Analysis and design process based on the viewpoint concept

HAIR Abdellatif

Abstract: The introduction of viewpoint in object-oriented design provides several improvements in modeling complex systems. In fact, it enables the users to build a unique model accessible by different users with various points of view, instead of building several sub-systems whose management is too hard to complete. Those concepts of view and viewpoint were implemented by VBOOL (View Based Object Oriented Language), the language which proposes a new relationship "the visibility". VBOOM (View Based Object Oriented Method) analysis and design method, integrates those concepts in an object-oriented modeling. In this work we propose firstly a new representation of the visibility relationship in UML (Unified Modeling Language), secondly the viewpoint oriented method (U_VBOOM based on UML). The latest method is based on the VBOOM method. The new representation of the visibility relationship encourages the multi-targets code generation and improves the development process proposed by the VBOOM method.

1 Introduction

Recently, two aspects have received a lot of attention in object-oriented development: the emergence of UML (Unified Modeling Language) as an unified notation for object-oriented analysis and design, and a growing consensus on viewpoint approaches to object-oriented modeling.

The UML [19] is a graphical language for specifying, visualizing, constructing, and documenting the artifacts of software systems. 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). UML does not define any specific process for software development, beyond some preliminary process description reported, for instance, in [13, 1].

The introduction of viewpoint approaches to software development provides several improvements in complex system modeling [3, 6, 16]. In fact, it enables the users to build a unique model accessible by different users with various viewpoints, instead of building several sub-systems whose management is too hard to complete. The viewpoint concept was first introduced by Shilling and Sweeny [11] as a filter of a global interface of the class, but the views are not separable or separately reusable. Harrison and al [17] proposed subject-oriented programming as a way to build integrated "multiple view" applications by composing application fragments, called subjects, which represent compilable and possibly executable functional slices.

To this effect, the VBOOL language has been defined which integrates the new relation ”the visibility” and its derived mechanisms (language that extend Eiffel) [15]. To implement those concepts in object-oriented methodology, the VBOOM method has been defined which extends the BON’s method [4, 12]. VBOOM propose 3 stages. The first one defines the system dictionary. The second one allows designer to develop their own model separately. The last stage generates a unique model thanks to the ”melting” process.

The aim of this work is, firstly to propose a new representation of the visibility relationship in UML [9, 1, 19]. Secondly, to propose the viewpoint oriented method U_VBOOM based on UML. The definition of the latest method is essentially based on VBOOM method.

The outline of the paper is as follows. In section 2, we define briefly the visibility relationship and its derived concepts in UML. Section 3, we present the 3 stages of U_VBOOM. Finally, briefs conclusions are given in section 4.

We are going to illustrate our subjects with the Media library Management example. The specifications of this example are more explained in [14]. The Media library system must allow its members to consult and to borrow various types of support for example 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 remove members, the exemplaries responsible who will seize the new exemplaries and to remove those damaged. According to the use types, the Media library system will be considered, as a set of COUNTS ADHERENTS, or of EXEMPLARIES, or a means to facilitate the LOANS. Thus, we identify 4 classes of the system, ”Media_Library”, ”Loans”, ”Exemplaries” and ”Counts_Adherents”.

2 The visibility relationship: associated mechanisms and syntax

By **visibility**, we mean that an entity can be seen upon several angles. It’s the fact of a satellite 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 [15], 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. In the Fig. 1, the ”Media_Library” class is a flexible class, which owns 3 views (”Loans”, ”Exemplaries”, ”Counts _Adherents”).

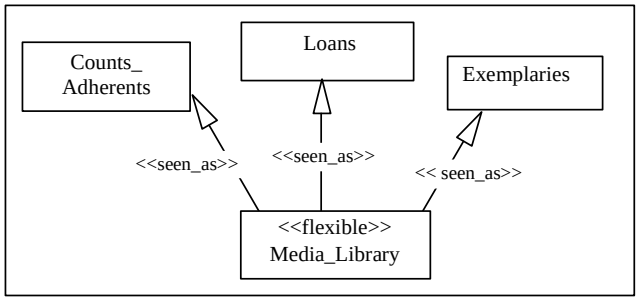

Figure 1. Visibility relationship and flexible class

The graphic representation of the flexible class and the visibility relationship are illustrated respectively by the ”flexible” stereotype added to the class symbol of UML and the stereotype ”seen_as” added to the inheritance relationship symbol of UML (Fig. 1).

The syntax proposed by VBOOL to declare views is the following:

```
flexible class Media_Library
  feature...
-- declaration of views
     seen_as Loans
     seen_as Exemplaries
     seen_as Counts_Adherents
.....
end class -- Media_Library
```

In flexible class the definition of new features implies specification of their export status. Some features can be exported to some client’s classes, other should be used only by specific instances with particular viewpoint on the flexible class. This restriction of visibility is expressed by the ”view_feature” keyword.

```
flexible class Media_Library
feature...
-- Views declaration
...
-- View feature declaration
view_feature Loans, Exemplaries, Counts_Adherents
     display do... end
...
end class -- Media_Library
```

Instances of a flexible class are defined by specifying a viewpoint as follows:

```
Media_library_librarian: Media_Library (Loans, Exemplaries,
 Counts_Adherents);
```

The ”Media_library_librarian” is an instance of ”Media_Library” having the viewpoint (Loans, Exemplaries, Counts _Adherents). Features declared in ”Loans”, in ”Exemplaries” and in ”Counts _Adherents” classes and those specified in ”view_feature Loans, Exemplaries, Counts _Adherents” or in ”feature” clauses of ”Media_Library” class are accessible by ”Media_library_librarian” instance.

The object to use the viewpoint concept is to define access rights to the model. If we consider librarian’s access for example the views ”Loans”, ”Exemplaries” and ”Counts_Adherents” concern him. Contrary to the adherents responsible, the librarian must not have the right to change the adherent’s feature *block_count_adherent*, so to call this feature of the ”Counts_Adherents” view. To solve this problem, the mutual exclusive views concept has been introduced, that means to specify the views that cannot be simultaneously in the same viewpoint. Thus, in the Media library Management example, we can create two views inheriting from ”Counts_Adherents”: the views ”Mod_Counts_Adherents” and ”Not_Mod_Counts_Adherents”. These two views are in mutual exclusive (Fig. 2). That is to say ”Mod_Counts_Adherents” view that will have access the exemplaries responsible and ”Not_Mod_Counts_Adherents” view that will have access the librarian. Of the same way, the librarian must not have the right to change the *number_available_exemplary* feature of a ”Exemplaries” view, contrary to the exemplaries responsible who can add new exemplary or to remove the damaged exemplaries by invocation the two features *add_exemplary* or *remove_exemplary* of the ”Exemplaries” view. Thus, we can also create two views inheriting from ”Exemplaries”, ”Mod_Exemplaries” view that will have access the exemplaries responsible and ”Not_Mod_Exemplaries” view that will have access the librarian (Fig. 2).

VBOOL propose others concepts such as:

- visibility derivation which enables a class to see a flexible class upon a particular viewpoint,
- viewpoint evolution, polymorphism.. etc.

3 Stages of U_VBOOM method

VBOOM is an analysis and design method, which integrates the multiview approach in a coherent and deductive method [4, 12]. VBOOM provides users the possibility to:

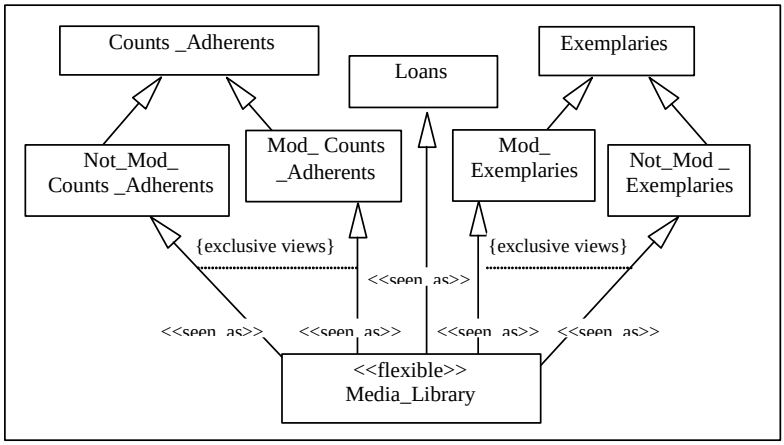

Figure 2. Example of mutual exclusive views

- deal strategically with problem space by identifying user's needs,
- improve the interaction expert/user and designer,
- establish partial models, operating different knowledge of the system. Those models, which make design easier, are called "model's view". Each one concern a specific viewpoint,
- generate a unique model integrating different view models.

Like has been made for the OMT [18], OOSE [10], BOOCH [2] methods. The VBOOM method must take in account the standard UML, i.e. to integrate the concepts and the notations used in the unified modeling language in the VBOOM method, by using its possibilities of specialization and extension [8, 7]. The U_VBOOM method presents an adaptation of VBOOM to UML. The development model of U_VBOOM is iterative, incremental, and piloted by the use case [10, 9]. U_VBOOM method proposes two kinds of models the static and dynamic one to describe the system and its behaviors. The establishment of those models is an incremental process of 3 stages. The first stage consists of defining model components, the second one enables designer to develop their models commonly or not, The final stage has the goal to generate a coherent document (global model, dictionary of components, scenarios).

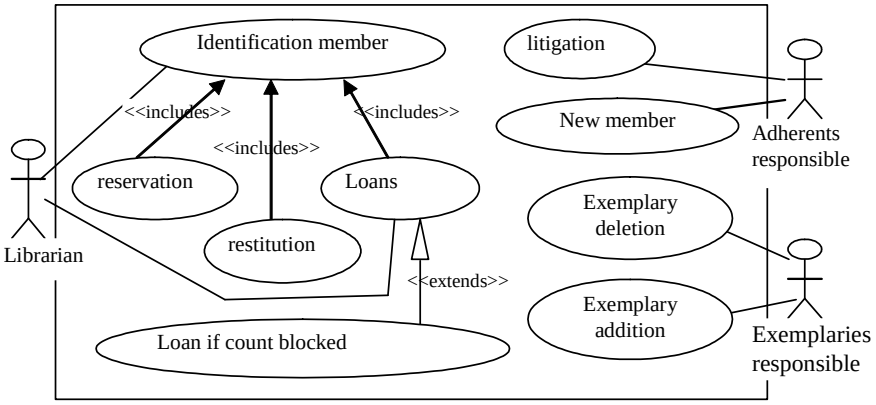

Figure 3. Use case diagram of MEDIA LIBRARY system

3.1 Stage1: global analysis

The object of the first stage is to define model components. It is a stage of global specification of development by U_VBOOM. It consists in providing a precise description of the different needs of the system users.

This stage begins by elaborating the use cases diagram for the system that consists to identify use cases and agents (users) interacting with (Fig. 3). The views elaboration of the system is supported by the use cases description technique [8, 7]. This technique consists to describe the use case as an action sequence that will make an agent to achieve his goal. An action is an effect produced by an agent (actor) acting in way given on the system or by the system itself [5]. Every action has a number and a label who are indicated in the "Decomposition in actions" column (Table 1).

The use cases identified for the agents and the intersection of their actions are going to permit to cut the viewpoints in views. A view corresponds to actions list to a given viewpoint or result of actions intersection of viewpoints group (Fig. 4).

The global analysis stage continues to identify objects and classes belong to the problem domain. Scenarios are acquired as collaboration diagrams for each use case. Collaboration diagrams concentrates on the structure of the interaction between objects and their inter-relationships rather than focuses the temporal dimensions of a scenario. Thereafter, potential classes of the system are identified and select flexible ones. At the end of this stage, the initial dictionary of components is elaborated. It contains simple and flexible classes and views (Fig. 5).

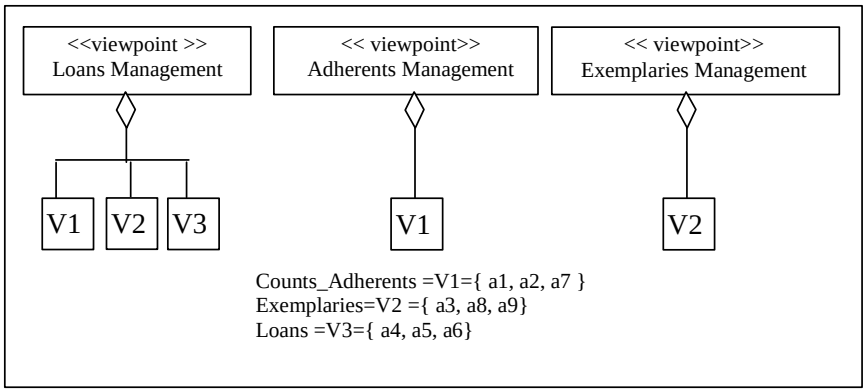

Figure 4. Viewpoint diagrams of the MEDIA LIBRARY system

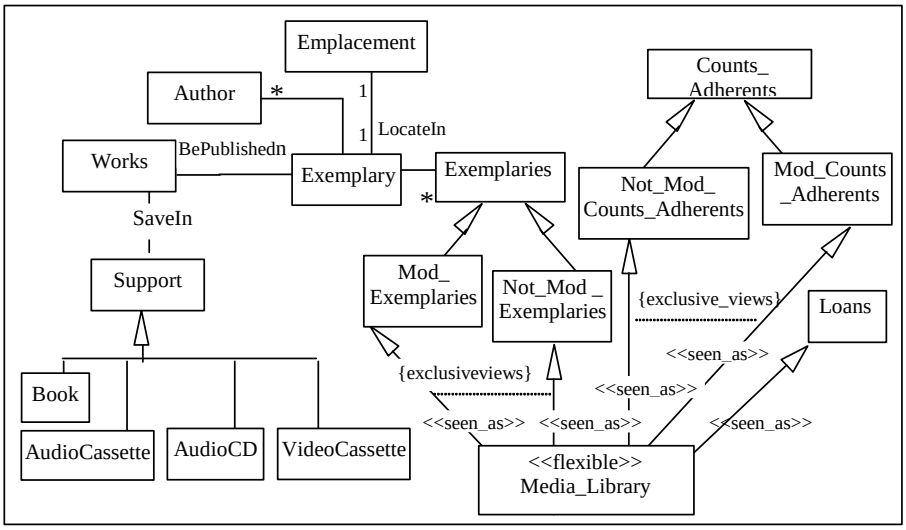

Figure 5. Initial class diagram of the MEDIA LIBRARY system

Table 1. The 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 an exemplary a4: To handle an exemplary (to validate the output of an exemplary) a5: To handle an 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 an exemplary a6: To reserve an exemplary
	Restitution	a1: To identify an adherent a3: To seek an exemplary a4: To handle an exemplary (to validate the input of exemplary) a5: To handle an adherent (to indicate the return of an exemplary)
	Loan if counts blocked	a1: To identify an adherent a2: Count adherent (To check right loan for member)
	Identification member	a1: To identify an adherent
adherents responsible	New adherent	a2: Count adherent (to Add adherent)
	Litigation	a1: To identify an adherent a2: Count adherent (Blocked count adherent) a7: To inform an adherent
Exemplaries responsible	Exemplary addition	a3: To seek an exemplary a8: To add an exemplary
	Exemplary deletion	a3: To seek an exemplary a9: To remove the damaged exemplaries

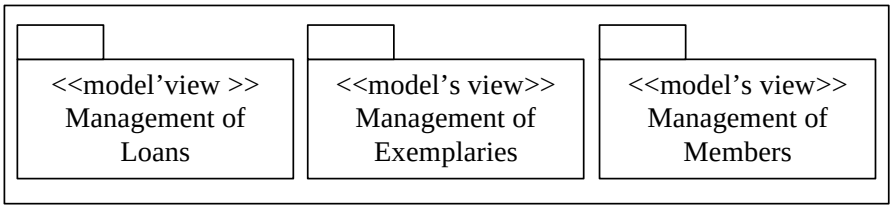

Figure 6. The 3 packages of the MEDIA LIBRARY system

On the final time, the packages can be identified to organize the modeling elements. The identified packages are gotten according to the logical criteria use type of an agent. Those packages are going to become thereafter the sub-systems (named model's views) (Fig. 6).

3.2 Stage2: The sub-systems design

The global analysis of the U_VBOOM method is elaborated and it will be enriched in second stage. The sub-systems (model's views) design permits the translation of the analysis model. These sub-systems constitute an essential artifact of the second stage of the U_VBOOM method. Indeed, the cutting out of the solution space in sub-systems permits to land the global system design to partials designs of the sub-systems. These partial designs of the sub-systems can be made in a disparate and autonomous way and be led in parallel or sequential.

For each sub-system, designer develops during this stage static and dynamic representation. Static representation is described by formal interfaces of classes derived from informal ones. If a class is present in different sub-system then each its interface corresponds to a partial description. Dynamic models are described by 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 of this stage, for each sub-system, partial dictionary of components is elaborated. The Fig. 7 represents the class diagram (partial) of the **Management of Loans** sub-system.

3.3 Stage3: Global model design

The goal of the final stage was to generate a coherent document (global model, dictionary of components). This stage consists of melting partial interfaces, resolving conflict and identifying modifications to reverberate between classes of different sub-system and

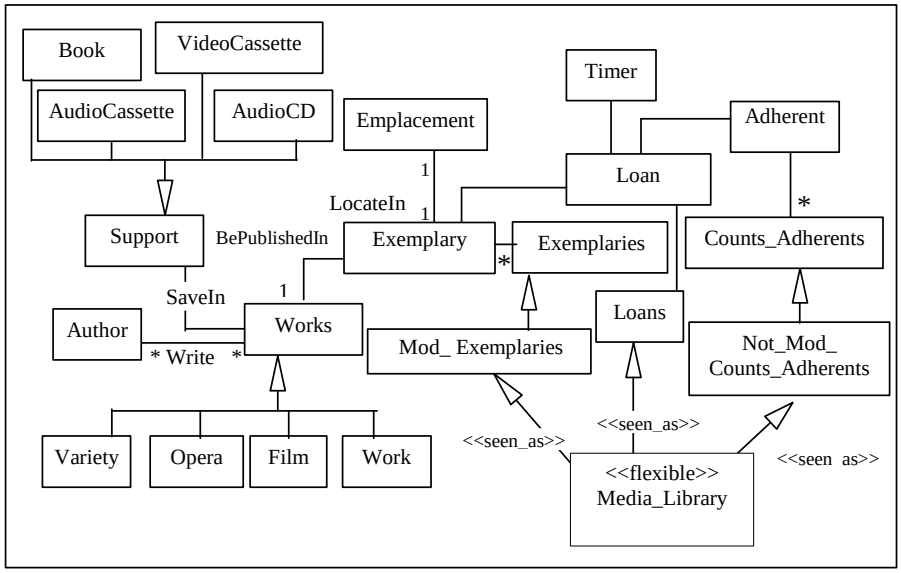

Figure 7. Class diagram of the Management of Loans sub-system

providing the global architecture of system. To lead this stage, the function named "melting_function" is purposed. This function provides an efficient and automatic process to melt sub-system, resolve conflicts and avoid redundancy. At the end of this stage, we validate the document and check coherency between global model and interfaces. During this stage, designers should be spectators unless their intervention is indispensable (Fig. 8). In our example, the global model is relatively close to the Management of Loans sub-system because this latest is predominant in the system, but this situation is not evidently a generality. The sub-systems obtained are tested and validated in order to be coded in the implementation activity.

4 Conclusion

In this paper, we have argued the viewpoint-oriented development. We have proposed the analysis and design method (U_VBOOM). This method extends the standard UML by incorporating new concepts like views, viewpoints, flexible, visibility relationships, etc. to object-oriented technology. In viewpoint-oriented modeling, establishing model's views is important as it makes global model design more easy and reusable. Moreover, model's views melting is an efficient and automatic possible. The U_VBOOM method is an adaptation of the

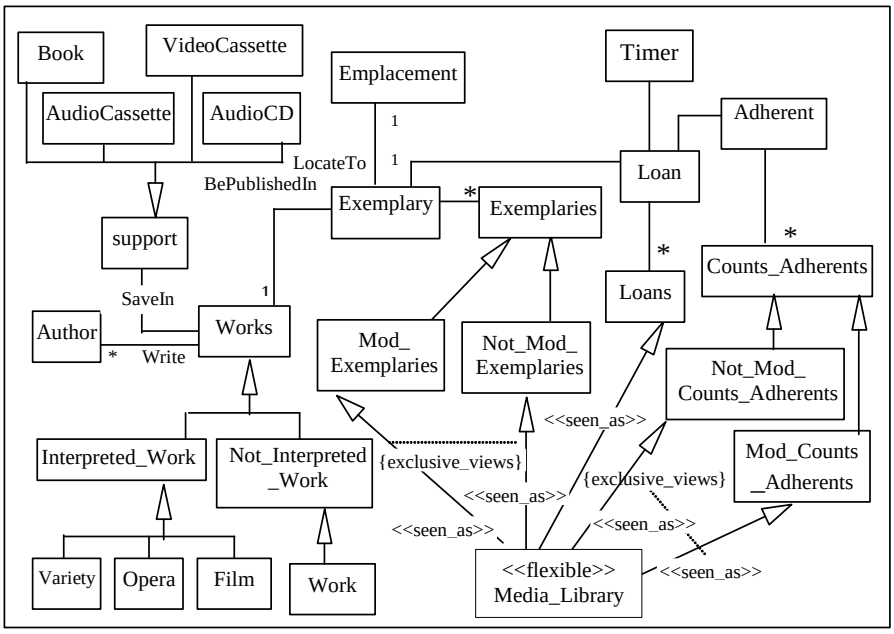

Figure 8. Class diagram of the Media library management

VBOOM method under the UML. The development process of U_VBOOM is incremental, iterative and piloted by the use cases. The decomposition of the system in packages, which became the sub-system (model's view), by the logical criteria use type permitted to land the global system design to the sub-system design.

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

- the definition of the composing multi-views notion as regrouping of multiview classes,
- the development of a basis of design pattern supporting the viewpoints approach,
- the realization of an environment support of U_VBOOM.

References

[1] Unified modeling language (uml), version 1.4, 2001. OMG Document formal/2001-09-07, `http://www.omg.org/cgi-bin/doc?formal/01-09-67`.

[2] G. Booch. *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, Redwood City, 1994.

[3] B. Carré and J. Geib. The point of view notion for multiple inheritance. In *Proceedings of the ECOOP/OOPSLA*, 1991.

[4] B. Coulette, A. Kriouile, and S. Marcaillou. L'approche par points de vue dans le développement orientée objet de systèmes complexes. *Revue l'Objet*, 10(5):13–20, Feb 1996.

[5] B. Dano, H. Briand, and F. Barbier. 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.

[6] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multi-perspective specifications. In *Proceedings of the ESEC'93 Conference*, pages 84–99, Garmish-Paternkirchen, 1993.

[7] A. Hair, A.Kriouile, and B. Coulette. Un processus d'analyse et de conception unifié basé sur le concept de point de vue. In *Proceedings of the 6^{th} Africain Conference on Research in Computer Science (CARI'02)*, pages 229–237, Yaoundé, Oct 2002.

[8] A. Hair, A. Kriouile, and B. Coulette. Vuml : Une méthode d'analyse et de conception orientée objet, intégrant uml et le concept de point de vue. In *Proceedings of the International Conference on Systems, Software Engineering and their applications (ICSSEA'2001)*, France, Dec 2001.

[9] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1999.

[10] I. Jacobson, M. Christerson, P. Jonhson, and G. Overgaard. *Object-Oriented Software Engineering, A Use Case Driven Approach*. Addison-Wesley, 1992.

[11] J.Shilling and P. Sweeny. Three steps to views. In *Proceedings of OOPSLA'89*, pages 353–361, New Orleans, LA, 1989.

[12] A. Kriouile. Vboom, une méthode d'analyse et de conception par objet fondée sur les points de vue, 1995. State thesis of sciences faculty, Rabat.

[13] P. Krutchen. *The Rational Unified Process - An Introduction*. Addison-Wesley, 2000.

[14] N. Lopez and E. P. J. Migueis. *To integrate UML in your projects*. Edition Eyrolles, 1998.

[15] S. Marcaillou. *Intégration de la notion de points de vue dans la modélisation par objets; Le langage VBOOL*. Thesis, Paul Sabatier University, Toulouse, 1995.

[16] H. Mili, J. Dargham, A. Mili, O. Cherkaoui, and R. Godin. View programming of oo applications. In *Proceedings of TOOLS'99*, USA, 1999.

[17] W. H. H. Ossher and A. K. M. Kaplan. Subject-oriented programming: a critique of pure objects. In *Proceedings of OOPSLA'93*, pages 411–428, Washington D.C., 1993.

[18] J. Rumbaugh. *OMT : Modélisation et conception orientées objet*. Prentice Hall, 1995.

[19] J. Rumbaugh, I. Jacobson, , and G.Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.