QVT Transformation Rules to Get PIM Model from CIM Model

Imane Essebaa and Salima Chantit

Abstract The Model Driven Architecture (MDA) approach introduces a clear separation of system requirements and their implementation. This approach uses models at the center of the development of software systems. The MDA approach is based on models transformations; in the literature, several works have summarized the MDA approach to the passage from PIM to PSM then to code. But very little works have contributed on CIM to PIM transformations. Thus, our proposal aims to provide a method of transforming CIM to PIM using QVT transformation rules and respecting the two levels aspects as specified by the Object Management Group (OMG). Our approach proposes to represent a CIM level by Business Use Case Diagram and Activity Diagram while after applying transformation rules, the PIM level is represented by Class Diagram and Sequence Diagram.

Keywords Model Driven Architecture · Models transformation · QVT · CIM · PIM · PSM

1 Introduction

The Model Driven Architecture approach aims to separate the views and concerns; MDA approach have three views represented by models: Computation Independent Model (CIM) contains system requirements independently of any computation information, Platform Independent Model (PIM) focuses on only the operation of the system without details of a particular platform, and finally a Platform Specific Model (PSM) that depends on technical platform.

I. Essebaa (✉) · S. Chantit
Laboratoire Informatique Mohammedia, Computer Science Departement, Faculté des Sciences et Techninques de Mohammedia, BP 146, 20650 Mohammedia, Morocco
e-mail: imane.essebaa@gmail.com

S. Chantit
e-mail: salima.chantit@gmail.com

Á. Rocha et al. (eds.), *Europe and MENA Cooperation Advances in Information and Communication Technologies*, Advances in Intelligent Systems and Computing 520, DOI 10.1007/978-3-319-46568-5_20

Models transformations from one level to the other one constitute the key of the MDA; several works have focused on the transformation between PIM and PSM then code generation, whereas few has addressed the CIM to PIM transformations, and the existing works don't propose a complete solution to ensure this transformation, existing works can be classified to works which cover all the aspects of the CIM and PIM level specified by the OMG but they don't define complete transformation rules or don't ensure the traceability and automation, and works that don't cover all the aspects of CIM and PIM.

In order to automate transformations between CIM and PIM, we investigate this paper to present an approach that covers the different viewpoints of CIM and PIM levels, a set of transformation rules are defined using QVT to build PIM from CIM.

This paper is organized as follows: Sect. 2 presents an MDA approach and their levels, it contains also a brief presentation of QVT transformation language. Section 3 presents our evaluation of some of related works, in Sect. 4 we present our proposal approach followed by Sect. 5 that illustrate our approach in a case study, we finally conclude with future works and conclusion of this work.

2 Model Driven Architecture

2.1 Modeling Levels in MDA

The MDA (Model Driven Architecture) is an initiative of the OMG (Object Management Group) released in 2000 [1] The basic idea of the MDA approach is the separation of the functional system specifications and its implementation on a particular platform.

The MDA approach lies in the context of the Model Driven Engineering which involve the use of model and metamodels in the different phases of development lifecycle of an application [2], MDA defines three viewpoints

- CIM (Computation Independent Model): the objective of this model is to represent the application in their environment independently of any computation information.
- PIM (Platform Independent Model): the role of the PIM is to give a static and dynamic vision of the application regardless of the technical conception of it.
- PSM (Platform Specific Model): This model depends on technical platforms; it represents a template of code that facilitates code generation.

2.2 Models Transformation

CIM, PIM and PSM models are the main levels of the MDA approach, each of them contain information necessary for the generation of the source code for the

application. The code is obtained by automatic generation from the PSM, the PSM is obtained by successive transformations of models CIM to PIM and PIM to PSM.

Execution of transformations ensure a link traceability between different models of the MDA approach. This link is a guarantee of quality in the software development process in MDA.

The MOF (Meta Object Facility) is normalized by OMG [3], it allows the definition of transformation rules and modeling languages, it also specifies the structure and syntax of metamodels. In this context the OMG proposes a standard transformation language QVT (Query/view/transformation) to define transformation rules from models to models.

3 Related Works

In the context of MDA approach, several methods of model transformations were proposed, specially between PIM level and PSM level, however limited works were done between CIM level and PSM one.

After the analyze of the previous works we deduced that the transformation rules proposed are not complete, indeed those methods may be classified into two categories; researches who cover different aspects of CIM and PIM but not defining complete transformation rules and researches that not cover the various aspects of the CIM and PIM levels.

Kherraf et al. in his proposition [4] uses to model the CIM by UML activity Diagrams which are detailed to get the system requirements which are transformed to be modeled as system components in PIM level. This method is based on modeling CIM using only the Activity Diagram, and after the transformation it doesn't cover the behavioral aspect.

In their method [5] Kardoš et al. modeled the CIM level by Data Flow Diagram (DFD), and PIM level was modeled by fours UML diagrams: Use Case Diagram, Activity Diagram, Sequence Diagram, and the domain models.

Wu et al. Method [6] describes how to get PIM level from CIM one, and the transformation PIM into PSM, in this method CIM is represented by Use Case Diagram, Activity Diagram and robustness diagram, while the PIM one was presented by Sequence Diagram and Class diagram, this method we note that the authors covered all the aspects of CIM and PIM but they didn't propose a complete transformation rules, neither traceability and automation of transformation.

In their proposition Kriouile et al. [7] modeled the CIM level by two diagrams: Business Process Model and Use Case Diagram, while the transformation of the CIM generates Domain Class Diagram and Sequence Diagram, this method covers the CIM and PIM aspects, but it doesn't ensure a completeness of transformation rules neither the automation of transformation.

It should be noted that in the literature that we have found, it exists many methods that are limited only in modeling CIM level without proposing transformation rules to get PIM level [8], it also exists works [9] that used a method called

TFM4MDA (Topological Functioning Modeling for Model Driven Architecture), it uses formal mathematical foundations of topological functioning model.

The results of our analyze are presented in Table 1, the columns represents an evaluation criterion while the rows represent studied methods.

4 Proposed Approach

This section presents the approach of the CIM Modeling and its transformation to PIM Model, our approach consists of:

- Modeling CIM through UML Business Use Case Diagram to cover the structural and functional aspect, and UML Activity Diagram to cover the dynamic aspect.
- Obtain the static viewpoint of PIM from CIM level through a vertical transformation C2P1 detailed in Table 2, the result of this transformation is represented with Class Diagram.
- Obtain the dynamic viewpoint of PIM from CIM level through a vertical transformation C2P2 detailed in Table 3, the result of this transformation is represented with Detailed Sequence Diagram.
- Transformation rules that map elements of one metamodel to the elements of another metamodel, are defined by using the QVT transformation language.

The Fig. 1 below describes this approach.

4.1 CIM Architecture in Our Approach

In his book "MDA in action" Xavier Blanc [10] consider the CIM level as the most important and complex entity, any changes of requirements in CIM level will reflect the PIM and the PSM levels.

After the analysis of the previous related works we didn't find any consensus and rules on how CIM should be presented and even how many models could represent CIM.

In practice, the first question to address in order to represent CIM level is "what should be represented in this model?".

According to the OMG specifications and criteria, we deduce that there are two topics of CIM level: System requirements and the process between system stakeholders, moreover those topics determine the three aspects of CIM level: Functional, Static and Behavioral.

In our approach we have opted to present this level by two models: Business Use Case Diagram to describe the functional requirements as well as a static view of the

Table 1 Results of analyze and evaluation

Methods studied	CIM level			PIM level		CIM2PIM transformation		
	Behavioral	Functional	Static	Dynamic	Static	Automation rules	Rules traceability	Completeness of rules
Kherraf et al. [4]	Y	N	N	N	Y	N	P	N
Kardoš et al. [5]	P	N	N	Y	Y	N	N	N
Wu et al. [6]	Y	Y	Y	Y	Y	N	N	P
Kriouile et al. [7]	Y	Y	Y	Y	Y	N	P	N
Sharifi et al. [8]	N	Y	Y					
Erika et al. [9]	N	Y	Y					

Legend: *Y* Yes; *N* No; *P* Partial

Table 2 UCD2CD transformation QVT rules

Rule	Transformation rule	Source model	Target model
1	Actor2Class	Actor	Class
2	DataObject2Class	DataObject	Class
3	Associations2Associations	Associations	Associations
4	UseCase2Operation	UseCase	Operation

Table 3 UCD&AD2SD transformation QVT rules

Rule	Transformation rule	Source model	Target model
1	Actor2Actor	Actor	Actor
2	UsecaseSystem2System	UsecaseSystem	System
3	Include2Ref	Include	Ref
4	Extend2Alt	Extend	Alt
5	DataObject2LifelineObject	DataObject	LifelineObject
6	Action2Message	Action	Message
7	NodeObject2LifelineObject	NodeObject	LifelineObject
8	FlowFinal2break	FlowFinal	Break

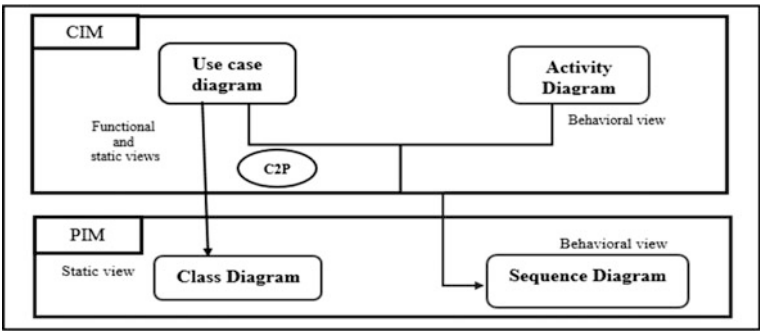

Fig. 1 Overview of the approach

system, and the Activity Diagram to represent different sequences and activities of the system.

UML Business Use Case Diagram

UML (Unified Modeling Language) defines a Use Case as: "the specification of a set of actions performed by a system, which yields an observable result that is, typically, of value for one or more actors or other stakeholders of the system" [11].

We choose Business Use Case diagram for several reasons; it identifies functionality of system and how the system works making links between actors and functionality which cover the functional aspect, and in order to cover the static aspect, we have extended the Use Cases diagram by adding "DataObject" element

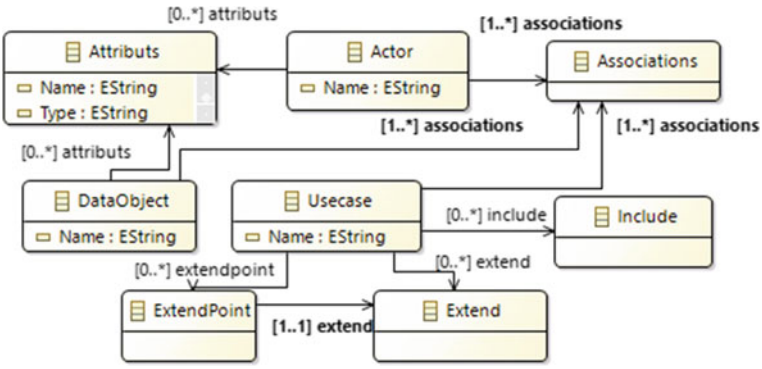

Fig. 2 Main fragments of metamodel UCD

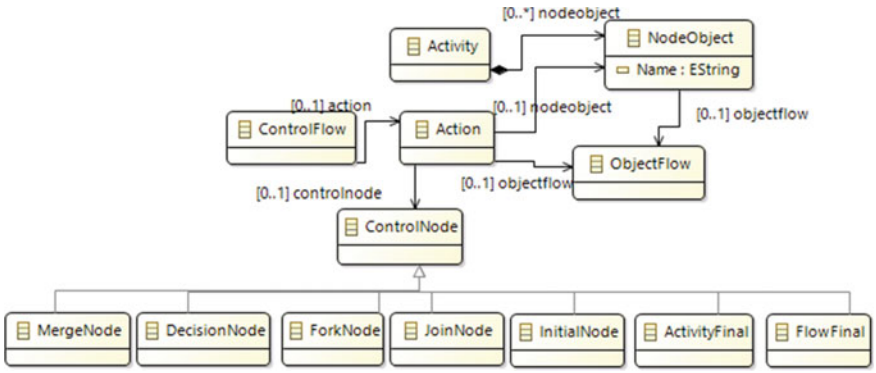

Fig. 3 Main fragments of metamodel Activity Diagram

A simplified version of the metamodel of the Business Use Case Diagram is represented in the Fig. 2.

UML Activity Diagram

Activity diagram is UML behavior diagram which shows flow of control or object flow with emphasis on the sequence and conditions of the flow.

We choose the activity diagram to represent a dynamic view of system in the CIM level, it allows to understand the process of different features of the system.

Figure 3 shows the metamodel of the activity diagram:

4.2 PIM Architecture in Our Approach

The PIM, called Model of analysis and design, represents the business logic specific for a system. It depicts the functioning of entities and services. It must be

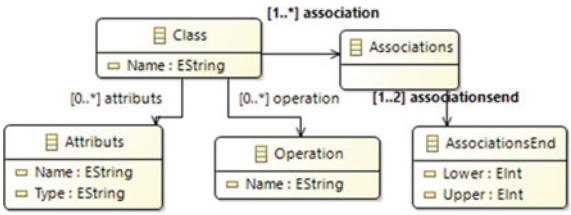

Fig. 4 Main fragments of metamodel DCD

sustainable over time. To respond to the OMG specifications, the adequate PIM model should cover two aspects:

- The structural aspect (Static) represented by Domain Class Diagram.
- The dynamic represented by Sequence Diagram.

UML Domain Class Diagram

The domain model is a representation of meaningful real-world concepts pertinent to the domain that need to be modeled in software. The concepts include the data involved in the business and rules the business uses in relation to that data.

In UML, the class diagram is used to represent the domain model, the domain class diagram represents in our approach a static view of the PIM.

The main fragments of metamodel of Domain class diagram are shown in Fig. 4.

UML Detailed Sequence Diagram

The sequence diagram models the collaboration of objects based on time sequence, the Fig. 5 represents metamodel of Detailed Sequence Diagram.

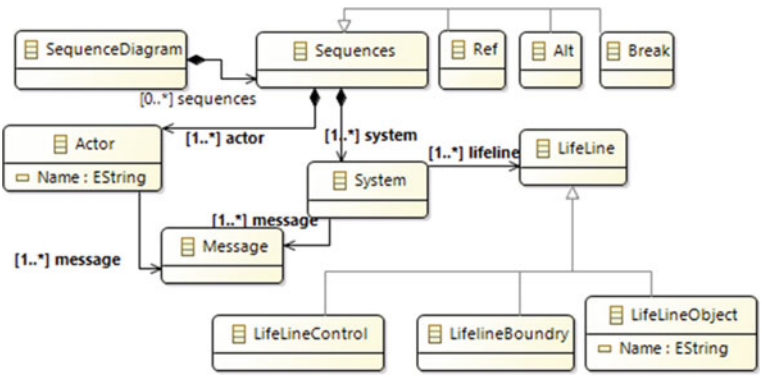

Fig. 5 Main fragments metamodel Detailed Sequence Diagram

```
mapping UseCaseDiagram::Usecase::UseCase2Operation() : Class::Operation{
    result.Name := self.Name;
    }
mapping UseCaseDiagram::Actor::Actor2Class() : Class::Class {
    result.Name :=self.Name;
    result.association +=self.associations.map associations2association();
    result.attributs +=self.attributs.map attributs2attributs();
    }
mapping UseCaseDiagram::DataObject::DataObject2Class() : Class::Class{
    result.Name := self.Name;
    result.attributs +=self.attributs.map attributs2attributs();
    result.association +=self.associations.map associations2association();
    }
mapping UseCaseDiagram::Attributs::attributs2attributs() : Class::Attributs {
    result.Name := self.Name;
    result.Type := self.Type;
    }
```

Fig. 6 Part of transformation QVT rules of UCD2CD

```
mapping UseCaseDiagram::Actor::Actor2Actor() : sequence::Actor{
    result.Name := self.Name;
}
mapping UseCaseDiagram::DataObject::DataObject2LifelineObject() : sequence::LifeLineObject{
    result.Name := self.Name;
}
```

Fig. 7 Part of transformation QVT rules of UCD&AD2SD

Transformation approach from CIM to PIM

CIM Modeling

In this step, the requirements of system are represented firstly through UML Business Use Case Diagram, and secondly their process is represented with UML Activity Diagram.

Obtaining the PIM static view from the CIM

$C2P_1$ is a vertical transformation which consists of transforming the UML-UCD conformed to its metamodel to a Class Diagram conformed to its metamodel. Those transformation rules are presented in Table 2.

A part of those rules are described with QVT Language in the Fig. 6.

Obtaining the PIM dynamic view from the CIM

The CIM2PIM transformation, noted $C2P_2$, is a vertical transformation aims to transform UML-UCD and UML Activity diagram to Detailed Sequence Diagram that represent the dynamic aspect of the PIM. Those transformation rules are presented in Table 3.

A part of those rules are described with QVT Language in the Fig. 7.

5 Case Study

In this section we are going to illustrate our approach throw an example, we choose as an example a system of E-Library that models an interaction between system and customers. Any customer can access to the web site and search one book, they can read it online or download it, they can also request a new book by filling a form, customers must connect with their account or subscribe if it is their first visit of the web site. To implement our approach, we must start by modeling the requirements with UML-UCD (Fig. 8).

After modeling all system requirements, we are going to focus on "Choose book" use case and implement an activity diagram to describe its process (Fig. 9).

UML-UCD transformation to UML Class diagram

From the Business Use Case model, by applying the mapping rules presented in Table 2 we get the Class Diagram bellow (Fig. 10)

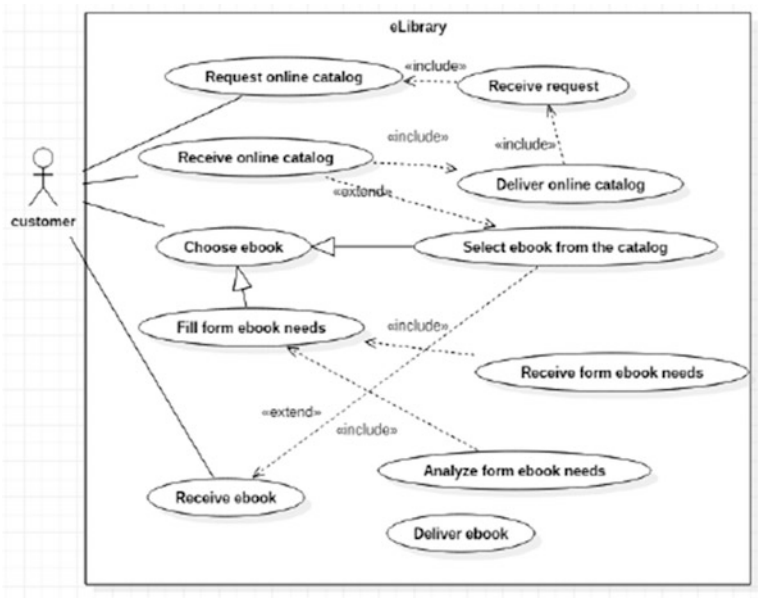

Fig. 8 Business Use case diagram of E-library

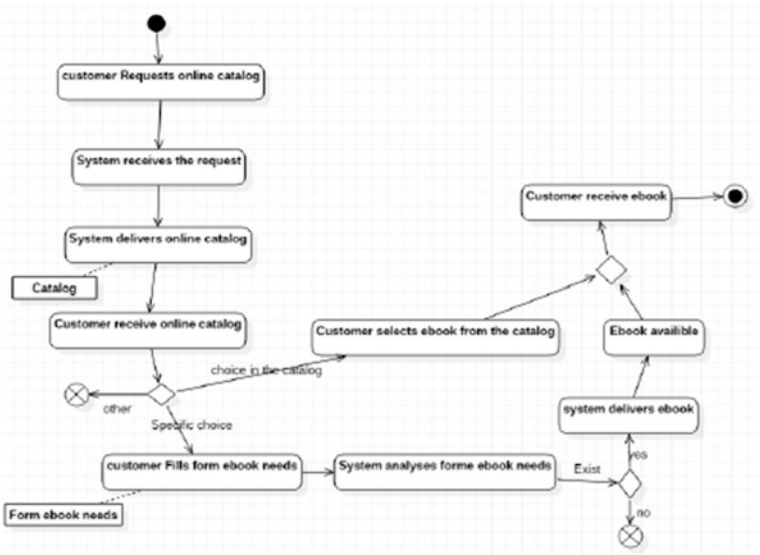

Fig. 9 Activity diagram of "choose book"

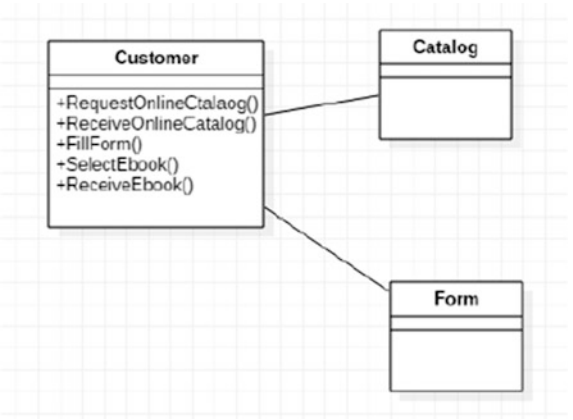

Fig. 10 Class Diagram of the system "E-Library"

UML-UCD and Activity Diagram transformation to Detailed Sequence Diagram The dynamic aspect of PIM is represented through a Sequence Diagram; this diagram it is a result of transformation $C2P_2$ presented in Table 3 which allow to transform UML Business Use Case Diagram and Activity Diagram to a Sequence Diagram (Fig. 11).

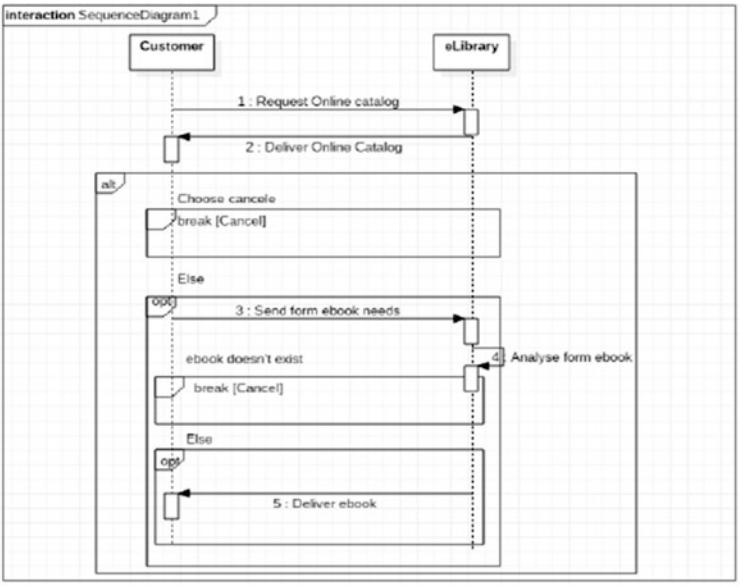

Fig. 11 Sequence diagram of "choose book"

6 Conclusion

In this paper we have proposed an original approach based on UML diagrams to model the CIM level, we also proposed a set of transformation rules using QVT that help to build PIM level which is also modeled with UML diagrams.

This approach provides a disciplined way towards automation of the CIM2PIM transformations.

In our future works we will focus on completing transformation rules to ensure more traceability and complete automation of transformation, we aim also to propose a continuation of this approach to cover all the MDA transformation namely PIM2PSM transformation and PSM2Code generation.

References

1. Miller, J., Mukerji, J.: MDA Guide Version 1.0.1. OMG (2003)
2. Soley, R.: Model driven architecture (MDA), draft 3.2. Rapport technique, disponible sur: http://www.omg.org/cgibin/doc?omg/00-11-05 (2000)
3. OMG: Meta Object Facility (MOF)2.0Query/View/Transformation Specification. http://www.omg.org/spec/QVT/1.0/PDF (2009)
4. Kherraf, S., Lefebvre, E., Suryn, W.: Transformation from CIM to PIM using patterns and archetypes. In: 19th Australian Conference on Software Engineering (2008)

5. Kardoš, M., Drozdová, M.: Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA). J. Inf. Organ. Sci. **34**, 89–99 (2010)
6. Wu, J.H., Shin, S.S., Chien, J.L., Chao, W.S., Hsieh, M.C.: An extended MDA method for user interface modeling and transformation. In: The 15th European Conference on Information Systems, pp. 1632–1641, June 2007
7. Kriouile, A., Addamssiri, N., Gadi, T.: An MDA method for automatic transformation of models from CIM to PIM. Am. J. Softw. Eng. Appl. **4**(1), 1–14 (2015). doi:10.11648/j.ajsea.20150401.11
8. Sharifi, H.R., Mohsenzadeh, M.: A new method for generating CIM using business and requirement models. World Comput. Sci. Inf. Technol. J. (WCSIT) **2**(1), 8–12 (2012)
9. Osis, J., Asnina, E., Grave, A.: Computation independent modeling within the MDA. In: IEEE International Conference on Software-Science, Technology and Engineering, 2007. SwSTE 2007, pp. 22–34. IEEE (2007)
10. Xavier, B.: MDA en Action: Ingénirie Logicielles Dirigée par les Modèles. Eyrolles, Paris (2005)
11. OMG: OMG Unified Modeling Language TM (OMG UML), Superstructure, http://www.omg.org/spec/UML/2.4.1/Superstructure. Accessed Aug 2011