Enrichment of Object Oriented Petri Net and Object Z Aiming at Business Process Optimization

Aliasghar Ahmadikatouli
Islamic Azad University, Sari Branch
Sari, Iran

Homayoon Motameni
Islamic Azad University, Sari Branch
Sari, Iran

***Abstract*—Software development process is on the basis of two important steps each of which has to be taken seriously, system requirement analysis and system modeling. There have been many different approaches in the literature that has their own strengths and weaknesses to tackle these two important steps, however, there is none comprehensive approach. Among them, formal methods by using their mathematical supporting background could achieve a precise, clear and in detail requirement analysis. However they were not able to illustrate graphically a system for stakeholders. On the other hand, semi-formal methods owning graphically representation of a system's behavior make it easy for the stakeholders to perceive thoroughly. In this paper we represent an integration of object Z formal language and a graphically modeling tool hierarchical object oriented petri net. The application of business process was used to be modeled by this intergraded language.**

Keywords—Object Z; Hierarchical object oriented Petri net; Formal methods integration; Business process; process improvement; process optimization

I. Introduction

Many efforts have been devoted to develop software systems that could meet all customer expectations and without any faults and ambiguities. Therefore, software development methods succeed to attract more interests among software mania in last decade. In the last 25 years we observe the emerge of different software development approaches such as structural methods, mathematical, object oriented and some approach to develop parallel and distributed systems. However, none of them could be applied as a comprehensive approach to describe all types of systems and different aspects of systems. Hence, researchers pursuits an integrated method that cover each other's weaknesses and bring new aspects. Method integration was studied in a number of contexts: combining formal and informal methods, combining multiple formal methods. Method integration can involve the definition of relationships between the processes of the methods being combined. It may happen to be involving constructing formal definition of the meaning of compositions written in different notation. Method integration can also be carried out via the linking of individual tools that support separate methods.

In this paper we intend to develop an integrated approach, OPOZ, which is an enrichment of two different languages, object Z and hierarchical object oriented petri net to model business processes. This new approach can complete business process optimization and their verification with less time and cost comparing to using only one of them. The importance of formal method for developing reliable and fault tolerant systems has been widely recognized in last decade. Therefore, many formal methods have been developed. Formal methods based on a formal foundation only used to specify some aspects (control, data, structure, behavior, etc.) and only some types (sequential, concurrent, distributed, real time and etc.) of software systems.

It is indispensible to say that there isn't a single formal language to satisfy the analysis of all functional and non-functional requirements. There exists several technical and philosophical reasons like, most formal methods are not able to describe non-functional requirements of systems or each formal language possess a separate set of properties to specify aspects of systems [1]. Integration of methods presents some advantages that mapping one of them to the other cannot achieve the same goal. Another way is that the system analyst modeled a system and describes its specification by different methods. Formal methods use different formalism and never an analyst could apply all these notations and formalisms on a single same system and if this happens it needed to model a system over and over which all these leads to consume time and cost. However, using integration language can help to diminish this gap in analyzing and modeling a system.

The rest of paper organized as follows: in section 2 we describe main aspects of Object Z and hierarchical object oriented petri nets. Section 3 describes the integrated model of these two languages. In section 4 we develop some reduction rules which will be uses in business process optimization and finally in last section we model a simple system by this integrated model.

Related Work

Formal methods up to now have been applied on safe and critical application. Formal method integration attracts many researches in recent years. Richard F. Paige [1] proposed a Meta model to integrate formal method and semi-formal method and analyze and evaluate the Meta model with some case studies. It is necessary to add that the way two methods must be chosen is based on the research [2] that defines when a method is incomplete and should be integrated. In [3] an integrated Object-Z and Use case diagram method has been proposed to insure the completeness and consistent of model. Soon-Kyeong Kim et. al. [4] develops an approach to express classes in class diagram via Object Z and they could reason a class diagram. Also the communications among class are fully defined as communications among OZ schemas. . A method to map use case diagram to Z schema has been proposed in [5] and a type checking by using ZTC tool was performed to extract ER diagram. Furthermore, though UML models graphically represent the structure of models and interaction among them, the lack of formal definition was immensely required, therefore, UML models were formalized by object Z schema in [7].

Researches carried out extensive research for business process modeling and analysis and many modeling and analysis techniques have been developed. However, the lack of a systematic approach that software designer could take its steps and produce an optimum model was the reason to propose methods to optimize business process models [8]. Zho and Chen [9] claimed that business process optimization can lead to less running cost and turnaround time and intensify quality of product and customer satisfaction. Vetschera and Hofacker [10] by using a genetic algorithm proposed a mathematical modeling to optimize a business process. Tiwari et al [11] and Vergdis et al [18,19] developed this mathematical model and proposed a multi-objective optimization algorithm to optimize a business process. Using formal methods to detect bottlenecks and redundant processes in a model can help designers to improve their models and optimize their predefined objective functions.

II. Principle Concepts

A. Hierarchical Object Oriented Petri Net:

HOOPN is a graphical representation of system using petri net and classes in object oriented paradigm [14]. A HOONet is a three tuple including (OIP, ION, DD) that OIP is a unique property and identification of a class. Inter Object Net (ION) is inter structural of system that represent the behavior of class and data dictionary (DD) declare the attribute of a class. The formal definition of a HOONet is [14]:

A HOONet is a three tuple (OIP,ION,DD) that must satisfy the following conditions

1) OIP is a specific place that is defined as a tuple (oip, pid, M_0,status)

- Oip is a variable for the unique name of a HOONet
- Pid a unique process identifier to distinguish multiple instances of a class
- M_0 is a function that gives specific value to the tokens
- Status is a flag variable to specify the state of OIP

2) Is a type of CPN that shows the variant of attributes and behavior of methods

3) DD is a dictionary of variables, token types and functions

B. Object Z Formal Method:

Applying formal methods in software development process is indispensible. We need to consider modularity and reusability concepts to deal with the complexity of today’s software and using methods. Therefore, using object oriented formal method that models a system based on objects and their interactions can help us.

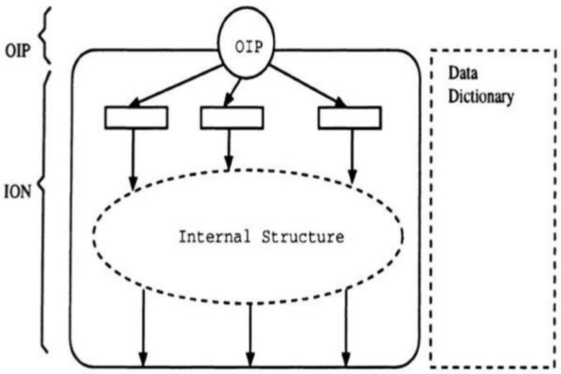
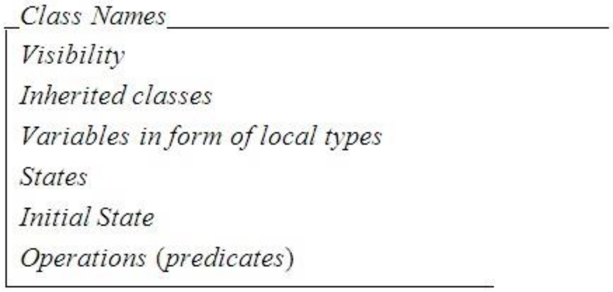


Fig. 1. The general structure of a HOONet [14]

Here, we summarize some important concepts of object Z which is an extension of Z and the key important feature added to it is the concept of class schema [16]. Class schema capsulate a schema with its all operations that may effect on its attributes. Figure 2 shows a class schema defined in object Z. Specifying a system in object Z, we must identify and specify the underlying objects and specify the system in terms of the communication between the underlying objects. However, an object may itself be a system of communication objects. A class can incorporates all the features of its inherited classes and their local types have based on Z syntax. Variables also called attributes in form of Z. All details of Object z specification and notation can be found on [19].



Class Names
Visibility
Inherited classes
Variables in form of local types
States
Initial State
Operations (predicates)

Fig. 2. Class schema defined in object Z

C. Business Process:

Today organizations consist of a set of business process that each one possess a unique functionality. Business process can be specified and model based on goals and missions in an organization. In other word, each business process defines a specific workflow in organization and workflow in basis of organization operation [13]. There have been many definitions of business process in literature. Harvey [6] defines a business process as a set of specific rules to analyze a business problem. In last decade, many researchers devoted their efforts on process modeling and many notations, methods and tools were developed [17,18]. From organization view, different goals can be considered in modeling a process. Modeling aiming at documentation or execution and etc. some of the most important business process modeling are UML, EPDL, EPC, WSDL, BPEL4WS, BPMN.

III. Object Z – Object Oriented Petri Nets (OPOZ)

In this section we present the integrated approach of object Z and object oriented petri nets. Main advantages emerge by object oriented petri net are:

1) Object oriented petri net is a good graphically tool to model concurrent and distributed system.

2) Object oriented petri net is able to comprehensively describe control structure and dynamic behavior of systems.

Also, main merits bring by object Z include suitable notations to describe and specify sequential system's function and appropriate to define types and logical and formal reasoning. In addition, advantages carries out by integrated methods are included into these:

1) A unified model for specifying different aspects of a system (structure, control flow, data types, and functionality)

2) A unified formal model for specifying different types of systems (sequential, concurrent, and distributed systems)

3) A rich set of complementary specification development and analysis techniques

4) New theoretical and methodological research problems resulted from the integration.

Similar to the [7,15] the general strategy is to use object oriented petri net to define structure and control and behavior of system and to use object Z schemas to define data abstraction , structure, system constraints and functional process. Therefore the structure of OPOZ net is as follow:

A OPOZ net is a tuple (Net, OZ, des) that

1) Net = (P,T,F) is a net structure, in which

a) P and T are non-empty finite sets satisfying P∩T= Ø (P and T are the sets of places and transitions of Net, respectively),

b) F⊆(P × T) ∪ (T × P) is a flow relation (the arcs of Net);

2) OZ = (Z_p, Z_T,Z_I) is a tuple of collection of OZ schemas

Also we uses three functions, Pro(Z), Signature(Z), Title(Z) to specify the name and signature and property of schema. Here a schema has to satisfy following conditions:

- ○ $\forall\, z_1, z_2 \in Z_P\,.(z_1 \neq z_2 \Rightarrow Signature(z_1) \cap Signature(z_2) = \emptyset)$;
- ○ $\forall\, z_1, z_2 \in Z_I\,.(z_1 \neq z_2 \Rightarrow Signature(z_1) \cap Signature(z_2) = \emptyset)$; and
- ○ $|Z_P| = |Z_I|$

The first two conditions show that the signature of OZ schema are pair-wise disjoingt, and the last condition depicts that the number of OZ schema in Z_p is the same as that in Z_I, i.e. one to one correspondence.

- ❖ des= (PM,TM,L,M_0) is a net inscription that associates a net element in N with its notation in Z
  - ○ PM: P→Z_P For each place p ∈ P, S maps p to a unique OZ schema oz ∈ Z_p such that p=Title(oz). The type of p is defined by the signature of oz
  - ○ TM: T→Z_T is one to one mapping providing the functionality definition of Net. For each transition t ∈ T, TM maps t to a OZ schema oz ∈ Z_T such that t=Title(oz).
  - ○ L: F→$\wp$Var is the control flow definition of Net, where Var is the set of all hidden variables (through quantification) in OZ_T .
  - ○ M_0: P→Z_I is an PM-respecting (i.e. signature(PM(p)) = signature (M_0(p)) initial marking of Net, where Z_I is a set of OZ schemas defining the initial state of the system.

The process of development of OPOZ model should be based on following steps:

1) Considering the requirement of systems that includes a list of events. Each event can be model by a separate petri net and then integrate by common places.

2) Using OZ schema to express data and function of system that defines the behavioral model of system.

3) Define a state schema with a specific type of each place

4) Define an initial state schema of each state schema

5) Define a functional schema for each transition and the predicate section of schema must be written based on the constraint on each transition

That is obvious that organizations need to adapt themselves with the changes in customer requirements and therefore they have to change their business processes. Besides, pervious processes and pervious models are not usable any longer and a modification in modeling is necessary. Therefore, modeling business processes using this integrated model avoid most of remodeling. We can change schema predicate if there exist any changes in business functionalities. Hence, we develop some rules to reduce or join processes. In addition, if function of only a process changed, therefore only correspond predicate in its schema needs to be modified.

Rule 1 – addition of a new predicate: Let P be a new place, Z1 be its type and state defining OZ schema, and Z2 be its initial state defining Z schema, addition place p to P results in:

a) Adding z1 to Z_p and Z2 to Z_I

b) Addition relationship (p,Z1) to S, and relationship (p,Z2) to M_0

Rule 2 – addition of a new transition: let t be a new transition needs to be added to the model and z be its OZ schema, adding t to T results in:

a) Adding z to Z_T and

b) Adding relationship (t,z) to TM

Rule 3- Merging two places with same preset and post set

Assuming p1 and p2 two places with exact equal preset and post sets and following condition (figure 3) :

$$\begin{cases} T_i I_i = P1 \\ T_i O_i = P2 \\ T_{i-1} O_{i-1} = P1 = T_i I_i \end{cases}$$

where I_i and O_i are input condition and output condition of transition T_i respectively. Therefore merging those results in:

a) Adding p to the P

b) Adding arc (p,t) to F. if $(p1, t) \in F$ *then adds* $\big((p,t), L(p1,t) \cup L(p2,t)\big)$ *to L*

c) Adding new arc (t,p) the same opposite to the b

d) Adding Z1 to the Z_p that we use schema conjunction on PM(p1) and PM(p2)

e) Adding Z2 to the Z_I that we use schema conjunction on M_0(p1) and M_0(p2) and adding (p,Z_0) to the M_0

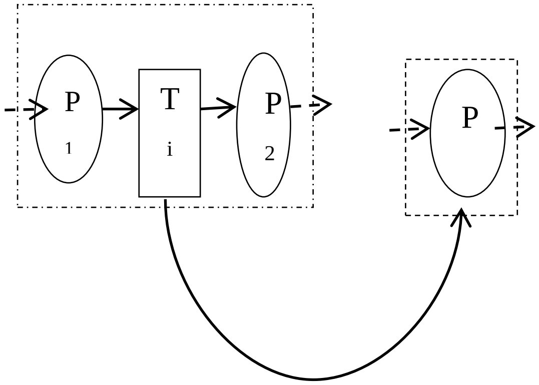


Fig. 3. Merging two places with the same presets and post sets conditions

Rule 4: Merging two serial transitions:

Assuming that t1 and t2 two transitions with the exact same preset and post set (figure 4).

Consider the following condition T_i . O_i = T_j .I_j =P_k

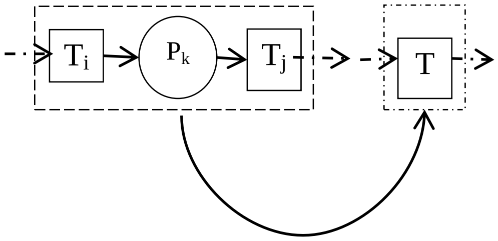


Fig. 4. Merging two Transition with the same presets and post sets conditions

Merging two T_i and T_j result in T and following constraints

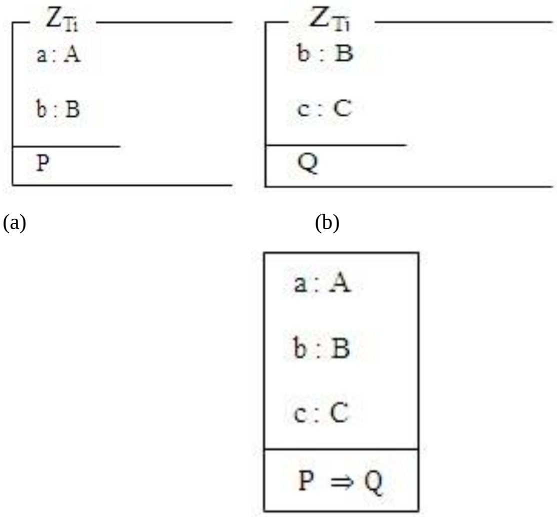
a) Adding t to the T

b) Adding z schema to the Z_T that define based on figure 5.

c) Adding new arc (p,t) to the F

d) Removing p from P and state schema p from Z_p

e) Removing t1,t2 from T and correspond schemas from Z_T and related arcs from F.



(a) (b)

(c)

Fig. 5. Merging two Z_{Ti} , Z_{Tj}, Z_T schemas(c)

Rule 5: Merging to parallel transitions:

Assuming T1 and T2 two parallel transitions with exact equal preset and post sets and with the schema t1 and t2 and following condition:

a) $T_i \,.\, I_{1,2,\ldots,n} = T_j \,.\, I_{1,2,\ldots,n} = T_n \,.\, I_{1,2,..,n}$

b) $T_i \,.\, O_{1,2,\ldots,n} = T_j \,.\, O_{1,2,\ldots,n} = T_n \,.\, I_{1,2,\ldots,n}$

The purpose of this rule is that the all function with the same input and output that apply simultaneously can be merged into one function that improves the overall performance of business process. The figure 6 shows this merging procedure.

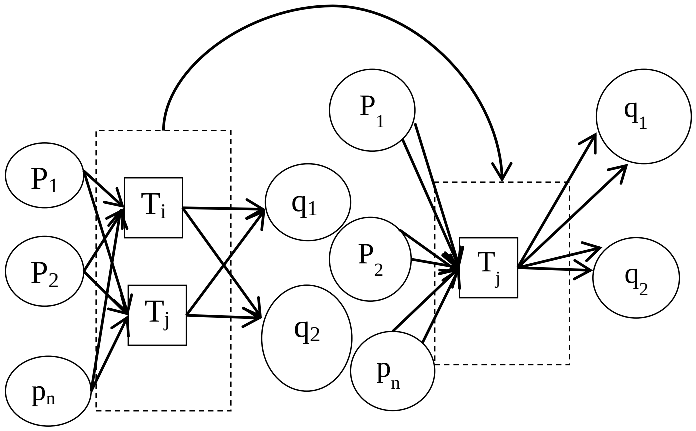


Fig. 6. Merging two parallel Transitions

Also, the functional states will be modified according to the figure 7 and it shows that predicates will be merged together and logical conjunction will be applied on predicate P and Q.

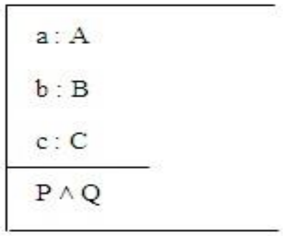


Fig. 7. The Modification of Two parallel OZ schemas

IV. CASE STUDY

Here we consider a car renting system and following interaction should be supported

1) Checking for renting availability of a car

2) Restoring a car to the garage

Also all cars should be available for renting or be rented already. According to the pervious section with the requirement analysis our system component includes customers, cars, car state, and contracts. The overall system model is depicted in figure 8.

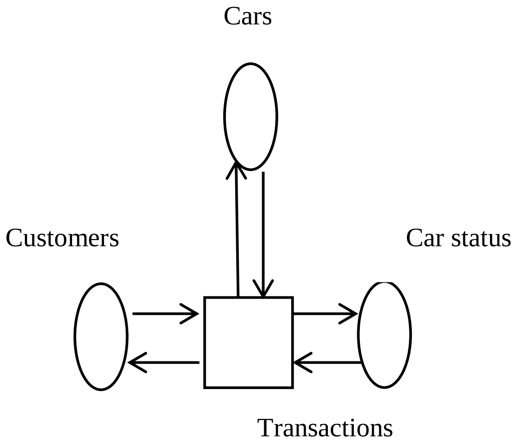


Fig. 8. Model of Car renting system

Based on the requirements of car renting system two events correspond to the renting a car and restoring to the garage have been noticed. Therefore, the HOONet structure of these two events are depicted in figure 9,10. The renting HOONet structure includes 1) customer validating 2) car availability 3) car status updating , and the HOONet structure of restoring a car contain 1) customer validating and 2) car status updating and then these two nets are integrated by common places and figure 11 are resulted.

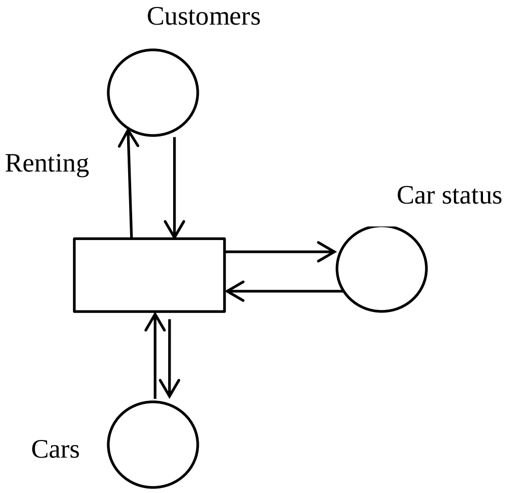


Fig. 9. Petri net model of Renting event

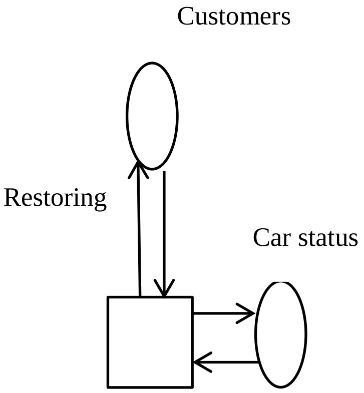


Fig. 10. Petri net model of Restoring event

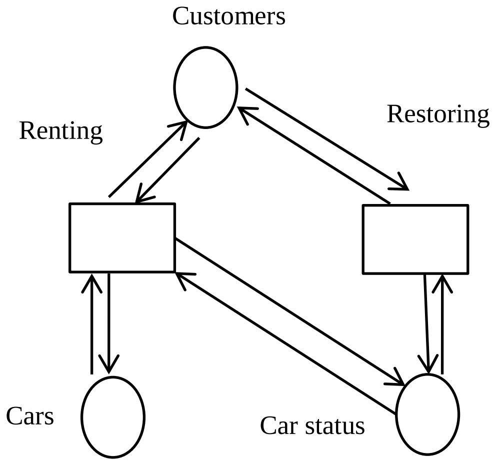
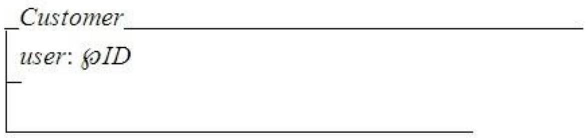


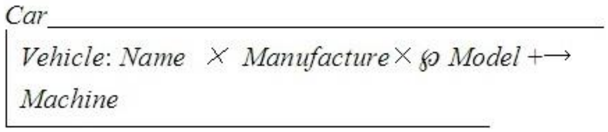
Fig. 11. Combining two petri net model (fig 9,10) based on common places

Based on the HOONet structure we propose three schemas in OZ for customers, cars and car status which are illustrated in figure 12,13,14 respectively.



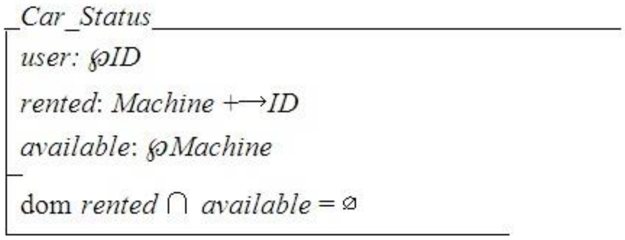
```
Customer
user: ℘ID
```

Fig. 12. Customer OZ



```
Car
Vehicle: Name × Manufacture× ℘ Model +→
Machine
```

Fig. 13. Car OZ schema

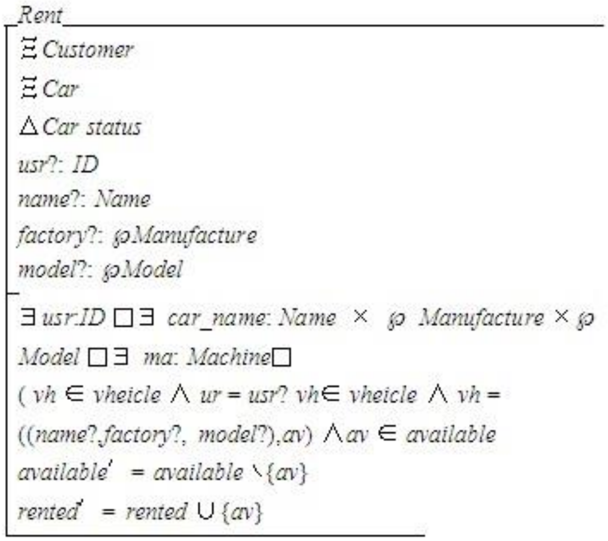


```
Car_Status
user: ℘ID
rented: Machine +→ID
available: ℘Machine

dom rented ∩ available = ∅
```

Fig. 14. Car_Status OZ schema

The last predicate in Car status shows that a car must be available for renting or have been rented already not both. Renting transaction requires both validate a customer and update a car status. The input of this transaction includes customer identification and car information and car status flag. The renting schema shows in figure 15.

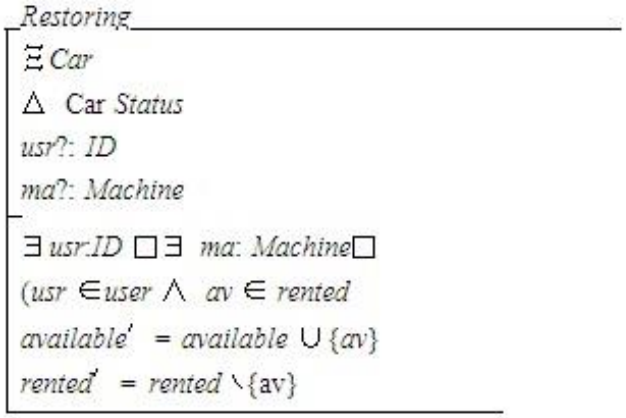


```
Rent
Ξ Customer
Ξ Car
Δ Car status
usr?: ID
name?: Name
factory?: ℘Manufacture
model?: ℘Model

∃ usr:ID □∃ car_name: Name × ℘ Manufacture × ℘
Model □∃ ma: Machine□
( vh ∈ vheicle ∧ ur = usr? vh∈ vheicle ∧ vh =
((name?,factory?, model?),av) ∧av ∈ available
available′ = available ∖{av}
rented′ = rented ∪ {av}
```

Fig. 15. Renting OZ schema

In the renting schema we do not need the information of cars and therefore the restoring schema shows in figure 16



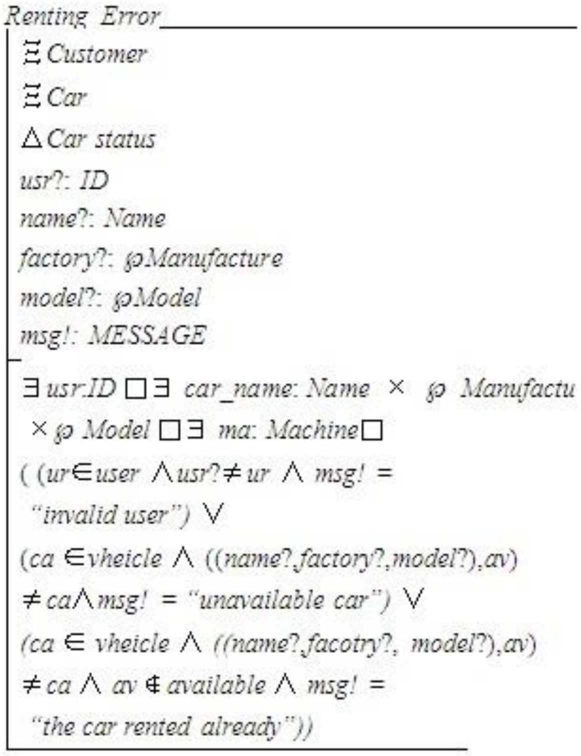
```
Restoring
Ξ Car
Δ Car Status
usr?: ID
ma?: Machine

∃ usr:ID □∃ ma: Machine□
(usr ∈user ∧ av ∈ rented
available′ = available ∪ {av}
rented′ = rented ∖{av}
```

Fig. 16. Restoring OZ schema

Assuming that we need to add a new transaction that shows if a car is not available or the customer is not valid the system should reported an error message.

Therefore adding this new transaction resulted in updating only one HOONet structure and designing a new schema which is depicted in figure 17 and figure 18 respectively.



```
Renting Error
Ξ Customer
Ξ Car
Δ Car status
usr?: ID
name?: Name
factory?: ℘Manufacture
model?: ℘Model
msg!: MESSAGE

∃ usr:ID □∃ car_name: Name × ℘ Manufactu
 × ℘ Model □∃ ma: Machine□
( (ur∈user ∧usr?≠ ur ∧ msg! =
 "invalid user") ∨
(ca ∈vheicle ∧ ((name?,factory?,model?),av)
≠ ca∧msg! = "unavailable car") ∨
(ca ∈ vheicle ∧ ((name?,facotry?, model?),av)
≠ ca ∧ av ∉ available ∧ msg! =
 "the car rented already"))
```

Fig. 17. Renting Error OZ schema

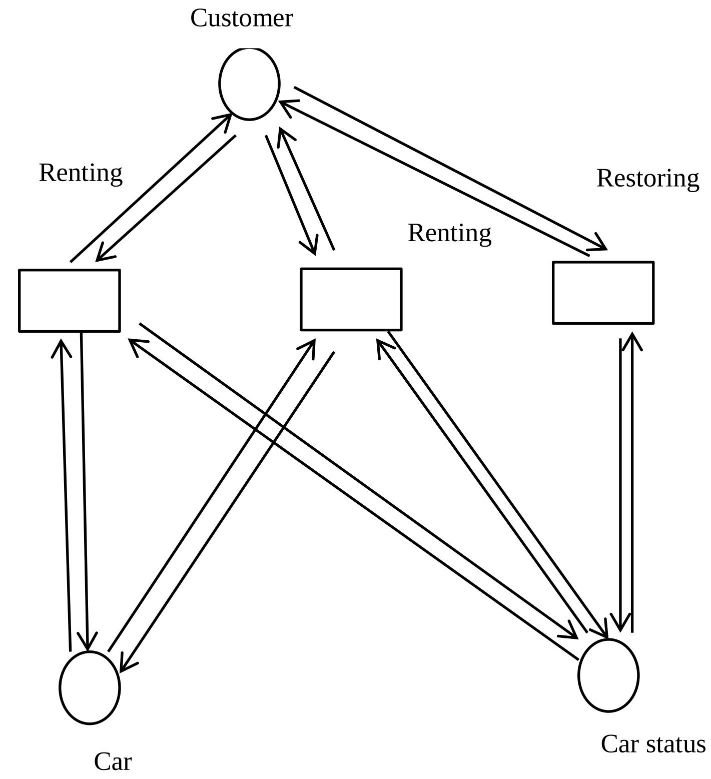


Fig. 18. New Petri net model after adding new Transition

V. CONCLUSION

Maturity and popularity of graphical and object oriented modeling expanded literary. One of the most important requirements of complex and large system modeling and analysis is their unambiguous and flawless designs. Therefore, experts develops variety of approaches to tackle this including formal methods, object oriented and etc. each of which could only model one of aspect of system. Here, we integrated two object oriented approach that could result both graphical representation and possess formal descriptions of system specification. In addition, we illustrate some rules could represent some changes in requirements that prevents remodeling of all designs. Future works in object oriented formal method integration can be done in integrating aspects like inheritance, polymorphism in both methods. So, we are aiming to develop an integrated definition that includes these both concepts.

REFERENCES

[1] Paige R. F., Case Studies in Using a Meta-Method for Formal Method Integration.

[2] Paige R. F. 1999. When are methods complementary? vol. 41. pp. 157–162.

[3] A. Moreira, “Generating Object-Z Specifications From Use Cases Object-Z,” no. Duke 1991, 1992.

[4] S. Kim and D. Carrington, “Formalizing the UML Class Diagram Using Object-Z,” pp. 83–98, 1999.

[5] Sabnam Sengupta , Swapan Bhattacharya, Formalization of UML Use Case Diagram-A Z Notation Based Approach, International Conference on Computing & Informatics (ICOCI), 2006.

[6] Havey, M. Essential Business Process Modeling. 2005. U.S.A., O’Reilly.

[7] Huaikou Miao, Ling Liu, and Li Li, Formalizing UML Models with Object-Z, Formal Methods and Software Engineering , Lecture Notes in Computer Science Volume 2495, pp 523-534 , 2002.

[8] E.K. Burke, J.D. Landa Silva, “The influence of the fitness evaluation method on the performance of multiobjective search algorithms” , European Journal of Operational Research 169 ,2006 , 875–897

[9] Zhou, Y. and Chen, Y. Project-oriented business process performance optimization. Proceedings of IEEE International Conference on System, Man and Cybernetics. 5, 4079-4084. 2003.

[10] Hofacker, I. and Vetschera, R. Algorithmical approaches to business process design. Computers & Operations Research 28, 1253-1275. 2001.

[11] Tiwari, A., vergidis, K., and Majeed, B. Evolutionary Multi-Objective Optimization of Business Process. Proceeding of IEEE Congress on Evolutionary Computing 2006. 3091-3097. 2006.

[12] K. Vergidis, A. Tiwari , B. Majeed , R. Roy , Optimization Of Business Process Designs:An Algorithmic Approach With Multiple Objectives, Int. Journal production Economics 109 , 2007 , 105-121.

[13] Vergidis K.,Tiwiari A., and Majeed, B. Business Process Improvement Using Multi-Objective Optimization . BT Technology Journal 24(2), 229-235. 2006.

[14] Jang-Eui Hong , Doo-Hwan Bae, Software Modeling And Analysis Using Hierarchical Object-Oriented Petri Net, Information Sciences ,pp.133-164, 2000.

[15] X. He, PZ nets, A formal method integrating Petri nets with Z,vol. 43, pp. 1–18, 2001.

[16] Graeme, P. S.1992. An Object-Oriented Approach To Formal Specification. PHD Thesis.

[17] K. Vergidis, A. Tiwari and B. Majeed , Business Process Analysis and Optimization: Beyond Reengineering , IEEE TRANSACTIONS ON SYSTEMS , VOL. 38, NO. 1, JANUARY 2008.

[18] R. Changrui, W. Wei , H. Ding, B. Shao , Q. Wang , Towards FLEXIBLE BUSINESS PROCESS MODELING AND SIMULATION ENVIROMENT ,

[19] Graeme Smith, The ObjectZ Specification Language, 2000,Springer.