THE FORMAL SPECIFICATION FOR THE INVENTORY SYSTEM USING Z LANGUAGE

Siti Halimah Bakri[1], Hanis Harun[2], Amera Alzoubi[3], and Rosziati Ibrahim[4]

[1]sitihalimahbakri@gmail.com , [2]muhdhanis08@gmail.com ,
[3]a3k_126@yahoo.com, [4]rosziati@uthm.edu.my,

ABSTRACT: Formal methods use mathematical notations to precisely express requirements specification. The formal specification removes ambiguity which is inherently present in natural language specification. Formal specification also addresses the software reliability. In this paper, we present the formal specification for the inventory system by using Z language. Based from the scenario of the inventory system, we present the Unified Modeling Language (UML) specification by means of use case diagram and class diagram. We then present the Z schema based from the UML specification. We also show the consistency between UML specification and Z schema for this inventory system. The Z schema can effectively improve system reliability and reduce defect in developing the system.

Keywords: formal method, formal specification, Z language, inventory system, UML diagram

INTRODUCTION

Z is a formal specification language, which is a set of conventions for presenting mathematical text, used for describing and modeling computing systems (Spivey, 1992). Z language is one of the widely used formal specification languages since people are already familiar with its syntax and semantics (Jonathan, 1996).

This paper defines requirements to establish a formal model of checking the inventory system, to deliver the item and invoice to the customer project. The project named as Stock System and Z language is used to describe the notation of every function from the UML class diagram. The rest of the paper is organized as follows. Section 2 presents the related work and Section 3 demonstrates the case study of inventory system. Section 4 proposed the Z schema based from the UML class diagram and finally Section 5 concludes the paper.

RELATED WORK

Yusufu and Yusufu (Yusufu and Yusufu, 2012) discussed and compared the properties of five formal specification methods (Z language, B method, UML, LOTOS, SDL) by designing a particular part of the ABM system for each method. They also discussed the two formal specification methods, SDL and LOTOS, by emphasizing some similarities among them and address their differences based on a particular part of the ABM system, and further compare these methods by analysing their strengths and weaknesses.

Latif et al. (Latif et al., 2007) used Z specification language to specify multimedia content in safety critical systems. The extensions in Z specification language are suggested to add

support for multimedia in Z. Z has been chosen because it is the most revered language in formal methods. It is one of the widely used formal languages so people are already familiar with its syntax and semantics. They also emphasized that multimedia application development is structural instead of object oriented approach therefore Z language is quite suitable for this kind of development.

Fukagawa et al. (Fukagawa et al., 1995) discussed structural mapping system from Object-Z to the programming language C++, and presented on its implementation on UNIX. The structural mapping translates an Object-Z specification consisting of classes into class interfaces of C++ such as data members and prototypes of member functions. Thus it is not intended as a code generation system, but rather as a tool for analysing specification (including syntax and type checking) and for aiding a software developer in obtaining code. Through the implementation of the mapping system several language features of Object-Z and C++ concerning object-orientation are clarified.

According to Yu et al. (Yu et al., 2008) using the specification language Z, to design and describe a formal mathematical model of a class scheduling system. Their formal model is using Z specifications that representing the system's data model, system state and operations. They claimed that the specification can effectively improve system reliability, design time and comprehensibility. Therefore it greatly improves the quality of system design and development. Cimatti et al. (Cimatti et al., 2013) proposed a formal method for hybrid systems using the scenario based.

THE CASE STUDY OF AN INVENTORY SYSTEM

In this paper we present a case study of an inventory system. A developer wants to develop a simple system to monitor his business of stock, if the stock is available, deliver the goods to customer and bill the invoice. The first step is to determine the requirements analysis for the scenario. The requirements analysis is presented by giving the algorithm for the inventory system which is as follows:

- Start
  - Read file from database.
  - Login.
  - If login success, these following activities can be done:
    - Monitor Stock
    - Update Stock
    - Get Order
    - Generate Invoice
    - View Order
  - Else, re-login or register for new user.
- End

Based on the algorithm, we then draw the main diagram of Unified Modeling Language (UML) mainly the use case diagram and class diagram. Figure 1 shows use case diagram for the inventory system. There are only one user to use the system, which can do activity for Login, and other activities such as Get Order, Monitor Stock, Update Stock and Generate Invoice.

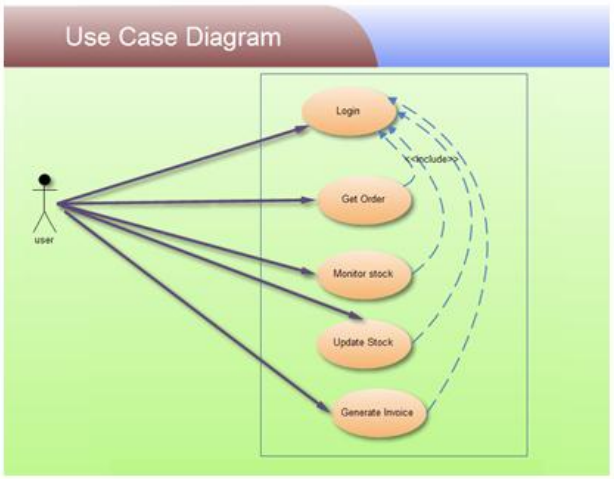

Figure 1. UML Use Case Diagram for Inventory System

From the use case diagram, Figure 2 shows the class diagram identified by the use cases from Figure 1. There are 8 functions generate by 3 classes. The three classes are User Class, Order Class and Product Class. Each function will transform to Z scheme as discussed in the next section.

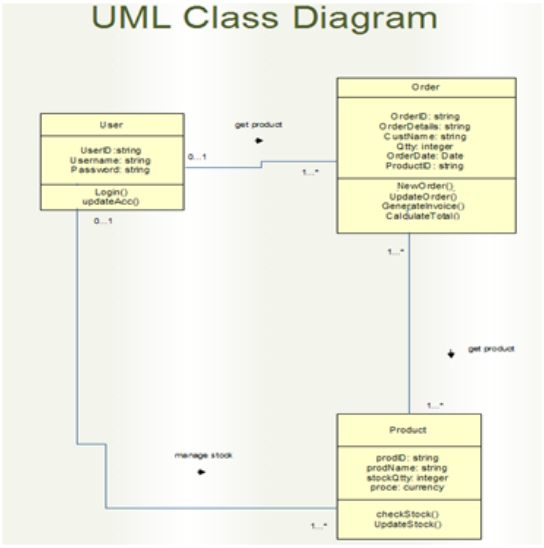

Figure 2. UML Class Diagram for Inventory System

FROM UML SPECIFICATION TO Z SPECIFICATION

In this project we use Z-EVES. Z-EVES includes a graphical user interface that allows Z specifications to be entered, edited, and analysed in their typeset form; supports the incremental analysis of specifications; and manages the synchronization of the analysis with modifications to the specification. Before implementing the UML specification, the database for the Inventory System should be identified. The Inventory System consists of three table object which are User, Order and Product. Each of the operation executed in the table object requires response from the system. So we represent the respond message by using variable Response. There are only two types of response which are Success or Failed. Success respond occurs when the overriding process or execution by user was successfully written in the database in whatever operation while Failed respond indicates that the system were not

override data in the database. This will be explained through Z scheme in the following sub-sections.

Database:

StockSys: P User // to represent attribute for class User

StockSys: P Order // to represent attribute for class Order

StockSys: P Product // to represent attribute for class Product

Response by system:

Response ::= Success, Failed

Z Schema for Class User

The following schema represents all methods for class User. The Inventory System provides login and update account for each of it user. Figure 3 shows the Z schema for the Login method. During the execution of the login method, system will require username and password to authorize access for the user. If the username and password matched, the system will respond as success and the user is authenticated to access to the system, while if the username and password are not matched, system will respond with failed notification.

When the user is authenticated to the system access, he/she is able to update account information by using UpdateAcc method. Figure 4 shows the Z schema for the UpdateAcc method. This method requires user data to be existed in the database to proceed.

login

$\Delta User$
Username, Password? : *User* //*input*
R! : *Response* //*output*

If username $\cap$ *password* $\in$ *User*
R! = *Success*
else
R! = *Failed*

Figure 3. Z Schema for Login method for class User

UpdateAcc

$\Delta User$ //*update information using User*
Username, Password : *U*? *User* // *input*
R! : *Response* // *output*

If UserID $\in$ *User*
username! = *username*?
password!=*password*?
R! = *Success*
else
R! = *Failed*

Figure 4. Z Schema for UpdateAcc method for class User

Z Schema for Class Order

Class order consists of four methods which will be used by User to generate and invoke customer order. To access to this class, user must be authenticated by the system. By using this class, user is able to place new order, update customer order, generate invoice and calculate total order. Figure 5 shows the Z schema for the NewOrder method. It allows user to add new customer order. Each of new order process will require new order value to be inserted in each of attributes for class Order. The system will respond success when all attributes were successfully written in database. Figure 6 shows the Z schema for the UpdateOrder method. Update order requires attribute OrderID to be a set of Order before the overwriting process can be done. Based on the Z schema, if the OrderID is not a subset of Order, the system cannot update the record which means the OrderID is not a subset from the class.

Another method in this class is GenerateInvoice. Figure 7 shows the Z schema for this GenerateInvoice method. In order to generate invoice, user must identify OrderID to be used as the primary key to which record the system should generate. The system will not generate the invoice if the OrderID does not exist in the class Order. Figure 8 shows the CalculateTotal method. CalculateTotal is a method in class Order used to calculate all transactions details to be paid by the customer. CalculateTotal also required OrderID to process the calculation.

New Order

Δ *User // add new order using User*
OrderID, OrderDetails, CustName, Qtty, OrderDate, ProductID : *S*? *Order //input*
R! : *Response //output*

If S? $\notin$ *Order*
S! = *S*?
R! = *Success*
else
R! = *Failed*

Figure 5. Z Schema for NewOrder method for class Order

Update Order

Δ *Order//update order using Order*
OrderID, OrderDetails, CustName, Qtty, OrderDate, ProductID : *S*? *Order //input*
R! : *Response*

If OrderID? $\in$ *Order*
  S! = *S*?
  R! = *Success*
Else
  R! = *Failed*

Figure 6. Z Schema for UpdateOrder method for class Order

Generate Invoice

Δ *Order // generate invoice using Order*
OrderID? : *Order //input*
R! = *Response*

If OrderID $\in$ *Order*
  R! = *Success*
Else
  R! = *Failed*

Figure 7. Z Schema for GenerateInvoice method for class Order

Calculate Total

Δ *Order // calculate total using Order*
OrderID? : *Order //input*
R! : *Response*

If OrderID $\in$ *Order*
  Total:*Qtty* * *Price*
  R! = *Success*
Else
  R! = *Failed*

Figure 8. Z Schema for CalculateTotal method for class Order

Z Schema for Class Product

Class Product consists of details information of stock exists in the business transaction for the Inventory System. Each of the transactions occurs in the system will be transformed to Z schema. Class Product consists of two methods, which are UpdateStock and CheckStock. Figure 9 shows the Z Schema for the UpdateStock method. Update stock requires product attribute detail to be written into the existing record. Each of the products to be update must be a set of class Product. Before the execution, system will check whether the ProdID is the set of Product or not. If it is not, then the system will respond as Failed. Figure 10 shows the Z Schema for the CheckStock method. Check Sock is a method to find any stock and its details quantity in class Product. This method works by using ProdID as a subset of class Product. If the quantity of the stock is zero, system will notify that the stock is unavailable to be purchased.

Update Stock

Δ *Product //update stock using SProduct*
ProdID, *ProdName*, *StockQtty*, *Price* : *P*? *Product //input*
R! : *Response*

If ProdID$\in$*Product*
  P! = *P*?
  R! = *Success*
Else
  R! = *Failed*

Figure 9. Z Schema for Update Stock method for class Product

Check Stock

Ξ *Product* // *find stock in Product*
ProdID? : *Order* //*input*
R! = *Response* //*output*

If ProdID? $\in$ *Product*
  If Qtty > 0
    R! = *Success*
  Else
    R! = *Failed*
Else
R! = *Failed*

Figure 10. Z Schema for Check Stock method for class Product

CONCLUSION

This paper uses formal method to design unambiguous formal model using the Z language for the inventory system. Z specification is translated from the UML specification. This will help the developer to implement the system more easily since the Z specification can be adapted from UML specification. Formal method is the optimum technique that helps the reduction of errors, particularly at the earlier stages. Therefore, the system design and development can improve the quality effectively and reduce cost as well.

ACKNOWLEDGEMENT

This project is under Science Fund grant VoteS019. We would like to thanks Universiti Tun Hussein Onn Malaysia (UTHM) and MOSTI for this project.

REFERENCES

Alessandro Cimatti, Sergio Mover, Stefano Tenetta (2013). SMT-Based Scenario Verification for Hybrid Systems, *Formal Methods in System Design.* 01/2013, 42(1), 46-66.

Jonathan, J. (1996). *The Way of Z: Practical Programming with Formal Methods*. London: Cambridge University Press.

Fukagawa, M., Hikita, T. & Yamazaki, H. (1995). *A Mapping System from Object-Z to C++*.IEEE Japan.

Muhammad Bilal Latif, Wajahat Noshairwan, Moeen-ud-din Tariq, Zafar. I. Malik (2007). *The Use of Z Language for Multimedia Systems.* IEEE, Pakistan.

Spivey J. M. (1992). *The Z Notation: A Reference ManualSecond Edition*. Prentice Hall.

Yusufu Munina and Yusufu Gulina (2012). *Comparative Study of Formal Specifications through a Case Study*. IEEE, China.

YU Jun, HU Zhi-yi (2008). *Using Formal Methods to Design a Class SchedulingSystem*. IEEE, China.