International Journal of Soft Computing 5 (6): 214-218, 2010
ISSN: 1816-9503

Towards Generic Pattern for Reusing Multiviews Components

M. Hain and A. Marzak
TIM Labo, IGSI Team, Department of Mathematics and Computer Sciences,
Ben M'sik College of Sciences, PB 7955, Casablanca, Morroco

Abstract: VUML is a method of analysis/and conception based on the view of the actor. The main contribution of this method in the UML is the concept of multiviews component which makes it possible to encapsulate the characteristics of components according to various points of view with each point of view corresponding to a given actor. A good re-use of the multiviews components is based on a representation independent of a given platform. This representation can target, thereafter, various specific technologies. The vision is to integrate the multiviews components in the MDA approach. The keys to success are: the compatibility of the platform targeted with the solution generated and the transformation of a multiviews component-based PIM-PSM.

Key words: UML, component, MDA, PIM, PSM, transformation

INTRODUCTION

The new tendency of software development makes it possible to find solution for complex systems which require the intervention of several competences by assembling the existing components (Szyperski *et al*., 2002; Meyer, 2000). A component is a fundamental part which can be seen and handled by several actors (An actor can be a man, a machine or a third program). Modeling driven Actor enables us to encapsulate the characteristics of a business in one component and to provide the services waited of this component according to the active viewpoint of the actor, promoting the re-use culture.

VUML is a method of analysis/and conception based on the actor's view (Nassar *et al*., 2003; El Asri *et al*., 2005). Researchers think that the evolution of a multiviews component depends on finding a suitable representation according to UML2.0 norms. Furthermore, this representation targets various specific technologies, in order to increase the pace of reusing multiviews components.

THE MULTIVIEWS COMPONENTS IN MDA

Concept of the multiviews component: UML2.0 (OMG, 2003a, b) defines a component as a decomposable and reusable unit which interacts with its environment via points of interactions called ports which encompass many interfaces. The latter are characterized by signatures of methods. Moreover, there are two types of interfaces: provided interfaces and required interfaces.

The internal contents of the component should be neither visible nor accessible differently than by its ports. In addition, assembling the components is achieved by means of connectors. In fact, two types of connectors exist: delegation connector and assembly connector. Figure 1 shows the metamodel of UML 2.0 component.

According to (El Asri *et al*., 2005), a multiviews component is an extension of UML 2.0 component. It combines the criterion of reuse via the concept of component and that of flexibility via the muti-view concept. A multiviews component has an internal structure and a whole of ports which structure the points of interaction with the internal and external environment. The specificity of a multiviews component is to provide interfaces whose definition changes dynamically according to an active view. Moreover, one multiviews component is equipped with a mechanism of managing the coherence of different views. Thus, a client handles a component according to a given point of view. He can also enable/disable the views during the execution (El Asri *et al*., 2005). Figure 2 shows a diagram of multiviews components through the simplified example of a formation in a remote educational system.

This Fig. 2 contains three multiviews components: formation, documentation and enrolment. A multiviews component publishes its attributes and its methods in interfaces that have a stereotype called base. Each base is shared by different points of views and a specification which is dedicated for a specific point of view that has a stereotype view. Researchers can schematize the MVIFormation multiviews interface like (Fig. 3). For example, the behavior method display changes according to an active view.

Corresponding Author: M. Hain, TIM Labo, IGSI Team, Department of Mathematics and Computer Sciences,
Ben M'sik College of Sciences, PB 7955, Casablanca, Morroco

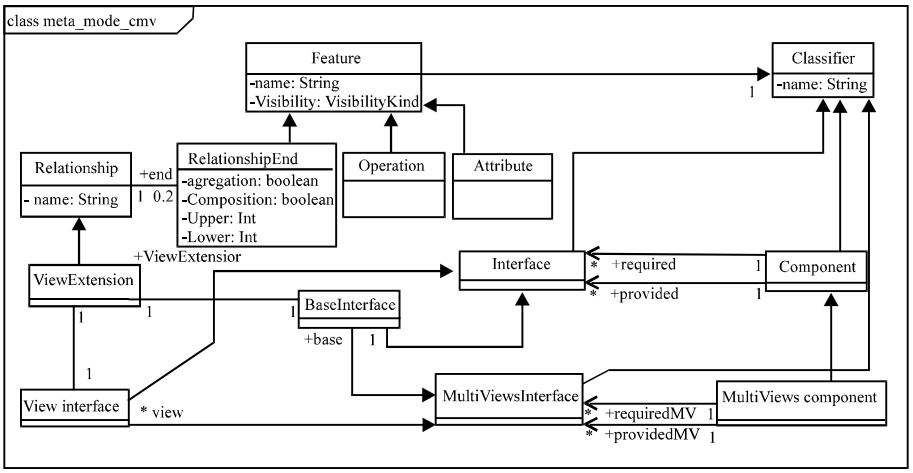

Fig. 1: Metamodel defining the concept of component in UML 2.0 and multiviews component

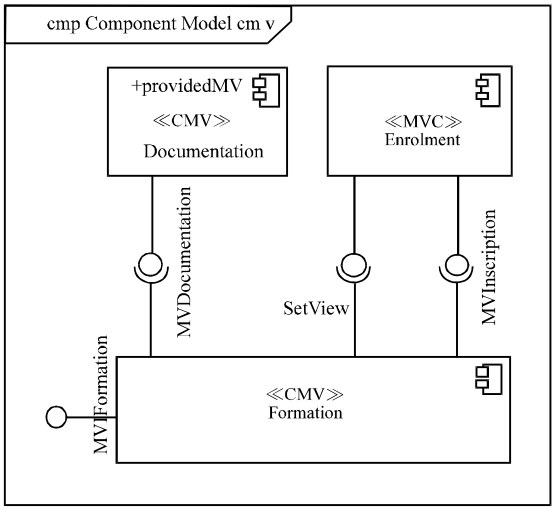

Fig. 2: Example of the multiviews components

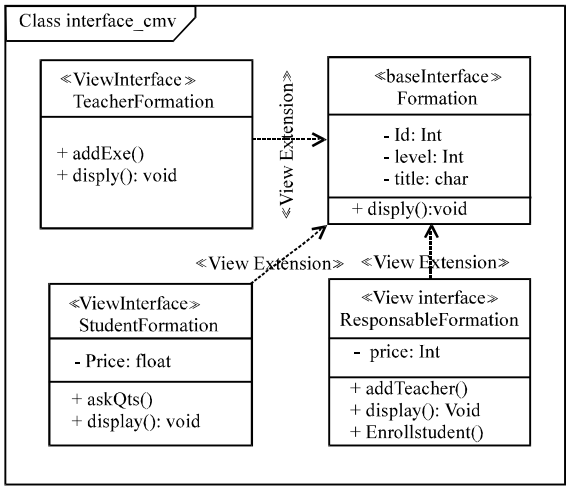

Fig. 3: Representation of MVIFormation multiviews interface

Indeed, a student can exploit this function in order to have more information. By using his own view, he can display the price of a certain product.

MDA and the multiviews components: Model-Driven Architecture (MDA) is a software design approach launched by the Object Management Group (OMG). MDA supports model-driven engineering of software systems.

MDA provides a set of guidelines for structuring specifications expressed as models. The MDA approach defines also the system of functionality using a Platform-Independent Model (PIM) which is translated into one or more Platform-Specific Models (PSMs) that computers can run.

MDA is an approach based entirely on models. We regard a model of abstract multiviews components as a starting PIM. The following step is specifying the process of transformation and refinement which is characterized by adding other information to the model so that it becomes more complete and detailed. For example, an abstract multiviews component can be refined into a whole of other more specific components. Figure 4 shows the process of development of the systems containing multiviews components in MDA.

Researchers suggest initially a PIM Model containing Multiviews Components (PIMMVC). Then, researchers shall transform PIMMVC into a PIM containing standard components UML 2.0, in order to have a standard model in conformity with the concept of

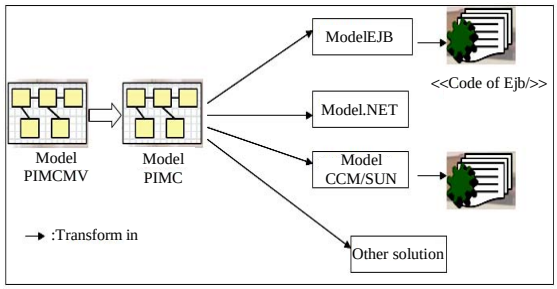

Fig. 4: The process of development of systems with multiviews components

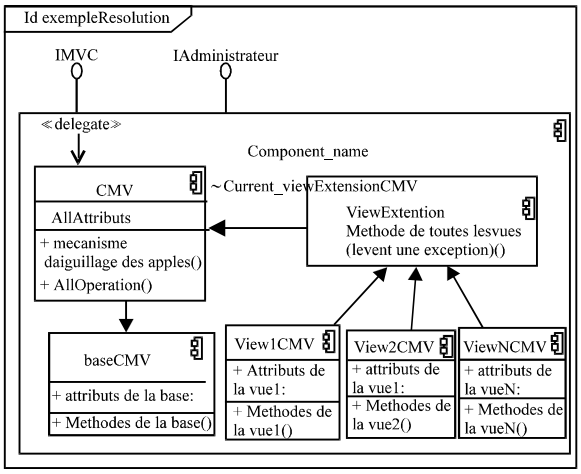

Fig. 5: The structure of the generic pattern of multiviews components

component. The resulting model is a model compatible with the standard of the OMG. This facilitates the projection of the model on the technological models of components: CCM (OMG), EJB and DCOM. To meet the need for a language of transformation of models, the OMG published a call with proposals (RFP, Request for Proposal) entitled MOF 2.0 Query/Views/Transformations (QVT). Presently, there are several products (commercial or open source) that claim compliance with the QVT standard. QVT defines a standard way to transform source models into target models. There are several ideas in this proposal. One is that the source and target models may conform to arbitrary MOF metamodels. The other is that the transformation program itself can be considered as a model. In consequence, it also conforms to an MOF metamodel. This means more precisely that the abstract syntax of QVT should conform to a MOF2.0 metamodel.

PRESENTATION OF THE PATTERN

Problem: The main target is to suggest a representation of the multiviews component according to the standard of the UML2.0 because the multiviews component contains types of the different relationships that cannot undergo implementation namely: ViewExtention, ViewDependancy, the base multiviews. The resulting solution is a composite component that conforms to the standard of the UML2.0 and which encompasses the mechanism of of the multiviews component functioning (Gamma *et al*., 1995; Nassar *et al*., 2003).

The motivations: Researchers would like to make the multiviews component solutions tangible and compatible with the UML2.0 standard. Therefore, a standard representation is used to convert a multiviews component into an UML2.0 component. The next step is targeting different commercial component i.e., CCM (OMG), EJB and DCOM. In fact, this approach helps us not to invent a new language that has something to do with the multiviews component approach which makes it difficult for the developer to handle this language easily. Researchers would like also to hide the interactions and the interdependences between the actors and the component in order to maintain the control of the point of view of each actor.

The solution suggested: Researchers suggest a modeling of the generic pattern according to the UML2.0.

Static structure: We represent a multiviews component as a composite component involving parts connected to each other. We use the delegation and the polymorphism to manage the rights of access to various views, ensuring the implementation of views for different operations published in the interface multiviews. Figure 5 shows the static structure of the suggestion.

Component_name: It represents the composite component embodying the notion of the multiviews component. It is essentially a black box equipped with interfaces, allowing communication with the external environment.

IAdministration: It is an interface that manages the views and the internal structure of multiviews components. Furthermore, it is endowed with two operations: the SetView() that allows to enable/or disable a view and the Getview() that provides us with the view and the propagation of the active view.

IMVC: The multiviews component communicates with the outside via at least, one interface which displays the various attributes and operations bearing the name of the interface multiviews IMVC.

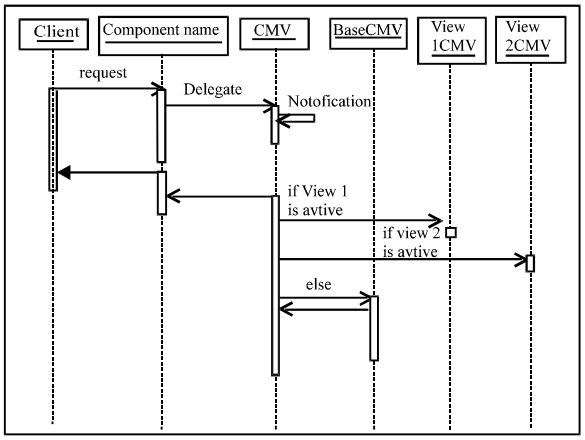

Fig. 6: The dynamic structure of the generic pattern of multiviews components

MVC: It is a component leaves in the composite component. It functions like a frontage of reception and switches the calls towards the concerned shares according to the active view.

BaseCMV: It is a component that a share contains allotted them and the common operations between all the actors.

View_Extention: Keep the trace call component MVC with the shares of the actors by a referential aggregation by CurrentView_ExtentionCMV.

ViewKCMV: Represent the specific point of view of each actor. He contains the attributes and the redefined operations removal of level of the base where exclusive for an actor. K represents an index of a given actor.

Dynamic structure: Components represented in the static structure usually have a dynamic behavior (Fig. 6). In fact, the client, who uses the multiviews components system, connects first with a composite component (Compoent_name) which sends the complaint received to an internal part component (MVC). Afterwards, the MVC identifies the source of the request. If the view of the actor is active, the MVC forwards the request to the concerned view.

On the contrary, the request is sent to BaseCMV if the view is not active. Needless to say, the BaseCMV facilitates sharing information. In addition, the composite component has various interfaces, namely component management interfaces which allow the administrator to manage his own MVC using them to enable or disable the views.

THE TRANSFORMATION OF MULTIVIEWS COMPONENT

The procedure of transformation represents the process of evolution from a model to another through a language of transformation. The first step of transformation depends on the correspondence between the elements of source model and target model. Whereas the source model is a PIM containing multiviews components: PIMVMC, the target model is a PIM containing standard component UML2.0: PIMC.

The transformation which converts a PIMMVC into a PIMC, consists of a set of rules. In order to express these rules, we use the free tool ATL (OMG, 2003b) (Jouault and Kurtev, 2006) which quickly answers best the problems of transformation. Let us recall that ATL is an official proposal which came in response to a call RFP from the OMG.

Like different other languages from MDA space, ATL is based on OCL language for writing expressions. Let us recall again that the OCL is a language of constraints associated with UML 2.0.

CONCLUSION

In order to maintain the concept of the actor's viewpoint during the design process, consider the viewpoint like business information, regardless of technology. The result is a model which is independent from a platform containing multiviews components, PIMCMV. The point is integrating the multiviews components into the MDA context. The keys of success are: the transformation and the compatibility of the platform with the generated solution. In this study, researchers suggested an intermediate PIM which provides a simplified translation of a model base of multiviews components into a model conforming to UML 2.0 component standard according to the MDA context. We currently research on a prototype of deployment for the components multiviews, in order to generate a PSM starting from a PIM containing components multiviews.

NOMENCLATURE

The syntax definition of a transformation rule can be presented as follows:

```
Rule MVComp2Comp{

From   MVC: PIMMVC! MVComp (Cond)
To C: PIMC! Comp
(
--Ex:        MVC.allinstances->exists
(comp_CVM      :MVC|comp_CVM.name      =
component_name);)
}
```

With,

PIMMVC = Name of the metamodel of the model of entry
MVComp = Name of an element of the metamodel
MVC = Local name of of MVComp
Cond = Condition of filtering on the system of entry to the MVComp2Comp rule
PIMC = Name of the metamodel of the model
Comp = Name of the element of the metamodel with instance in the model
C = Local name of the system created at exit of the rule

REFERENCES

El Asri, B., M. Nassar, B. Coulette and A. Kriouile, 2005. Multiviews component for development. Proceedings of the 7th International Conference on Enterprise Information Systems, May 24-28, Miami, USA., pp: 217-225.

Gamma, E., H. Richard, R. Johnson and J. Vlissides, 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Massachusetts.

Jouault, F. and I. Kurtev, 2006. On the architectural alignment of ATL and QVT. Proceedings of the ACM Symposium on Applied Computing, April 23-27, Dijon, France, pp: 1188-1195.

Meyer, B., 2000. What to compose. Software Development, March 2000. http://www.citeulike.org/user/gvdh/article/6494628.

Nassar, M., B. Coulette, X. Cregut, S. Marcaillou and A. Kriouile, 2003. Towards a view based unified modeling language. Proceedings of the 5th International Conference on Enterprise Information Systems, April 22-26, Angers, France, pp: 257-265.

OMG, 2003a. UML 2.0 superstructure final adopted specification. Document-ptc/03- 08-02. http://www.omg.org/cgi-bin/doc?ptc/2003-08-02.

OMG, 2003b. UML 2.0 OCL final adopted specification. Document-ptc/03-10-14. http://www.omg.org/cgi-bin/doc?ptc/2003-10-14.

Szyperski, C., D. Gruntz and S. Murer, 2002. Component Software: Beyond Object-Oriented Programming. 2nd Edn., Addison-Wesley, New York, ISBN: 0-201-74572-0.