Transforms Class to Formal Specification By Object-Z

Wasun Khan-am[1]

[1]Associate Professor: Dept. of Information Systems
Faculty of Business Administration, RMUTT
Pathum thani, Thailand 12110
e-mail: wasun_k@rmutt.ac.th

บทคัดย่อ — บทความนี้นำเสนอตัวอย่างของการเปลี่ยนสัญลักษณ์คลาสในแผนภาพคลาสเป็นรายละเอียดแบบเป็นทางการด้วยการใช้สัญลักษณ์ซีด และวัตถุแบบซีด ในการสร้างแผนภาพคลาส มีการกำหนดสัญลักษณ์ที่ใช้เป็นตัวแทนของคลาส สัญลักษณ์ที่ใช้เป็นตัวแทนของคลาสจะถูกแยกออกเป็นสามส่วน แต่ละส่วนของสัญลักษณ์คลาสจะถูกนำไปเปลี่ยนเป็นแต่ละส่วนที่อยู่ในวัตถุแบบซีด เมื่อดำเนินการจนครบจะได้วัตถุแบบซีดที่สอดคล้องกับสัญลักษณ์คลาสในแผนภาพคลาส

คำสำคัญ: สัญลักษณ์แบบซีด, การแปลงคลาส, รูปแบบอย่างเป็นทางการ

Abstract — this paper presents an example for transforming class symbol in class diagram to formal specification using z-notation and object-Z. In class symbol, there are three parts in class. Each part of given class transforms to each section of Object-Z model. To sum up, the result is found that transform class to formal specification by z-notation and object-Z is possible as show in this paper.

Keywords Z-notation; object-Z; transform class; formal specification

I. INTRODUCTION

Formal Specification is a kind of mathematics language. The benefit of formal specification is concise due to using mathematics symbol.

Object-Oriented Design is a famous method which use to design proposed system or application. The detail design output which is essential for development are a class diagrams and interaction diagrams including: sequence and communication diagram. The detail design output is show a vision of purposed system called model.

Although the model have been obtained, but sometime model is vague for system development. Programmer may or may not be able to create correctly program corresponds to a designed model. It should be made as an obvious model. The formal specification of purposed system need to be generated with a mathematical models. The final detail model is clear for application development.

At the last ACTIS'2016, my presentation about apply JavaScript module to create pie graph and for making model precisely, I need to create a formal specification for my application as show in this paper.

II. LITERATURE REVIEW

This content of this paper relevant with formal specification, notation for creating formal specification; and class design, by using UML tools.

A. Formal Specification [1]-[2]

Formal specification is process that use for creating definitely model of a proposed system. It is an unambiguous description of system function, ease of implementation, and be verifiable such as a mathematical proof.

B. Z-notation [1]-[2]

Z-notation is a one approach to generate formal specification. It normally use to transform structure or state diagram to specification. The transformed specification is transform again to program. The advantage of z-notation are to make a specification clear and precise.

The main idea of Z is to decompose a specification into small pieces, called schemas. After break up, the pieces need to describe in formal mathematics. There are two aspects of Z are: static aspects including state, and invariant relationships; and dynamic aspects including operation, relationship, and state transition.

C. Object-Z [3]-[5]

Object-Z is expanded from the Oxford formal specification language Z and works under object-oriented context. By using Object-z, developer can specify a set of states that system be moving from one state to other. This legally way be called as system's operations.

The main extension in Object-Z is class which encapsulates of state schema and operation affecting to state. The symbolic represent class show in the next below.

ClassName[*generic parameters*]
inherited classes
type definitions
constant definition
state schema
initial state schema
operation schema

history invariant

D. UML [6]

UML is an object-oriented tool. The word "UML" stands for Unified Modeling Language, and is a two-dimension language to model target system in object-oriented style. UML consists of a diagram for documenting target system, and presenting an interaction among objects; and dynamics of those object.

In UML, There are fourteen diagrams as: use case, class, object, state machine, activity, interaction overview, sequence, communication, component, deployment, profile, timing, composite structure, and package diagram.

Class diagram consists with class and relationship among those class or object instanced from class. The class represents by the follow symbol.

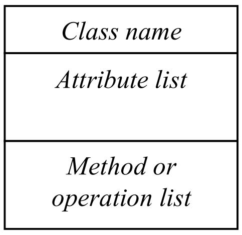


Figure 1. Class symbol

Figure 1 show a class symbol with 3 compartment are:

- The top compartment refer to class name
- The middle compartment refer to class' attribute
- The low compartment refer to class' method or operation or behavior

III. Result

In this section, a class design and result of class transformation are presented.

A. Class design

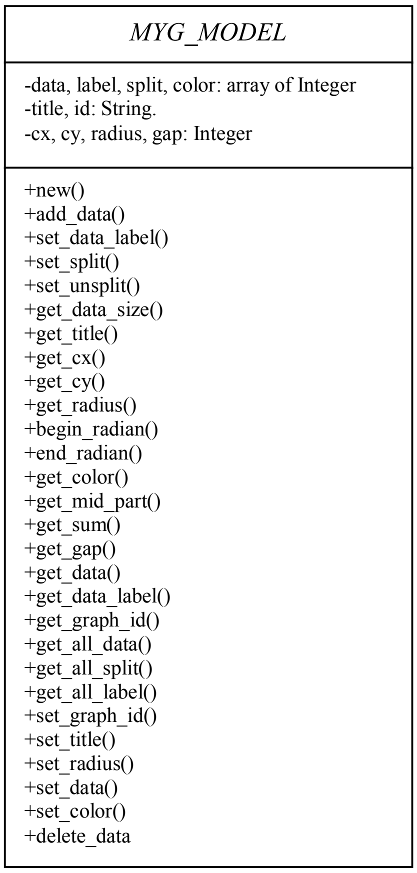
In class diagram, the MYG_MODEL class is designed as a kind of model class and is shown in the figure 2.

B. Class result

Firstly, the transformation are starting from translate class to formal specification as follow:

MYG_MODEL
{Place for attribute}
{Place for operation}

In the formal specification above, there are two compartments. First compartment is for attribute and another space is for operation. Both of space need to fulfill at the end of transformation.



MYG_MODEL

-data, label, split, color: array of Integer
-title, id: String.
-cx, cy, radius, gap: Integer

+new()
+add_data()
+set_data_label()
+set_split()
+set_unsplit()
+get_data_size()
+get_title()
+get_cx()
+get_cy()
+get_radius()
+begin_radian()
+end_radian()
+get_color()
+get_mid_part()
+get_sum()
+get_gap()
+get_data()
+get_data_label()
+get_graph_id()
+get_all_data()
+get_all_split()
+get_all_label()
+set_graph_id()
+set_title()
+set_radius()
+set_data()
+set_color()
+delete_data

Figure 2. MYG_MODEL represent MODEL in MVC of application.

In figure 2, there are 10 attributes, 4 array of integer; 2 string and 4 integer; and 28 methods including 8 set methods; 16 get methods and 4 other methods; in MYG_MODEL class

C. *Attribute result*

Secondly, all attribute of class are transformed to *type definition, constant, and state schema* and replace back to the class specification. The output is shown next.

MYG_MODEL

$[CHAR]$
$STRING: \mathrm{seq}\ CHAR$

$cx, cy, radius, gap: \mathbb{N}$
$id, title: STRING$
$label: \mathbb{N} \rightarrow STRING$
$data, split: \mathbb{N} \rightarrow \mathbb{N}$

$cx > radius \land cy > radius$
$radius > 0 \land gap > 0$

D. *Init-operation*

When attribute transformation have been finished, Init operation is an essential operation that have to be created. The Init-Operation is a special objective operation that use for initializing proper value to all attribute and running special tasks.

Init

$cx, cy, radius, gap: \mathbb{N}$
$id, title: STRING$
$label: \mathbb{N} \rightarrow STRING$
$data, split: \mathbb{N} \rightarrow \mathbb{N}$

$cx > radius \land cy > radius \land cx > 0 \land cy > 0 \land$
$radius > 0 \land gap > 0$
$cx = cy = 300$
$radius = 100$
$id = title = \langle\rangle$
$data = split = label = \varnothing$

E. *Other operation*

After Init-Operation has been declared, the next task are to transform other operations in class to formal specification. The result of this operation show as below.

add_data

$\Delta\ (data, label, split)$
$num: \mathbb{N}$

$num = \#data$
$data' = data \cup \{num \mapsto 0\}\ ⨾$
$split' = split \cup \{num \mapsto 0\}\ ⨾$
$label' = label \cup \{num \mapsto \text{‘data ’} + num\}$

set_split

$\Delta\ (split)$
$num?: \mathbb{N}$

$num < \#data$
$split' = split \oplus \{num? \mapsto 1\}$

set_unsplit

$\Delta\ (splitl)$
$num?: \mathbb{N}$

$num < \#data$
$split' = split \oplus \{num? \mapsto 0\}$

get_data_size

$data\_size!: \mathbb{N}$

$data\_size! = \#data$

set_data_label

$\Delta\ (label)$
$num?: \mathbb{N}$
$label?: STRING$

$num < \#data$
$label' = label \oplus \{num? \mapsto label?\}$

begin_radian

$num?, before, radian!: \mathbb{N}$

$num? < \#data$
$before = 0\ ⨾$
$\forall\ i : 1..num?-1 \bullet before' = before + data(i)⨾$
$sum = self.get\_sum()⨾$
$radian! = 2 * \pi * before / sum$

end_radian

$num?, end, radian! : \mathbb{N}$

$num? < \#data$
$end = 0 \fatsemi$
$\forall\, i : 1..num? \bullet end' = end + data(i) \fatsemi$
$sum = self.get\_sum() \fatsemi$
$radian! = 2 * \pi * end / sum$

get_graph_id

$id!, id : STRING$

$id! = id$

get_title

$title!, title : STRING$

$title! = title$

get_cx

$cx!, cx : \mathbb{N}$

$cx! = cx$

get_cy

$cy!, cy : \mathbb{N}$

$cy! = cy$

get_color

$color!, num? : \mathbb{N}$
$color : \mathbb{N} \rightarrow \mathbb{N}$

$num? < \#data$
$color! = second(\{num?\} \lhd color\})$

get_mid_part

$sum!, sum, num? : \mathbb{N}$

$\#data > 0 \wedge num? < \#data$
$sum = 0$
$\forall i : 1..\ num? - 1 \bullet sum' = sum + data(i)$
$sum! = sum + data(num?) / 2$

get_all_data

$data!, data : \mathbb{N} \rightarrow \mathbb{N}$

$data! = data$

get_gap

$gap!, gap : \mathbb{N}$

$gap! = gap$

get_sum

$sum!, sum : \mathbb{N}$

$\#data > 0$
$sum = 0 \fatsemi$
$\forall i : 1..\#data \bullet sum' = sum + data(i) \fatsemi$
$sum! = sum$

get_data

$data!, num? : \mathbb{N}$
$data : \mathbb{N} \rightarrow \mathbb{N}$

$num? < \#data$
$data! = second(\{num?\} \lhd data\})$

get_radius

$radius!, radius : \mathbb{N}$

$radius! = radius$

get_data_label

$num? : \mathbb{N}$
$label! : STRING$
$label : \mathbb{N} \rightarrow STRING$

$num? < \#data$
$label! = second(\{num?\} \lhd label\})$

get_split

$split!, num? : \mathbb{N}$
$split : \mathbb{N} \rightarrow \mathbb{N}$

$num? < \#data$
$split! = second(\{num?\} \lhd split\})$

get_all_label

$label!, label : \mathbb{N} \rightarrow STRING$

$label! = label$

get_all_split

$split!, split : \mathbb{N} \rightarrow \mathbb{N}$

$split! = split$

set_graph_id

$\Delta\,(id)$
$id, id?: STRING$

$id = id?$

set_radius

$\Delta\,(radius)$
$radius, radius?, cx, cy: \mathbb{N}$

$radius? < cx \wedge radius? < cy \wedge radius? > 0$
$radius = radius?$

set_data

$\Delta\,(data)$
$num?, data?: \mathbb{N}$
$data : \mathbb{N} \rightarrow \mathbb{N}$

$num? < \#data$
$data \oplus \{num? \mapsto data?\}$

set_color

$\Delta\,(color)$
$num?, color?: \mathbb{N}$
$color : \mathbb{N} \rightarrow \mathbb{N}$

$num? < \#data$
$color \oplus \{num? \mapsto color?\}$

delete_data

$\Delta\,(data)$
$delete\_set?, tmp, data : \mathbb{N} \rightarrow \mathbb{N}$
$label?, tmp1 : \mathbb{N} \rightarrow STRING$

$tmp = \varnothing$
$tmp1 = \varnothing$
$\forall\, i: 1..\#delete\_set? \bullet$
$\quad tmp' = tmp \cup \{delete\_set(i) \mapsto data(i)\}$
$data' = data \setminus tmp$
$tmp = \varnothing$
$\forall\, i: 1..\#delete\_set? \bullet$
$\quad tmp' = tmp \cup \{delete\_set(i) \mapsto split(i)\}$
$split' = split \setminus tmp$
$\forall\, i: 1..\#delete\_set? \bullet$
$\quad tmp1' = tmp1 \cup \{delete\_set(i) \mapsto data(i)\}$
$label' = label \setminus tmp1$

set_title

$\Delta\,(title)$
$title, title?: STRING$

$title = title?$

set_cx

$\Delta\,(cx)$
$cx, cx?: \mathbb{N}$

$cx? > radius \wedge cx? > 0$
$cx = cx?$

set_cy

$\Delta\,(cy)$
$cy, cy?: \mathbb{N}$

$cy? > radius \wedge cy? > 0$
$cy = cy?$

F. Compose the result

When structure of class was been built and both of attributes and operations have been transformed, those attribute and operation should be compose back to an appropriate position where they should be in. The result of put them back to the specification is full formal specification of MYG_MODEL class. Consequently, the specification is ready to pass to program implementation phase.

IV. CONCLUSION

Transform class in class diagram to formal specification have been successful by using both of Z-language, and Object-Z. All class attributes and class operations of MYG_MODEL have been transformed. By using both of the transformed formal specification and a class diagram not only will assist programmer to create program better than using only class diagram, but also help system designer to design a better interaction diagram. The transformed specification is clear and testable.

REFERENCES

[1] J.M. Spivey., *The Z notation: A reference Manual, 2nd ed.* Oriel College. Oxford, OX1 4EW:England. 1998.

[2] ISO, *ISO/IEC 13568:2002(E):Information technology-Z formal specification notation- Syntax, type system, and semantics,1st ed..* Switzerland. 2002.

[3] R. duke, P. King, G.Rose, and G. Smith. *The Object-Z Specification Language: Version 1.* Queensland 4072:Australia. 1991.

[4] A. Hussey and D.carrington. *Using Object-Z to Compare the MVC and PAC Architectures.* Proceedings of BCS-FACS Workshop on Formal Aspects of the Human Computer Inteface, Sheffield Hallam University, 10-12 September 1996.

[5] J. Derrick, E.A. Boiten, H. Bowman and M. Steen. *Translating LOTOS to Object-Z.* Proceedings of the 2nd BCS-FACS Northern Formal Methods Workshop, Ilkley, 14-15 July 1997.

[6] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide,* Addison-Wesley. 1999.