UDC:004.415.2
Original scientific paper

A UML PROFILES FOR VIEWPOINT-ORIENTED MODELING

Abdellatif Hair
Faculty of Sciences and Technics, Beni-mellal, Morocco
hair@fstbm.ac.ma

Abstract: *The viewpoint-oriented modeling consists in providing an unique model accessible by users with various access rights according to their needs. The system is represented as sub-systems (called sub-models). This cutting makes the design of system easier by an autonomous design of its sub-systems.*

UML is a standard modeling language and supports extension mechanism to allow tailoring UML to fit the needs of a specific domain. UML Profile is a predefined set of extension mechanisms. UML profile allows the stakeholders of a certain domain to express the semantics of their systems using a well-defined set of extensions. In this paper we propose an initial discussion on UML profile for viewpoint oriented modeling, which extends the standard UML by incorporating new concepts like views, viewpoints, flexible, visibility relationships, etc.

Keywords: *Modeling, viewpoint, view, visibility relationship, profile, UML.*

1. INTRODUCTION

The development of complex software systems involves many users (agents) with different perspectives of the system they are trying to describe. These perspectives, or viewpoints, are usually partial or incomplete. Software engineers have recognized the need to identify and use this multiplicity of viewpoints when creating software systems [1][5][15].

For that, the viewpoint concept has received significant attention and has been used for different purposes such as a filter, subject, role, state, perspective, and view [8][20][2]. We recover these different concepts in object-oriented databases, knowledge representation systems and object-oriented analysis and design methods. In object-oriented analysis and design, one of the results more succeed is the one of VBOOM which is a method integrating the viewpoint approach in the object-oriented modeling [3][11]. And also the work that consists in elaborating an unified oriented-viewpoint method (named U_VBOOM) based on the VBOOM method [6][7].

The Viewpoint-Oriented Modeling (VOM) is concerned with providing mechanisms for explicitly capturing crosscutting user's need, and therefore, enhancing modularity. In VOM, systems are represented as sub-systems (called sub-models). Each one reply a particular user's need. Systems, however, have properties that cut across their functional components increasing their interdependencies. Viewpoint-oriented is a paradigm that complements the

object-oriented concept and helps in avoiding the endless circle of reengineering entire software projects for minor changes.

Since UML is a graphical language for specifying, visualizing, constructing, and documenting the artifacts of software systems [23][24]. UML is a standard modeling language endorsed by Object Management Group (OMG) in 1997 as UML 1.1. UML is the graphical notation that defines the semantics of the object meta-model (model of UML itself), and defines a notation for capturing and communicating object structure and behavior. UML supports extension mechanisms [19][21][23][24] (stereotypes, tagged values, and constrains) to allow tailoring UML to fit the needs of a specific domain. UML Profile [16][10][12] is a predefined set of extension mechanisms. UML profile allows the stakeholders of a certain domain to express the semantics of their systems using a well-defined set of extensions. The Profile does not extend UML by adding a new concept. Instead, it provides connections for applying and specializing UML to a specific domain.

This paper addresses VOM at the analysis and design phases. The purpose of the paper is to present a UML profiles for VOM. The viewpoint concept will be represented as standard object-oriented artifacts, which may be integrated in object-oriented modeling. We will use an UML-like notation [19] to illustrate our subjects.

The paper is organized as follows: section 2 describes and defines the terms used in the VOM. Section 3 discusses briefly the UML profiles. The proposed UML profiles for VOM is described in section 4 and this section gives an example that illustrates the practicality of the VOM. Finally, section 5 presents our conclusions.

In this paper, we are going to illustrate our subjects with the **Media Library** Management example. The specifications of this example are more explained in [13]. The **Media Library** system must allow its members to consult and to borrow various types of support: books, video and audio disks, audio CD, etc. Only one member of the library can borrow books, reviews, etc. The borrow is limited in time. The potential users of the **Media Library** system are: the **librarian** who manages the loans, the **adherents responsible** who will add and withdraw members, the **exemplaries responsible** who will seize the new exemplaries and to withdraw those damaged, and finally the **system engineer** who ensures the good exploitation of system for the users.

2. VIEWPOINT-ORIENTED MODELING

The development of complex software systems requires the building of several partials models, each one reply a particular user's need. The use of multiple viewpoints in complex system modeling provides a big profit. But each author deals differently with those notions. Our approach in VOM consists in providing an unique model accessible by users with various access rights according to their needs and supporting multiple and evolutive viewpoint. This approach guarantees knowledge centralization, consistency checking and non-redundancy information and supports multiple and evolutive viewpoint. We will first define briefly the visibility relationship and its derived concepts.

By visibility, we mean that an entity can be seen upon several angles. It's the fact of a **Media Library** system, which can be studied under the Exemplaries responsible angle, Adherents responsible or Librarian one. The syntax and the semantic of this concept is detailed in [14], we summarize here the principal characteristic.

A **view** is an abstraction of the model. It constitutes the unity of visibility; it is the result of factorizing user's needs.

A viewpoint is the sight that has a user to the model. It 's a combination of views.

A **flexible class** is a class that declares more than two views; its instances, which must specify a particular viewpoint, take various appearances. The Media_Library class is a flexible class, which owns 3 views (Loans, Exemplaries, Counts_Adherents).

A sub-**model** is a partial model or sub-system, operating different knowledge of system. Each one concerns a specific viewpoint. In the **Media Library** system, we distinguish three sub-models***:*** *Loans management*, *Exemplaries management*, and *Adherents management*. Those sub-models are associated respectively to viewpoints descended of three actors: *Exemplaries responsible*, *Adherents responsible* and *Librarian*.

3. UML PROFILES

UML [23] is the graphical notation that defines the semantics of the object meta-model (model of UML itself), and defines a notation for capturing and communicating object structure and behavior. In its meta-model architecture, UML supports extension mechanisms that allow tailoring it to fit the needs of a specific domain. These extension mechanisms are the means by which UML can adapt to new technologies, domains and methodologies. The Object Management Group (OMG) defines UML modeling language in a four-layer architecture as shown in Figure 1 [23][24]. The meta-meta-model level defined by the MOF (meta object facility), the UML meta-model is defined and standardized on top of the MOF, the meta-model layer specifies the modeling language, the model layer defines the model, and the user object layer defines instances of the model. The standard meta-models represent areas under which specific domains exist that require a specialization of their domain. In order to provide mechanisms for defining domain standards specializing the standard meta-models, UML introduces the notion of Profile, which is specified in white paper [16]. UML provides the means by which one can extend it's meta-model (model of UML itself) to fit the needs of a modeling concern. It provides extension mechanisms that allow customizing and extending UML model elements (i.e. <<class>>). The UML extension mechanisms are stereotype, tagged values, and constrains. Stereotypes are classification of an existing UML model element, which has its own properties that are expressed as tagged values, constrains are restrictions placed on the stereotypes. UML profiles are predefined set of stereotype, tagged values, constrains, and graphical icons to allow modeling a specific domain.

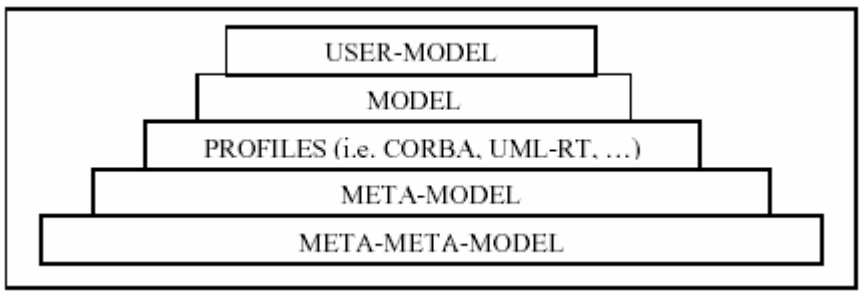

Figure 1: UML architecture

There have been many UML profiles proposed to OMG, UML profile for real time systems UML-RT [23] is a UML extension to support Real time intensive application, the profile itself is based on concepts from [22] the profile introduces the notions Capsules, ports, connectors, protocols, and protocol role to UML. CORBA Profile [17] CORBA allows applications to communicate with each other regardless of the location and design, the UML Profile for CORBA provides a standard means for expressing the semantics of CORBA IDL using UML artifacts, which enable expressing these artifacts with UML tools. UML Profile for CORBA has been adopted by OMG in October of 2000.

In the following section we show our proposed UML Profiles to support viewpoint-oriented software development. Our proposed Profiles introduce basic VOM terminology to UML to enable the visual representations of viewpoint-oriented systems in the design models.

4. UML PROFILES FOR VIEWPOINT-ORIENTED MODELING

UML profiles for VOM will provide conventions and guidelines for applying and specializing standard UML to the graphical notation for VOM (Specialize UML meta-model for VOM). It would help in capturing every desired terminology of VOM and provide "native" support for VOM in UML. The profiles will set a common means of expressing VOM artifacts visually for all VOM stakeholders. We need to create a set of extensions (stereotypes, tagged values, and constraints) to support VOM and not a particular implementation of VOM. We believe that the extension mechanisms are capable of adapting UML semantics to express VOM terminology without changing the UML meta-model itself.

4.1. STEREOTYPES AND RELATIONSHIPS FOR VIEWPOINT-ORIENTED MODELING

Table 1 starts the list of stereotypes that we suggest for VOM. The 1st and 2nd row define <<flexible>> and <<view>> as stereotypes. These are stereotypes of the UML "Class" model element. The <<flexible>> stereotype is used to indicate a multi-views class. The <<view>> stereotype is used to indicate that a class is view of flexible class. Viewpoints or use types of an agent (user) through the system are concerns that cut across a system's main components. These components are called sub-models. These sub-models are sub-systems of system's main. The VOM attempts to modularize sub-models and uses an efficient process to melt those sub-models in order to get a coherent global model. For that the 3rd row of Table 1 defines <<sub-model>> as a stereotype, this stereotype is used to indicate that the use type of an agent through the system classifies packages (sub-systems).

Table 1: Candidates stereotypes for VOM

Base UML meta-model element	Tagged Value	Explanation
class	<<flexible>>	The <<flexible>> stereotype is used to indicate a flexible class (Figure 2).
class	<<view>>	The <<view>> stereotype is used to indicate that a class is view of a flexible class (Figure 2).
package	<<sub-model>>	The <<sub-model>> stereotype is used to indicate that the packages are classified by the use types of agents.

Table 2 starts the list of relationships that we suggest for VOM. The UML generalization relationship and the <<seen_as>> stereotype represent the relation between a flexible class and its views. This relation is named visibility relationship. Each flexible class has at least two-visibility relationship because the flexible class is a class that declares more than two views.

A View might have dependency relation over another view, that is to say, when the information of view source depends of other information of view target. If the design does not specify dependency relation, this means that the views are not dependent. To specify

that a view has dependency relation than another, this extension extends the UML "association" meta-model element with the stereotype <<view_dependency>> linking two views. The association's arrow comes from the view source to the view target. Table 2 shows the <<view_dependency>> stereotype. This mechanism of dependence between the views is called repercussion of modification.

Table 2: Candidates' relationships for VOM

Base UML meta-model element	Tagged Value	Explanation
generalization	<<seen_as>>	It's visibility relationship. This relation can be used between flexible class and its views. The relation arrow points from the flexible class to its views. For example, Media_Library is flexible class and it is linked to its 3 views Loans, Exemplaries, and Counts_Adherents by the visibility relationship (Figure 2).
association	<<view_dependency>>	This relation can be used between two views. The relation arrow points from the view source to the view target. For example, when an adherent does a loan, numbers of Exemplaries available to the library must be modified. Then, the association with <<view_dependency>> stereotype must be used between two views Loans and Exemplaries (Figure 2).

The extension mechanisms are based mainly on the stereotype concept, which provides a way of classifying elements so that they behave as if they were instances of the new meta-model. For example when defining <<flexible>> as a stereotype of the classifier <<class>> it is guaranteed that <<flexible>> behaves the same way if it was created as an instance of <<class>>. The newly constructed stereotype becomes a model element and can participate in all relationships a model element can (i.e. Dependency relationship, generalization, etc.).

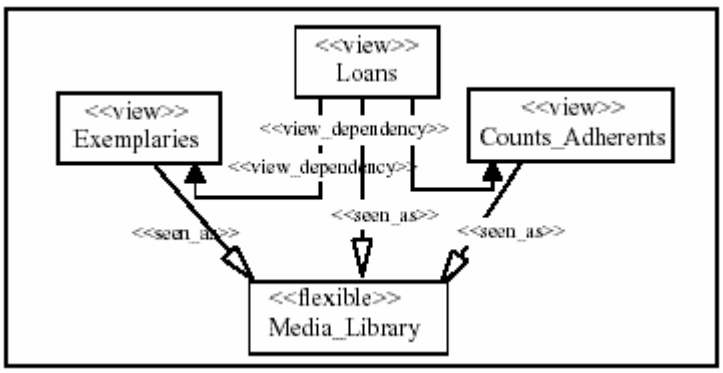

Figure 2: Flexible class and its views representation

3.2. ANALYSIS AND DESIGN PROFILES FOR VIEWPOINT-ORIENTED MODELING

In this section, we propose the practicality of VOM systems, we present it on the analysis and design profiles and we apply it on the **Media Library** example. The elaboration of these two profiles is based on the U_VBOOM method. U_VBOOM is an analysis/design method based on UML, which integrates the viewpoint approach in a coherent and deductive process [6][7]. U_VBOOM proposes two kinds of models: static and dynamic one to describe the system and its behaviors. U_VBOOM provides users the possibility to:

- deal strategically with problem space by identifying user's needs,
- improve the interaction expert/user and designer,
- establish sub-models (partials models),
- generate an unique model integrating different sub-models.

3.2.1. Analysis profile

The object of Analysis Profile is to define model components. In this profile, the analyst begins by elaborating the use cases diagram for system (Figure 3). Then, he identifies the different viewpoints (viewpoint per agent) and their combination of views. Three viewpoints are identified for **Media Library** system: *Librarian_viewpoint, Exemplaries_responsible_viewpoint,* and *adherents_responsible_viewpoint.*

The views elaboration of system is supported by the use cases description technique [9][7]. This technique consists in describing the use case as an action sequence that will make an agent to achieve his goal (Table 3) [4]. An action is an effect produced by an agent according the given way on the system or by the system itself. The use cases identified for agents and the intersection of their actions are going to cut the viewpoints in views. A view corresponds to set actions of one viewpoint or the intersection actions result of viewpoints group (Figure 4).

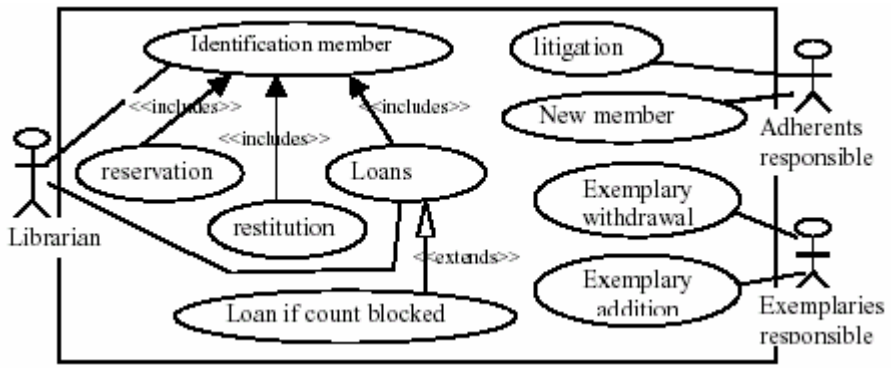

Figure 3: Use case diagram of MEDIA LIBRARY system

Table 3: Use cases decomposition in actions

Agents	Use case	Decomposition in actions
Librarian	Loan	a1: To identify an adherent a2: Count adherent (To check right loan for member) a3: To seek for an exemplary a4: To treat an exemplary (to validate the output of an exemplary) a5: To treat adherent (to indicate the loan by adherent)
	Reservation	a1: To identify an adherent a2: Count adherent (To check right loan for member) a3: To seek for an exemplary a6: To reserve an exemplary
	Restitution	a1: To identify adherent a3: To seek for an exemplary a4: To treat an exemplary (to validate the input of exemplary) a5: To treat adherent (to indicate the return of an exemplary)
	Loan if counts blocked	a1: To identify adherent a2: Count adherent (To check right loan for member)
	Identification member	a1: To identify adherent
Adherents responsible	New adherent	a2: Adherent account (to Add adherent)
	Litigation	a1: To identify adherent a2: Count adherent (Blocked count adherent) a7: To inform adherent
Exemplaries responsible	Exemplary addition	a3: To seek for an exemplary a8: To add copy
	Exemplary withdrawal	a3: To seek exemplary a9: To withdraw the damage exemplary

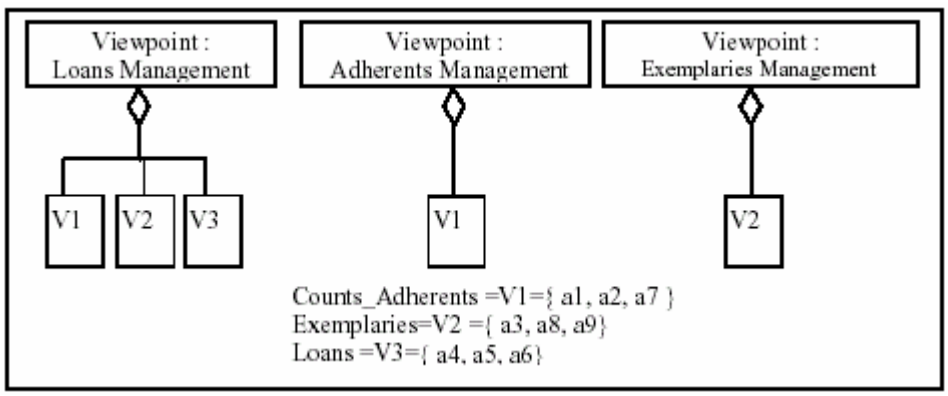

Figure 4: Viewpoint diagrams of MEDIA LIBRARY system

After, the Analysis Profile must enable the analyst to acquire scenarios as collaboration diagrams for each use case. The Collaboration diagrams concentrates on the structure of interaction between objects and their inter-relationships rather than focuses the temporal dimensions of a scenario. Thereafter, the potential classes of our system are identified and

the flexible classes are selected. At the end, the initial dictionary is elaborated and contains simple and flexible classes and views (Figure 5).

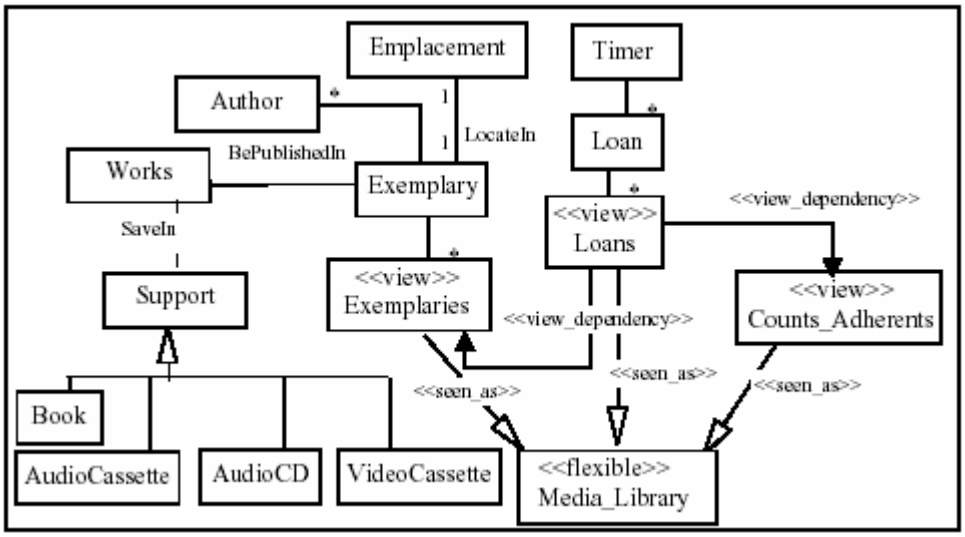

Figure 5: Initial class diagram of MEDIA LIBRARY system

3.2.2. Design profile

The analysis has been done with guidelines of the Analysis Profile. The Analysis Profile is enriched and is refined under the control and guidelines of the Design Profile.

The Design Profile must permit firstly to make the sub-models design, secondly to melt and to validate these sub-models. These sub-models constitute an essential artifact on the Design Profile.

3.2.2.1. Sub-models design

The Design Profile must allow the designers to develop their sub-models commonly or not. Three sub-models are identified for **Media Library** system: *Loans management*, *Exemplaries management*, and *Adherents management*. The designers must develop the static and dynamic representation for each sub-model. The Design Profile proposes the class diagrams to describe the static representation. Each class in the class diagram defines its own static structure (Interface) including attributes and methods. If a class is present in different sub-models, then its static structure corresponds to a partial description. For example, **Media_Library** flexible class of our system possesses a partial description in three sub-models. For the dynamic representation, the Design Profile proposes the sequence diagram and statechart diagram. A sequence diagram shows interactions among a set of objects in temporal order, which is good for understanding timing issues. A statechart diagram shows the sequence of states that an object goes through during its lifecycle in response to stimuli. Generally a statechart diagram may be attached to a class of objects with an interesting dynamic behavior. At the end, for each sub-model a partial dictionary is elaborated. Figure 6 represents the class diagram (partial) of Loans Management sub-model.

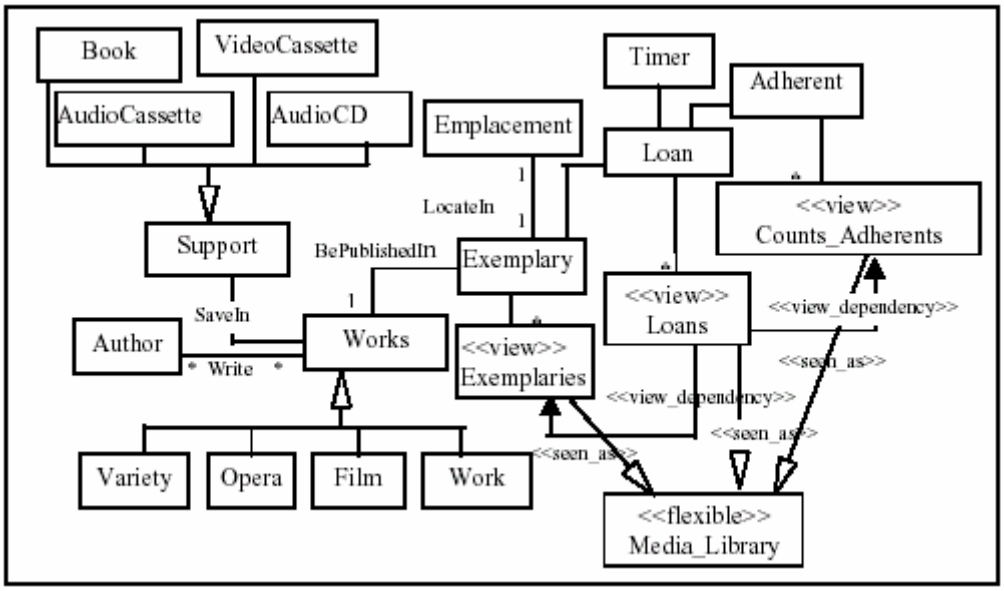

Figure 6: Class diagram of Loans Management sub-model

3.2.2.2. Melting and validation of sub-models

The Design Profile must also enable the designer to generate a coherent document (global model, dictionary of sub-models). That is to say, to allow the designer to melt partial interfaces, resolve conflicts and identify modifications to reverberate between classes of sub-models and provide the global architecture of system. To lead this operation of melting, an efficient process is purposed. This process is based on the following rules (A more detailed description of this process is in [7][11]):

- Same concepts must be represented by the same names: The melting rules will use names to define the semantics of the involved concepts. Thus, if two different sub-models had names Adherent and Member to represent the same concept, a melting procedure would have to be used to rename one of them. The renaming can be applied to classes, attributes, and methods.
- Attribute types must be consistent: attributes representing the same concept should also have the same type. That means that if the same attribute is an integer in one sub-model and a float in other, the melting rules will deal with these variations.
- Cyclic hierarchies must be avoided: the melting of sub-models cannot generate an inheritance cycle. If there is a sub-model-i within a certain class hierarchy such that class-sub is a subclass of class-super and a sub-model-j in which class-super is a subclass of class-sub there is an inconsistency and the melting rules can be applied.

At the end, the Design Profile must enable to validate the document and check coherency between global model and interfaces.

In our example, the global model is relatively identical to the **Loans Management** sub-model because this last is predominant, but this situation is not a generality (Figure 7).

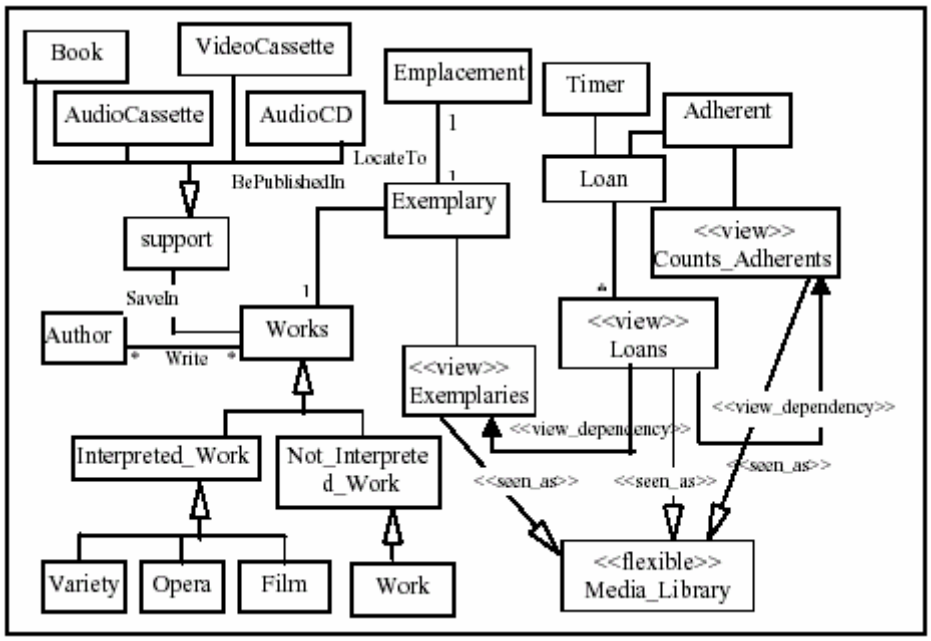

Figure 7: Class diagram of MEDIA LIBRARY system

4. CONCLUSION

In this paper, we have argued for a UML profile for viewpoint-oriented software development. We have proposed the Analysis and Design Profiles, which are based on U_VBOOM method. These profiles extend the standard UML by incorporating new concepts like views, viewpoints, flexible, visibility relationships, etc. to object-oriented technology.

In VOM, establishing sub-models is important as it makes global model design easier. Moreover, the sub-models melting are an efficient and automatic.

The Rational Rose is used to support the proposed extension. Rational Rose provides several ways for extending and customizing its capabilities to meet specific software development needs. A detailed discussion of Rational Rose Extensibility Interface can be found in [18]. We used Rose Script to generate a VBOOL skeleton for the class diagram built using our proposed extension.

The work describes herein is part of a project which define a methodology of development of components multi-views objects. Among the tasks remaining to achieve in this project, we can mention:

- the definition of the composing multi-views notion as regrouping of multi-views classes,
- the development of design pattern supporting the viewpoints approach.

REFERENCES

[1] Ainsworth M, Cruickshank A H, Groves L J, Wallis P J L. Viewpoint specification and Z. Information and Software Technology 1994; 36(1):43-51.
[2] Carré B, Geib J. The Point of View notion for Multiple Inheritance. In Proceedings of the ECOOP/OOPSLA, 1991.
[3] Coulette B, Kriouile A, Marcaillou S. Viewpoints approach in the object-oriented development of complex systems. Object Review 1996; 2(4): 13-20.
[4] Dano B, Briand, H, Barbier F. An Approach Based on the Concept of Use Cases to Produce Dynamic Object-Oriented Specifications. In Proceedings of the Third IEEE International Symposium on Requirements Engineering 1997.
[5] Filkelstein A, Gabbay D, Hunter A, Kramer J, Nuseibeh B. Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. International Journal of Software Engineering and Knowledge Engineering, 1993; 2(1):31-58.
[6] Hair A: Analysis and design process based on the viewpoint concept. RITA - Revista de Informática Teorica e Aplicada -, vol. X n.2, 2004, pp 63-75.
[7] Hair A, Sbihi B, Ettalbi A: Object-oriented modeling by viewpoint using UML ; AMO - Advanced Modeling and Optimisation - journal , vol. 5 n. 2, 2003, pp 107-115.
[8] Harrison W, Ossher H, Kaplan M, Katz A. Subject-oriented programming: a critique of pure objects. In Proceedings of OOPSLA 1993; 28(10): 411-428.
[9] Jacobson I, Christerson M, Jonhson P, Overgaard G. Object-Oriented Software Engineering, A Use Case Driven Approach, Addison-Wesley, 1992.
[10] Jacobson I, Booch G, Rumbaugh J. The unified software development process. Addison-wesley, 1999.
[11] Kriouile A. VBOOM: Analysis and Design object method based on par objet on the viewpoint concept. Ph.D. thesis faculty of sciences, Rabat 1995.
[12] Krutchen P. The Rational Unified Process - An Introduction. Addison-Wesley 2003.
[13] Lopez N, Migueis J, Pichon E. To integrate UML in your projects. Edition Eyrolles, 1998.
[14] Marcaillou S. , Integration of the viewpoint concept in the object modeling; The VBOOL language. Thesis of Paul Sabatier university, Toulouse 1995.
[15] Mili H, Dargham J, Mili A, Cherkaoui O, Godin R.View programming of OO applications. In Proceedings of TOOLS 1999.
[16] OMG White Paper on the Profile mechanism V1.0. OMG Document ad/99-04-07.
[17] OMG, UML Profile for CORBA Specification V1.0, OMG Document FormaI/02-04-01, 2001
[18] Rational Rose 2003. Rose Extensibility User's Guide. Copyright © 2003 Rational Software Corporation.
[19] Rumbaugh J, Jacobson I, Booch G, The Unified Modeling Language Reference Manual, Second Edition. Addison-Wesley, 2004.
[20] Shilling J, Sweeny P. Three Steps to Views. In Proceedings of OOPSLA 1989; 1:353-361.

[21]Sinan S.A., Extending UML. Distributed Computing (1998); (1):28-32.

[22]Slic B, Gullekson G, War P. Real-Time OO Modeling. John Wiley &sons, 1995.

[23]UML Syntax and Semantics Guide V2.0, OMG Document format, http:// www.rational.com/UML, 2004.

[24]Unified Modeling Language, Version 2.0. Object Management Group, http://www.omg.org, 2004.

Received: 22 November 2004
Accepted: 16 December 2005