Compositional Formal Verification for Business Process Models with Heterogeneous Notations Using Colored Petri Net

C. Dechsupa, W. Vatanawood, and A. Thongtak

Abstract—**The main drawback of a huge software process verification is the variety of the sub-system models. The designers have to transform the design models described in semi-formal modeling language to be an abstract model expressed in the formal modelling language. The complexity of the design model results in the time-consuming and may leads to the incorrect abstract model. In this paper, we propose an alternative fashion to verify the business process models that contains sub-process models designed in the heterogeneous modeling languages, BPMN, BPEL and UML Activity diagram. The partial and hierarchical verification techniques are proposed. The design model are mapped into Colored Petri Net (CPN) models using transformation rules. Next, we validate the model's properties the obtained CPN models using our framework.**

Index Terms—**Formal Verification, Business Transaction, Colored Petri Net, BPMN, BPEL, UML Activity Diagram**

I. Introduction

A model checking [1] has been used for verifying software models. The designers have to model the abstract model in formal modeling language using the automatic transformation framework or the manual creation. The huge business process model contains heterogeneous representation of sub-systems, which each sub-system may be represented in different semi-formal modeling languages such as BPMN, BPEL, and UML activity diagram. It results in a cumbersome procedure of the formal model abstraction for the designers. Moreover, the abstract model used in the validating stage may produce tremendous state space graph and faces with the state space explosion problem [2].

CPN is an outstanding formal modeling language for verifying the concurrent system. It is suitable for complex and huge system verification since the modelers can perform the construction of compact and parameterized models, is able to entirely verify the procedural logic in business process models. The CPN color sets and inscriptions can be used for determining data flows, data objects, and primitive data types. However, the modelers must have the CPN background knowledge and how to create an abstract model form the existing process models or from the software requirement specifications. To make compositional formal verification easier, the automated transformation and verification tools are a precious equipment for the designers who are not familiar with CPN language. They advocate a transparency of the complicated transformation procedures and also provide functionality advoiding the state space explosion problem.

In this paper, we propose a CPN model automated framework and verification techniques supporting the business process models described in BPMN, BPEL and UML activity diagram. In our verification processes, the existing design models are transformed into the CPN models. The transformation rules are extended from our previous works [3, 4, 5], and they are implemented as a revision of CP4BPMN tool [4]. We generate state spaces from the obtained CPN model and analyze the state spaces using the exploration queries formalized from the user requirements to check that the model satisfies safeness, soundness and transaction properties or not.

The remaining of this paper is organized as follows: Section II describes background of General Ledger System. Section III reviews the related researches. Section IV discusses the proposed approach and section V illustrates our implementation with a simple case study. Section VI is the study's conclusion.

II. Background

A. Formal Verification using Model Checking

A model checking is an automatic verification technique for testing the software and hardware design models. The verification procedure is divided into two main stages, the model abstraction stage and properties checking stage. For the state space analysis method, all possible states are generated as state space graph. In case of huge abstract model, the hierarchically structural re-arrangement can reduces the complexity of an abstract model and size of a state space. It can be applied in conjunction with other existing techniques to avoid a state space explosion problem such as sweep-line method [6] and partial verification method.

B. Business Process Model Notation (BPMN)

Business Process Model Notation is commonly known as BPMN [7]. It is graphical notations that are used for modeling a business process or describing procedural logic

C. Dechsupa received Ph.D. degree in computer engineering from Chulalongkorn University, Bangkok, 10330 Thailand (e-mail: Dechsupa_chan@yahoo.com).

W. Vatanawood is an associate professor at department of computer engineering faculty of engineering, Chulalongkorn University, Bangkok, 10330, Thailand (e-mail: wiwat@chula.ac.th).

A. Thongtak is an assistant professor at department of computer engineering faculty of engineering, Chulalongkorn University, Bangkok, 10330, Thailand (e-mail: arthit.t@chula.ac.th).

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

of software. BPMN contains four main elements groups: 1) flow elements divided into events, activities and gateways, 2) connecting flows, 3) pools and lanes and 4) artifacts. The BPMN elements can express both structural and behavioral models, there are many tools and model types. *Process Diagram* can express the procedural logic in low-level, which a data flow and control flow can be detailed event if the data types through an activity. Furthermore, *Collaboration Diagram* is an extended model of process diagram by integration of interfaces among organizations into the model.

Because of the lack of BPMN standard semantics, that can cause a system crash or desirable properties dissatisfaction. There are many researches provided tools and methodologies to cope with the issues of BPMN design model verification.

C. Unified Modeling Language :UML Activity Diagram

UML activity diagram [8] is used for presenting the activity sequence of a workflow. It focuses on the control flows perspective. The most common components of an activity diagram include *Activity*, *Decision* and *Control flows* (gateways), *Start* and *End*. We can simplify and improve any process by clarifying complicated business process using activity diagram, supporting the expression both sequential processing and concurrent processing of activities using a gateway symbols. In UML 2.0, the activity diagrams were re-formalized to be based on Petri net-like semantics.

D. Web Service Business Process Execution Language

Business Process Execution Language is commonly known as BPEL or WS-BPEL [9], which is used for business processes definition as coordinated sets of Web service interactions to achieve business goals. It uses an XML-based language supporting the web services technology stack, including SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination, and WS-Transaction. These standard languages are used to define business process definition, process model including the process grammar for describing the application behaviors based on interactions between the service partners.

E. Colored Petri-Net

Colored Petri-Net or CPN [10] is a formal modeling language for verifying the concurrent systems. It is a combination of the classical Petri-net and programming language. CPN provides functionalities to address the variables declaration, data types, data manipulation including the hierarchically structural representation. The data types declared in a CPN model are called a color set. *Place* represents a state of a model, containing the data objects called *token*, whereas the data value of a token is called a token color. *Transition* is used to represent the state change, which is located between places and their connections are an *Arc*. A state space generator is required to compute the reachability graph and needs temporal logic [11] queries for validating the model's properties.

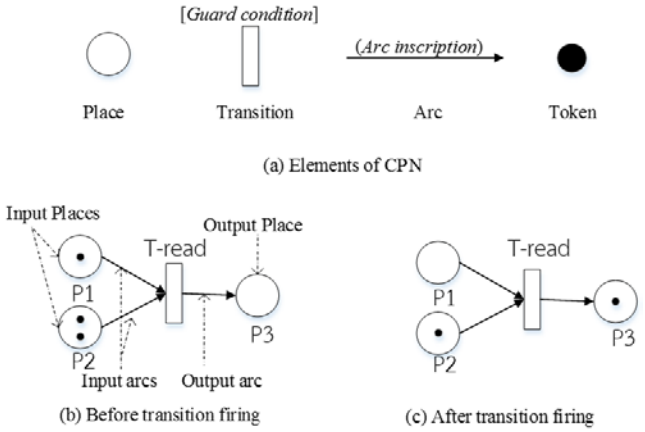


Fig. 1. Core elements and transition firing of CPN [10].

III. Literature Review

In the area of BPMN model verification, the works of [12, 13] proposed the CPN based representation for the formal model abstraction of BPMN design model. They used CPN tools to verify the model properties. The control flows were considered but they did not focus on the data flows perspective. In [14] focused on the loop, sub-process and transaction of the BPMN model. CPN places were used for representing the task properties but they did not detail a colored set handling and implementation. Dechsupa et al. [4, 5] provided the transformation rules and framework for verify the BPMN design model, solving the gaps of transformation rules of previous related works.

In the area of UML activity diagram verification. In [15] considered a variant of UML activity diagram. They addressed the issues by providing a formal semantics with Petri nets as the semantical domain of interpretation. Eshuis et al. [16] compared the design choices between Petri nets and UML activity diagram. They defined two formal semantics for UML activity diagrams and illustrated only the advantage of their UML activity diagram semantics but they did not verify the model properties. Likewise, D. Foures et al. [17] checked the invariant properties in activity diagram using TINA toolbox, and *selt*.

There are various verification techniques applied to verify the composite service. Almost all research in this area focus the service composition verification on the service interaction designed in BPEL, works of [18, 19]. Artifacts in service models transformed into formal mathematical models that were allowed the designer to verify the structure and behavior of service model. CPN tools framework were used for modeling and analyzing the web service models. The service models and its additional files were mapped into CPN model. Next, the temporal properties written in Computational Tree Logic (CTL) were used to verify their properties. Wei Tan et al. [20] provided an approach to verify the compatibility of web services composition. They transformed BPEL description written BPEL processes to CPN model and analyzed the message passing of the mediator of web service composition. Likewise, Yingmin LI et al. [21] diagnosed of faulty activities and data in orchestrated web services. The inequations solving algorithm is proposed to improve the fault detection.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

IV. Methodology

The overview of our verification technique is shown in Fig. 2. A system template will be created to fill in the components or sub-system of a whole system. The model types are determined in order to use in the transformation step. The model files are imported accordingly with a model type determined (BPMN, BPEL and UML Activity diagram). The elements of all models in the system template are extracted and considered. Each element is transformed into CPN structures depended on the transformation rules. The connection edges between sub-systems are manually identified by the designers in order to merge them together. Next, the designers assign CPN model inscriptions and take the obtained CPN model to generate a state space and to validate the model properties using the state space analysis technique.

Since we extend the capacity of CP4BPMN tool by adding transformation rules supporting the notation of UML activity diagram and BPEL. We use the template model to classify the input model type, and design the one-to-one mapping rules to transform input element to be CPN constructs. On the basis of the formal definitions in [4], we present formal definitions involving an extension. Section A describes the BPMN transformation rules, and Section B and C show the transformation rules of BPEL and UML activity diagram respectively.

A. Transform BPMN to CPN.

We extended the transformation rules of [4, 5]. The transaction and boundary event elements can be addressed by our transformation rules. We define the formal definition to show the relationships of elements as bellows.

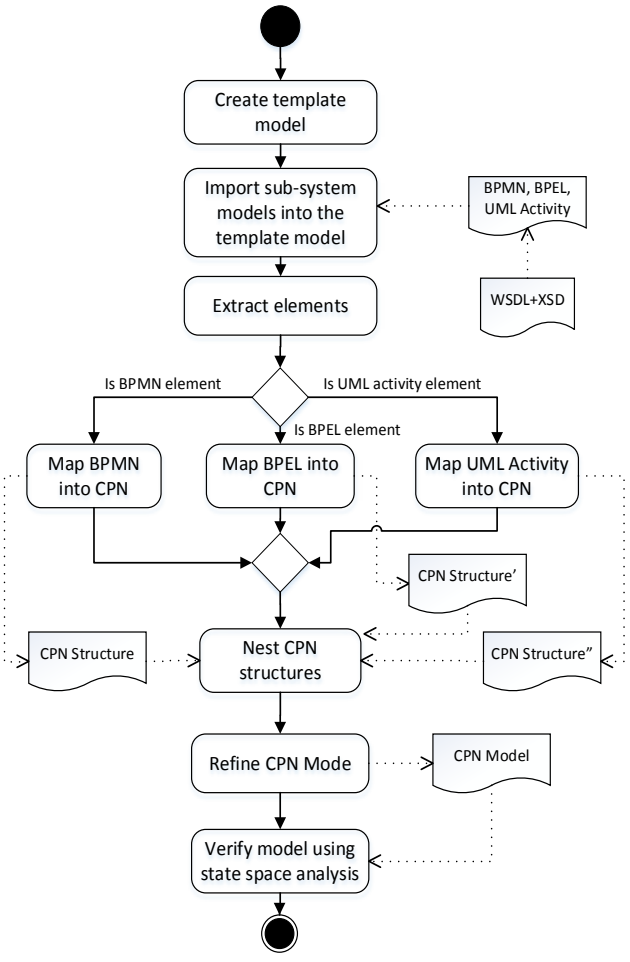


Fig. 2. Compositional Formal verification process.

Definition 1*: Extended Ordinary BPMN process.*

An extended ordinary BPMN process is a tuple $\mathcal{O}' = (\mathcal{O}, fE_T, fEs_T)$ where

$\mathcal{O}$ is ordinary BPMN process.

fE_T is a mapping function revised that is used to indicate the type of an intermediate event, fE_T: $E_I \longrightarrow$ {*catch, throw, boundary, boundary non-interrupt*}.

fEs_T is a mapping function used to indicate the type of an intermediate boundary event, fEs_T: $E_I \longrightarrow$ {*Error, compensation, Cancel, Signal, Timer*}.

Definition 2*: Extended Hierarchical BPMN process (BPMN process with sub-processes or transaction).*

A hierarchical BPMN process is a tuple $\mathcal{H}' = (\mathcal{H}, T\mathcal{O}, fN_{ST})$, where

$T\mathcal{O}$ is a set of ordinary BPMN processes that are sub-process determined as the transactional processes.

fN_{ST} is a mapping function, $fN_{ST}: A \longrightarrow T\mathcal{O}$.

The basis of transformation rules BPMN to CPN relies the transformation rules of [4, 5]. The CPN structure of the extended rules likes that of the existing rules, and arcs connected between CPN structures are adjudged. In boundary events transformation rule, if the boundary event occupies on an activity or sub-process determined as the transactional processes, the structure of the boundary events are linked from all CPN transitions of such activity to the transition of boundary events by the CPN arcs.

B. Transform BPEL to CPN.

On the basis of the transformation rules in [3], the transformation rules are revised in order to obtain the CPN constructs that can be composed with CPN contracts derived from the other transformation. The transformation rules from BPEL elements to CPN structures are as follows:

1) For the set of consecutive basic activities, two CPN transitions are defined to represent the state change: input reading state and output writing state.
2) For a partner link in Plnk, an additional dummy web service is defined by a CPN transition.
3) For structured activity of "*If-Else*", CPN transitions are defined along with their guard conditions.
4) For each pair of activities or the pair of activity and dummy web service, a CPN place is defined to show that such activity has been computed already.
5) Between a transition and a place is connected by an arc. The direction of the arc relies on the direction of the *pertnerLink.*
6) For each variable in BPEL is defined in CPN model.
7) For each type of variable is defined as a color set in CPN model.
8) WSDL is used to create the arc inscriptions depending on input and output definition.

C. Transform UML Activity diagram to CPN.

The definitions and simple rules of transformation of UML activity diagram into CPN model are proposed.

Definition 3*: Ordinary UML Activity Diagram.*

An ordinary UML activity diagram is a tuple $\mathcal{OA} = (B, T, B_S, B_E, B_G, B_F)$ where

B is a finite nodes.

T is a finite set of activities or atomic tasks, $T \subseteq B$.

B_S is a set of start events, $B_S \subseteq B$.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

B_E is a set of end events, $B_E \subseteq B$.

B_G is a set of gateways, $B_G \subseteq B$. There are three symbols: *Fork*, *Join* and *Decision*.

B_F is a set of connector symbols, $B_F \subseteq (B \times B)$.

Var is a set of variables in $\mathcal{OA}$.

Definition 4*: Partition UML activity diagram.*

A UML activity diagram is a tuple $\mathcal{PA} = (\mathcal{OA}, S_L, fS_M)$ where:

$\mathcal{OA}$ is a set ordinary UML activity diagram.

S_L is a set of swim-lanes.

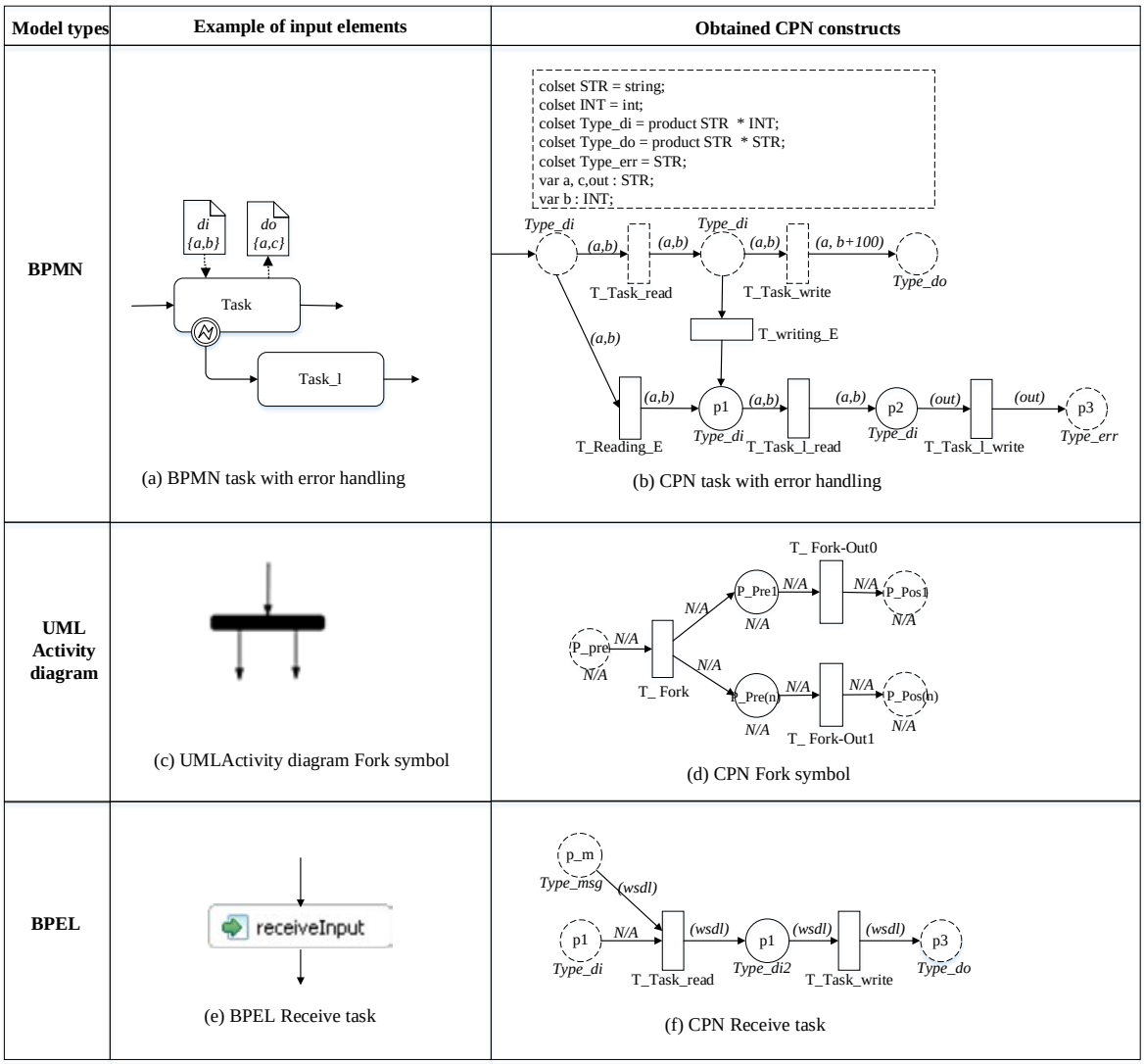
fS_M is a mapping functions used to indicate the pool of BPMN process, $fP_M: \mathcal{OA} \longrightarrow S_L$.

According to the definitions of UML activity diagram and CPN models [4], we define the simple rules of UML activity diagram transformation into CPN.

Transformation rules from $\mathcal{PA}$ to CPN:

1) For the sets of the start and end event, a transition in T_T is defined in CPN.
2) For the set of the activities, the state transitions of an activity are spited into two states, input reading state and output writing state. A transition in T_T is defined in CPN.
3) For a gateway, all connectors outgoing the gateways, a transition in T_T is defined in CPN, and a guard condition on the connector symbol is copied to be the transition's inscription.
4) For each pair of activities or the pair of activity and event, a CPN place is defined to show that such activity has been computed already.
5) CPN arc is used for connecting between a place and transition, which is depended on the direction of the connector symbol.
6) For each variable in *Var*, a corresponding variable in *VV* is defined in CPN.
7) For each type of variable in *Var*, a corresponding Σ is defined in CPN.

The elements in a business process model are arranged likely a directed graph. The graph of design models must conform to the well-defined process [4]. The elements in the business process model are transformed into CPN structures, and they will be connected to the other CPN constructs by using the concatenation rule of [4]. The excerpt of the transformation rules are illustrated in Fig. 3. The designers will determine the interface between sub-systems in the model refinement stage. Including, the initial marking, inscription expression and hierarchical structural re-arrangement. These steps must be proceeded before the state space construction. In the refinement procedure, the designers can concentrate at certain sub-systems or sub-processes in a sub-system. The sub-system represented in



(a) BPMN task with error handling

(b) CPN task with error handling

(c) UMLActivity diagram Fork symbol

(d) CPN Fork symbol

(e) BPEL Receive task

(f) CPN Receive task

Fig. 3. The excerpt of transformation rules into CPN.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

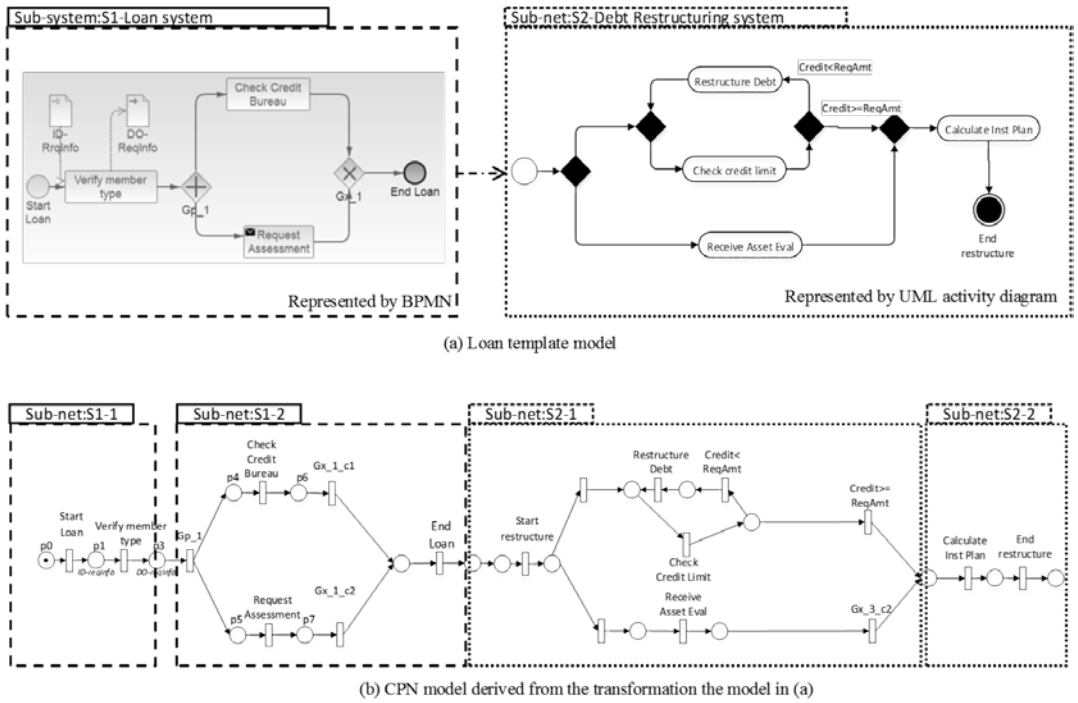


Fig. 4. An example of the template model and obtained CPN model derived from the transformation rules.

CPN is called *Sub-nets*. The designer can select the sub-nets and can also re-arrange them in deferent abstraction level that is called the hierarchical structure by reducing some of the sub-nets to be a black box process.

The example of a CPN model derived from the transformation stage is shown in Fig. 4. The template model of a mortgage loan contains two sub-systems, *Loan system* and *Debt restring system*. The loan system is designed in BPMN while the debt restricting system is described in UML activity diagram. Both sub-systems are transformed into CPN model shown in Fig. 4(b). Each sub-system may contains semi sub-nets that are derived from the partitioning algorithm of CP4BPMN tool such as S1:{S1-1, S1-2}. The sub-nets can be arbitrarily chosen to be the abstract model for the state space construction. And the designer can use the hierarchical verification technique to reduce the state space size. Since the state space generator of CP4BPMN tool implemented the sweep-line method in the state space construction algorithm to store only certain fragment states.

To validate the model properties, the state space exploration queries are formalized form the user requirements in term of temporal logic. The tool provides the functional commands for creating the exploration queries. We verify the model properties such as safety property, completeness property, including specific properties.

V. Implementation

A case study is used to illustrate the detail of our implementation. The template model of a retail system is shown in Fig. 5. The template model comprises six sub-systems, the dashed-line arrows represent the interactions between sub-systems. The sub-system models are represented in deference semi-formal modeling languages. For example, *Core process of POS* is modeled in BPMN model while *Insurance process* is expressed in BPEL model.

We create the template model using CP4BPMN tool. All sub-system models are uploaded into the template model. Next, the tool transforms all elements in all sub-system models into CPN models. We refine the obtained CPN models before generating state space graph. Due to the huge design model, we verify the retail system by combination of partial and hierarchical verification techniques to reduce the state space size. For instance, we re-arrange the CPN model in hierarchical structure by determining the sub-net S-3, S-4 and S-5 into substituted transitions if they are inconsiderable sub-systems, or selecting only sub-net S-1, S-2 and S-6 to be an abstract model. Fig. 6 shows the CPN model of sub-net S-1, S-2 and S-6 in hierarchical structure by the sub-net S-6 is a substituted transition. The model contains 16 places, 13 transitions and 32 arcs.

We test the implementation by combination of various sub-systems for five case studies. Case 1: {S-1, S-2, S-6}, Case 2: {S-1, S-2, S-5, S-6}, Case 3: {S-1, S-3, S-5, S-4}, Case 4: {S-1, S-2, S-3, S-4} and whole sub-systems. The CPN models are refined at their inscriptions and structures including the initial markings. Next, the state spaces of the obtained CPN models are constructed. The state spaces are tested by the exploration queries. The Excerpt results of the CPN model verification are detailed in Table 1.

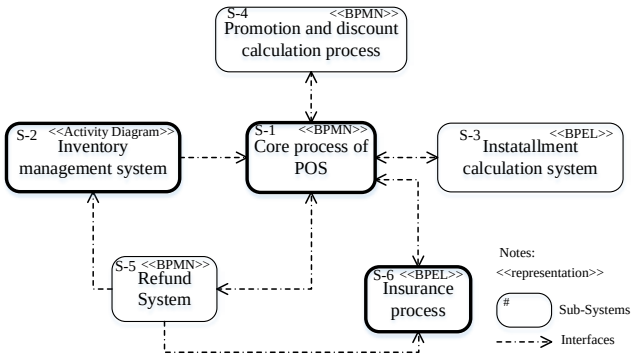


Fig. 5. The template model of a retail system.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

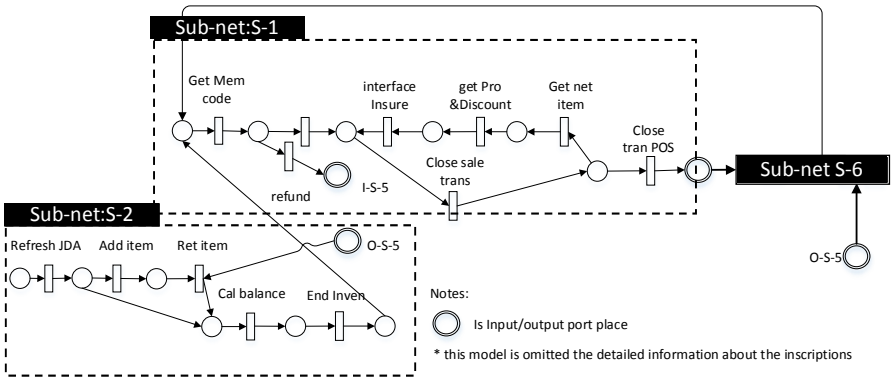


Fig. 6. An example of hierarchical CPN model of retail system considered only sub system, S-1, S-2 and S-6 (Case 1).

TABLE I
THE EXCERPT RESULTS OF CPN MODEL VERIFICATION

No	Requirements : Queries	Results
1	Soundness checking : SOUND()	Satisfied
2	Invariant checking: INV()	Satisfied
3	Find unreachable activities: -UNREACH()	Satisfied
4	The system must calculate the inventory amount of item; next, the processes of the add item check the duplicated item before adding new amount.: EF_AND_EF('t12', 't5')	Unsatisfied
5	All sale transactions have to proceed three processes: get item, get promotion and interface insurance date respectively. : EF_IMPLY_EF('t4',AND('t8', 't9'))	Satisfied
6	The closes of sale transaction and POS transaction are mandatory processes for the point of sale termination. : EG_AND('t15', 't17')	Satisfied

The 't*' in the query is the CPN transition's name representing an activity.

VI. Conclusion

The business process models should be verified for achieving desired properties. These models may have heterogeneous representation such as BPMN, BPEL and UML activity diagram. If the design model contains many sub-systems, and they are represented in deferent representation, it means that the designer have to transform the design models into the formal model represented in the same formal modeling language. We propose the method for verifying a composite design models written in BPMN, BPEL, and UML activity diagram in the early stage of the low level design process using the model checking technique. We present the transformation rules of input elements into CPN models, which is extended from our previous works. The state spaces are constructed and they are explored with the soundness, safeness, and specific properties queries. All stages of verification are activated by using CP4BPMN tool. Form the experiment, we observe that the limitation of the UML activity diagram transformation rules are clumsily procedure because of the lack of data flows in a UML activity diagram. Thus, CPN constructs derived from the UML activity diagram are without inscriptions. Our future works will include the additional files describing the data flows of UML activity diagram to be the data sources for inscription creation. We will adjust the template model creation process and will provide the flexible functionality to determine the communication flows between the sub-systems as well.

References

[1] C. Baier and J.-P. Katoen, Principles of Model Checking. MIT Press, 2008.
[2] C. E. M., K. William, N. Miloš, and Z. Paolo, Model Checking and the State Explosion Problem. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–30.
[3] C. Dechsupa, W. Vatanawood, and A. Thongtak, "Formal verification of web service orchestration using colored petri net," IMECS 2016, pp. 398–403, 2016.
[4] C. Dechsupa, W. Vatanawood, and A. Thongtak, "Transformation of the BPMN design model into a colored petri net using the partitioning approach," IEEE Access, vol. 6, pp. 38421–38436, 2018.
[5] C. Dechsupa, W. Vatanawood, and A. Thongtak, "Hierarchical Verification for the BPMN Design Model Using State Space Analysis," 2019, DOI: 10.1109/ACCESS.2019.2892958.
[6] K. Jensen, L. M. Kristensen, and T. Mailund, "The sweep-line state space exploration method," Theoretical Computer Science, vol. 429, pp.169–179, 2012.
[7] Object Management Group, "OMG Unified Modeling Language TM (OMG UML) version 2.5," 2015.
[8] UML Revision Task Force. OMG Unified Modeling Language Specification, Version 1.4 (final draft), 2001.
[9] OASIS, "Web Services Business Process Execution Language Version 2.0," 2015.
[10] K. Jensen and L. M. Kristensen, ''Colored Petri nets: A graphical language for formal modeling and validation of concurrent systems,''. ACM, vol. 58, pp. 61–70, May 2015.
[11] M. Fisher, An Introduction to Practical Formal Methods Using Temporal Logic. Hoboken, NJ, USA: Wiley, 2011.
[12] C. Ou-Yang and Y. D. Lin, "Bpmn-based business process model feasibility analysis: a petri net approach," Journal of Production Research, vol. 46, no. 14, pp. 3763–3781, 2008.
[13] M. M. Ibrahim, "Formal semantics of bpmn process models using cpn," IREIT. J, vol. 5, no. 3, 2017.
[14] M. Ramadan, H. G. Elmongui, and R. Hassan, "Bpmn formalisation using coloured petri nets," The Global Science and Technology Forum.
[15] D. Fahland, "Translating UML2 Activity Diagrams to Petri Nets," 2008.
[16] R. Eshuis and R.Wieringa, "A Comparison of Petri Net and Activity Diagram Variants," 2007.
[17] D. Foures, V. Albert and J.C. Pascal, "ACTIVITYDIAGRAM2PETRINET:Transformation-based Model in Accordance with the OMG SYSML Specifications," ESMC 2011, p.429-433, 2011.
[18] M. Perepletchikov "A Formal Model of Service-Oriented Design Structure", Software Engineering Conference, 2007. ASWEC 2007. 18th Australian, pp.71 -80.
[19] H. Guan, S. Ying and C. Wang, "A Correctness Verification Approach of the BPEL Exception Handling CPN Model Based on Temporal Property," Journal of Networks, Vol.9, pp. 2743-2750, 2014.
[20] W. Tan, Y. Fan, and M. Zhou, "A Petri Net-Based Method for Compatibility Analysis and Composition of Web Services in Business Process Execution Language," IEEE Trans. on Automation Science and Engineering, Vol.6, pp. 94–106, 2009.
[21] Y. Yan, P. Dague, Y. Pencole and M. -O. Cordier, "A Model-based Approach for Diagnosing Faults in Web Service Processes", JWSR. , vol. 6, pp. 87-110, 2009.

ISBN: 978-988-14048-5-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)