TOWARDS FORMALIZING OO MODELS: A CASE STUDY

Kok Meng Yew, Helena Bulbul and Mashkuri Hj. Yaacob
Faculty of Computer Science and Information Technology
University of Malaya,
50603 Kuala Lumpur, Malaysia
email: helena@siswazah.fsktm.um.edu.my

ABSTRACT

Object Oriented System Development (OOSD) is gaining vast popularity among the software developers and researchers. However, the main challenge is to build a system which is reliable and less error-prone. The application of graphical informal Object Oriented (OO) technique for developing an OO system lacks validity checking though it can represent the system with more clarity. On the other hand, formal specification can facilitate the development of a correct implementation through automated reasoning techniques. The incorporation of formal specification together with informal technique helps in building complex and safety critical system.

In this paper, we analyzed a case study system applying both formal and informal techniques. The informal technique for developing object model is Object Modeling Technique (OMT), the notation used is of Unified Modeling Language (UML) and the formal specification language used is Object-Z.

Keywords: Formal method, Object-Z, OMT

1.0 INTRODUCTION

The use of graphical informal techniques for representing system's architecture and behavior particularly with respect to object oriented methodologies is gaining vast popularity among the software developers today. While providing system's information in a user friendly way and with more clarity, these types of methodology suffer from a lack of mathematical precision that allows misinterpretation. The semantics of the graphical languages associated with these methodologies are typically ambiguous, and thus imprecise. The business logic executed by the system is not predictable if the informal modeling technique is used. On the other hand, formal technique to the task of system development offers significant advantages. Formal methods are amenable to machine manipulation and their precision facilitates detailed methods of analysis [1],[2]. As both formal and informal techniques have their merits in the proper context during system development, it is better to incorporate both the techniques in the course of system development. The informal technique will provide system's information visually and with clarity, whereas the formal technique will ensure the correctness and reliability of the system developed with the informal technique through complex validation and extensive testing at every stage of development which is a must [3] to produce a reliable and less error-prone system.

The main goal of this paper is to use a formal modeling technique together with an informal technique in order to model and analyze the "LoanSch Processing System" of University of Malaya. The informal technique which we used to develop the graphical model of the system is Rambaugh's OMT and the formal technique which we have chosen is Object -Z.

Fig. 1 illustrates the process followed in developing and validating the object model.

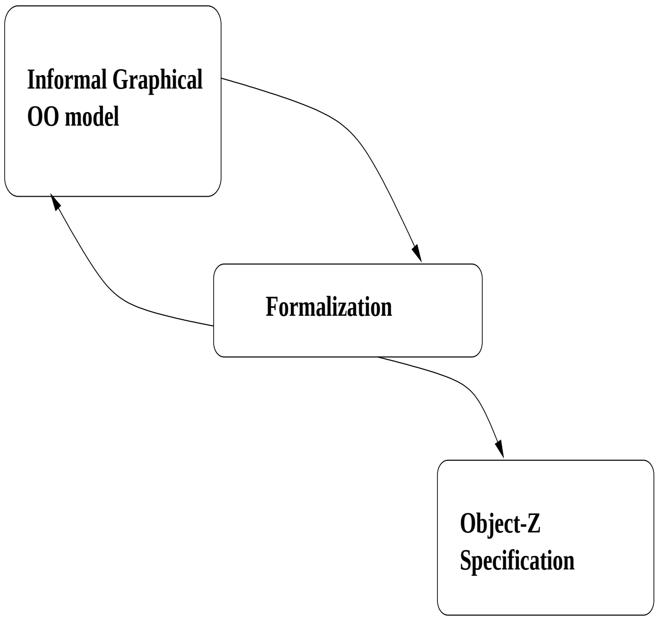

Fig. 1: The development and validation process

The formalizing activity starts in transforming the semantics captured by the graphical model into Object-Z specifications. The formalization was done manually. Typically, formalization helps in finding problems related to missing information and ambiguous structures. Once the flaws are diagnosed the informal model can be corrected to produce more clarified and enhanced model.

This paper reveals our experience in applying the development process to the case study system. Section 2 gives an overview of the notations and techniques used and also presents a brief overview of the case study system. Section 3 describes our experience in applying the development process to some scenarios of the case study problem. The conclusion is drawn in section 4 regarding our overall experiences throughout the development process.

2.0 BACKGROUND

2.1 The OMT Method (Notation UML)

The Object Modeling Technique (OMT), developed by J. Rumbaugh[4], can be seen as a design methodology which consists of a collection of predefined techniques and notational conventions in order to support the process, which consists of several phases(e.g. analysis, design and implementation) for the organized production of software. OMT, like many other popular modeling and development methods uses diagrams or graphical notations to represent system's architecture and operational behavior. This methodology uses three kinds of models to describe a system, each capturing different aspects of the system, on the whole describing the complete system. The object model describes the static, structural data aspects of the system. The dynamic model represents the temporal, behavioral, control aspects of the system whereas the functional model represents the transformational "function" aspects of the system.

2.2 Object-Z : An Overview

Object-Z [5] is a formal specification language which is basically an extension to the Z-specification to incorporate object orientation. The main motivation towards the extension is to improve the clarity of large specifications through enhanced structuring.

A Z-specification is typically based on predicate logic and set theory. The primary structuring construct of Z is the schema. A schema has two parts: a declaration and a predicate part. Schemas are used to model both static and dynamic aspects of the system. A state schema groups together variables and defines the relationship that holds between their values. The operation schema defines the relationship between the 'before' and 'after' states corresponding to one or more state schema. To know the operations which affects a particular schema, it needs to go through all the operation schema signatures which is impracticable for large specifications.

Object-Z overcomes this drawback of Z by grouping the operations to refer to one state schema which is closer to the concept of class. It supports the concept of inheritance and instantiation which builds towards a class representing the entire system. However, Object-Z does not explicitly specify the association between two classes.

Syntactically, a class definition is a named box, optionally with generic parameters. In this box, the constituents of the class are defined and related. Possible constituents are a visibility list, inherited classes, type and constant definitions, a state schema, an initial state schema, operation schemas, and a history invariant.

```
ClassName
  visibilitylist
  inheritedclasses
  typedefiniions
  constantdefinitions
  stateschema
  initialstateschema
  operationschemas
  historyinvariants
```

2.3 The LoanSch Problem Scenario

This section briefly describes the LoanSch system scenarios.

1. A sponsor offers many scholarships, for which a student may apply. A student may apply more than one scholarship. A student is said to be applied when he/she applies by filling in the application form and submit to the Student Affair division.
2. A student is said to be awarded if the Sponsor selects him/her after an interview and is said to be rejected if the sponsor does not select. An awarded student has to sign an agreement with the sponsor.
3. Keeping and managing sponsor information as well as student information updating is within the scope of the system.

3.0 FORMALIZATION OF CASE STUDY OO GRAPHICAL ANALYSIS MODEL

In this section we present our case study system requirement model and outline the rationale underlying the model.

3.1 The Concepts/Classes of the Case Study System

Sponsor: The attributes of a Sponsor which are of concern for the system are its name and address. The sponsor has an aggregation relationship with one or more scholarships. The aggregation in between the sponsor and the scholarships can be modeled as a data attribute of the sponsor. An instance of a scholarship must belong to a unique sponsor.

Scholarship: The properties of a scholarship which are relevant are its amount, duration, its bond type. A scholarship is a part of a sponsor.

Student: The attributes which are of concern for the student are his/her name, faculty name, metric no., year of study as well as his/her address. A student can apply for more than one scholarship. There is an association in between scholarship and student.

Notice: A notice has to be produced after the sponsor declares offerings. It has a deadline as its attributes as well as the sponsor's information as a reference.

Interview schedule: The interview schedule class has date,time, venue as attributes. Different interview schedule are produced and allocated to different students who applied.

Agreement schedule: After the interview is over and the student is selected, an agreement has to be signed between the student and the sponsor. The agreement schedule class generates the agreement signing date, time and venue information as well as allocating a particular schedule to a particular student.

The class diagram reflecting the above scenario is given in Fig. 2. The notation used is of UML[6] because it is an industrial standard and it is very effective in modeling a system [7].

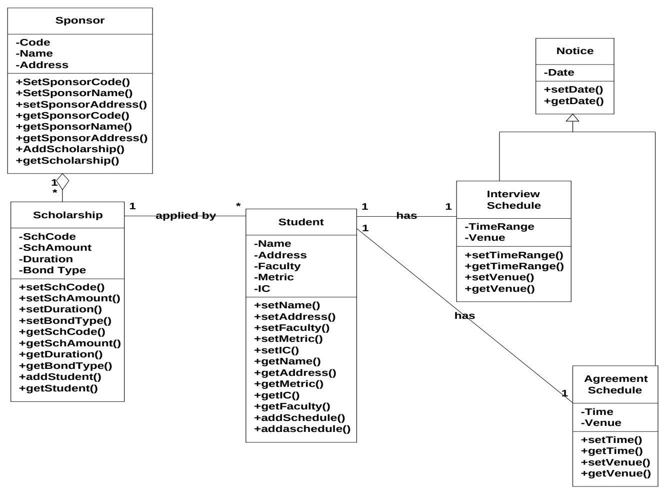

Fig. 2: The class diagram of "LoanSch Processing System"

3.2 Formalizing the Analysis Model Using Object- Z Notation

Formalization of graphical OO model can be done in one of the three ways namely supplemental, OO-extended formal notation, and method integration approaches [8]. Among them, we followed the method integration approach. In this section, we present the Object-Z formalization of our Case Study analysis model. The formalization was carried out according to the guidelines presented in section 2. Formalization starts its first step by formalizing the parts of the class diagram and then composing them in an Object-Z schema formally characterizing the system view of the class diagram.

The Object-Z formalization of the class diagram is given below:

The class schema for Notice class is as follows:

$[DATE]$

Notice

$Date : \mathbb{P}\, DATE$

INIT

$Date = \varnothing$

setDate

$\Delta(Date)$

$Date? : \mathbb{P}\, DATE$

$Date = \varnothing$

$Date' = Date?$

getDate

$Date : \mathbb{P}\, DATE$

$Date \neq \varnothing$

$Date = Date!$

$setDate \mathrel{\widehat=} setDate$

$getDate \mathrel{\widehat=} getDate$

The class schema for agreement schedule class is stated as follows:

$[TIME, VENUE]$

Ashedule

Notice

$Time : \mathbb{P}\, TIME$

$Venue : \mathbb{P}\, VENUE$

INIT

$Time = \varnothing$

$Venue = \varnothing$

setTime

$\Delta(Time)$

$Time? : \mathbb{P}\, TIME$

$Time = \varnothing$

$Time' = Time?$

getTime

$Time : \mathbb{P}\, TIME$

$Time \neq \varnothing$

$Time = Time!$

setVenue

$$\begin{array}{l} \Delta(Venue) \\ Venue? : \mathbb{P}\, VENUE \\ \hline Venue = \varnothing \\ Venue' = Venue? \end{array}$$

getVenue

$$\begin{array}{l} Venue : \mathbb{P}\, VENUE \\ \hline Venue \neq \varnothing \\ Venue = Venue! \end{array}$$

$$\begin{array}{l} setTime \mathrel{\widehat{=}} setTime \\ getTime \mathrel{\widehat{=}} getTime \\ setVenue \mathrel{\widehat{=}} setVenue \\ getVenue \mathrel{\widehat{=}} getVenue \end{array}$$

The class schema for the class Student is as follows:

$$[STDNAME, FACULTY, METRIC, YSTUDY, IC]$$

Student

$$\begin{array}{l} Name : \mathbb{P}\, STDNAME \\ Faculty : \mathbb{P}\, FACULTY \\ Metric : \mathbb{P}\, METRIC \\ Ystudy : \mathbb{P}\, YSTUDY \\ Ic : \mathbb{P}\, IC \\ aschedule : \mathbb{P}\, Aschedule \end{array}$$

INIT

$$\begin{array}{l} Name = \varnothing \\ Faculty = \varnothing \\ Metric = \varnothing \\ Ystudy = \varnothing \\ Ic = \varnothing \end{array}$$

setName

$$\begin{array}{l} \Delta(Name) \\ Name? : STDNAME \\ \hline Name = \varnothing \\ Name' = Name? \end{array}$$

getName

$$\begin{array}{l} Name : \mathbb{P}\, STDNAME \\ \hline Name \neq \varnothing \\ Name = Name! \end{array}$$

$$\begin{array}{l} setName \mathrel{\widehat{=}} setName \\ getName \mathrel{\widehat{=}} getName \end{array}$$

The class schema for the class scholarship is as follows:

$$[SCHCODE]$$

Scholarship

$$\begin{array}{l} schcode : \mathbb{P}\, SCHCODE \\ schamount : \mathbb{R} \\ schduration : \mathbb{N} \\ schbond : \mathbb{B} \\ nos : \mathbb{P}\, \mathbb{N} \\ student : \mathbb{P}\, Student \end{array}$$

INIT

$$\begin{array}{l} schcode = \varnothing \\ schamount = \varnothing \\ schduration = \varnothing \\ schbond = \varnothing \\ nos = \varnothing \end{array}$$

SetScholarshipCode

$$\begin{array}{l} \Delta(schcode) \\ schcode? : SCHCODE \\ \hline schcode = \varnothing \\ schcode' = schcode? \end{array}$$

getScholarshipCode

$$\begin{array}{l} schcode! : \mathbb{P}\, SCHCODE \\ \hline schcode \neq \varnothing \\ schcode = schcode! \end{array}$$

AddStudent

$$\begin{array}{l} \Delta(nos) \\ student? : Student \\ \hline nos' = nos + 1 \end{array}$$

$$\begin{array}{l} setScholarshipCode \mathrel{\widehat{=}} setScholarshipCode \\ getScholarshipCode \mathrel{\widehat{=}} getScholarshipCode \\ AddStudent \mathrel{\widehat{=}} AddStudent \end{array}$$

The class schema for the class Sponsor is as follows:

$$[SPCODE, SPNAME, SPADDRESS]$$

Sponsor

$$\begin{array}{l} spcode : \mathbb{P}\, SPCODE \\ spname : \mathbb{P}\, SPNAME \\ spadd : \mathbb{P}\, SPADDRESS \\ nosch : \mathbb{N} \\ scholarship : \mathbb{P}\, Scholarship \end{array}$$

INIT

$$\begin{array}{l} spcode = \varnothing \\ spname = \varnothing \\ spadd = \varnothing \\ nosch = \varnothing \end{array}$$

SetSponsorCode

$$\begin{array}{l} \Delta(spcode) \\ spcode? : SPCODE \\ \hline spcode = \varnothing \\ spcode' = spcode? \end{array}$$

getSponsorCode

$$\begin{array}{l} spcode! : \mathbb{P}\, SPCODE \\ \hline spcode \neq \varnothing \\ spcode = spcode! \end{array}$$

AddScholarship

$\Delta(nosch)$

$scholarship? : Scholarship$

$nosch' = nosch + 1$

$setSponsorCode \mathrel{\widehat=} setSponsorCode$

$getSponsorCode \mathrel{\widehat=} getSponsorCode$

$AddScholarship \mathrel{\widehat=} AddScholarship$

4.0 CONCLUSION

The goal of using formal specifications together with informal OO modeling is to develop an object oriented system which is less error-prone and reliable. Graphical OO model as derived using Object Modeling Technique (OMT) has its advantage in closely representing real world concepts with more clarity. Once the model is specified in formal language, the model can be checked using the corresponding language type checker. For developing our case study system, both formal and informal methods have been used. Most of the concepts of our case study system could be easily specified in Object-Z. However, the central role of associations between classes such as Sponsor and Scholarship as represented in the OMT analysis model raises problem for the formalization process. The problem arises because Object-Z and none of the formal object oriented specification languages have an explicit representation of associations between object classes[9]. Hence the design decision may be quite immature as its dependency on the analysis specifications. As the available formal object oriented specification is quite incomplete in specifying all features of an object model as represented graphically, the specification is not complete too. Full formalization of these models requires more research in the concurrent and dynamic properties of objects in the formal specification languages. However, the integration of formal specification together with informal technique will enable rigorous consistency and refinement checks that are not possible with the sole use of the graphical modeling techniques.

REFERENCES

[1] M. W. Jeannette, "A Specifier's Introduction to Formal Methods," *IEEE Computer*, September 1990, pp. 8–24.

[2] H. C. C. Betty, "Applying Formal Methods in Automated Software Development," *Journal of Computer and Software Engineering*, Vol. 2, No. 2, 1994, pp. 137–164.

[3] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. McGraw Hill, 3rd ed., 1992.

[4] J. Rambaugh, M. Blaha, W. Premerlaniand, F. Eddy, and W. Lorenson, *Object Oriented Modeling and Design*. Prentice Hall, 1991.

[5] D. Carrington, D. Duke, R. Duke, P. King, G. A. Rose, and G. Smith, "Object-Z: An Object-Oriented Extension to Z," in *Formal Description Techniques, II(FORTE'89)*, 1990, pp. 281–296, North-Holland.

[6] T. U. Group, *Unified Modeling Language. Version 1.0.* Rational Software Corporation, Santa Clara, CA-95051, USA, January 1997.

[7] Zahidur R.M., Bulbul H., Zaitun. A.B., and M. Yaakob, "Object Oriented Modeling: An Experience Using The UML Approach," in *REDECS97: National Conference on Research and Development in Computer Science and its Applications*, 1997, pp. 69–75, School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia.

[8] R. France, A. Evans, K. Lano, and B. Rumpe, "The [UML] as a Formal Modeling Notation," *submitted to Computer Standards and Interfaces*, 1998.

[9] K. Lano and H. Haughton, eds., *Object-oriented Specifications Case Studies*. The Object-Oriented Series, Prentice Hall International(UK) Limited, 1994.

BIOGRAPHY

Kok Meng Yew was an Associate Professor of the Faculty of Computer Science and Information Technology at University of Malaya.

Helena Bulbul received B.Sc. degree in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology in 1992. She is currently pursuing her Master degree in the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia.

Mashkuri Hj. Yaacob is a Professor of Computer Science and Dean of the Faculty of Computer Science and Information Technology, University of Malaya. He joined University of Malaya in 1976. He has published over 130 research papers and presented papers at both local and international conferences. He is a member of the IASTED Conference Organizing Committee and IEEE Computer Society. His research interests are software engineering, and computer architecture which includes ATM network architecture, virtual circuit management performance and others.