Meta-model Based Model Organization and Transformation of Design Pattern Units in MDA

Chang-chun YANG(杨长春)[1,2], Zi-yi ZHAO(赵子艺)[2], Jing Sun(孙 婧)[2]
(1. *School of Economics and Management, Nanjing University of Science and Technology, Nanjing* 210094, *China*;
2. *School of Information Science and Engineering, Changzhou University, Changzhou* 213164, *China*)

Abstract – To achieve the purpose of applying design patterns which are various in kind and constant in changing in MDA from idea and application, one way is used to solve the problem of pattern disappearance which occurs at the process of pattern instantiation, to guarantee the independence of patterns, and at the same time, to apply this process to multiple design patterns. To solve these two problems, the modeling method of design pattern units based on meta-models is adopted, i.e., to divide the basic operations into atoms in the meta-model tier and then combine the atoms to complete design pattern units meta-models without business logic. After one process of conversion, the purpose of making up various pattern units meta-model and dividing business logic and pattern logic is achieved.

Key words – ***MDA***; ***PIM***; ***design pattern***; ***model organization***

Manuscript Number: 1674-8042(2010)02-0183-05

dio: 10.3969/j.issn.1674-8042.2010.02.20

1 Introduction

The Integration of Heterogeneous Systems has always bothered the development of large-scale software. MDA (Model Driven Architecture) is a framework for model organization and management in the process of software development proposed by OMG. It's a new way to solve the problem of the Integration of Heterogeneous Systems, and allows the integration of different systems on different middleware platforms. Its main idea is separating the tight coupling relationship between the analysis and design of business function and the implementation platforms, thus minimizing the impact on the system from the changing of technology and platform[1]. For systems' heterogeneity, the authors use the idea of design pattern, upgrade the abstraction level of the systems, so as to design sub-system in a unified way.

Design patterns use classes and methods in the object-oriented language to achieve some program objective, they are the "blade" of object-oriented technology[2], they can provide reusable design solutions to object-oriented software development and help to improve software reusability and system maintainability[3]. Apply it to the MDA and combine their advantages can improve development efficiency, save development time and accelerate the development of MDA.

However, the applications of design patterns in MDA often face the following two issues.

1) Pattern disappearances: When instantiating the design patterns, their implementations will change according to the environment, eventually lead to the pattern disappearances[4].

2) Pattern explosions: a complete development often involves a variety of design patterns and multiple pattern-unit automatic conversions[3]. Patterns are continuously updating and developing, the number will be more and more, finally, applying design patterns in the software development will be an arduous task.

Aiming at the above two issues, Ref.[5] has adopted a role-based modeling method, defined the role in the design patterns to independent class and separated the independent class from the application class, then used a Role-Of relationship to connect the two classes, thus achieved the separation of business logic and pattern logic.

Ref.[5] gave the Meta-Model of Role-Of relationship at the Meta-Model level, but the business logic and pattern logic are at the model level, the definition of pattern logic at Meta-Model level was not given. Ref.[3] applied one kind of design patterns as a whole unit to the development process of MDA, achieved the definition of pattern logic at Meta-Model level, and also realized the combination-binding of business logic and pattern logic, but did not achieve the combination-realization of pattern units Meta-Model(PUMM). Ref.[6] presented a method of subdividing the operations of model elements as atomic mappings, this method can achieve combination of a variety of design patterns, automatically convert source model which does not contain the design pattern to target model that contains design pattern. Drawing on such thinking, the authors can combine the PUMM and simultaneously solve these two problems.

* **Received**: 2010-04-01
Corresponding author: Chang-chun YANG(yjsywmycc@126.com)

2 Meta-model

In the development process of MDA, modeling is an important part. The quality of PIM directly impacts on the conversion efficiency and quality from PIM to PSM. A.

Large number of engineering practices show that, the efficiency of Meta-Modeling based modeling is 10 times higher than the UML based[4], so the Meta-Modeling based MDA has greater potential than the single UML based MDA[7]. Meta-modeling is a mechanism used to define language in the MDA environment, its core elements are: meta-meta model, meta model and modeling tool integration[7]. As the core products of Meta-modeling, meta-model's construction is very important for the following reasons: first, meta-model is used to define the language; second, changing rules use themeta-model of source language and target language to define the transformation. The following part focuses on the Construction of PIM, including its uilding foundation at meta-model level.

Meta-Model explicitly defines the elements of modeling language and their relationships, its model object is language. OMG takes four-tier meta-model architecture, M_0 tier corresponds to the objective world, M_1 tier is model tier, M_2 tier is meta-model tier, M_3 tier is meta-meta model tier, lower is the example of upper.

1) MOF (Meta Object Facility) locates at the M_3 level, it is an OMG standard and the core technology which can realize MDA. It defines the language of modeling language. It is the standard language describing meta-model.

2) UML is the standard modeling language at the M_2 level. It is defined by MOF and can be used to build PIM and PSM. UML meta-model is the instance of MOF model. Core UML is platform-independent, so extended UML language is needed when building PSM, the usual two ways are as follows: UML Profile approach and meta-modeling approach[8].

3) UML Profile mechanism uses the graphic presentation and OCL text query. It is defined as a set of Stereotypes, a group of Tagged Values, a group of Constraints. Their function is defining a specialized variant of UML for specific target. Stereotype-defined meta-model don't merge with original one, its extension of meta-model is on the surface, and the relationship between the concept of model abstract lexical extension and original concept is only inherited but not associated, that is to say it can be directly extended but define new association[8].

4) OCL is a query and expression language of UML, as well as one part of UML standard, currently, it has become a complete query language. Using OCL in UML will make the models of system more complete and accurate[9].

3 The modeling support environment of design pattern

Using the idea of reusing design patterns from meta-model level, the authors can take design pattern units as pattern models and separate the pattern models with business models. PUMM based on extended meta-meta model of MOF, the authors should extend two meta-meta models of Epattern and Erole to define PUMM[3].

3.1 Epattern and Erole pattern meta-meta model

Epattern and Erole are the subclass of Classifier in the MOF meta-meta model, Classifier inherits from Type which is included in Package, and Package is used to organize meta-model. One instance of Epattern is a PUMM. Erole defines the roles in pattern, one role is an Erole instance. Eole can have multiple operations and attributes in grammar by extending.

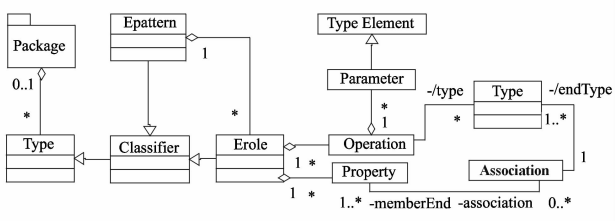

Fig. 1 Epattern、Erole pattern meta-meta model

3.2 The meta-model of RoleOf relationship

Role-Of is a dependency relationship from Role to Class, Role is the role class in design pattern, and Class is the application class defined with modeling language. A role design pattern and business model has the function relationship. A business model can be repeatedly bound and it includes all the methods as well as properties in the binding Role. RoleOf relationship can achieve the binding of business model and pattern model after their separation and can avoid the problem of pattern disappearances. The symbol of this binding relation is dependency indicated by virtual arrows added with a "bind".[3,7]

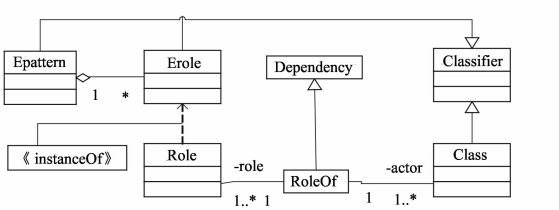

Fig. 2 The meta-model of RoleOf relationship

4 The construction and transformation of PUMM

The extended MOF meta-meta model has been given earlier, next step is carrying out the subdivision and combination of PUMM. The purpose of combined-conversion is combing source model without design pattern to target model with design pattern[9,5]. Transformation rules are defined in the meta-model layer, the combination processes are at the M_2 level, as shown in Tab. 1. Using UML as

the model representation language, using OCL language to describe the rules and constraints between elements so as to enhance the expression of the dynamic modeling of UML, using UML Profile to extend the model description of a specific platform[10,6,11]. Focus on the vertical model organization shown in the Tab.1.

Tab.1 Model Organization And Transformation

Model level	Source model	Operation	Target mode
M_3	MOF meta-meta model	Extend MOF meta-meta model	Epattern、Erole meta-meta model
M_2	Meta-model	Apply QVT transformation rules	Struts meta-model
M_1	PIM	Apply QVT transformation rules	Struts PSM
M_0	instance	Apply QVT transformation rules	instance

4.1 The UML extension of Observer pattern unit meta-model

Take observer pattern for example. According to the standard and simplified view of Observer pattern which has been rewritten and without business logic, the authors instantiate Epattern and Erole pattern meta-meta model, so as to get the Observer PUMM shown in Fig.3.

The contained meta-model elements are: Package, Class, Association, Property, and Operation.

The core relations includes: Class2Class, Operation2Operation, Class2Association, Class2Generalization, Class2Operation, and Class2Property. Operation includes Operation1 and Operation2, Property includes Property1 and Property2, this constraint is to avoid naming conflicts.

The contained four roles are: Subject, Observer, Concrete Subject and Concrete Observer. They are the roles of the Observer pattern, respectively extended to tagged values in UML (Observer _ Class _ Subject, Observer _ Class _ Observer, Observer _ Class _ Concrete Subject, Observer _ Class _ Concrete Observer).

Then the elements of Observer pattern can be expressed in UML. Name meanings is: pattern name _ meta class _ role name[6].

4.2 The combination of PUMM

Source model and target meta-model are both the extended UML meta-model, the class whose role of stereotypes named Subject will transform into Observer pattern. Take the operations such as add, delete, modify between pattern elements as atom mappings, each atom mapping is a velocity template, combing these atom mappings to model transformation rules and inputting parameters from users, then the model transformation code can be acquired.

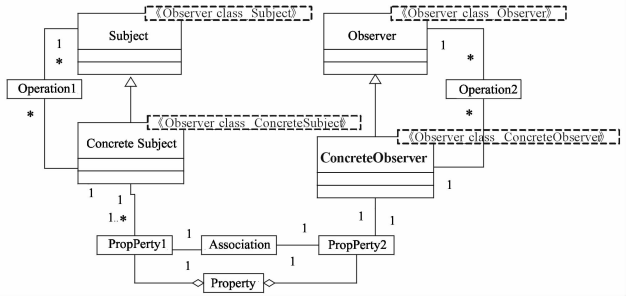

Fig.3 Observer pattern unit meta model

Tab.2 Atom mapping table

Source model element	Atom mapping name	parameter
Class	AddClass2Class/ DelClass2Class	ClassName, StereotypeName
srcClass, tarClass	AddGeneralization/ DelGeneralization	SrcClassName, TarClassName
Class	AddClass2Property1/ DelClass2Property1	ClassName, QropertyName,type
Class	AddClass2Operation1/ DelClass2Operation1	ClassName,OperationName, type,returnType,accessability,
srcClass, tarClass	AddAssociation/ DelAddAssociation/	SrcClassName,TarClassName, ConstraintNode
srcClass, TarClass	AddAggregation / DelAggregation	SrcClassName, TarClassName

Simply design the atom mappings table, as Tab.2. The first line of Tab.2 indicates that the source model element is Class, and target model element is the same Class, parameter ClassName is target class name, StereotypeName indicates stereotype name, this atom mapping means adding or removing another class. The second line means adding or removing inheritance relation between two classes.

The third line means adding or removing properties in the class, PropertyName is property name, type is property type.

The fourth line means adding or removing methods in class, OperationName indicates method name, type is method type, returnType shows return value type, accessability expresses the accessibility of approach.

The fifth line means adding or removing associated class between two classes, ConstraintNode shows the constraints association relation, for example, one-to-many.

Identify all of the atoms according to the transformation rules, and then the authors can construct different shift templates for different design patterns to meet requirements through the way of combining or modifying parameters.

4.3 The combination of Observer PUMM

The transformation rules of these pattern units meta-model are as follows:

As shown in Fig.3, its source model consists of a Subject Class with an Observer _ Class _ Subject stereo-

type, so convert it to Observer design pattern structure according to the transformation rules: (1) Create a class Observer with construction type Observer _ Class _ Observer, add Operations2 and insert a one-to-many association between Observer and Operation2; (2) Create class Concrete Observer with construction type Observer _ Class _ Concrete Observer, add every operations of Observer and attribute Property2; (3) Add a one _ to _ one association between property2 and Concrete Observer, add a one-to-many association between methods and Concrete Observer. (4) Create a model element Property, put in a shared aggregation relationship between Property and Property2. (5) Add an inheritance relationship between Observer and Concrete Observer. (6) Create a class Concrete Subject with structure type Observer _ Class _ Concrete Subject, join in all the methods Operation1 of Subject and add attribute Property1. (7) Respectively add one-to-one and one-to-many associations between Property1 and Concrete Subject, Operation1 and Concrete Subject. (8) Put in a shared aggregation relationship between Property and Property1. (9) Add inheritance relationship between Subject and Concrete Subject. (10) Create a one-to-one association between Property1 and Property2, so as to guarantee that one Concrete Subject can correspond to many Concrete Observer in an instance of the pattern model.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="operation">
  <xsl:element name="operation*">
    <xsl:attribute name="ClassName">$TarClassName</xsl:attribute>
    <xsl:attribute name="OperationName">$OperationName</xsl:attribute>
    <xsl:attribute name="type">$Type</xsl:attribute>
    <xsl:attribute name="returnType">$ReturnType</xsl:attribute>
    <xsl:attribute name="accessibility">$Accessibility</xsl:attribute>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

Fig. 4 The XSL code of atom mapping AddOperation

```
<xsl:template match="class">
 <xsl:choose>
  <xsl:when test="./@StereoType='Observer_Class_Subject'">
   <xsl:element name="class">
     <xsl:attribute name="Subject">false</xsl:attribute>
     <xsl:attribute name="ID"><xsl:value-of select="@id">_1</xsl:value-of></xsl:attribute>
     <xsl:attribute name="ClassName">Observer</xsl:attribute>
     <xsl:attribute name="StereoType">Observer_Class_Observer</xsl:attribute>
   </xsl:element>
  </xsl:when>
 </xsl:choose>
</xsl:template>
<xsl:template match="class">
 <xsl:choose>
  <xsl:when test="./@ClassName='Observer'">
   <xsl:template match="operation">
    <xsl:element name="operation2">
     <xsl:attribute name="ID"><xsl:value-of select="@id">_2</xsl:value-of></xsl:attribute>
     <xsl:attribute name="ClassName">Observer</xsl:attribute>
     <xsl:attribute name="OperationName">Operation2</xsl:attribute>
```

Fig. 5 Parts of XSL code of PUMM

Then the atom mappings are got as follows: AddClass2Class, AddGeneralization, AddOperation, AddProperty, AddAssociation and AddAggregation.

Each atom mapping is a velocity template wrote into XSL document. Organize the atom mappings and input parameters according to the above conversion rules, different XSL code files can be got, that is the conversion code of different PUMM.

Fig. 4 and Fig. 5 respectively shows the AddOperation XSL code and parts of PUMM XSL code.

4.4 The instance of pattern

When the construction of Observer PUMM has been done, next step is instantiating it to Observer pattern model. Decide the concrete form of Observer pattern according to business logic, create model element E, remove business logic B, mark E to ME, input ME and parameters(such as the number of specific classes, specific methods and B)into PUMM, generate pattern model. Then bind pattern model and logic model to get the complete PIM located at the M1 level through the RoleOf relationship[6].

5 Instance of PIM model construction

If a large supermarket has the online business and real business at the same time, consumers apply for membership cards to own the right of accessing to the online supermarket. When members login, the system sends a welcome e-mail, displays the current visit-number of members from the first one, verifies the address of members, and expresses these results at the interface. Membership is constantly being added, and member information is also constantly increasing and changing and updating, and the system itself will also have new needs, such as sending a letter with coupons to the members within the vicinity of 20 miles away from the supermarkets, so that the system needs to update data according to the changing, such problems can be resolved by using Observer pattern.

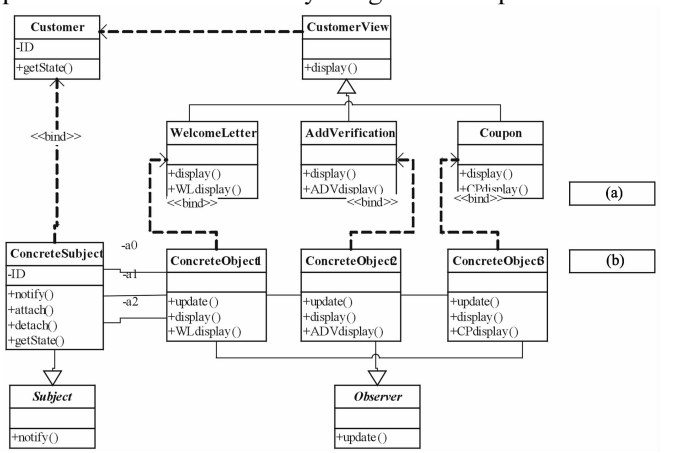

Fig. 6 The PIM of member information service

Here the updates of member information view related to the data layer and presentation layer, Customer is the member information model, CostomerView defines an abstract class used to display information, there are three sub-classes to achieve their respective functions, which are the business model, as shown in Fig. 6(a).

Fig. 3 has shown the Observer pattern unit metamodel, pattern model shown in Fig. 6(b) can be got by instantiating it, PIM showed in Fig. 6 can be got by binding

the two models.

6 Conclusions

The constructed models are all meta-meta models and meta-model located in or above M_2 layer, except for the pattern model generated by instantiation and the business model constructed by modeling language. The meta-model of Epattern and Erole pattern is obtained by extending MOF meta-meta model, Observer pattern unit meta-model is got by instantiating the meta-model of Epattern and Erole pattern. Then all of the meta-models share a same meta-meta model, which makes them have common communication base at the M_2 level and makes the mutual conversion possible.

Therefore, the main modeling environment is meta-model level, the design idea based on role is also from meta-model level to reuse design patterns, apply design patterns as complete units to MDA, then combine the pattern units meta-model. The combination idea is: subdivide the basic operations to indivisible atom, combine them into transformation rules, form a variety of design PUMM. Distinguish the business logic and pattern logic at the time of combination, remove the part of business logic in model elements, thus, the combined PUMM is the model contains only pattern logic and its instance is the pattern model completely not contain business logic. At last, bind the two kinds of models by RoleOf relation, so as to achieve the construction of PIM with separated business logic and pattern logic.

The point is the construction of PIM and prior-period combination of PUMM. The constructed PIM not only contains a clearly identifiable design pattern but the pattern is obtained by combination rules. The next step is to achieve the conversion from PIM to different PSM in Tab.1 on horizontal direction.

7 Acknowledgments

Yang Chang-chun and Zhao Zi-yi thank to MDA based model transformation technology Research in Software engineering, Wang Xue-bin, and MDA based design patterns modeling and model transformation, He Cheng-wan.

References

[1] Jin-kui Hou, Jian-cheng Wan, Yu-yan Zhang, 2007. MDA-supported Model Transformation Approach. *Computer Engineering*, 33(8).

[2] Alan Shalloway, James R. Trott write, Xiong Jie interpret, 2005. Design Pattern Precise analysis. Tsinghua University Press, Beijing, p. 200-202.

[3] Tian Zhang, Yan Zhang, 2008. MDA based design patterns modeling and model transformation. *Journal of Software*, 19(9): 2203-2217. http://www.jos.org.cn/1000-9825/19/2203.htm.

[4] MetaEdit Inc, 2005. Domain-Specific modeling with MetaEdit +10 times faster than UML. White Paper.

[5] Cheng-wan He, Ke-qing He, 2006. Aroel-based approach to design pattern modeling and implementation. *Journal of Software*, 17(4): 658-669.

[6] Xue-bin Wang, Quan-yuan Wu, 2006. MDA based model transformation technology Research in Software engineering. National University of Defense Technology.

[7] Hui Liu, Zhi-yi Ma, Wei-zhong Shao, 2008. Progress of Research on Metamodeling. *Journal of Software*, 19(6): 1317-1327.

[8] Ying Zhou, Guo-liang Zhng, Xuan-dong Li, 2005. Uml Model Transformation Inmda Context: From Function Models To Implementation Models. Computer Applications and Software.

[9] Cheng-jia Diao, 2007. UML System Modeling and Analyse and design. Engineering Industry Publishing House, Beijing, p. 7.

[10] Xue-bin Wang, Quan-yuan Wu. Research and Implementation of Design Pattern-Oriented Model Trans formation. ICCGI, 07.

[11] Li-ting Zhang, 2008. Research and Implementation of MDA Development Pattern. Beijing Jiaotong University.

[12] M. Elaasar, LC. Briand, Y. Labiche, 2006. A meta modeling approch to pattern specific ation. In: Proc. of the 9th Acm/IEEE Int'l Conf. on Model Driven Engineering Languages and Systems (MoDELS2006). LNCS 4199, Berlin, Heidel berg: Springer-Verlag, p. 484-496.