Viewpoints Diagram: Towards an innovative diagram in the UML Language

Ahmed Ettalbi[†], Mahmoud Nassar[†] and Boubker Sbihi[††]

[†]University of Mohamed V-Souissi, ENSIAS, Rabat, Morocco
[††]Information Sciences School, Rabat, Morocco

Summary
The objective in this paper is to put forward a modeling of multiview classes by making use of the Petri networks formalism. The object approach aims at modeling complex systems and allows to take into account the views of the different users of the system and their access rights. Thus, in the present work we propose a representation of multiview classes by Petri networks. This representation allows the administrator to follow the dynamic evolution of viewpoints and to monitor the access rights of users.

Key words:
Complex systems, UML, View and Viewpoint, Modeling, Petri networks.

1. Introduction

The modeling of complex systems cannot be carried out by a unique viewpoint because of the different needs and rights of access to information that are specific to each user. The object approach has shown its power in the field of software development. Among the approaches that have been inspired by the object approach, there is the object approach by viewpoints that is interested in the development of complex systems. The viewpoint approach is based on the notion of multiview class. Several research studies have investigated the possibility of incorporating the viewpoints in UML.

The remainder of this paper is organized as follows. Section Two introduces complex systems and their characteristics. Section Three presents the notion of view and viewpoint and its use in software development. In Section Four, we deal with views and viewpoints in the object approach. Section Five tackles Petri networks and some areas of their use. In Section Six we put forward our approach which aims at modeling a multiview class by Petri networks. Section Seven provides a brief conclusion and gives an overview of our future works.

2. 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 [12]. The complexity of a system is due to two main dimensions:

The temporal dimension: it is the fact that a complex system is always unpredictable in time, deterministic (not reducible to a finite state system), and the same observer, with the same viewing angle but at different times (or sessions) can make different representations of the observed system. This can be referred as the dynamicity or metamorphosis of the system.

The spatial dimension: depending on the place of observation and the observer (or profile), the amount of information that the observer has on the system is different. Indeed, steeped in a system of value, culture and endogenous factors, each observer understands, thinks and acts towards a system in a different way. The relationship between the amount of information the observer has on the system and the amount of information actually contained in this system can range from 0 (chaotic system) to a (simple).

This complexity is apprehended by the interested observer (view / time). The modeler, trying to make sense of the actual information of the system, artificially represents the latter according to the design and the view that it sets up and receives with respect to this system. A system can be represented by a set of possible varieties according to the projection of the actor, the purpose of the system, etc.

Every complex system is, therefore, inherently multiview, and each interaction observer / system can be regarded as an interactive experience. As such, we can argue that every space-time provided by the observer, is associated with a space-time (views) of the system.

When developing a complex system [12], the construction of a comprehensive model taking into account both the needs of all players is impossible. In reality, either several partial models are developed separately and coexist with the associated risks of inconsistency, or the global model must be frequently challenged, sometimes deeply, when the users' needs evolve.

Manuscript received August 5, 2012
Manuscript revised August 20, 2012

3. Views and viewpoints in software development

In the object approach, a class is a template from which we can create physical representatives called object or instance. The object represents a real world entity characterized by attributes and methods. Attributes model the state of the object and the methods its behavior.
In the object approach by viewpoint, a view is defined as being a partial abstraction of the model; it is a sub-model. A viewpoint is a user's view of the model. A viewpoint is, thus, a combination of several views.
The modeling of complex systems cannot be carried out using a unique point of view. This is due to the different needs and rights of access to information specific to each user. It is, therefore, necessary to take into account the viewpoints of the different users of the system during its modeling. Indeed, each user has his own profile that defines the set of needs and access rights associated with that user.
The introduction of the notion of viewpoint in object-oriented modeling of complex systems can elaborate a unique model that is shareable and accessible following next several viewpoints [4]. The advantage of this new approach appears at the consistency of data, deletion of some redundancy, enhancement of the multi-model approach and the definition of access rights.
In the object viewpoint approach, a flexible class is defined as a class in which we declare a set of views to be selected when choosing one viewpoint at instantiation.
The notion of views has been addressed in the area of programming including programming by topics [19], aspects [10] and by object [2] [5] [13] [14] [22] [28]. It was also studied in the case of systems such as TROPES LOOPS [27] in the field of knowledge representation, role models [8] and also in O2Views, MultiView and COCOON systems [21] for databases.

4. Views and viewpoints in the object approach

In the object approach, various research works have dealt with the concept of views in order to integrate it. These include the work of [4], the VBOOM method (View Based Object Oriented Method) proposed by [11] and VBOOL language (View Based Object Oriented Language) put forward by [13] and VBOOL compiler [16]. After the emergence of the Unified Modeling Language (UML) [1] and its standardization by the OMG [18], and given the interest to take into account in any modeling a standard modeling language used by the scientific community, it is necessary to migrate the notion of viewpoints to UML.
In this vein, several approaches have been proposed to integrate the notion of visibility in object modeling. We can also cite the work of Clarke [3], those of Desmond and Wills [6], the VUML profile (View based Unified Modeling Language) proposed by [17] and the method proposed by U_VBOOM [9]. On the other hand, we proposed in [23] [24] [25] [26] a filtering mechanism to move from a VBOOM class diagram to a UML class diagram in order to target object languages other than VBOOL (single target language of VBOOM) such as Java and C++.
At the formalization level, we proposed in [7] an approach for moving from one viewpoint to an ordinary Petri network in which places represent the viewpoints and transitions of viewpoints activation or deactivation.

5. Petri networks and areas of use

Petri networks actually allow to study complex dynamic systems. They were proposed in the 60's by Carl Adam Petri [20], then developed at MIT (Massachusetts Institute of Technology) in 1975 [15]. They are now used to specify, model and understand the systems in which several processes are interdependent.
A Petri network consists mainly of place and transition. Places represent states and transitions events. An arc can connect a place to a transition or a transition to a place. Each place contains a positive integer or zero marks or tokens. The tokens are usually resources in the modeled system. M Marking defines the state of the system described by the network at any given time. It is a column vector of dimension the number of places in the network. The i^{th} element of the vector is the number of tokens contained in the i^{th} place.
A transition is passable when all the input places of the transition contain at least one token. The clearing of a transition consists of removing a token from each of the input places and adding one token to each output place of the same transition.
Petri networks are widely used in modeling, specification and verification of the behavior of competitive systems, dynamic systems, real-time systems and distributed systems. They are also used to validate communication protocols, man-machine interfaces and to synchronize processes and production systems.
The following figure shows an example of a Petri network. The network models a system comprising two computers competing for the use of shared memory. Initially, the two computers do not need memory (one token in PBM place) and, hence, the memory is available in this state.

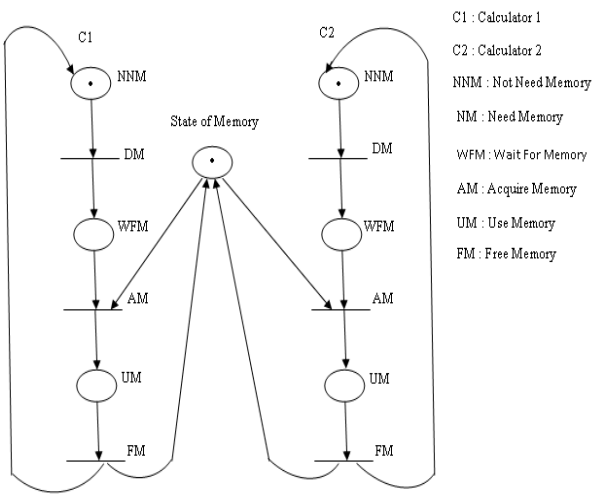

Figure 1: Example of Petri networks

There is a variety of Petri networks. These include colored networks, stochastic networks, temporal networks and inhibitor arcs networks. This actually gives the choice to adopt the type of Petri networks that is best suited to model a given problem.

6. Using Petri Networks to model viewpoints

In this section we focus on the application of our approach to model multiview classes by Petri networks. This approach is illustrated below:

PLACES

At every viewpoint i, we associate two places:
- VPiDS: corresponds to the deactivated state of Viewpoint i,
- VPiAC: corresponds to the activated state of Viewpoint i,

At each view i, we associate a place:
- Vi corresponds to the activated or deactivated state of View i,

TRANSITIONS
- AVPi: corresponds to the event Activate Viewpoint i,
- AVVPi: corresponds to the event Activate Views of Viewpoint i,
- DVPi: corresponds to the event Deactivate Viewpoint i.

INITIAL STATE

Initially, all viewpoints are deactivated. So, VPiDS places each contain each a token, other places do contain no token.

Below, we will apply our approach on two different examples. The first is the multiview class Article that supports three viewpoints: that of the Client, another for the Cashier and the third concerns the Seller. In the second example, we are interested in the class Course that also supports three viewpoints: that of a Student, a second is related to the Tutor and the third is that of the Responsible. In each example, we first determine for each viewpoint the fields that the user has and for which he has access to depending on his viewpoint. Then, we highlight the views of our class. After, we establish the views associated with each user on the basis of highlighted views. Finally we apply our approach to this multiview class and we give the Petri network associated with it.

6.1 Example 1: Multiview Class : Article

This multiview class 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	***Cashier***	***Seller***
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 views.

Table 2: Views related to the class Article

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

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 3: Composition of viewpoints in terms of views

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

Within the proposed approach, the Petri network associated with the class Article is illustrated in Figure 2 below.

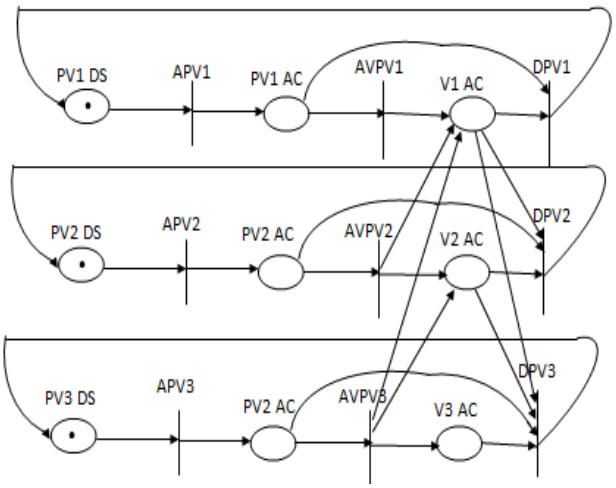

Figure 2: Petri network associated with the multiview class Article

6.2 Example 2: Multiview Class Course

This multiview class 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	**Tutor**	**Responsible**
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

ViewV1	**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

PV1: Student	**PV2: Tutor**	**PV3: Responsible**
V1 + V2 + V3	V1 + V3 + V4	V1 + V4

Under our approach, the Petri network associated with the class Course is shown in Figure 3 below.

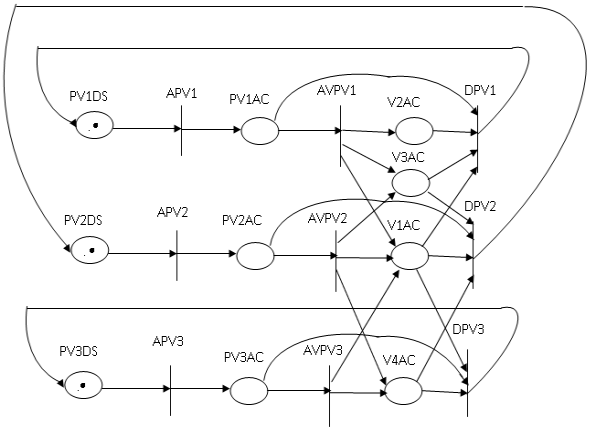

Figure 3: Petri network associated with the multiview class Course

7. Conclusion and perspectives

In this paper we presented an approach for modeling a multiview class by 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 Petri network. Moreover, we applied our approach on an example, namely the multiview class Article that supports three different viewpoints.

In our future works, an attempt will be made to extend our approach so as to systematically translate a multiview classes diagram to a Petri network. Our perspectives also include the use of colored Petri networks to reduce the complexity of a regular network as well as the use of simulation tools of Petri networks such as CPNTool. Another perspective might be to propose a new diagram in the UML object modeling language to represent multiview classes.

References

[1] Booch G., Rumbaugh J., Jacobson J., Le guide de l'utilisateur UML, http://www.omg.org/spec/UML/2.4.1/2000.

[2] Carré, B. Dekker, L. et Gei b, J., Multiple and Evolutive Representation in the ROME Language, TOOLS2, p.101-109, 1990.

[3] Clarke, S., Extending standard UML with model composition semantics, Science of Computer Pogramming, Elsevier Science, 2002.

[4] 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.

[5] Debauwer, L. Caron, O. Carré, B., Contextualization of OODB Schemas in CROME., 11th International Conference DEXA septembre 2000, London.

[6] Desmond, S. and Wills, S., Objects, Components and Frameworks With UML : The Catalysis Approach, Addison-Wesley, 1999.

[7] Ettalbi A., Sbihi B., Hair A., La modélisation des classes multivues par les réseaux de Petri, Conférence Internationale sur les Mathématiques Appliquées et les Sciences de l'Ingénieur, CIMASI'2002, 23-25 Octobre, 2002, Casablanca, Maroc.

[8] Gottlob, G. Schrefl, M. et Rock, B., Extending Object-Oriented Systems with Roles, ACM Transactions on Information Systems (TOIS), page 268-296, 1996.

[9] Hair A., Sbihi B., Ettalbi A., Object-oriented modeling by viewpoint using UML, Advanced Modeling and Optimisation, Volume 5, Number 2, pp : 107-115, 2003.

[10] Kiczales G., Lamping J., Mendhekar A., Maeda C., Videira Lopes C., Loingtier J.-M., and Irwin J., Aspect-Oriented Programming, In European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag LNCS 1241. pp. 220-42, june 1997, Finland.

[11] 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.

[12] Le Moigne J.L., la modélisation des systèmes complexes, Dunod, 1990.

[13] 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.

[14] Mili, H. Dargham, J. Mili, A., Views : A Framework for Feature-Based Development and Distribution of OO Applications, Proceedings, Thirty-Third Hawaii International Conference on System Sciences, Honolulu, HI, january 2000.

[15] http://web.mit.edu.

[16] Nassar, M. Kriouile, A. Coulette, B., Programmation par objets et points de vue – le compilateur VBOOL, 6e Conférence Maghrébine des Sciences Informatiques MCSEAI'2000, Fès, Maroc, novembre 2000.

[17] Nassar, M., Analyse/Conception par points de vue : le profil VUML, Thèse de l'Institut National Polytechnique de Toulouse, 2005.

[18] OMG, Unified Modeling Language (UML), version 1.4, OMG Document formal/2001-09-07, septembre, http://www.omg.org/cgi - bin/doc?formal/01-09-67.

[19] Ossher, H. Kaplan, M. Harrison, W. Katz, A. and Kruskal, V., Subject-oriented composition rules, in Proceedings of OOPSLA'95, Austin, TX, p. 235-250, 1995.

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

[21] Rundensteiner, A., A Classification Algorithm for Supporting Object-Oriented Views, Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM' 94), pages 18-25, Gaithersburg, Maryland, USA, ACM Press, 1994.

[22] Sbihi B., Kriouile A., Ettalbi A., Hair A., Marzak A., **La génération du multicible en VBOOM**, Conférence Internationale sur les Mathématiques Appliquées et les

Sciences de l'Ingénieur, CIMASI'2002, 23-25 Octobre, 2002, Casablanca, Maroc.

[23] Sbihi B., Kriouile A., Ettalbi A., Coulette B., **Toward a generation of code multi-target for the VBOOM method**, International Conference on Software Engineering, Research and Practice, 23-25 June 2003, Las Vegas, SERP'03, USA.

[24] Sbihi B., Kriouile A., Ettalbi A., Marzak A., **L'implémentation en UML du diagramme final de VBOOM**, The International Conference on Image and Signal Processing, ICISP'2003, 25-27 Juin, Agadir, Morocco.

[25] Sbihi B., Kriouile A., Ettalbi A., Nassar M., **The implementation in UML of the points of view's notion in a Distance Education System**, The 4th International Conference on Information Technology Based Higher Education and Training, ITHET'2003, 7-9 July, Marrakech, Morocco.

[26] Sbihi B., Hair A., Ettalbi A., L'implémentation en UML du diagramme final de VBOOM, The International Conference on Image and Signal Processing, ICISP'2003, 25-27 Juin, Agadir, Morocco.

[27] Sbihi B., Hair A., Ettalbi A., The implementation in UML of the points of view's notion in a Distance Education System, The 4th International Conference on Information Technology Based Higher Education and Training, ITHET'2003, 7-9 July, Marrakech, Morocco.

[28] Tropes, Tropes 1.0 reference manual, INRIA Rhônes-Alpes IMAG-LIFIA, Grenoble, France, 1995.

[29] Vanwormhoudt, G., CROME : un cadre de programmation par objets structurés en contextes, PhD thesis, Laboratoire d'Informatique Fondamentale de Lille I, Lille, 1999.