International Journal of Computer Applications (0975 – 8887)
Volume 26– No.10, July 2011

Transformation of UML Class Diagram for Object Oriented Database System

Dr. Vipin Saxena
Department of Computer Science
B.B. Ambedkar University (A Central University)
Rae Barely Road, Lucknow-25, U.P. (India)

Ajay Pratap
Department of Computer Science
B.B. Ambedkar University (A Central University)
Rae Barely Road, Lucknow-25, U.P. (India)

ABSTRACT

Now-a-days use of objects is a common approach in all aspects of computing. An Object-Oriented database can utilize the benefits of both the design and implementation of any application. Object-Oriented database System is the fusion of Database Management System with Object-Oriented System. UML provides stable and industry standard notation for object-oriented analysis and design models. A case study of NSC scheme of Indian Postal Services is considered for storing the data using object database modeling. UML class diagram and sequence diagram are constructed for the post office system and then the transformation of class diagram is done for the ease of understanding and working.

General Terms

Equivalence, Transformation, Algorithms et. al.

Keywords

UML, OODBMS, Class Diagram, Sequence Diagram, OCL.

1. INTRODUCTION

The benefits of employing object technology in application design and development are now well known. Viewing real life problems in terms of objects can often lead to a more natural design and thus a better implementation [1]. Many recent research efforts have been devoted to the definition of object-oriented database models. These are data models based on the notions of object, class, attribute, and method, where classes and attributes are used to describe the structural aspects of object and methods are used to represent their dynamic aspects [2]. UML is a language to specify, construct, present and document artifacts of both software systems and business processes and other systems that are not strictly software [3]. Object-oriented software development has been in wide use for some time. There is now a stable, industry-standard notation for object-oriented analysis and design models – the Unified Modeling Language (UML) [4], [5].

The class diagram shows the static structure of classes in the system and it can also be used to measure the performance of any system [6]. Martin Gogolla and Mark Richters have presented the rules to make equivalences of various notations [7]. Security technique on the designed data cube is explained by saxena et al. [8]. Object Management Group [9] is an active group for the production of various kinds of static and dynamic types of UML diagram. In the Non-Postal Services, Indian Postal Services has Public Provident Fund, National Savings Certificate, Kisan Vikas Patra, Savings Bank Account and Recurring Deposit Account [10].

Relational database management systems (RDBMSs) are not designed to handle the complex data. The complexity of data is increasing and so the object-oriented database management systems (ODBMSs) would soon become the primary database technology.

2. THE UML CLASS DIAGRAM

In recent years, the Unified Modelling Language (UML) has emerged as the defacto standard for the representation of software engineering diagrams (Rumbaugh et al.1999) [5]. The UML class diagram contains classes, interfaces, collaborations, and dependencies, associations and interface relationships [6]. The UML class diagrams are used to describe the static view of any application. Classes and their relationships are the main constituents of class diagram. A class is a description of a concept which may have attributes and operations associated with it. Classes are represented as rectangles and relationship between two classes is drawn as a line.

3. TRANSFORMATION PROCESS

UML is very complex modeling language. The objective of transformation is to explain more complex features in terms of basic ones. Its static modeling with class diagram involves many description primitives such as qualifiers, cardinality constraints, association classes, compositions, aggregations and generalizations. The n-ary associations and OCL constraints are used for the transformation process.

The Object Constraint Language (OCL) is used to define instructions for specifying constraints and actions in any model. A constraint is a restriction on an element that limits the usage of that particular element. The OCL is a notational and formal mathematical expression language. When we view entire modeling details and relationship of any class diagram it appears complex but by using OCL, the complexity can be reduced. Relationships in class diagram are removed after they have been defined by the OCL expressions [6].

Multiplicities of all relationship types are not given but they are required only when statistics of data are calculated. By using these classes containing only binary associations and OCL constrains can be transformed into equivalent class diagram that involves cardinality constraints, qualifier, aggregation, composition, generalization and association. The rules used to design equivalent class diagram are called equivalence rules. In this research paper we are implementing the transformation from n-ary association to binary association.

4. CASE STUDY: POST OFFICE SYSTEM

In this section, we are designing the class diagram and sequence diagram for the National saving Certificate scheme and then the class diagram is transferred into equivalent class diagram.

4.1 The Class Diagram

Static representation of the system is done by UML class diagram. The class diagram for Indian Postal Services is represented in Figure 1. Two types of classes are present in this class diagram: Persistent and Transient. Persistent classes are those classes which can be stored in the database and object can return its value between different executions of programs.

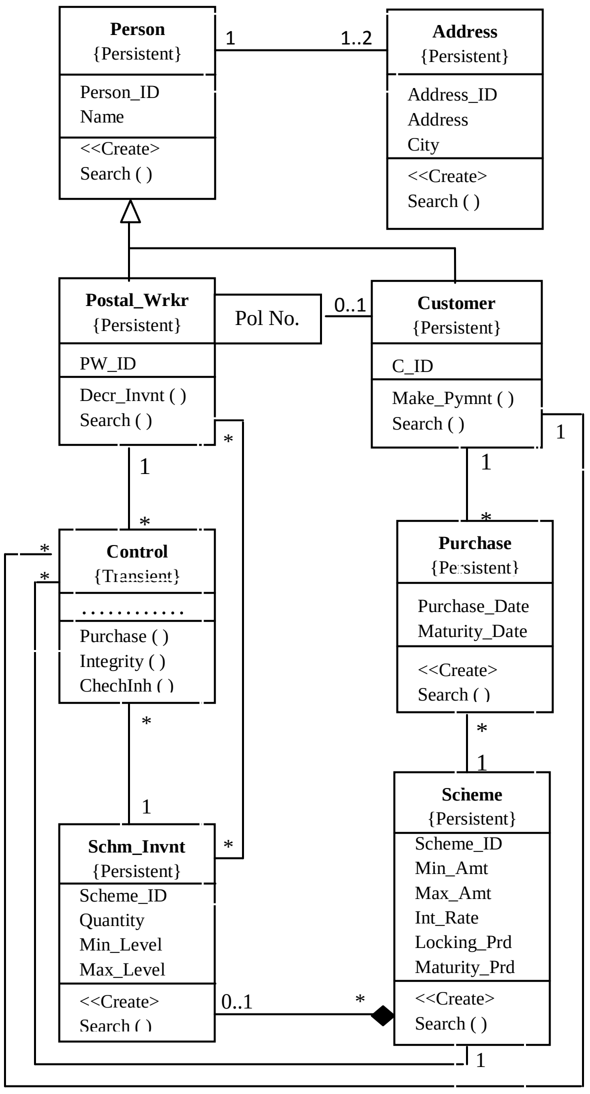

Fig 1: UML Class Diagram for Indian Postal Services

This class diagram contains seven persistent classes (Person, Address, Customer, Postal Worker, Scheme, Schm_Invnt and Purchase). These classes are connected to each other by various relationships with their multiplicities. Diagram contains one transient class (Control) for the smooth flow of interaction between classes. Subclasses Customer and Postal Worker are inherited from non-abstract class Person, which is disjoint and incomplete inheritance. This shows the process of generalization. Class postal worker has a qualifier Policy Number. A qualifier is an attribute or set of attributes of association. The postal worker can get details of customer by the attribute Policy Number. The diagram also shows the association between Scheme and Schm_Invnt classes by the composition because each scheme has its own scheme inventory, which cannot be shared by another one. The strong form of aggregation is known as composition.

4.2 The Sequence Diagram

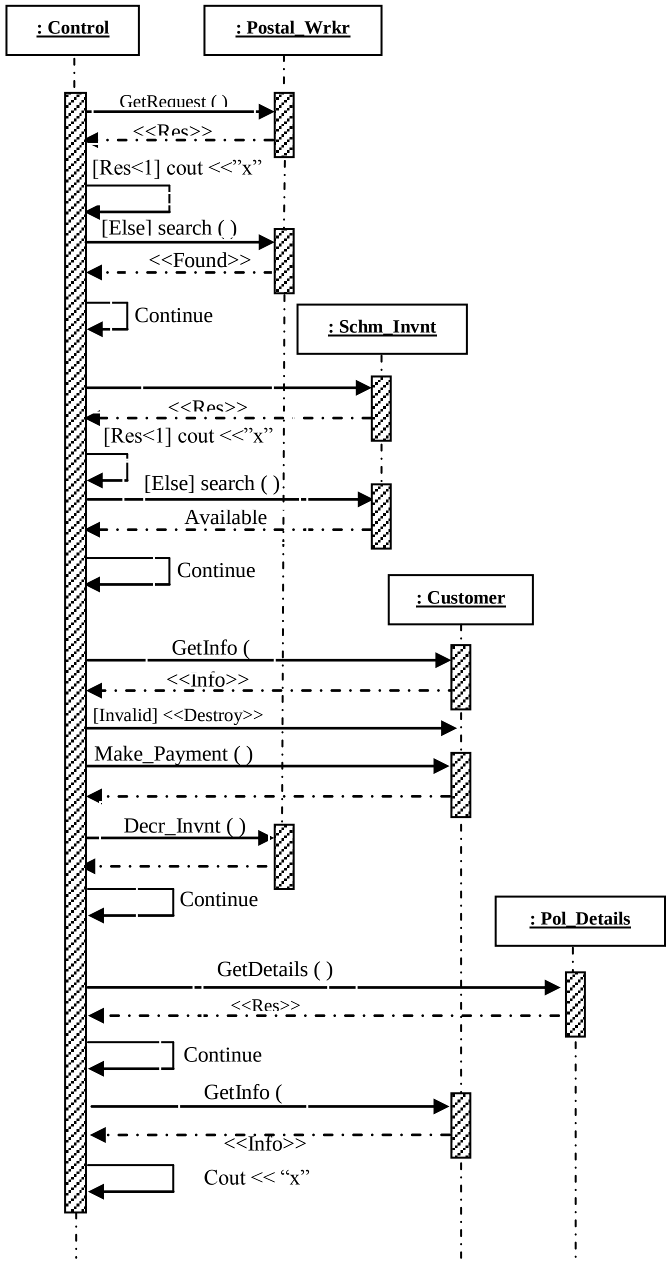

Fig 2: UML Sequence Diagram for Indian Postal Services

Sequence diagram tells how objects interact with each other i.e. how messages are being send and received. The control object is used for the smooth functioning of the system and it handles all objects from different classes. With the help of control object the customer requests for any scheme to the postal worker by using operation GetRequest( ). If the request is valid then integrity check is performed on scheme inventory object by using integrity( ). Interactions such as continue and destroy control the flow.

The customer is informed about the availability of scheme and makes payment by using Make_Payment( ). Decr_Invnt( ) decreases the number of schemes from the inventory after selling the scheme. The details of policies sold, are stored in Policy Details and a unique Policy Number is generated. Postal worker or the customer can get information from the policy details by using GetDetail( ) and GetInfo( ).

4.3 Designing the Equivalent Class diagram

Static modeling with UML class diagram involves many description primitives such as qualifiers, cardinality constraints, association classes, compositions, aggregations and generalizations [7]. The objective of transformation is to explain more complex features in terms of basic ones. Transformation process generates an equivalent class diagram from the existing one. Here the class diagram in Fig 1 is transformed into equivalent class diagram.

The class diagram presented in Fig 1 shows following complexities:

a. Attribute Policy No is used as quantifier;

b. Presence of association class;

c. Composition between the class Scheme and Scheme Inventory;

d. Generalization relationship between the class person and sub classes Customer and Postal Worker.

4.3.1 Equivalence Rule for Qualifier

The qualifier is translated into association class if it is already present.

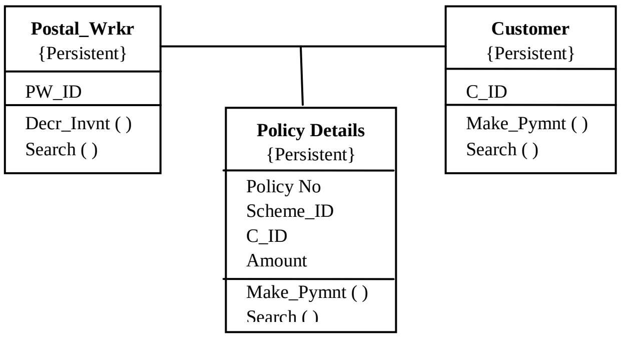

Fig 3: Equivalence rule for qualifier

The constraints require that the size of the set of customer object determined by the combination of postal worker object and the value of attribute Policy No is restricted by given lower and upper bound.

```
Customer : Postal_Wrkr -> Set (Customer),
Postal_Wrkr : Policy details -> Postal_Wrkr,
Customer : Policy details -> Customer,
Policy Details : Postal_Wrkr -> Set (Policy
Details),
Policy Details : Customer -> Set (Policy
Details).
```

4.3.2 Equivalence Rule for Association Classes

The translation of association classes are done into ternary associations, which is shown into Fig 4. A ternary association GetDetails is used between postal worker, customer and policy details. The constraint demands that a policy number is related to exactly one post office and to exactly one customer, and there cannot be a different policy numbers with the same links.

```
Sales Detail->forAll( p |
p.Postal Worker->size=1 and p.Customer->size=1
and
Sales Detail->forAll( p' | ( p. Postal Worker =
p'. Postal Worker and p.Customer=p'.Customer )
implies
p=p' ) )
```

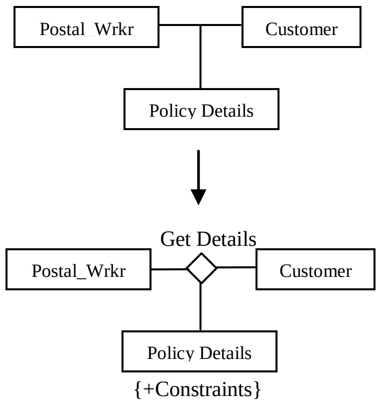

Fig 4: Equivalence rule for Association Class

4.3.3 Equivalence Rule for Composition

Fig 5 shows that the composition relation is translated into binary association. We require the part to be existentially dependent for the aggregate and a strong form of forbidding sharing.

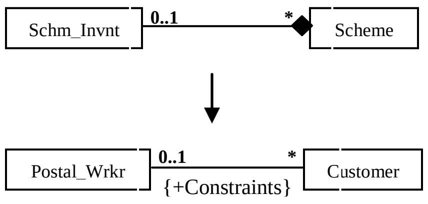

Fig 5: Equivalence rule for Composition

4.3.4 Equivalence Rule for Generalization

As shown in Fig 6, UML generalizations are transformed to special binary associations. The cardinalities make sure that each specialized object is related with exactly one general object. We can say that a special object is associated with a unique general object.

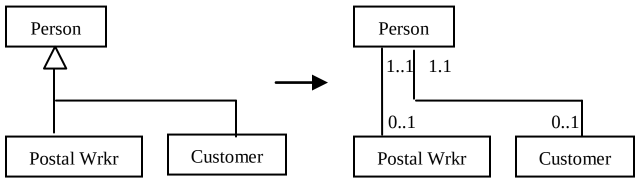

Fig 6: Equivalence rule for Generalization

The multiplicity between person and subclasses is 1..1 and 0..1 that means the any object from class person has at most one association with any object of sub class.

After that, we have to determine the average multiplicity and number of objects for each association end and each class extension. For this, we have to take information from the system. The data is not in clustered form and so the extensions of classes are stored.

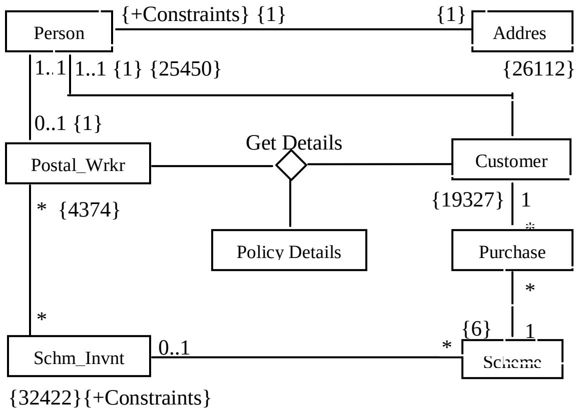

Fig 7: Equivalent Class Diagram of Fig 1

Fig 7 presents the final equivalent class diagram of the class diagram presented in Fig 1. The complex features of the class diagram such as qualifiers, association classes, compositions and generalizations are transformed into equivalent easy representations.

5. CONCLUSION AND FUTURE WORK

UML diagrams are designed for an OODBMS and here we have presented an approach to transform the existing UML class diagram into equivalent class diagram. Our results suggest that some of the UML class diagram concepts increases the complexity and works as a shortcut for existing one. We have translated few UML features such as qualifiers, association classes, compositions and generalizations into association relations with additional constraints because the association concept is a very general and it is able to model many situations. Further, it can be used for the performance analysis of the software system using class diagram.

6. ACKNOWLEDGMENTS

The authors are thankful to Prof. B. Hanumaiah, Vice-Chancellor, B.B. Ambedkar University (A Central University) Lucknow for providing the excellent facility in the computing lab of B. B. Ambedkar University, Lucknow, India. I am also thankful to Mr. Martin Gogolla and Mr. Mark Richters for his support to understand the concepts used in the design and implementation of our project.

7. REFERENCES

[1] A Case Study of an Object Database Implementation, Billy Gibson, APM. 1608.00.02 Draft 10th October 1995.

[2] Diego Calvanese, Maurizio Lenzerini. Making Object-Oriented Schemas More Expressive. in Proc. of PODS'94,

[3] Luca Vetti Tagliati, Carlo Caloro. UML and Object Oriented Drama, Online at http://www.jot.fm. Published by ETH Zurich, Vol. 7, No. 1, January-February 2008.

[4] Fowler, Martin (2004) UML Distilled, Third Edition: A Brief Guide to the Standard Object Modeling Language. Addison- Wesley, Boston.

[5] Rumbaugh, James, Ivar Jacobson, and Grady Booch (2005) The Unified Modeling Language Reference Manual, 2nd ed. Addison-Wesley, Boston.

[6] Ahmad Alsaadi, "A Performance Analysis Approach Based on the UML Class Diagram"

[7] Martin Gogolla and Mark Richters, "Equivalence Rules for UML Class Diagrams"

[8] Saxena, V., Verma, N.M.P. and Pratap, A., "A Data Mining Technique for a Secure Electronic Payment Transaction", International journal of Economics and finance, Vol. 2, No-4, ISSN 1916-9728, November 2010.

[9] Object Management Group, Unified Modeling Language: Superstructure, v.2.1.2. OMG, Needham, MA, USA, 2007

[10] "Indian post offices to go hi-tech" 25 November 2010. www.indiapost.gov.in.