Transformation of Coloured Petri Nets to UML 2 Diagrams

Ayman Yassin and Hoda Hassan

Computer Science, Arab Academy for Science and Technology, Cairo, Egypt
Ayman.Yassin@Gmail.com, huda.hassan@aast.edu

Abstract. Business Process Modeling Notation is used by business modelers to model business processes logic and artifacts. However, it is inadequate in expressing the execution semantics of business processes and takes a process-oriented approach for modeling systems. UML, on the other hand, is known for its expressiveness to present the object-oriented approach for modeling software-based system. There is a rising need to transform business process models to flawless UML models. This paper proposes a modeling transformation technique for transforming a business process-modeling notation model to different UML diagrams, using Coloured Petri Nets (CPN) as a formal intermediate step to ensure flawless transformation. This transformation would allow modeler to take advantages of the presentation power in BPMN as well as the implementation power in UML. Furthermore, this step will bridge the gap between the different modeling notations previously mentioned.

Keywords: Business Process Modeling Notation, Coloured Petri Nets, UML, Transformation, Use Cases Diagram, Activity Diagram.

1 Introduction

BPMN is one of the most popular Business Processes modeling language nowadays. BPMN is more suitable for business process modeling due to its high presentation power [1, 2]. With A Business Process Modeling Diagram, modelers can simulate the business artifacts to know the bottlenecks in the system being visualized, as well as identify the metrics of the system under development [3, 4].

BPMN is a Process Centric Approach that lacks the execution semantics required for process modeling implementation [2, 4]. In that respect, UML, which is an object oriented modeling language, is more suitable and preferred by software engineers [2, 4]. Yet, most of the stakeholders come from a non-technical background, and are ALIEN towards UML notations and the different OOP constraints that the language expresses. Furthermore, the UML model generated based on the BPMN process model is usually validated at a late stage by the business modelers leading to late detection of flaws, which results in more effort and cost to correct [5].

Therefore, there is a pressing need to bridge the gap between the modeling tools used by business modelers (BPMN) and that used by the software engineers (UML),

Á. Rocha et al. (eds.), *New Perspectives in Information Systems and Technologies, Volume 2*, Advances in Intelligent Systems and Computing 276,
DOI: 10.1007/978-3-319-05948-8_13,

while ensuring flawless transformation from one tool to the other. Defining formal semantics for business process modeling notation (BPMN) using Coloured Petri Nets (CPN) helps the modeling tools vendors to automatically validate the business model. Further, it helps business analysts simulate the business process behavior to enable detection of flaws [5].

In this paper, we propose a model to transform a formal CPN model transformed and mapped from BPMN model to different common UML diagrams as shown in Fig. 1. This would allow modeler to be able to take advantages of both the presentation powers in BPMN and the implementation power in UML. Accordingly, business modelers will only have to focus on their expertise in process modeling, while software engineers will only have to focus on implementation tasks. Furthermore, it will help to bridge the gap between the formal model represented by CPN and the semi-formal model represented by UML. The transformation will be based on one-to-one mapping. Our focus in this paper will be on the mapping of CPN to use cases and activity diagrams, which are considered two of the common dynamic UML models.

1.1 Coloured Petri Nets

According to [5] Coloured Petri Net (CPN) is a graphical language for constructing models of concurrent systems and analyzing their properties. CPN is a discrete-event modeling language combining Petri Nets and the functional programming language CPN ML that is based on Standard ML. Standard ML provides the primitives for the definition of data types, describing data manipulation, and for creating compact and parameterized models. CPN is the most well known kind of high-level Petri Nets. CPN incorporates both data structuring and hierarchical decomposition - without compromising the qualities of the original Petri Nets.

A CPN model consists of data, places, transitions and arcs. Data are defined as data types, data objects and variables that hold data values. A place is a location for holding data. A transition is an activity that transforms data. An arc connects a place to a transition to specify data flow paths. A CPN model uses data types, data objects and variables. A CPN data type is called a color set such as integer, real, string, and Boolean that is available in a computer language [6, 9].

1.2 UML

Unified Modeling Language (UML) is a standardized general purpose modeling language in the field of object-oriented software engineering. UML is maintained and developed by the object management group (OMG). UML is widely adopted and considered as de facto standard for software modeling and design. Thus, UML provides us with an integrated way to specify the problems, facts, requirements, or software structure, using a set of expressive UML diagrams. Since recent large-scale software development focuses more on modeling than on programming, modeling is one of the most important key activities for successful software projects [4]. UML diagrams can be divided into two groups, namely static or structural diagrams and dynamic or behavioral diagrams.

Use cases diagram is one of the most common UML behavioral diagrams. Use cases diagrams are a means for specifying required usages of a system. Typically, they are used to capture the requirements of a system. The key concepts associated with use cases diagram are actors, use cases, and the subject. The users and any other systems that may interact with the subject are represented as actors. The required behavior of the subject is specified by one or more use cases, which are defined according to the needs of actors. An instance of use case refers to an occurrence of the emergent behavior that conforms to the corresponding use case type. Such instances are often described by interaction specifications [7].

UML provides mechanisms for reusing and adding on to use cases and actors. Actor capabilities can be expanded or can replace entire use cases using generalization. Use cases common elements can be factored by using included use cases, or can be added on to base use cases using use case extension [8].

UML 2 activity diagrams are important structured visual modeling notations useful for describing different types of behavior found in computer and information systems.

UML 2 activities are suitable for modeling the diverse requirements of many traditional scenarios. Activities provide for visual modeling that can be easily understood. UML 2 activities are based on Petri Net like semantics [11].

An activity diagram is a directed graph that consists of actions and flows. The basic structure of an activity model is similar to that of a Petri net, however there are supplementary model components provided in UML, which CPN does not provide. Those components include initial/final nodes, fork/join nodes, and decision/merge nodes. In addition to those components, there could exist expansion regions in UML, which manipulate collections of input data [4].

The choice of using UML diagrams as the target model in this paper is motivated by the fact that UML is one of the most widely used industry-standard modeling tools for software development. Furthermore, the proposed model attempts to bridge the gap between the modeling tool used by the business modelers (BPMN) and that used by the software engineers (UML), with the early detection of the business process flaws using the semantic model of CPN as shown in Fig. 1. The mapping of BPMN model into its corresponding CPN model was proposed in [5].

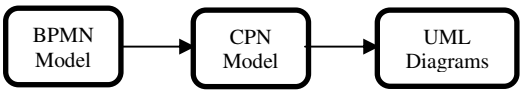

Fig. 1. The transformation process from BPMN to UML

The rest of this paper is organized as follows. Section 2 presents the mapping from CPN to UML diagrams. Examples of such mapping are shown in section 3. Finally, Section 4 and Section 5 discuss the related work and the conclusion respectively.

2 The Mapping of CPN to UML Diagrams

The proposed mapping process aims to extract the use cases diagram and the activity diagram. In particular, activity diagrams can be used to describe the dynamics of the system while use cases diagram can be used to describe the functionality of the system components. This will produce formal and structured use cases that can be used directly in the use case driven approach [10].

2.1 Mapping CPN to Use Cases Diagrams

In UML, use case diagram shows actors, use cases and their dependency relationships. In this paper, all relationships between the actor and the use case and between different use cases have been considered. A CPN model can be transformed to the use case diagram using the mapping process shown in Fig. 2 that should be applied in order and taking in consideration the following:

- As the transition of the CPN expresses a task, action or computation thus, it will be mapped to a use case and consequentially, CPN place will be mapped to an actor as shown in Fig. 3(a) and 3(b).
- The additional information of CPN (e.g. the transition guard and the arc expressions) will be mapped to the use case pre-conditions.
- The color function of an initial place, which is the place with no-inbound arcs, will be also mapped to the use case pre-condition.
- A final place is the place with no outbound arcs, and its color function represents the post-condition or the results of alternative steps within the use case.
- The predicate of the transitions will be also mapped to post-conditions.
- When a marked token is in a place and connected to a transition this means that the place is interacting with the transition. This place will be mapped to the caller actor and the trigger of the use case. The transition firing with this place means that the use case is ready for its task and will return to the caller actor after execution as shown in Fig. 3(c).
- The mapping from CPN to the use cases dependences (e.g. include and extend) based on the token functions, arc expressions, transition predicate and the multiple fusions of CPN. An include relationship is a relationship in which one use case (the base use case) includes the functionality of another use case (the inclusion use case). Include relationship can be extracted from CPN as shown in Fig. 3(d) such that P1, T2 represents use case (UC1) and P2, T2 represents use case (UC2). T1 output arc with its arc expression marking P2 with the required token and value, which are required for firing the transition T2. Respectively, an extend relationship is used to specify that one use case (extension) extends the behavior of another use case (base) as shown in 3(e) such that when UC1 is being executed, the execution of UC2 is optional. The condition of this selection is assigned to transition T1.

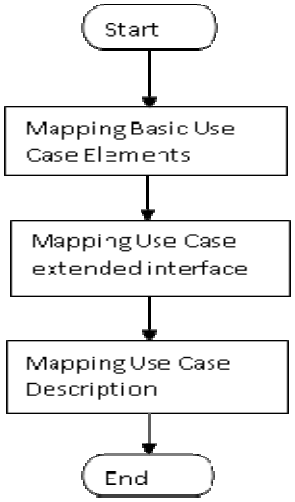

Fig. 2. The mapping Process of Use Case

Fig. 3(a). Mapping CPN Places

Fig. 3(b). Mapping CPN Transition

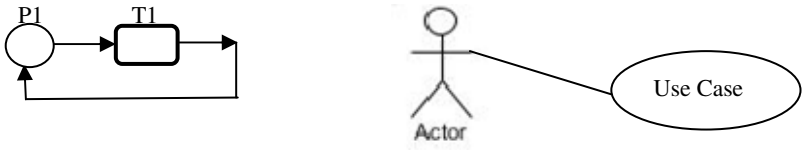

Fig. 3(c). Mapping CPN to Simple Use Case

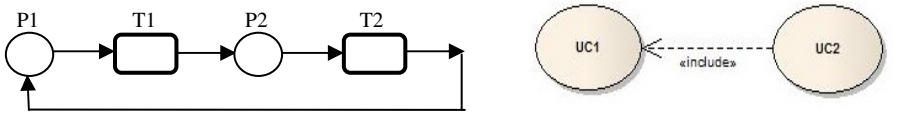

Fig. 3(d). Mapping CPN to Use Cases Include Relationship

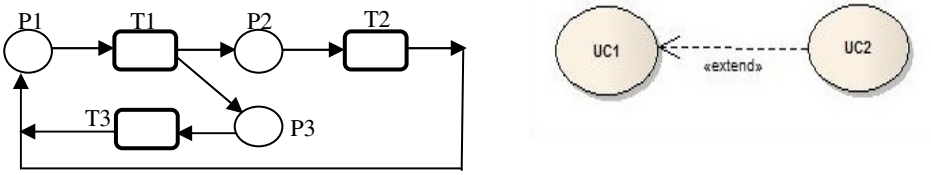

Fig. 3(e). Mapping CPN to Use Cases Extend Relationship

2.2 Mapping CPN to Activity Diagrams

In order to transform CPN to activity diagram (AD) the proposed rules and the idea presented by [12] is applied with a little adjustment while, using the reverse engineering of the rules. In all presented diagrams character P denotes places, character m denotes arc expression and character F denotes transition fusion. We cannot elaborate more description due to the limited paper size.

1) General CPN to Activity

In general, each CPN transition is mapped to an action. A place with a single inbound and a single outbound arcs is mapped to a control flow. The activity diagram presented in Fig. 4(a) represents an interaction without high-level operators. There are two Lifelines and one message between them. Each action represents activity state.

2) CPN to Activity (forming fork)

CPN transition branched to parallel transitions are mapped to correspondence AD parallel interaction as shown in Fig. 4(b). Fork node is a control node that splits a flow into multiple concurrent flows. A fork node has one incoming edge and multiple outgoing edges.

3) CPN to Activity (forming join)

CPN parallel transitions combined into one transition later are mapped to two parallel ADs combined later into one AD that denotes the end of parallel processing as shown in Fig. 4(c). Join node is a control node that synchronizes multiple flows with multiple incoming edges and one outgoing edge.

4) CPN to Activity(forming decision)

CPN with transition condition function is mapped to AD decision. AD decision node accepts tokens on an incoming edge and presents them to multiple outgoing edges. Which of the edges is actually traversed depends on the evaluation of the guards on

the outgoing edges as shown in Fig. 4(d). The AD represented shows that if condition Action1, then do action 2, else do action 3. This signifies If-Else statement.

5) CPN to Activity (forming merge)

CPN transition with merging places is mapped as transition from two Parallel Activities to one Activity using merge as shown in Fig. 4(e).AD Merge node is a control node that brings together multiple alternate flows. It is not used to synchronize concurrent flows but to accept one among several alternate flows. A merge node has multiple incoming edges and a single outgoing edge.

6) CPN to AD Looping Transition

AD Looping Transition node is a structured activity node that represents a loop with setup, test, and body sections. Looping occurs in second condition of decision i.e. in Action3 indicates that While condition true do Action 3 as shown in Fig. 4(f).

7) CPN to AD Precedence Transition

AD Precedence means Action 1 should precede action 3 as shown in Fig. 4(g).

8) CPN to AD Timing Transition

As shown in Fig. 4(h) the result value contains the time at which the occurrence transpired. Such an action is informally called a wait time action.

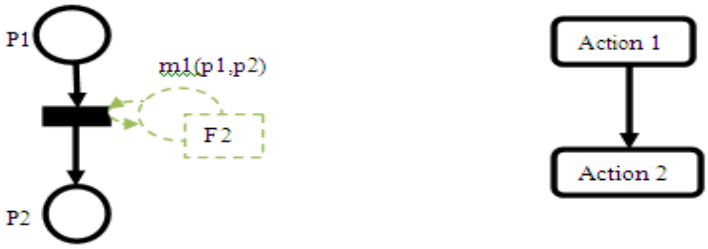

Fig. 4(a). General CPN to Activity

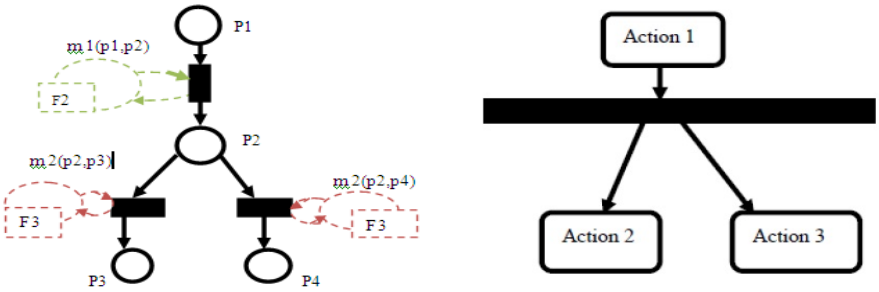

Fig. 4(b). CPN to Activity (forming fork)

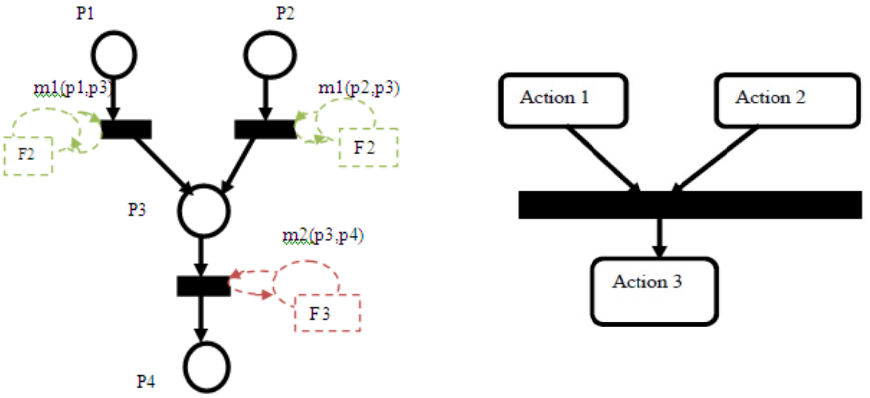

Fig. 4(c). CPN to Activity (forming join)

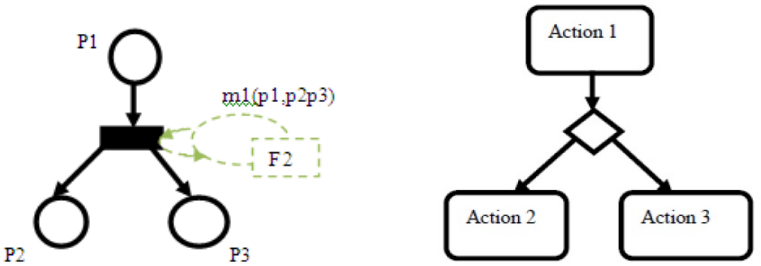

Fig. 4(d). CPN to Activity (forming decision)

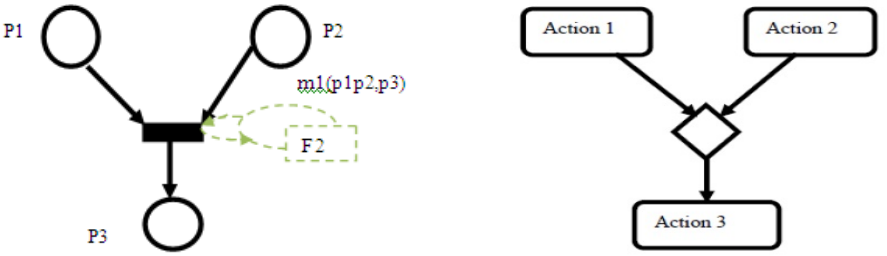

Fig. 4(e). CPN to Activity (forming merge)

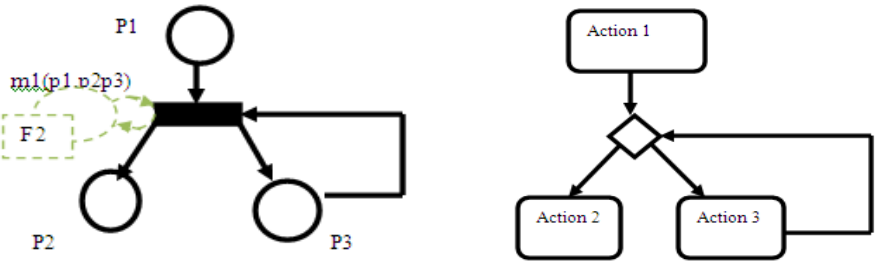

Fig. 4(f). CPN to AD Looping Transition

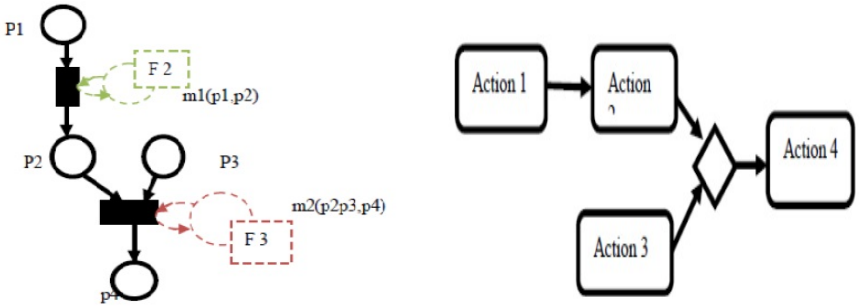

Fig. 4(g). CPN to AD Precedence Transition

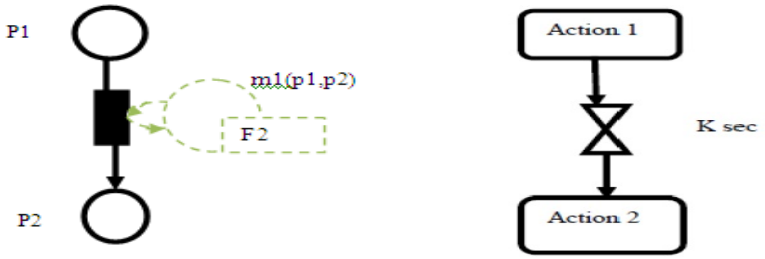

Fig. 4(h). CPN to AD Timing Transition

3 Example

The coloured petri net in Fig. 5 models a part of an automatic teller machine (ATM) with a bank at the backend. When the ATM is in the ready state, a client can ask for a certain amount of money. The ATM communicates this amount to the bank and waits for approval. When the approval arrives, the money is given to the client. Meanwhile,

the requested amount is deducted from the client's account. If the account is deficient, approval will not be granted. In the model, a rejection message would be sent from the bank to the ATM, leading to an error message from the ATM to the client. Using the mapping from CPN to use case and activity diagrams described in the previous section, ATM System process in Fig. 5 is mapped to the use case diagram in Fig. 6 In addition, the activity diagram Fig. 7 is produced.

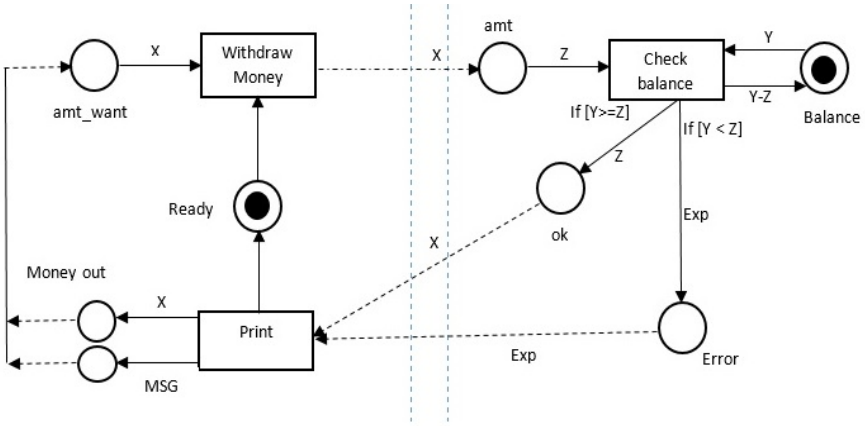

Fig. 5. CPN ATM Example

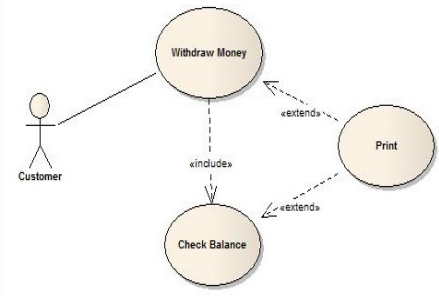

Fig. 6. Use Case for ATM Example

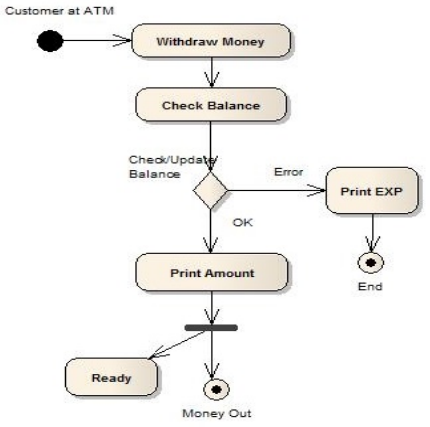

Fig. 7. Activity Diagram for ATM Example

4 Related Work

Based on our knowledge most of the researches and approaches are transforming and presenting different UML diagrams notations to Petri Nets or Coloured Petri Net (CPN) for simulation, verification and validation purposes depending on its formality and analysis capabilities in contrast to the work proposed in this paper. There are two mainstream approaches to transform UML constructs into CPN, which are informal approaches and formal approaches. Not all Approaches use CPN.

In [11], the authors present the transformation of UML 2.0 activity diagram into Petri Nets and Coloured Petri Nets. One of the formal mappings proposed was using the triple graph grammar (TGG). Although TGG can represent forward and backward transformation but the set of rules used in the backward transformation is not the same as those used in the forward one. Accordingly, in complex models the transformation presented will require the application of different set of rules recurrently.

In [4], the authors propose a complementary modeling process to UML modeling, which is used to keep the consistency between heterogeneous UML models based on CPN. The proposed informal transformation model is abstract as the focus was on preserving the consistency between UML models.

In [13], the authors propose a simple rule based bi-directional transformation of UML 2 activity diagram to Petri Net. In that transformation, the rules can be implemented in a generic informal manner or formally using triple graph grammar (TGG) notations. However, the idea presented was not validated through implementation. In addition, the proposed mapping is to Petri Net not to CPN. The backward transformation needs elaboration on the rules required.

In [14], the authors propose an algorithm to transform a software architecture described by use case, sequence and component diagrams into an executable model based on different extensions of Petri Nets. This work tries to fill the gap between software architect and non-functional requirement analyst. However, the idea presented in the mapping of use case diagrams is to Petri nets not CPN that yields to extra places and transitions in the Petri Nets. In addition, the idea did not present clearly how the Use case additional information (e.g. pre-conditions, post-conditions and trigger) are mapped to its corresponding Petri Nets.

5 Conclusion

This paper presents a formalization and a mapping process of Coloured Petri Nets (CPN) model to its corresponding UML models with respect to use case and activity diagrams. The proposed process is a step towards bridging the gap between different modeling notations, as that exists between BPMN 2.0 and UML 2. BPMN is used by business process modelers, and UML 2.0 is used by software engineers for further software development. The transformation proposed uses CPN for validation and simulation of the previously mentioned models. This helps vendors of modeling tools to validate the business process model automatically and generate the corresponding software UML model. Future work will aim to better tune the proposed mapping

rules, transforming CPN to other common UML diagrams and build an automated tool for this transformation. Such automation should help to reduce the communication time between business process modelers and software engineers as well as, to synchronize business process with implementation seamlessly.

References

1. White, S.A.: Introduction to BPMN. IBM Corporation
2. Bào, N.Q.: A proposal for a method to translate BPMN model into UML activity diagram. Vietnamese-German University, BIS 2010 (2010), 2010,contact@nqbao.com
3. Birkmeier, D.: An Empirical Comparison of the Usability of BPMN and UML Activity Diagrams for Business Users. University of Augsburg, Germany, dominik.birkmeier@wiwi.uni-augsburg.de
4. Shinkawa, Y.: Inter-Model Consistency in UML Based on CPN Formalism. Faculty of Science and Technology, Ryukoku University, 1-5 Seta Oe-cho Yokotani, Otsu 520-2194, Japanshinkawa@rins.ryukoku.ac.jp
5. Ramadan, M.: BPMN Formalization using Coloured Petri Nets. Computer Science. Arab Academy for Science and Technology, Cairo, mohamed.e.ramadan@aast.edu
6. Shin, E., Levis, A.H., Wagenhals, L.W.: Transformation of UML-based System Model to Design/CPN Model for Validating System Behavior, Michael. Department of Computer Science, Texas Tech University, Lubbock, TX 79409-3104, USA, Michael.Shin@coe.ttu.edu
7. OMG Unified Modeling LanguageTM (OMG UML), Superstructure Version 2.4.1 (August 2011)
8. Pilone, D., Pitman, N.: UML 2.0 in a Nutshell A desktop Quick Reference. O'REILY (2005)
9. Jensen, K.: Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use, vol. 1-3. Springer (1992, 1994, 1997)
10. Jacobson, I.: Object-Oriented Software Engineering, A Use Case Driven Approach. Addison-Wesley (1992)
11. Staines, A.S.: Intuitive Transformation of UML 2 Activities into Fundamental Modeling Concept Petri nets and Colored Petri Nets. University of Malta, Malta
12. Agarwal, B.: Some Rules to Transform Activity Diagrams into Colored Petri Nets. International Journal of Recent Technology and Engineering (IJRTE) I(5) (November 2012) ISSN: 2277-3878
13. Spiteri Staines, A.: Rule Based Bi-Directional Transformation of UML2 Activities into Petri Nets. International Journal of Computers 5(2) (2011)
14. Emadi, S., Shams, F.: A new executable model for software architecture based on Petri Net. Indian Journal of Science and Technology 2(9) (September 2009) ISSN: 0974- 6846; Computer Engineering Department, Science Research Branch, Islamic Azad University, Tehran, Iran, Computer Engineering Department, Shahid Beheshti University, GC, Tehran, Iran emadi@srbiau.ac.ir1; f_shams@sbu.ac.ir2