Improving Relationship between UML and Petri Nets for Analyzing System by Applying Object Oriented Petri Nets

Meysam Aminzadeh
Department of computer engineering, Mazandaran University of Science & Technology, Babol, Iran
E-mail: Aminzadeh_m_t@yahoo.com

Hamidreza Salmani Mojaveri and Somayeh Shafeiha
Master of Business Administration, Science and Research Branch, Islamic Azad University, Tehran, Iran.
E-mail: Mazisalmani@yahoo.com, Somayeh_shafei@yahoo.com

***Abstract*— Today, by developing technology and presenting Object-Oriented and Concurrent systems, new modeling languages with powerful mathematical and formulaic base are needed. UML as an Object-Oriented modeling language is needed a powerful mathematical and formulaic base for its symbols, besides, Petri Nets as a language for Concurrent systems need to have symbols for representing Object-Oriented models. In many complex systems, model presenting by Petri Nets caused model complexity and designer's perplexity and also due to wide changes in such systems, its part by part presenting by means of UML diagram is not possible. The aim of the paper is to present interface model called Object Oriented Petri Nets and its relevant software for converting Petri Nets complex model to various UML diagrams, in order to benefit from advantages of Petri Nets and UML model. In this model, Object Orienting main parts such as Object, Class, Encapsulation and Inheritance are presented with special symbols**.

***Index Terms*—Petri Nets, UML, Object Oriented Petri Nets, Class, Inheritance.**

I. Introduction

Petri Nets have been a popular technique for describing concurrent and parallel systems since their invention in 1962 [1]. They can be used to intuitively describe the dynamic properties of concurrent systems and the restriction for the allocation of the system resources. Petri Nets also feature a graphical interface that is easy to represent and understand[9]. With the popularization of Object Oriented (OO) theory over the 1980s, the Petri Net community also appealed to OO concepts in that object orientation features abstraction, encapsulation, and inheritance, which may help to build layered Petri Net models instead of "flat" ones [2]. Until recently, a tremendous number of Petri Net modeling languages have emerged embodying more or less OO concepts, such as G-Net [3, 4], COOPN [5], LOOPN [6], PROT [7], OOCPN [8]. The efforts indeed helped to avoid state explosion to some extent; however, problems still remain. Some languages claim to be object-oriented simply by regarding token as objects, and some otherwise require profound mathematical knowledge [9].

Unified Modeling Language (UML) specifies a modeling language that incorporates the object-oriented community's consensus on core modeling concepts. The behavioral specifications in UML are based on State Charts [10]. A Statechart diagram models the behavior of a single object over its lifetime [11]. An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages (or events) that may be dispatched among them [12].

In this paper, we have presented a specific model of Object Oriented Petri Nets and simulated software for displaying relevant symbols in order to convert complex models from variety of Petri Nets such as colored Petri Nets, time Petri Nets and classic Petri Nets to UML diagrams, by the help of this model. Converting these complex models to UML diagrams have one benefit: system designer or system analyzer encounter with a complete object oriented display of a system and the analyzing process is easier, moreover any part which is designed by Petri Nets had strong mathematical base and this conversion also give this feature to presented Petri Nets model. In addition, if a part of a system is changed, this is only enough to determine the changed part of the system from Object Oriented model, then by entering to its details (which is hidden by a symbols of the model) desired changes are implemented without section by section investigating of the entire system. This is caused designers' perplexity and wasting time. In the second part, we have introduced suggested model and we review its symbols and its structure. In the third part, we review simulated software which is relevant to suggested model symbols. With one example, we convert a model which is designed by Petri Nets, to UML diagram by the help of suggested Object Oriented Petri Nets. Finally, we should deal with the conclusion and future works.

doi:10.4304/jait.4.1.1-7

II. Object Oriented Petri Net Model (OOPN)

For simplifying drawn models by means of all kinds of Petri Nets, special symbols has presented in table 1. Moreover, Using of this symbol has another benefit and it is that easy conversion of these symbols to UML symbols. In this suggested model we tried to put 4 main features of Object Oriented language in order to face with a complete Object Oriented model.

According to the former notations and some notations such as place, transition and arc the following structure could be presented for the proposed model:

The OOPNS structure is a 7-tuple

C = (P, T, OB, E, IN, I, O):

P is a finite set of places (P = {p1, p2, …, pn}),

T is a finite set of transitions (T = {t1, t2, …, tn}),

P and T are disjoint ($P \cap T = \varnothing$)

OB is a finite set of object (OB = {ob1, ob2, …, obn}),

E is a finite set of encapsulation (E = {e1, e2, …, en}),

IN is a finite set of inheritance (IN = {in1, in2, …, inn}),

I: is the input function:

O: is the output function:

The input and output functions in general are $T \rightarrow Pn$ and as soon as creation of notations as OB, E and IN, it may change[13].

A. Notation

The significance of object-oriented technology is that it enables programmers to design software in much the same way that they perceive the real world [10]. All object-oriented programming languages provide mechanisms to allow you to perform the model of your Object-oriented. This model includes: encapsulation, class, inheritance.

Encapsulation

Encapsulation is a major feature of OO, and only the public methods or attributes of an object are accessible to others; however, OO does not impose constraints upon the access, such as the order of several method invocations, which thus complicates the relationships amongst objects[9].

Class and object

According to OO theory, the state of an object is made up by the values of all its attributes. State transformations occur when any of such values changes; OO, however does not provide facilities for representing the state transformations according to the inherent application logic[9]. A class may have multiple instances or objects and a class may have attributes and each object of this type is associated with its own corresponding values [9].

A class could be simple or composite. A simple class is defined by a subnet, whereas a composite one is in turn made up of several smaller classes. An OOPN system consists of multiple OOPN classes with instances specified that may pass messages to one another.[9]

By considering introduced symbol for Class concept, a class input or output can get 4 bellow modes which are displayed in table 2[13].

Inheritance

Inheritance is usually regarded as the unique characteristic that distinguishes OO [14], Like various programming languages, where inherited methods are automatically available in a subclass, every OOPN subclass receives the Petri Nets structure of its superclass, and hence the internal behavior and external interface. Since every OOPN object is also regulated by

meaningful sequential behavior, more detailed specification is necessary to define the semantics of inheritance rather than merely specifying a superclass.[9]

TABLE I.
NOTATION WHICH IS ILLUSTRATED BY THEIR PROPERTIES

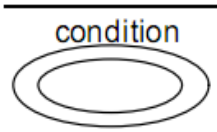
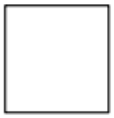
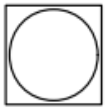
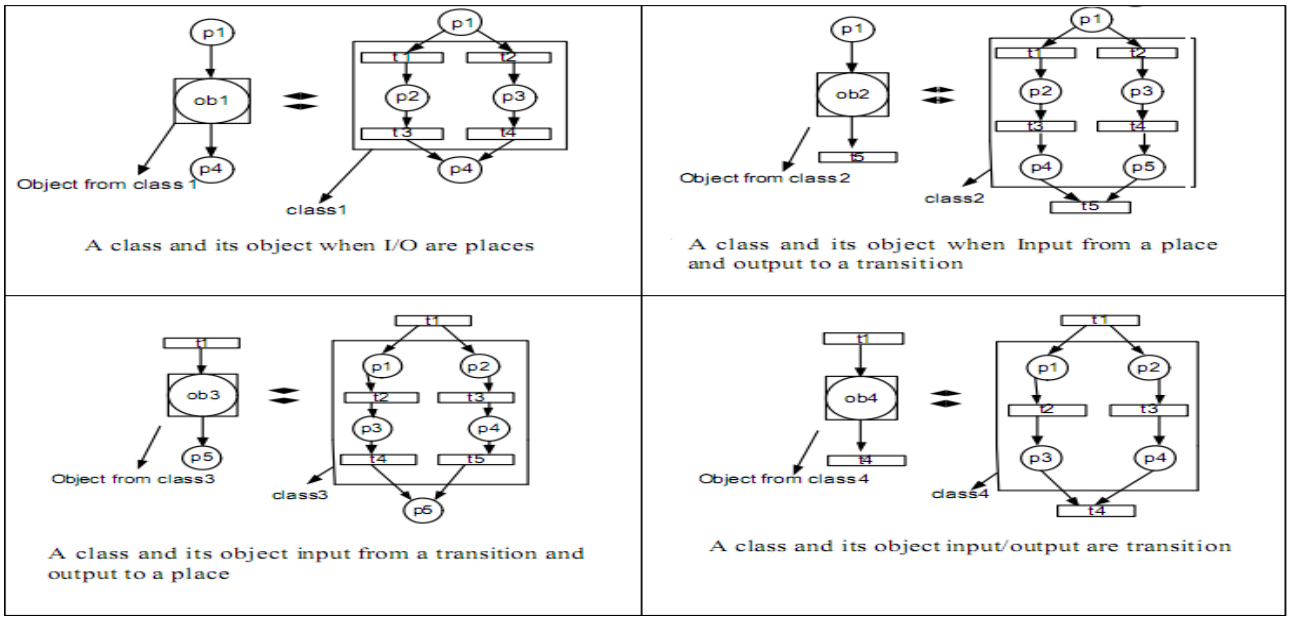
			
To show the encapsulation specification	To show the class specification	To show the object specification	To show the inheritance specification

TABLE II.
FOUR MODEL OF IN PUT / OUT PUT CLASS



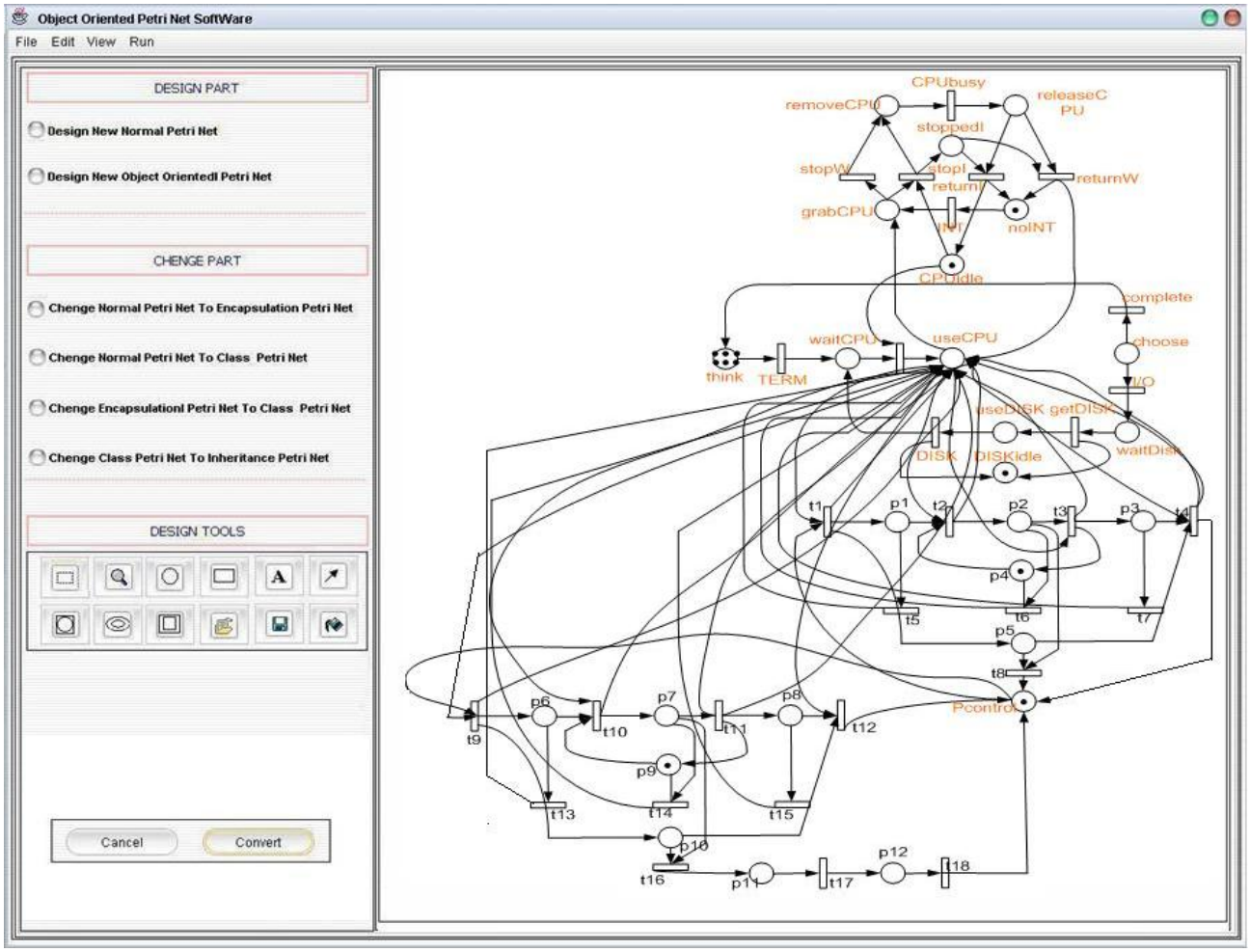


Figure1.a

Figure1. Compare before and after of using notation

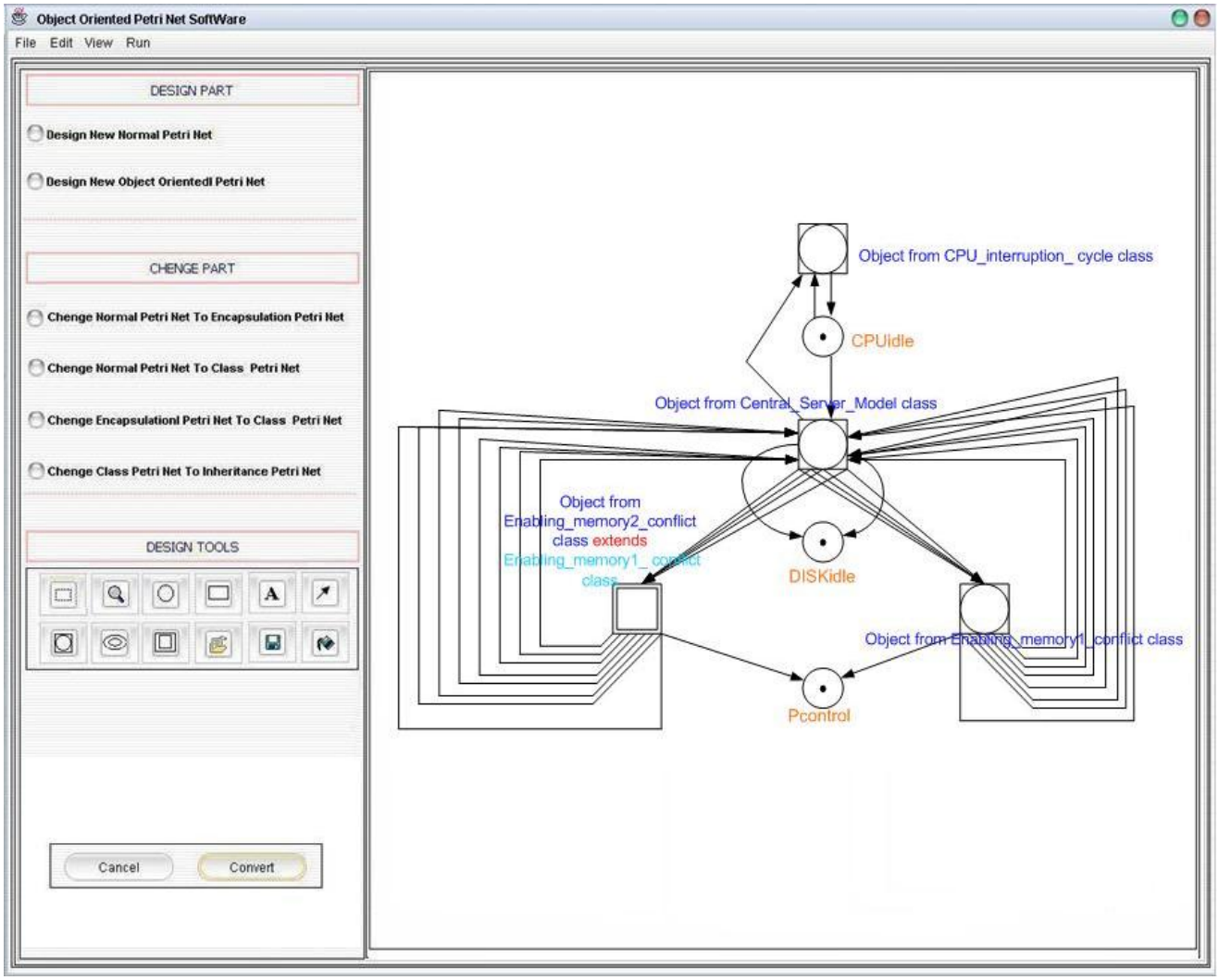


Figure1.b

Figure1. Compare before and after of using notation

III. Object Oriented Petri Net Software

Models which are designed by Petri Nets are displayable and implementable through deferent softwares. In this paper for presented model, we also have presented simulated software. This software is deferent from other existing softwares; this software could only convert one Petri net in deferent level to Class diagrams in UML and unable to have step by step implementation. The suggested simulated software is formed from two sections. The First section is designing and the second section is conversion. In the first section, designer draw system, which may be drawn with deferent types of Petri Nets (colored, classic and time Petri Nets), by using of a classic Petri Nets symbols on software, then by the help of second section and relevant symbols, designer convert model to suggested Object Oriented Petri Nets model.

By using of figure 1.a and 1.b, we have compared the benefit of these conversions. Figure 1.a is a Petri Nets model before conversion. As you see in the figure, model which is presented via Classic Petri Nets were a complex model and a little change on it, caused perplexity and wasting time. By selecting some parts from Classic Petri Nets and convert them by the help of its symbols, Object Oriented Petri Nets model make a model that is easier and more understandable. This model is shown in fig 1.b.

By clicking on each symbol of this Object Oriented model, hidden part is visible. This matter is shown in figure 2.

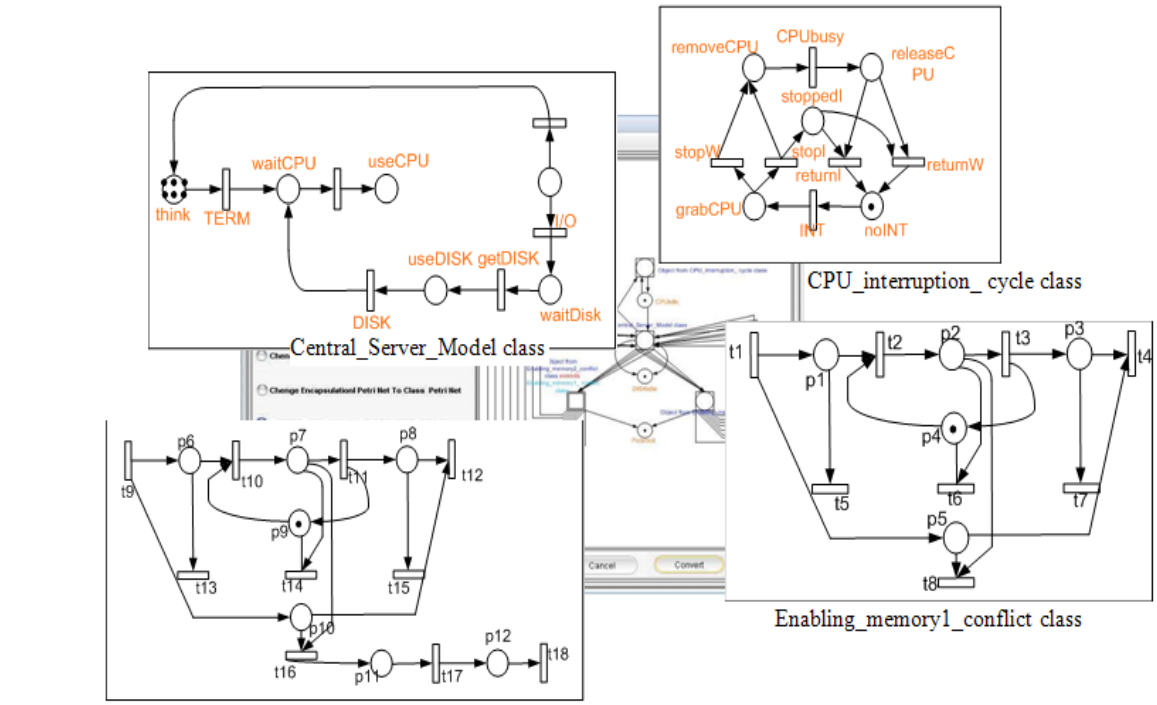


Figure2: Clicking on each symbol to detect hidden par of notation

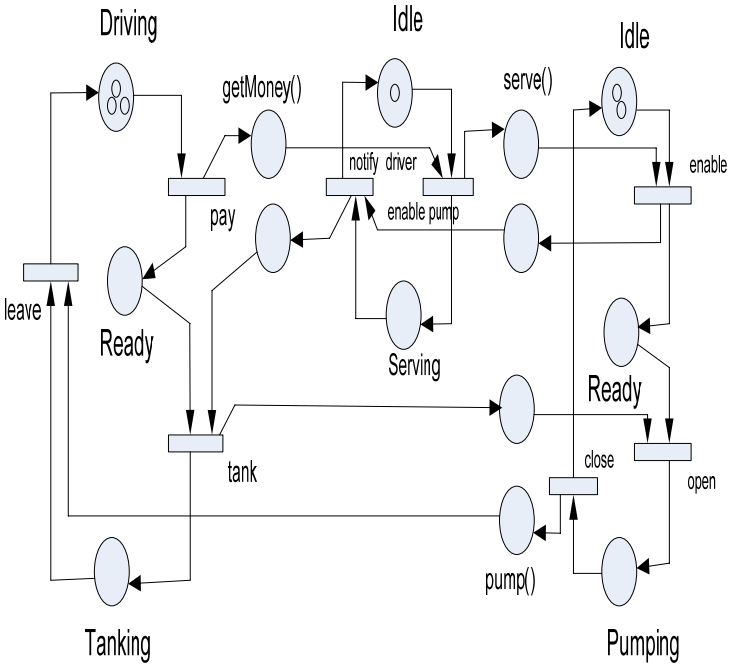


Figure3.a

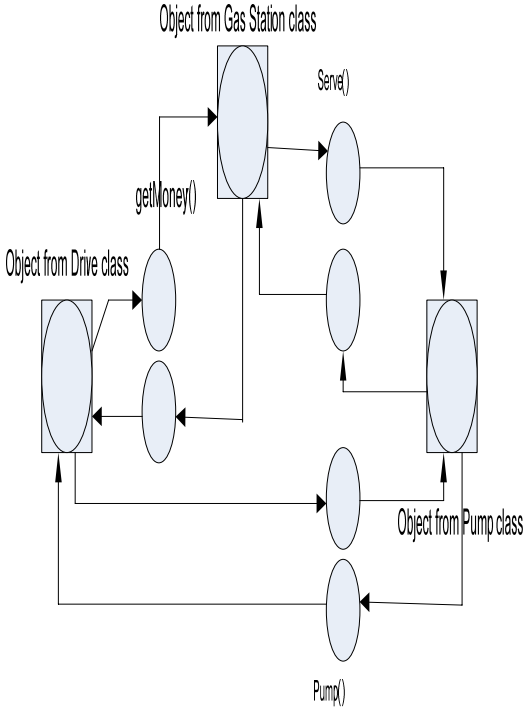


Figure3.b

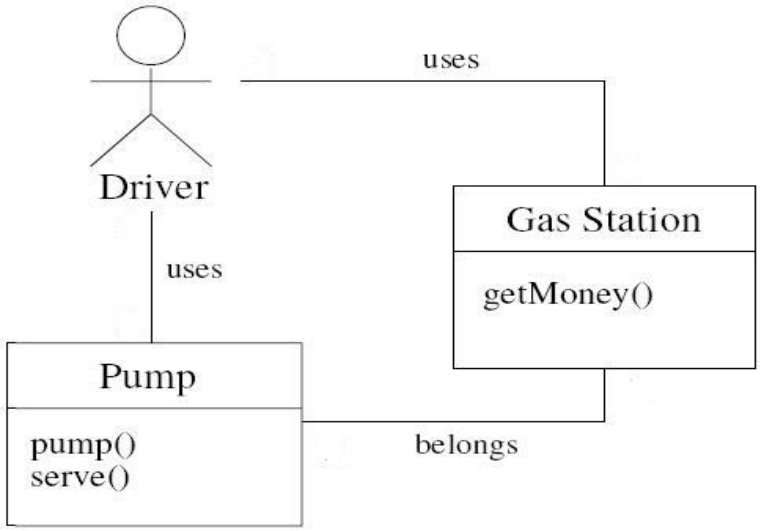


Figure3.c

Figure3: Converting Pn To UML With Oopn

Converting Object oriented symbols to UML symbols, makes designer's Petri Nets model and UML model close together. This matter is shown in figure 3. This figure shows a gas station. Figure 3.a is a model that is designed by the help of Petri Nets. Figure 3.b is a model that is presented and changed by Object Oriented Petri Nets and figure 3.c shows UML model which is obtained from Object Oriented Petri Nets.

IV. CONCLUSION

By converting a classic Petri Nets to UML diagrams, we will be able to not only enjoy the benefits of both of them, but also cover the disadvantages of each of them. We can consider suggested model as a model that have benefited from a Petri Nets model and UML diagrams all together. The advantages of Petri Nets model are:

1. We can display it in the graphical form which makes the method attractive.
2. It is not designed for any specific system type and it is multipurpose model that appropriate for all systems.
3. They have little components but they are effective that make easier to work with it.
4. It is possible to define hierarchy by Petri Nets; they make a big Petri net from linking several small Petri Nets.

Disadvantages:

1. Petri Nets model is very big complicated system that this issue makes the complicated system analysis difficult.
2. This is impossible to convert all the UML diagrams to Petri Nets.

The advantages of UML are:

UML is a non-proprietary modeling and design tool that offers an Object-Oriented modeling framework. In automation, methodological aspects of UML have been used within several applications for modeling as well as they have been used for simulation and implementation purposes: modeling and implementation of object relational databases for the traceability in batch process [15]; specification and validation of scheduling policies in agile production systems [16]; modeling and implementing production control systems [19]; designing and implementing simple network management protocol agents for remote control [21]; modeling and optimizing DCSs based on the industrial bus CAN [20]; modeling and validation of mechatronics systems [17]. The Object-Oriented approach is usually considered a basic principle for many modeling and also analysis and design methodologies in various engineering areas, and UML is considered a "drastic way to revolutionize and improve the deficiency of software development process", allowing its easy modification, extension and maintenance [18].

By considering the benefits of two methods, we observed that presented model can have the benefit of both two methods and as an interface model easily have a role in converting Petri Nets model to UML diagrams. We couldn't convert UML diagrams to Petri Nets models by the help of suggested model and this is one of the disadvantages of this model. This idea is considered as a future works; moreover we are trying to make software for implementing suggested Object Oriented Petri Nets.

REFERENCES

[1] T. Murata, Petri Nets: Properties, Analysis and Applications, Proc. of the IEEE, 77(4), 1989, 541-580.
[2] Michael Zapf, Armin Heinzl, Techniques for Integrating Petri-Nets and Object-Oriented Concepts, Working Papers in Information Systems, University of Bayreuth, 1999.
[3] Yi Den, S. K. Chang, Jorge C. A. de Figueired, Angelo Perkusich, Integrating Software Engineering Methods and Petri Nets for the Specification and Prototyping

of Complex Information Systems, Proceedings of the 14th International Conference on Application and Theory of Petri Nets, Chicago, USA, June 1993, 203-233.

[4] Yun Wu, Hanyu Yang, Aihua Ren, Wenlong Yang, The Development of Integration of OO and Petri Nets, China Computer World, Sep. 20th, 1995.

[5] Olivier Biberstein, Didier Buchs, An Object Oriented Specification Language based on Hierarchical Algebraic Petri Nets, Proc. of ISCORE-1994, Amsterdam, Holland, 1994, 47-62.

[6] C. D. Keen, C. A. Lakos, A Methodology for the Construction of Simulation Models Using Object-Oriented Petri Nets, Proc. of the European Simulation Multi-conference, 1993, 267-271.

[7] Sarah L Englist, Colored Petri Nets for Object-Oriented Modeling, Ph.D. Dissertation of University of Brighton, June 1993.

[8] Jianling Hu, Theory and System Modeling of Object-oriented Colored Petri Nets, Ph.D. Dissertation of Institute of Mathematics of Chinese Academy of Sciences, July 1996.

[9] Jinzhong Niu, Jing Zou, Aihua Ren, OOPN: An Object-Oriented Petri Nets and its Integrated Development Environment, In Proceedings of 2003 IASTED International Conference on Software Engineering and Applications (SEA'2003), ACTA Press, Marina del Ray, USA, November 2003.

[10] David Harel. "Statecharts: A Visual Formalism for Complex Systems", Science of Computer Programming 8 (1987), pp. 231-274.

[11] Booch et.al. The Unified Modelin Language User Guide, Addison-Wesley.

[12] John Anil Saldhana, Sol M, ShatzUML Diagrams to Object Petri Net Models: An Approach for Modeling and Analysis, (IJCNS) International Journal of Computer and Network Security, May 2010.

[13] H. Motameni, A. Movaghar, B. Shirazi, M. Aminzadeh , H. Samadi, Analysis Software with an Object-Oriented Petri Net Model, World Applied Sciences Journal 3 (4): 565-576, 2008.

[14] Xiyao Cai, Ping Chen, Object Theory (Xi'an, China: Xidian University Publishing House, 1995).

[15] O.R., Natale, L., Glielmo, F., Vasca, (2002), "Data modeling for batch processes data with application to winemaking,", *Proceedings of the 41st IEEE Conference on Decision and Control*, 4, 4101–4106.

[16] T., Tsai, R., Sato, (2004) , "A UML model of agile production planning and control system, *Computers in Industry* , 53, 133–152.

[17] C., Secchi, M., Bonfe, C., Fantuzzi,(2007), "On the use of UML for modeling mechatronic systems,", *IEEE Transactions on Automation Science and Engineering*, 4 (1) , 105–113.

[18] K.C., Thramboulidis,(2001) "Using UML for the development of distributed industrial process measurement and control systems,", *Proceedings of the 2001 IEEE International Conference on Control Applicaion*, 1129–1134.

[19] H.J., Kohler, U., Nickel, J., Niere, A., Zundorf, (2000) "Integrating UML diagrams for production control systems," *22nd International Conference on Software Engineering*, 241.

[20] J., Kotzian, V., Srovnal,(2004), "Design and optimization of distributed control systemusing UML model,", *Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, 469–476.

[21] L., Baresi, M., Pezze ,(2001), "Improving UML with Petri nets", in: *Electronic Notes in Theoretical Computer Science* 44 No.4