A Study of the Software Tools Capabilities in Translating UML Models to PN Models

Bassam Atieh Rajabi
Software Engineering Department
Faculty of Computer Science and Information Technology, University of Malaya
Kuala Lumpur Malaysia
bassam_rajabi@perdana.um.edu.my

Sai Peck Lee
Software Engineering Department
Faculty of Computer Science and Information Technology, University of Malaya
Kuala Lumpur, Malaysia
saipeck@um.edu.my

***Abstract*— Integrating Object Oriented (OO) modeling language and Petri Net (PN) modeling is important to gather complementary benefits of these two modeling languages. OO technology is used to describe the static aspects of systems because of its powerful structuring capabilities, whereas PN is used to model the behavioral aspects for concurrent and non deterministic applications. PN is an executable modeling language; the behavior of the model can be validated and verified automatically using formal specifications and mathematical foundation. In this research, a comparative study for the representation capabilities of the software tools in translating OO models to PN models will be provided and discussed.**

***Keywords*- Object Orientation, Object Oriented Petri Net, Petri Net, Software Tool, UML.**

I. INTRODUCTION

Petri Nets (PNs) are said to be "a powerful instrument for modeling, analyzing, and simulating dynamic systems with concurrent, asynchronous, distributed, parallel, and nondeterministic behavior" [1]. They provide a formal method with strong mathematical foundation and graphical representation [2]. A PN is a "directed graph that mainly consists of two different nodes, places and transitions" [3]. High Level PNs (HPN) and Low Level PNs (LPN) are specialized types of PN. Data types and data processing techniques are supported in HPN, but not supported in LPN [4].

A Coloured Petri Net (CPN) [3] [5] [6] incorporates both data structuring and hierarchical decomposition. The hierarchical structure of the CPN is useful in representing object dynamics, inheritance, and checking models correctness. Timed PNs [7] introduce time in PNs.

According to Bhushan Bauskar and Boleslaw Mikolajczak [5], OO methodology is "an established technique for structured software design. It supports polymorphism, inheritance, and dynamic binding to design comprehensible, maintainable, and flexible software". Object orientation introduces objects that represent real-world entities. An object has a state and behavior. A set of similar objects is called a class. Message passings are requests for communication between objects [3].

Unified Modeling Language (UML) is a "language for specifying, visualizing, constructing and documenting the artifacts of OO software systems, as well as for business modeling" [5] [8]. UML is increasingly being seen as the standard for software modeling and design. UML 2.2 is the most recent version of UML [9].

The main advantages expected from Object Oriented Petri Net (OOPN) formalisms are better representation of systems complexity, and greater ease to adapt, correct, analyze, or reuse a model [10]. OOPN is based on a combination of the best characteristics of PN and OO design methods [11] [12] [13] [14]. UML is used to describe the static aspects of systems because of its powerful structuring capabilities, and PN is used to model the system dynamics and behavioral aspects. It is efficient for implementing concurrent and non deterministic applications; because the behavior of the models can be validated and verified automatically using formal specifications and mathematical foundation. The graphical representation of PN aids in the understanding of such formal specification, through its automated analysis techniques.

The integration of PNs and OO models can minimize or avoid the negative aspects of object orientation such as communication and synchronization between the objects in

the large and scalable systems, and formal validation and verification [15]. Some concepts related to OOPN such as object nets, token, class, method nets, and synchronous ports are introduced in [16].

Modular CPNs are similar to Hierarchical CPNs, but include the notion of super places. Object-Based PNs allow tokens to have values which are the object identifiers of subnet instances [12]. OOPNs enhance Object-Based PNs with the notion of inheritance. Object Petri Nets (OPNs) enhance OOPNs with test and inhibitor arcs, thus providing the foundation for access functions.

In this research, a comparative study for the representation capabilities of the software tools in translating UML models to PN models will be provided and discussed.

The rest of the paper is organized as follows: In Section II, we will provide a brief overview about related works. In Section III, a comparative study about the representation capabilities of the software tools in translating UML models to PN models will be discussed and analyzed. Finally, we will summarize the research results and future work.

II. Related Works

UML to CPN transformation framework was proposed in [17]. A dynamic model analysis is provided in this framework by mapping state chart models and collaboration models to a CPN. The state chart models are converted to CPN models, and then the collaboration models are used to connect these models into a single CPN model. The state in the state chart is transformed to a place in PN, and the transition is translated to a PN transition. Arcs, events and actions are translated to tokens. The framework did not consider complex features of UML state chart, such as concurrent composite state.

Translating UML models into OPNs models approach is suggested in [18]. This approach defines the rules for transforming UML models into a HPN model. It uses a subset of UML models: the class model, the state charts of all the classes, and the collaboration model.

Object PN Models (OPMs) [19] are used to generate a PN model from UML specifications. The generation is based on two steps: the generation of objects and linking these object models to create a system model. The final system model is represented using CPN. Formal PN analysis techniques are used to analyze the OPMs. OPMs do not address UML classes as the starting point to derive the properties of the system [18]. The representation of objects in CPN is as follows [20]: object classes and states classes are represented by places, object instances are represented by tokens, and the object value state is represented by a function returning the state instance object.

The technique to transform UML state models to object CPNs was proposed by [21]. An Object CPN is created by constructing an object net for each object in the UML state model. Then these object nets are transformed into class nets connected together through communication channels to create a PN model.

Abstract Node approach [5] is used to present the technique to transform an OO design into a hierarchical CPN model. Abstract Node is a unified abstraction construct of PNs. There are two types of abstract nodes: abstract places to hold data and abstract transitions for data processing. Abstract Node can be used to represent objects. Class models and sequence models in OO design can also be represented.

Translating UML 2.0 Sequence Models (SDs) into CPN models was presented in [22] [23]. The translation to a CPN model is based on a Use Case model and a set of sequence diagrams for each use case [22].

A Hierarchical Object Oriented Petri Net (HOOPN) is suggested in [24] [25] to integrate HPN with OO concepts. HOOPN is developed to complement the weakness of the PN formalisms in terms of modularity, and reusability. A HOOPN model is a “PN representation that corresponds to a class in object orientation” [25].

III. Software Tools for Translating UML Models to PN Models

UML supports a number of model types to model the system from different views or perspectives. These views and perspectives described the static aspects and behavioral aspects [26].

Static models are: class and implementation models (components and deployment), and use cases. Dynamic behaviors models are activity model, state chart, and interaction models (sequence and collaboration models).

There are many software tools that support the translation of UML static and dynamic models to PN models. In this section, a brief description and comparative study between some of these tools will be provided.

ArgoSPE tool [7] [27] is a performance evaluation tool of software systems. It is based on translating the UML state machines, activity models and interaction models into Generalized continuous-time SPN (GSPN). The class and the implementation models are used to collect some system parameters. Stochastic Petri nets (SPNs) are used to represent the time in PNs to analyze the system performance. The GSPN model associates an instantaneous or exponentially distributed firing time with each transition. It can be used to model and analyze parallel systems lacking in SPN models.

WebSPN tool [28] [29] is a new modeling tool for the analysis of SPNs. It provides a discrete time approximation of the model stochastic behaviour to analyze a wider class of PNs. WebSPN 3.3 allows deriving PNs from UML models, with the purpose of analyzing, through the use of PNs. These models are: deployment, use case, and activity models.

A prototype tool was developed by Calderon [30] to transform the UML use case, class, and collaboration

models to a CPN model. Then the CPN model is used to check the correctness of use case models and concurrency analysis.

LoLA (A Low Level PN Analyzer) tool [31] is used to check the PNs model after translating it from the UML 2.0 activity models.

Baresi approach [18] defines the rules for transforming a UML model into a HPN model. This approach uses a subset of the UML models: the class model, the state charts of all the classes, and the collaboration model.

UML to CPN transformation framework was proposed in [17]. A dynamic model analysis is provided in this framework by mapping state chart models and collaboration models to a CPN. The state chart models are converted to CPN models, and then the collaboration models are used to connect these models into a single CPN.

Bokhari and Poehlman proposed a mechanism to map UML state models to object CPNs. This technique results in a PN model consisting of class nets matching the standard practice followed by most programming languages [21].

Meta-Modeling and Formalism transformation framework is a general framework for the analysis of software systems using Model-Checking [32]. This framework transforms the UML model which is composed of classes, state charts, and sequence models into PNs for further analysis.

Abstract Node approach is used to present the technique to transform an OO model into a hierarchical CPN model [5]. Using the concepts of Abstract Node, models like class models and sequence models in object orientation can be transformed to CPN.

Fernandes et al designed a tool for supporting the translation of UML use case and sequence models to CPN model [22]. The approach by Shin et al [33] is to model transformation of UML use case, class, and collaboration models to a CPN model.

A meta-level and highly automated technique based on graph transformation approach is presented in [34] to formally transform UML models to PNs for verification. This approach can be used to transform UML state charts and behavioral models to PNs.

Table 1 summarizes the comparison between these tools and approaches.

TABLE 1
TOOLS COMPARISON FOR UML MODELS TO PNS TRANSLATION

Tool Name	Model Name						
	Class	Use Cases	Activity	Calibration	State chart	Sequence	Deployment
ArgoSPE [27]	√		√		√		√
WebSPN [28] [29]		√	√				√
Calderon Prototype [30]	√	√		√			
LOLA [31]			√				
Baresi approach [18]	√			√	√		
UML-CPN approach [17]				√	√		
Bokhari and Poehlman approach [21]					√		
Meta-Modeling approach [32]	√				√	√	
Abstract Node approach [5]	√					√	
Fernandes et al [22]		√				√	
Shin, et al Approach [33]	√	√		√			
graph transformation approach [34]					√		

However, this study indicates that the transformation approaches have certain weaknesses, such that, each transformation approach uses only a subset of the UML models. Most of these transformations are based on the behavioral UML models, and the tools and techniques discussed in this paper do not cover translation of all different UML models to PN models.

IV. SUMMARY AND FUTURE WORK

OOPN emerged from the combination of the benefit of maintainability and reusability of OO modeling together with the advantages of PN graphical interface and theoretical background. In order to combine the best characteristics of PN and UML design methods, a software tool is required to translate UML models to PN models. This research provides a brief description and comparative study between some of the software tools that support the translation of the UML static and dynamic models to PN models. As a result of this study, this translation is partially supported, and the tools and techniques discussed in this paper do not cover translation of all different UML models to PN models. Extending the existing tools or building a new tool to support a full translation for UML models to PNs is considered as a research issue and future work.

REFERENCES

[1] Michael Zapf and Armin Heinzl, "Techniques for Integrating Petri-Nets and Object- Oriented Concepts," 1999.

[2] Vedran Kordic, *Petri Net, Theory and Applications*. Vienna, Austria, Vienna, Austria : I-Tech Education and Publishing, 2008.

[3] Ruth Sara Aguilar-Sav'en, "Business process modelling: Review and framework," *International Journal of Production Economics*, vol. Vol 90, no. 2, pp. 129-149,

2004.

[4] Toshiyuki Miyamoto and Sadatoshi Kumagai, "Application of Object-Oriented Petri Nets to Industrial Electronics," in *The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Taipei, Taiwan, 2007, pp. 64-69.

[5] E. Bhushan Bauskar and Boleslaw Mikolajczak, "Abstract Node Method for Integration of Object Oriented Design with Colored Petri Nets," in *Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)*, Las Vegas, 2006, pp. 680 - 687.

[6] K Jensen, L M Kristensen, and L Wells, "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems," *International Journal on Software Tools for Technology Transfer*, pp. 213-254, 2007.

[7] Mark Holliday and Mark Vernon, "A Generalized Timed Petri Net Model for Performance Analysis," , vol. 13 (12), 1987, pp. 1297-310.

[8] Nickl Russel, Wil M.P van der Aalst, Arthur H.M ter Hofstede, and Petia Wohed, "On the Suitability of UML 2.0 Activity Diagrams for Business Process Modeling," in *Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling*, vol. 53, 2006, pp. 95-104.

[9] Object Management Group. (February 2009) Documents associated with UML Version 2.2. [Online]. http://www.omg.org/spec/UML/2.2/

[10] G.A Lewis, "Producing network applications using object-oriented petri nets," University of Tasmania, Master Thesis 1996.

[11] Boleslaw Miknlajczak and Charles Sefranek, "Integrating Object-Oriented Design with Petri Nets - Case Study of ATM System," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 2003, pp. 1499 - 1505.

[12] Charles Lakos, "Object Oriented Modelling with Object Petri Nets," in *Concurrent object-oriented programming and petri nets: advances in petri nets*.: Springer-Verlag New York, Inc. Secaucus, NJ, USA , 2001, pp. 1 - 37.

[13] Bassam Rajabi and Sai Peck Lee, "Change Management in Business Process Modeling Survey," in *International Conference on Information Management and Engineering (ICIME 2009)*, vol. 13, Kuala Lumpur,Malaysia, 2009, pp. 37-41.

[14] Bassam Rajabi and Sai Peck Lee, "Runtime Change Management Based on Object Oriented Petri Net," in *International Conference on Information Management and Engineering (ICIME 2009)*, vol. 13, Kuala Lumpur, Malaysia, 2009, pp. 42-46.

[15] Jinzhong Niul, Jing Zou, and Aihua Ren, "OOPN: An Object-Oriented Petri Nets and its Integrated Development Environment," in *Proceedings of IASTED International Conference on Software Engineering and Applications (SEA 2003)*, 2003.

[16] R. Koci, V. Janousek, and F Zboril, "Object Oriented Petri Nets Modelling Techniques Case Study," in *Second UKSIM European Symposium on Computer Modeling and Simulation EMS '08.* , 2008, pp. 165-170.

[17] Zhaoxia Hu and M Shatz, "Mapping UML Diagrams to a Petri Net Notation for System Simulation," in *Procedings of International Conference on Software Engineering and Knowledge Engineering (SEKE'04)*, Banff, Canada, 2004, pp. 213-219.

[18] Luciano Baresi, "Some preliminary hints on formalizing UML with Object Petri Nets," in *Procedings of the 6th World Conference on Integrated Design and Process Technology (IDPT2002)*, Pasadena (USA), 2002.

[19] J Saldhana and SM Shatz, "UML Diagrams to Object Petri Net Models: An Approach for Modeling and analysis," in *International Conference on Software Engineering and Knowledge Engineering*, Chicago, Illinois, 2000.

[20] Chakib Tadj and Toufik Laroussi, "Dynamic Verification of an Object-Rule Knowledge Base Using Colored Petri Nets," *Journal of Systemics, Cybernetics and Informatics*, vol. Vol 4 , No.3, 2006.

[21] Asghar Bokhari and Skip Poehlman, "Translation of UML Models to Object Coloured Petri Nets with a view to Analysis," in *Software Engineering and Knowledge Engineering (SEKE:2006)*, San Francisco, CA, USA, 2006, pp. 568-571.

[22] João M Fernandes, Simon Tjell, Jens Bæk Jorgensen, and Óscar Ribeiro, "Designing Tool Support for Translating Use Cases and UML 2.0 Sequence Diagrams into a Coloured Petri Net," in *Sixth International Workshop on Scenarios and State Machines (SCESM'07)*, 2007.

[23] Binsan Khadka, "Transformation of Live Sequence Charts to Colored Petri Nets," University Of Massachusetts Dartmouth, A Masters Project Report 2007.

[24] J.-E Hong and D.-H Bae, "High-level Petri net for incremental analysis of object-oriented system requirements," *IEE Proceedings of Software*, vol. Vol 148, no. 1, pp. 11-18, 2001.

[25] Feng Xiaoning, Wang Zhuo, and Yin Guisheng, "Hierarchical Object-Oriented Petri Net Modeling Method based on Ontology*," in *International Conference on Internet Computing in Science and Engineering(ICICSE 08)*, 2008, pp. 553 - 556.

[26] Jens Bæk Jorgensen, "Coloured Petri Nets in Development of a Pervasive Health Care System," in *Applications and Theory of Petri Nets 2003, Lecture Notes in Computer Science*.: Springer Berlin / Heidelberg, 2003, vol. Volume 2679/2003, pp. 256-275.

[27] E Go'mez Marti'nez and J Merseguer, "ArgoSPE: Model-based software performance engineering," in *27th International Conference on Application and Theory of Petri Nets and Other Models Of Councurrency*, 2006.

[28] TGI group. Petri Nets Tool Database. [Online]. http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/complete_db.html

[29] WebSPN 3.3 Web-accessible non Markovian Petri net tool. [Online]. https://mdslab.unime.it/webspn/

[30] MARTA EUNICE CALDERON, "Model Transformation Support For The Analysis Of Large Scale Systems," Texas Tech University, Master Thesis in Software Emgineering 2005.

[31] LoLA - A Low Level Petri Net Analyser. [Online]. http://wwwteo.informatik.uni-rostock.de/ls_tpp/lola/

[32] Esther Guerra and Juan de Lara, "A Framework for the

Verification of UML Models. Examples using Petri Nets," in *Proceding VIII Jornadas Ingenier´ıa del Software y Bases de Datos (JISBD 2003)*, 2003, pp. 325-334.

[33] Michael E Shin, Alexander Levis, and Lee Wagenhals, "Analyzing Dynamic Behavior of Large–Scale Systems through Model Transformation," *International Journal of Software Engineering and Knowledge Engineering(IJSEKE)*, 2005.

[34] Yu Zhao et al., "Towards Formal Verification of UML Diagrams Based on Graph Transformation," in *IEEE International Conference on E-Commerce Technology for Dynamic E-Business*, Beijing, 2004, pp. 180-187.

Bassam Atieh Rajabi received his B.S. degree in Computer System Engineering from Palestine Polytechnic University, Hebron, Palestine, in 2001 and the M.Sc. degree in Computer Science from Alquds University, Jerusalem, Palestine, in 2005. Currently, he is a PhD student in Computer Science at University Malaya, Malaysia. From 2001 to 2004, he was a Research and Teaching Assistant with the Computer Science Department, Alquds University, Jerusalem, Palestine. From 2001 to 2005 he was a Lecturer with the Computer Science Department, ORT College, Jerusalem, Palestine. He was a Lecturer and Dean Assistant for Administrative Affairs from 2005 to 2008 with Wajdi Institute of Technology, Jerusalem, Palestine. His areas of interest are Software Design and Modeling Techniques.

Sai Peck Lee is a professor at Faculty of Computer Science & Information Technology, University of Malaya. She obtained her Master of Computer Science from University of Malaya, her Diplôme d'Études Approfondies (D. E. A.) in Computer Science from University of Pierre et Marie Curie (Paris VI) and her Ph.D. degree in Computer Science from University of Panthéon-Sorbonne (Paris I). Her current research interests include Software Reuse, Application and Persistence Frameworks, Requirements and Design Engineering, Object-Oriented Techniques and CASE tools. She has published more than 80 research papers in local and international journals and conferences.