Using Colored Petri Networks for Modeling Multiview Class

ZEDDARI Abderrazzak, ETTALBI Ahmed and NASSAR Mahmoud

University of Mohammed V-Souissi, ENSIAS, Rabat, Morocco

Summary
Unified Modelling Language (UML) has some limits for modelling complex systems, such as access right, coherence and redundancy problems during multiview sub models fusion. In this paper, we propose an approach, using colored Petri networks when modelling a complex system to a multiview class Diagram. Our approach consists of reducing the complexity of ordinary Petri networks by using the concept of color and associating one color with each point of view.

Key words:
Complex systems, UML, View and Viewpoint, Modelling, Petri networks, colored Petri nets.

1. Introduction

In an UML class diagram, each actor has potentially the same access rights to information and methods encapsulated in classes. However, actors do not have the same needs and the same responsibilities.
Controlling access rights can not be realized in the class diagram but only in the dynamic diagrams (sequence diagram, collaboration diagram…), and in this case the control must be programmed in the class methods. Our objective is to put the access rights information to a high level of abstraction, particularly, during the analysis phase.
To achieve this goal, a new type of class, called the multiview class, has been proposed which allows users to define views associated with actors' profiles.
The main goal of this paper is to model a multiview class using a colored Petri net. This approach is better for being more visual, understandable (communication), scalable, executable. It also provides many techniques for automatic verification of model properties.
In the next sections, we will discuss

- The UML limits when modeling a complex system.
- The view point approach.
- The Petri networks
- Our Approach

2. UML limits in modelling complex systems

A complex system is defined as a system that is irreducible to a finite model regardless of its size, the number of its components and of the intensity of their interaction [8].

The Unified Modelling Language (UML) is a family of design notations that is rapidly becoming a standard software design language. UML provides a variety of useful capabilities to the software designer, including multiple, interrelated design views, a semiformal semantics ex-pressed as an UML meta model, and an associated language for expressing formal logic constraints on design elements.
Despite all the benefits listed above, UML has its limits though: it does not provide opportunities for formal verification of specific properties (for example: show that a signal always arrives before another). Another UML limitation is the lack of a performance evaluation procedure of the system being specified and designed. It is not possible to formally prove that certain situations will never be reached, which negatively affect the system safety. Similarly, it is not possible to prove that some states will still be met if necessary impinging on the system liveliness. Also it cannot model a multi-view system.
For these reasons the notion of point of view in modeling object-oriented complex systems has been introduced, it consists on a single model available after the fusion of several shareable views. The advantage of introducing point of view in modeling object-oriented complex systems is the following: the consistency of data, removing some redundancy, the enrichment of multi-models and access rights definition [10].

3. View Point Approach

A viewpoint-based approach is that all information about the system requirements cannot be discovered by considering the system from a single perspective. Rather, we need to collect and organize requirements from a number of different viewpoints.
A viewpoint is an encapsulation of partial information about a system's requirements. Information from different viewpoints must be integrated to form the final system specification.
A method of analysis and design by object and view point has been proposed in [11] supporting the concepts of view and point of view, these include the work of [1]. This method allows modeling a system according to the views of various users. This leads to develop a number of partial models called visual models. Once these models are made, a step of fusion of these models is the method proposed by

Manuscript received August 5, 2013
Manuscript revised August 20, 2013

VBOOM. VBOOM has been applied to modeling many systems: a subset of the Ariane IV box [10] and a system of management of the interaction transportation / environment in Morocco [2].

4. Petri networks

A Petri networks (also known as a place/transition net or P/T net) is one of several mathematical modeling languages for the description of distributed systems. A Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e. events that may occur, signified by bars) and places (i.e. conditions, signified by circles). The directed arcs describe which places are pre- and/or post conditions for which transitions (signified by arrows) occurs. Petri nets were invented in August 1962 by Carl Adam Petri [11] for the purpose of describing chemical processes.

A Petri net consists of places, transitions, and arcs. Arcs run from a place to a transition or vice versa, never between places or between transitions. The places from which an arc runs to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition.

Graphically, places in a Petri net may contain a discrete number of marks called tokens. Any distribution of tokens over the places will represent a configuration of the net called a marking. In an abstract sense relating to a Petri net diagram, a transition of a Petri net may fire if it is enabled, i.e. there are sufficient tokens in all of its input places; when the transition fires, it consumes the required input tokens, and creates tokens in its output places

In practice, it is necessary to model any kind of system, like communication protocols, production systems, reactive systems, real-time systems. This variety of systems has led researchers to expand PNs theory by introducing many concepts in each area application. These efforts have led to high-level PNs such as colored PNs, hierarchical PNs, timed PNs and others.

4.1 Colored Petri Networks (CPN)

Colored PNs [4], [6] include brands which colors are assigned. They are a class of networks whose intuitive perception is less clear as for ordinary PN.

They are of great interest to model data structures of objects in a system.

4.2 Hierarchical Petri Networks (HPN)

Hierarchical PNs [9] can model a modular way as a programming language. HPN is composed of a plurality of modules (pages), where in each module is a sub network. A main module references all other secondary modules. Referencing a module is done by a transition of substitution with the name of module. The use of HPNS is recommended for modeling large problems.

4.3 Temporal Petri Networks (TPN)

A timed PN is used to describe a system whose operation depends on the time. The timed PN are useful for performance evaluation of a system. Timers are associated either to places (PN P-timed) or transitions (T-timed PN).

Examples 1: Building a model for three philosophers with an ordinary Petri network.

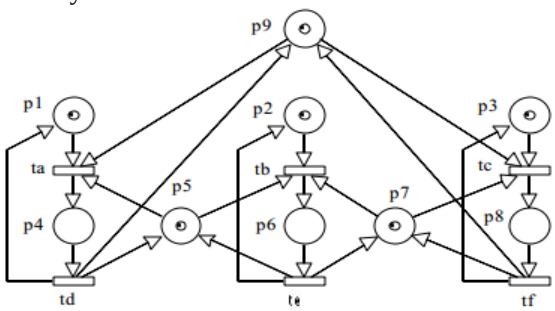

Fig. 1 PN of three philosophers example

Places **p1**, **p2** and **p3**, respectively, represent the philosophers **Ph1**, **Ph2** and **Ph3** in the initial state: thinking. So these are Ph_think statements for **Ph1**, **Ph2** and **Ph3**. Places **p4**, **p6** and **p8**, respectively, represent the philosophers **Ph1**, **Ph2** and **Ph3** eating. So these are **Ph_eat** statements for **Ph1**, **Ph2** and **Ph3** respectively.

Places **p9**, **p5** and **p7**, respectively, represent the spoons, **spoon1**, **spoon2** and **spoon3** in spoon_free state.

Transitions **ta**, **tb** and **tc** represent the change of philosophers state **Ph_think** states to **Ph_eat** states. **Td** transitions, **te** and **tf** represent the change of philosophers state **Ph_eat** to **Ph_think**.

The **P4** is the place **Ph_eat** state for **Ph1** and is also a busy spoons state for **spoon1** and **spoon2**. **P6** is the place **Ph_eat** state for **Ph2** and is also a busy spoons state for **spoon2** and **spoon3**. **P8** is the place **Ph_eat** state for **Ph3** and is also a busy spoons state for **spoon3** and **spoon2**.

Example 2: Building the same model for three philosophers with a colored Petri network.

We will fold the three Philosophers objects **(p1, ta, p4, td**), **(p2, tb, p6, te)** and **(p3, tc, p8, tf)** to form a single class **(p11, t1a, p12, t1b)**. Similarly, we will fold the three spoons objects **(p9, ta, p4, td, tc, p8, tf)**, **(p5, ta, p4, td, tb, p6, te)** and **(p7, tb, p6, te, tc, p8, tf)** to form a single class **(p13, t1a, p12, t1b**). Note that both states occupied by spoons (for example p4 and p8 for the spoon whose state is p9) are folded over each other.

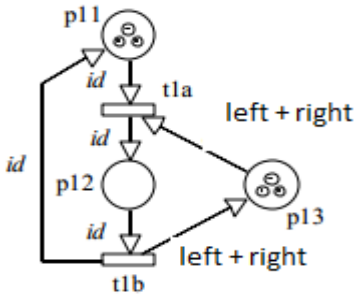

Fig. 2 CPN of three philosophers example

The colors set is: **Col = {Ph1, Ph2, Ph3, spoon1, spoon2, spoon3}.**

Because p11 results from folding of p1, p2 and p3, we have a function F: **F (p11) = {Ph1, Ph2, Ph3}**

Since p13 results from folding p9, p5 and p7 we have:

F (p13) = {spoon1, spoon2, spoon3}.

To the solution is compatible with the network of Figure 2 given functions "id" worn by arcs, we must have

F (t1a) = F (p12) = F (t1b) = F (p11) = {Ph1, Ph2, Ph3}.

This means that the names of the spoons are "forgotten" when philosophers eat. This is possible because each philosopher always use the same spoons (it is the one on his right and the one on the left). There is a way of eating. The name of the philosopher eating is enough to deduce the spoons used.

5. New approach for modelling multiview class

In [3], a new approach has been proposed to model a multiview class using ordinary Petri network, this approach consists of associating viewpoints to places and associating the viewpoint activation events to transitions. Although this approach is effective, it has its limit. The more complex the system becomes, the larger the network becomes.

To resolve this issue, we propose a new approach using Colored Petri Networks; it allows having a small Petri network, when modelling a very complex system. Our approach consists of several steps that are described below:

Step 1: Domain Color and Places

At every viewpoint, we associate a color and two places:
- VPDA: deactivated state of Viewpoint i,
- VPAC: activated state of Viewpoint i,

At each view, we associate a colored token:
- V corresponds to a View.

Step 2: Arcs and functions

At each arc, we associate a function to determine instances of tokens (views) necessary, activated and deactivated in crossing a transition.

Step 3: TRANSITIONS

At each ViewPoint, we associate two transitions:
- AVP: the event Activate Viewpoint i,
- DVP: the event Deactivate Viewpoint i.

INITIAL STATE

Initially, all viewpoints are deactivated. Place VPDA contains all colors; the other places do not contain any color.

Below, we apply our approach on 3 examples, the first two examples are used in [3] with ordinary Petri network. The 3rd example corresponds to a multiview class car.

5.1 Multiview class Article

In [1], an example of modelling a multiview class Article using ordinary Petri Network has been developed.

The class Article includes the following fields:
- **AN**: Article Number
- **Name**: Article Name
- **UPP**: Unit Purchase Price
- **USP**: Unit Selling Price
- **QS**: Current Quantity in Stock
- **MinTQS**: Minimum Threshold Quantity in Stock
- **MaxTQS**: Maximum Threshold Quantity in Stock

This class supports three viewpoints: that of the Client, another associated with the Cashier and the third is related to the Seller. Table 1 shows the fields accessible for each user according to his views.

Table 1: Accessible fields for each viewpoint

Client: VP1	***Cashier***: VP2	***Seller: VP3***
AN	AN	AN
Name	Name	Name
USP	USP	USP
	QS	QS
	MinTQS	MinTQS
		UPP
		MaxTQS

In Table 2, we present the views of the class Article. Three views can be distinguished: V1, V2 and V3. Each view contains fields. The views are then grouped to give point of views.

In Table 3, we present for each viewpoint, the views composing it. Therefore, the viewpoint of Client consists of View V1. That of the Cashier is composed of Views V1

and V2 whereas Views V1, V2 and V3 are the Seller's viewpoint.

Table 2: Views related to the class Article

View V1	*View V2*	*View V3*
AN	QS	UPP
Name	MinTQS	MaxTQS
USP		

Table 3: Composition of viewpoints in terms of views

VP1: Client	*VP2: Cashier*	*VP3: Seller*
V1	V1+V2	V1+V2+V3

After applying my approach, the obtained Petri network is below:

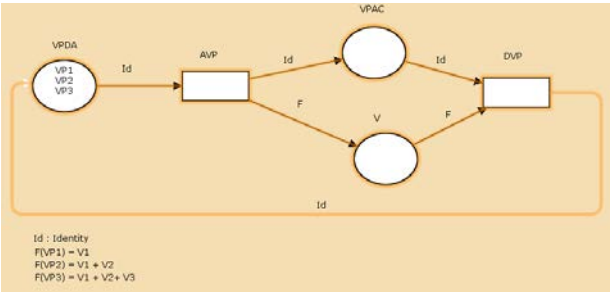

Fig 3 Colored Petri network associated with the multiviewclass Article using CPNTool.

5.2 Multiview class Course

In [1], a 2nd example of modelling a multiview class Course using ordinary Petri Network has been developed. The class course has the following fields:

- **Id**: Course Identifier
- **Title**: Course Title
- **Responsible**: Person in charge of the Course
- **Remarks**: List of remarks on the Course
- **Exam**: Exam associated with the Course
- **Exercises**: List of exercises
- **Resources**: Course resources
- **Difficulties**: Difficulties associated with the Course
- **Questions**: List of students' questions related to the Course
- **Answers**: List of the tutor's answers to asked questions
- **Fees**: Course fees
- **EnrolledStudents**: List of students enrolled in the Course

It supports three viewpoints: that of the Student, another associated with the Tutor and the third is related to Responsible. Table 4 shows the accessible fields for each user depending on his views.

Table 4: Accessible fields for every viewpoint

Student : VP1	*Tutor : VP2*	*Responsible : VP3*
Id	Id	Id
Title	Title	Title
Responsible	Responsible	Responsible
Remarks	Remarks	Remarks
Exam	Exam	Exam
Exercises	Exercises	Exercises
Fees	Resources	EnrolledStudents
Resources	Difficulties	
Difficulties	Questions	
Questions	Answers	
Answers	EnrolledStudents	

In Table 5, we present the views of our class Course. There are four views: V1, V2, V3 and V4. Each VIEW contains fields. The views will be then grouped to give the viewpoints of users.

Table 5: Views associated with the class Course

View V1	*View V2*	*View V3*	*View V4*
Id	Fees	Resources	EnrolledStudents
Title		Difficulties	
Responsible		Questions	
Remarks		Answers	
Exam			
Exercises			

In the table that follows (Table 6), we present for each point of view, the views that compose it. Thus, the view of the Student consists of Views V1, V2 and V3. That of the Tutor consists of Views V1, V3 and V4, whereas the views V1 and V4 are the Viewpoints of the Responsible.

Table 6: Composition of viewpoints in terms of views

VP1: Student	*VP2: Tutor*	*VP3: Responsible*
V1 + V2+ V3	V1+V3 + V4	V1+V4

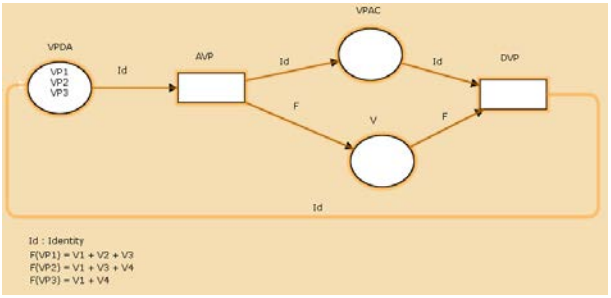

Fig. 4 Colored Petri network associated with the multiview class Course using CPNTool

5.3 Multiview Class: Car

Below, we will apply our approach on a multiview class Car; this class includes the following fields:
Ref : String ,represent car reference
Brand : String, represent the car brand

Color : String, represent car color
Fuel : String, represent the car fuel
Consumption: real, represent the consumption of the car
Discount: real, the discount in the price of the car
SellingPrice: real, represent the selling price of the car
RecommendedPrice: Real, recommended price

This class supports three viewpoints: that of the Client, another associated with the Commercial and the third is related to the Mechanic.
Table 7 shows the fields accessible for each user according to his views.

Table 7: Accessible fields for each viewpoint

Client: VP1	*Commercial: VP1*	*Mechanic: VP1*
Ref	Ref	Ref
brand	brand	brand
Color	color	color
Fuel	fuel	fuel
consumption	consumption	consumption
Discount	discount	
sellingPrice	sellingPrice	
	RecommendedPrice	

In Table 8, we present the views of the class Car. Three views can be distinguished: V1, V2 and V3. Each view contains fields.
The views are then grouped to give point of views.

Table 8: Views related to the class Car

V1	*V2*	*V3*
Ref	discount	RecommendedPrice
brand	SellingPrice	
color		
fuel		
consumption		

In Table 9, we present for each viewpoint, the views composing it.

Table 9: Composition of viewpoints in terms of views

VP1: Mechanic	*VP2: Client*	*VP3: Commercial*
V1	V1+V2	V1+V2+V3

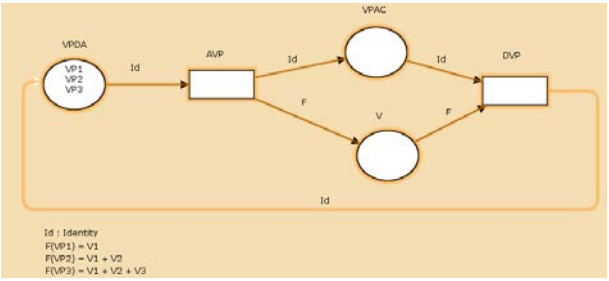

Fig. 5 Colored Petri network associated with the multiview class Car using CPNTool

6. Conclusion and perspectives

In this paper we have presented an approach for modeling a multiview class using colored Petri networks. After briefly describing complex systems, we introduced the notion of view and viewpoints, its interest for complex systems and its integration in software development including object modeling. Then, we presented our approach which attempts to associate each multiview class to a colored Petri network.
Our perspective is to include other high level Petri networks to reduce the complexity of modeling complex system such as Temporal Petri Networks, Stochastic Petri Networks and Hierarchical Petri Networks.
Also we hope to add a new diagram in the UML object modeling language to represent multiview classes by colored Petri networks.
Other perspective of our work is to take into account methods of classes in translating multiview classes to a Colored Petri Networks.

References

[1] Coulette, B. Kriouile, A. Marcaillou, S., L'approche par points de vue dans le développement orienté objet des systèmes complexes, Revue l'Objet, vol. 2, n°4, p. 13-20, 1996.
[2] El Asri B., Kriouile A., Boulmakoul A., Coulette B., "Application de l'approche objet orientée point de vue à la modélisation des Interactions Transport-Environnement", actes du 4ème Colloque Africain sur la Recherche en Informatique (CARI'98), Dakar (Sénégal), 12-15 octobre 1998, pp. 497-508.
[3] Ettalbi A., Nassar M., Sbihi B., Viewpoints Diagram: Towards an innovative diagram in the UML Language, International Journal of Computer Science and Network Security (IJCSNS'2012), ISSN: 1738-7906, Vol.12, No.8, August 2012, pp: 49-54, http://paper.ijcsns.org/07_book/201208/20120809.pdf,
[4] Jensen K., Coloured Petri Nets and the Invariant-Method. Theoretical Computer Science, Vol. 14, No. 3, 1981, p. 317-336
[5] Jensen K., High level Petri Nets, Anastasia Pagnoni; Grzegorz Rozenberg. Vol. 66 Springer, 1983. p. 166-180
[6] Jensen K., Kristensen L.M., Coloured Petri Nets, DOI 10.1007/b95112, (C) Springer-Verlag Berlin Heidelberg 2009
[7] Kriouile, A., VBOOM, une méthode orientée objet d'analyse et de conception par points de vue, Thèse de doctorat d'Etat, Université Mohammed V de Rabat, 1995.
[8] Le Moigne J.L, la modélisation des systèmes complexes, Dunod, 1990.
[9] Machado R.J, Hiararchy in object-Oriented petri nets for the specification of digital systems, Nov/96
[10] Marcaillou, S, Intégration de la notion de points de vue dans la modélisation par objets – Le Langage VBOOL, Thèse de l'Université Paul Sabatier de Toulouse, 1995.

[11] Petri C.-A., Communication par les automates, thèse de doctorat de l'université technologique de Darmstadt, Allemagne, 1962.

ZEDDARI Abderrazzak PhD student software development Engineer at The Institut National de Poste et Telecommunication, Rabat. Associate Degree in Science (Honors). Java Senior Developer.

ETTALBI Ahmed Professor at Software Engineering Department of the Higher National School of Computer Science and Systems Analysis (ENSIAS) Rabat. His main research interests Object Modeling with Viewpoints, Software Architecture and Business Process Modeling architecture.

Mahmoud NASSAR Professor Head of Software Engineering Department. Ph.D. in Computer Science from the National Polytechnic Institute of Toulouse, 2005. Research Theme: Context-Aware Service-Oriented Computing, Engineering Component-based, Model-Driven Engineering Subjects taught: Service Oriented Architecture (SOA), bases distributed Administration Database Management Systems database (ORACLE), distributed object systems, Compilation, Algorithms and Programming data.