(s). A step by step tour of state space analysis using CPN tools can be located at [18]. The main advantages of state space methods is that they can provide counter examples and reasoning as to why an expected property does not hold [19]. The main disadvantage of using state spaces is the state explosion problem. Even relatively small systems may have an astronomical or even infinite number of reachable states. This problem escalates severely, when the models include Time. A lot of effort has been invested in the development of reduction methods to alleviate this problem. Reduction methods avoid representing the entire state space of the system or represent the state space in a compact form [19]. The detail discussion of these methods is out of scope of this paper, however we rely on these methods, to alleviate the state explosion problem, in case the model under consideration becomes large and resource intensive.

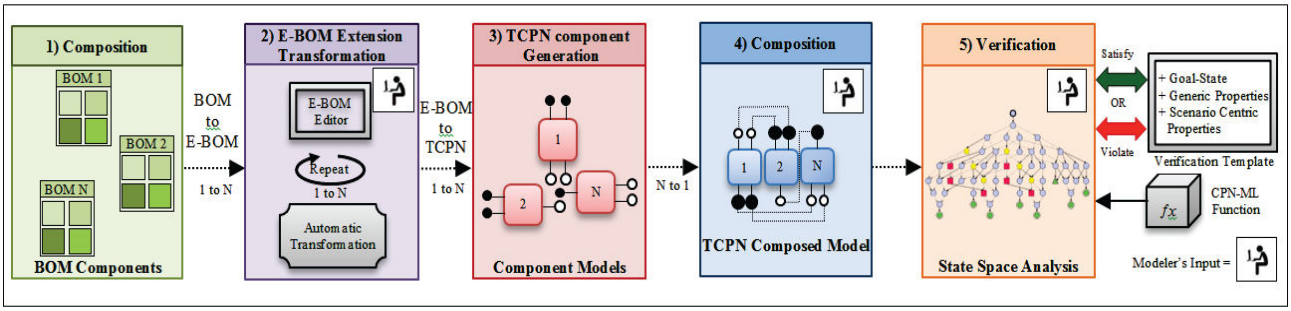


Figure 1. Composability Verification Process

III. Composability Verification Process

In this section we revisit our previously proposed approach [20] of dynamic semantic composability verification and extend it with additional features, particularly to support the verification of real-time system models. Before we discuss these additional features, we summarize our previous contributions as follows.

We proposed a process for the verification of BOM based composed models at the dynamic semantic level. Fig. 1 illustrates the entire process. We suggested to extend the BOM components into Extended BOMs (E-BOM) using our E-BOM editor, to include state-variables and more detailed transitions, with events, guards and actions. Once a standard BOM component is extended into E-BOM, which is transformed to our CPN based component model. The details of our proposed component model can be found in [20]. Fig. 2 represents a generic example of our proposed CPN based component model with three layers namely: (i) Structural Layer (ii) Behavioral Layer and (iii) Communication Lauer. Each layer is initialized with required initial markings that fulfill the condition for the enabling of the transition T0 hence the component can make progress by firing T0.

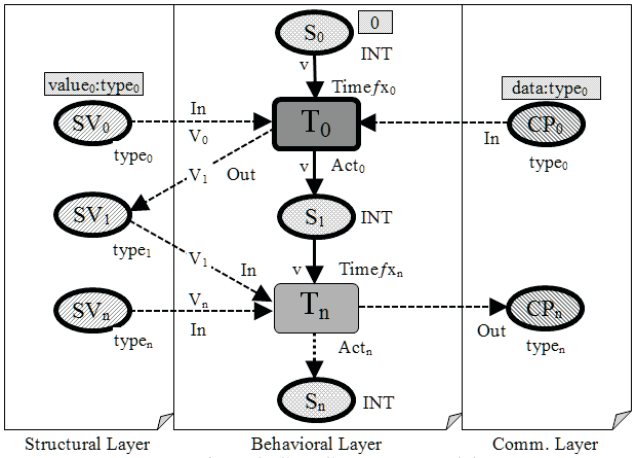


Figure 2. CPN Component Model

We proposed an automatic transformation method to convert E-BOM into CPN model. When all components are transformed, modeler can assemble them as a composed model using CPN hierarchy tools. The resultant model is executable in CPN environment and can be analyzed using state space analysis. For the purpose of verification, we proposed to use a Verification Template that consists of a set of properties representing Goal states, Generic system behavioral properties and scenario centric properties as requirements specification. The composed model is said to be verified at dynamic semantic level if it satisfies all the properties in the verification template.

In this paper, our main contribution (differentiated from the previous paper [20]) is summarized as follows. We upgrade our previously proposed verification process by extending its different parts. Since standard BOM doesn't support Time, so we present additional features in E-BOM to let the modeler specify time functions, for capturing the time specific system behavior. We upgrade our automatic transformation tool; to generate Time based CPN models. We extend our Behavioral Layer and allow Time inscriptions in the transitions, which are either constant non-negative numerical values or random numbers based on assigned probability distributions. (Time inscriptions include delay or interval functions, discussed later in this section). We develop a library of state space query functions for the verification of our model specific properties mentioned in verification template. Also we include time properties as specification in our verification template. (We provide examples in our case study section). Beside time related extensions, we provide modifications at different parts to improve the overall verification process.

A. E-BOM Extension

We propose the time functions as extensions in the BOM standard (in addition to the ones previously proposed [20]). Time modalities do not exist originally in standard BOM. But when the modeling of a real-time system is under consideration, where time plays a key role, we need to provide time functions. We define three types of time functions, which can be assigned to a transition as shown in Table 1.

Wait means the time taken by the transition to occur. It has constant non-zero integer parameter 'duration'. CPN tools provide this feature of assigning time delay to a transition. Deadline is similar to wait. The difference is that in deadline the component remains active and can make progress until a given deadline. Time out means the time between the enabling of a transition and a certain specified future time, during which it will remain enabled. If other components are interacting with a component having a time out transition, they can only progress if they communicate just within this interval.

Time Function	Usage and Explanation
Wait[*duration*]	Wait is assigned to model the delay in an activity. An enabled transition waits for the given duration before it is fired.
Deadline[*duration*]	A transition is constrained to fire when the deadline is reached. The difference between **Wait[]** and **Deadline[]** function is that the former makes the system inactive, i.e., it cannot do anything but wait, whereas when the latter is used the system is active and can respond to events etc., until its deadline is reached.
Time Out [*duration, next*]	When a timeout function is assigned to a transition, it waits for an event to occur. If the event occurs before timeout, it transits to the next state described in the transition definition; otherwise it transits to the next state described in the timeout parameters.

Table 1. Time Functions

(We provide example in our case study section). This feature is not available in CPN. Therefore for its implementation, we propose to attach a "timer" to the transition for evaluating the lapse of the specified interval as shown in Fig. 3.

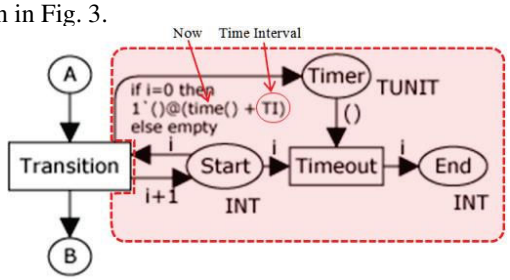


Figure 3. Transition with Time out

The shaded area represents our implementation of a timer, in CPN. Whenever the place A receives token(s), it enables the transition, which when fired sets the timer to run, starting from the current model time. This transition will remain enabled and can be fired multiple times, until TI is reached.

B. Transformation

When all BOMs are extended to E-BOMs with the proposed additional elements, our automatic transformation tool, transforms them into corresponding TCPN-component models (producing TCPN code for the three layers). The output is a .CPN file, with all the components generated as sub-modules. The modeler then composes all the components into a TCPN-Composed Model using CPN hierarchical tool. Then the model can be executed using CPN simulator and analyzed by performing state-space analysis.

C. State Space Analysis

In order to generate state-space of the entire model we use CPN state space calculation tool. When the state-space is generated, different query functions can be used to probe the state space graph for various verification questions. We proposed functions to perform our model specific queries. In order to verify a composed CPN model, we propose a verification template that consists of the verification questions in form of following three groups of properties.

1) General System Properties

State-Space analysis technique is very useful technique to verify general system properties such as freedom of deadlock, live-lock, starvation, or existence of boundedness, mutual exclusion, fairness, sequentiality, time-synchronization etc. Choice of these properties as verification criteria depends on the modeling objectives and their fulfillment become necessary conditions for the correctness of the composition. The solution for verifying a generic property involves specification of the property in CPN terms, and definition of a query function (or algorithm), to reason its satisfiability or violation e.g., freedom of deadlock property is specified in CPN terms as: "An absence of a marking with no-out going arcs, in the entire state-space graph". A verification function: **`ListDeadMarking()`** is used to evaluate this assertion which returns a set of all those markings (if any) having no outgoing arcs. If the result of this query is an empty list, then we assert that the model is deadlock free.

2) Goal Reachability

We propose to define the desired outcome of the composed model in form of a "Goal state", and use a *"Goal reachability"* function to assess if it is reachable in the state space or not. The goal state can be viewed as a CPN based translation of the requirements specification. A typical goal state could be certain desirable values of state-variables in structural layer, reaching of particular state(s) in behavioral layer or producing some required data at output port(s) of the communication layer (or a combination of all the three), in one or more components of the composition. A composed model may have multiple goals.

3) Scenario Centric Properties

We propose to define some safety (or unsafe) assumptions, which are particular to the scenario. They represent certain desirable (or un-desirable) situations which must (or must not) occur in order to satisfy (or violate) the requirements. These properties are not the ultimate goal(s), but they may become necessary conditions in order to reach the goals.

4) State space Query Functions

We develop a library of custom functions, using CPN-ML to perform verification of the properties, specified in the verification template. Some of these functions are explained as follows:

`GoalState()`: Finding goal state reachability is not a standard operation, and depends on the way Goal-State is defined. Most commonly, we make use of our library functions to define a "predicate", that serves as a goal state reachability condition, and then use SearchNode() function to find those nodes, which satisfy the predicate. If one or more nodes are found, then it is verified that the goal is reachable. In cases, where it is important to know "how a goal is reachable", and which sequence(s) of the occurrence of transitions, lead to the goal(s), we use SearchArc()function with the predicate. Similarly, **`minTime()/maxTime()/Interval()`** are used for finding nodes, having timed multi-sets with timestamps greater or lesser then a certain value, or between a certain time interval. These functions work on Timed CPN models.

When, a composed model satisfies all the required system properties, qualifies its goal state reachability, and fulfills the scenario centric safety criteria, we say that it is verified at dynamic semantic composability level.

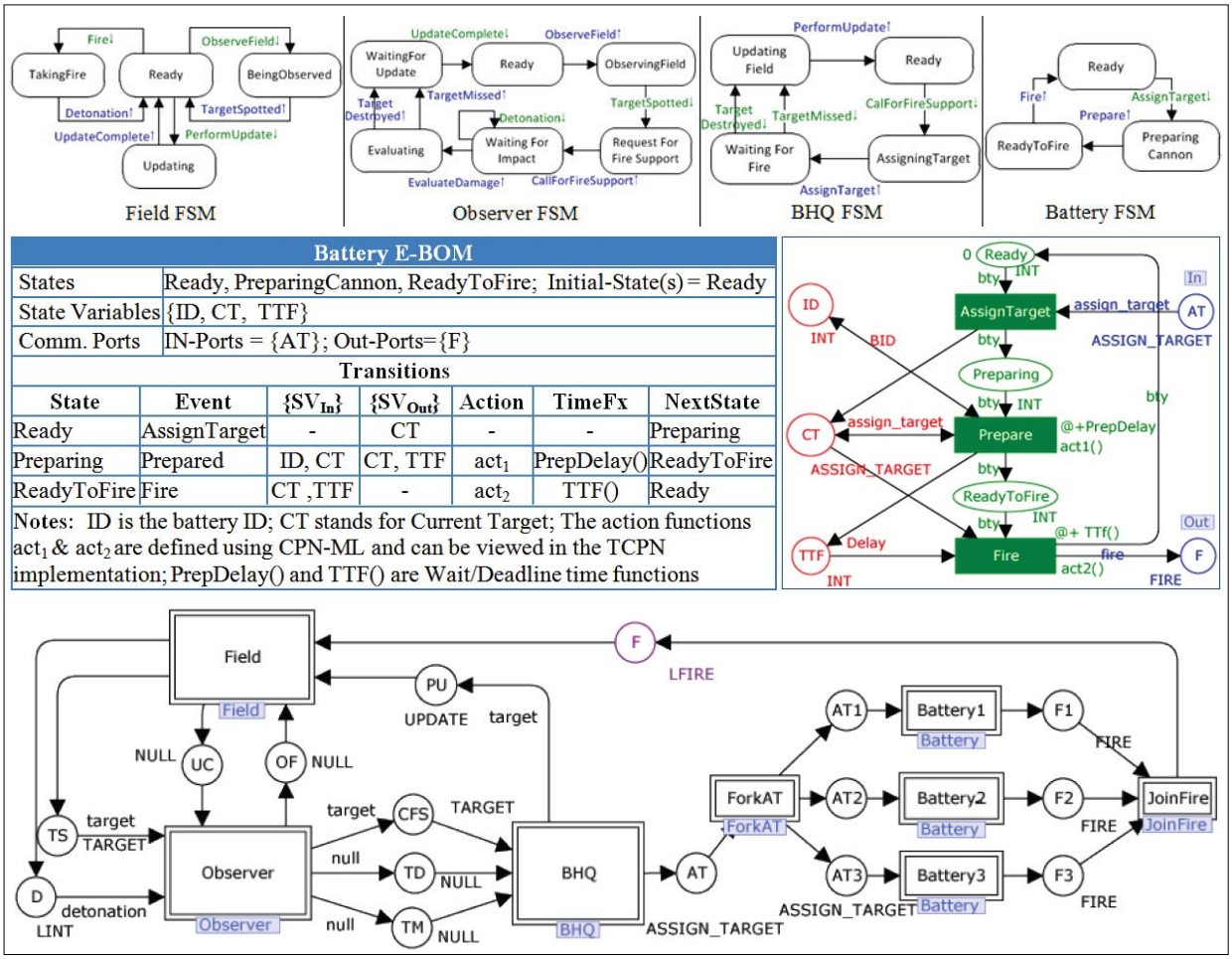
IV. Case Study: Field Artillery

In this section we present a case study of a Field Artillery model, to explain our verification process. This case study consist a scenario, based on indirect fire, where the target is out of sight, and artillery unit is requested for fire support by the forward observer. Following components are composed in this scenario:

- **Field component:** Where enemy and friendly units are deployed.
- **Observer:** A soldier who observes enemy units at the forward location and coordinates fire support.
- **BHQ:** BHQ, supervises the entire operation of fire support at the battalion level.
- **Battery:** Three units of artillery batteries (cannons and crew) actually responsible to hit the target.

We assume that a soldier observes the field and detects enemy units. When a target is spotted, he calls BHQ for fire support and provides the target details. "Time-On-Target" (TOT) is the military coordination of artillery fire observed by multiple firing units, so that all the munitions arrive at the target at precisely the same time (or plus or minus three seconds from the prescribed time of impact). This is done in order to achieve maximum target destruction. BHQ assigns target to the batteries, and also schedules a certain "TOT" for the batteries to comply. Each battery fires according to its target assignment. When field component receives fire, and if the detonation is within a destruction radius, then the target is said to be destroyed otherwise it is missed. If all batteries mange to hit the target within TOT±3 we say, that a desirable property have been satisfied. In order to proceed with our verification process, we assume that the BOM composition is given as an input. Fig. 4a represents BOM state-machines of each component in the composition. We extend each BOM to E-BOM, and transform it to CPN component model using our transformation tool. Fig. 4 presents the E-BOM and CPN Component Model representation of Battery component (as an example). [For further details we refer the interested readers to download complete TCPN implementation of Field Artillery scenario from: http://web.it.kth.se/~imahmood/FieldArtillery].

Fig. 4b describes the E-BOM including state-variables, communication ports and the extended transitions, whereas



Battery E-BOM

States	Ready, PreparingCannon, ReadyToFire; Initial-State(s) = Ready
State Variables	{ID, CT, TTF}
Comm. Ports	IN-Ports = {AT}; Out-Ports={F}

Transitions

State	Event	$\{SV_{In}\}$	$\{SV_{Out}\}$	Action	TimeFx	NextState
Ready	AssignTarget	-	CT	-	-	Preparing
Preparing	Prepared	ID, CT	CT, TTF	act_1	PrepDelay()	ReadyToFire
ReadyToFire	Fire	CT ,TTF	-	act_2	TTF()	Ready

Notes: ID is the battery ID; CT stands for Current Target; The action functions act_1 & act_2 are defined using CPN-ML and can be viewed in the TCPN implementation; PrepDelay() and TTF() are Wait/Deadline time functions

Figure 4. (a) Field Artillery BOM State-machine (b) Battery E-BOM (c) Battery CPN Component (d) CPN Compoased Model

Fig. 4c, illustrate one of the generated CPN component models. When all components are transformed, they are composed in a TCPN composed model as shown in Fig. 4d. The circles represent socket places that connect components with each other. The rectangles represent components that are transformed form E-BOM. The composed CPN model can now be executed and verified. We define the verification template in Table 2 and show how each property is verified using the state-space query functions:

Property	CPN Translation and verification method
All enemy units are destroyed	Search all nodes, where **Data** place of the **Field** components has an empty list
Deadlock freedom	There is no node that has no outgoing arc (except the goal states).
TOT as safety property.	Check if there is any node in the state space, where **F** place has three tokens (meaning all three batteries have delivered fire) and at least one of these tokens deviates from the TOT more than 3 time units, then the TOT property is violated. If no such node is retrieved, then TOT property is said to be satisfied. We apply minTime() and maxTime() functions to verify this property.

Table 2. Verification Template

For verification, state-space is calculated using CPN tools. We use our state-space analysis library functions to perform property verification according to our verification template. If all three properties are satisfied, we say that the model is verified at dynamic semantic level, and hence justifies the necessary condition of the overall composability.

V. SUMMARY & CONCLUSION

In this paper we present a process for the verification of composability at dynamic semantic level, with a focus on models of Real-time systems. We propose to use Base Object Model as a conceptual modeling framework, and Colored Petri Nets as an executable modeling standard. We suggest extending standard BOM model into E-BOM, so that it contains necessary behavioral details, required for its implementations, specially the time function. We provide an automatic transformation method to convert E-BOM into our proposed Timed CPN component model, which is useful to represent a model component in CPN language, yet it preserves the model structure and behavior conceptually intact. For the purpose of dynamic-semantic composability verification we suggest a verification template, as assessment criteria, which can be used to verify our executable model using State-space analysis. Lastly, we discuss a case study of Field Artillery to show how our framework verifies a given composition.

Our proposed verification framework expedites the process of composability verification and provides suitable environment for finding out defects in the model composition. Moreover, because of numerous analysis methods and verification algorithms contributed by the CPN community over a couple of decades, CPN provides a significant improvement on efficient and accurate reasoning regarding the model correctness. We intend to fully automate the construction of TCPN composed model from the generated CPN components to further depreciate the manual human effort in the development.

REFERENCES

[1] Marko Hofmann, "Component based military simulation: lessons learned with ground combat simulation systems," in Proceedings 15th European Simulation Symposium, Delft, Netherlands, 2003.

[2] Mikel D. Petty and Eric W. Weisel, "A theory of simulation composability," Virginia Modeling Analysis & Simulation Center, Old Dominion University, Norfolk, Virginia, 2004.

[3] Balci, J D Arthur, and W F Ormsby, "Achieving reusability and composability with a simulation conceptual model," Journal of Simulation, vol. 5, no. 3, pp. 157-165, August 2011.

[4] Paul K. Davis and Robert H. Anderson, Improving the composability of department of defense models and simulations.: RAND National Defense Research Institute, 2003.

[5] Andreas Tolk, "Interoperability and Composability," in Modeling and Simulation fundamentals Theoretical Underpinnings and Practical Domains.: John Wiley, 2010, ch. 12.

[6] Paul Gustavson, "Building and Using Base Object Models (BOMs) for Modeling and Simulation (M&S) focused Joint Training," in Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), Orlando, Florida, 2005.

[7] Paul Davis, "Composability," in Defense Modeling, Simulation, and Analysis: Meeting the Challenge. Washington, D.C.: The National Academies Press, 2006.

[8] Farshad Moradi, Rassul Ayani, Shahab Mokarizadeh, Gholam Hossein Akbari Shahmirzadi, and Gary Tan, "A Rule-based Approach to Syntactic and Semantic Composition of BOMs," in 11th IEEE Symposium on Distributed Simulation and Real-Time Applications, Chania, 2007.

[9] Osman Balci, "Verification, Validation and Accreditation of simulation models," in Proceedings of the Winter Simulation Conference, Atlanta, GA, 1997.

[10] Mikel D. Petty, "Verification and Validation," in Principles of Modeling and Simulation.: John Wiley & Sons, 2009, ch. 6.

[11] Ernst-Rudiger Olderog and Henning Dierks, Real-time systems - formal specification and automatic verification. Oldenburg, Germany: Cambridge University Press, 2008.

[12] Alexandre David, Oliver M Möller, and Wang Yi, "Verification of UML Statecharts with Real-Time Extensions," Uppsala, Sweden, Tehnical Report Wang.

[13] Hesham Saadawi and Gabriel Wainer, "Verification of real-time DEVS models," in Proceedings of the Spring Simulation Multiconference, San Diego, CA, USA, 2009.

[14] Claudia Szabo, Yong Meng Teo, and Simon See, "A Time-based Formalism for the Validation of Semantic Composability.," in Winter Simulation Conference, TX, USA, 2009, pp. 1411-1422.

[15] Kurt Jensen, "An Introduction To The Practical Use Of Coloured Petri Nets," in Lectures on Petri Nets II: Applications, Advances in Petri Nets. London, UK: Springer, 1998, pp. 237-292.

[16] Kurt Jensen and Lars M. Kristensen, Coloured Petri Nets Modelling and Validation of Concurrent Systems, 1st ed.: Springer, 2009.

[17] CPN Tools. [Online]. http://cpntools.org/

[18] Kurt Jensen, Søren Christense, and Lars M Kristensen, "CPN Tools State Space Manual," Aarhus , Denmark, Manual 2006.

[19] Lars Michael Kristensen, "State Space Methods for Coloured Petri Nets," Department of Computer Science, University of Aarhus, Aarhus, Denmark, Ph.D. Dissertation 2000.

[20] Imran Mahmood, Rassul Ayani, Vladimir Vlassov, and Farshad Moradi, "Verifying Dynamic Semantic Composability of BOM-based composed models using Colored Petri Nets," 26th Workshop on Principles of Advanced and Distributed Simulation, Zhangjiajie, China, 2012, pp. 250-257.