Effective System analysis and design by using the Unified Modeling Language (U M L)

Daljeet Singh[1], Palak Mahajan[1], Vivek Kumar[1]
[1]*Guru Nanak Dev Engineering College, Ludhiana- 141006, Punjab, India*

Abstract—This paper elaborates all the design and analysis process of effective software development in the field of Computer Science and Engineering. There are three models under unified modeling Language in which all the diagrams of unified modeling Language are well explained. It is important to differentiate the UML model and the all set of diagrams of a system to be developed. A diagram is a partial graphic representation of a system's to be developed. The set of diagrams need not completely cover the model and deleting a diagram does not change importance of the model.

Keywords— *UML; Class Diagram; Genralisation; Relationship;Aggregation.*

I. INTRODUCTION

This UML stands for Unified Modeling Language which is used in object oriented software engineering. Although typically used in software engineering it is a rich language that can be used to model an application structures, behaviour and even business processes UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP). In this paper we discuss various diagrams of UML

II. CLASS MODELING

A. Class Diagras

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematics of the application, and for detailed modeling translating the models into programming code. Class diagrams are fundamental to the object modeling process and model the static structure of a system.

B. Objects

Objects are model elements that represent instances of a class or of classes. We can add objects to your model to represent concrete and prototypical instances. Concrete instances represent an actual person or thing in real world. For example, a concrete instance of a Customer class represents an actual customer. A prototypical instance of a Customer class contains data that represents a typical customer.

C. Relationship

In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between model elements.

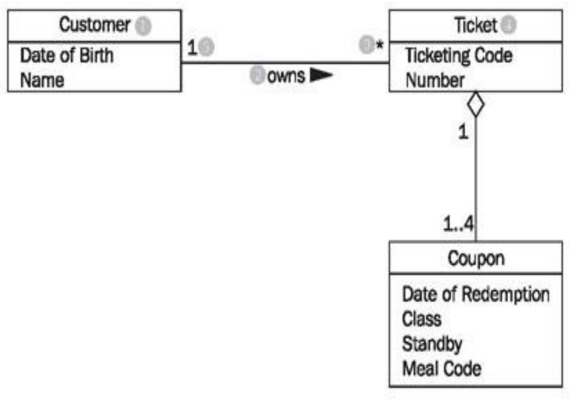


Fig.1: Example of Class diagram

Fig.1 shows a class diagram from our case study with the classes customer, ticket, and coupon, and their associations: Looking at the class diagram in Figure 1, you can read the association between the classes customer and ticket as follows:

- One (this sentence always begins with "one") object of the first class has an association with a number of objects of the second class.

The appropriate values from the diagram have to be inserted into this first abstract formulation, which can be universally applied. The name of one class is customer; the name of the other class is ticket. The name of the association is owns.Since associations usually are not directional, meaning usually go both directions, our association also has a meaning in the other direction:

D. Generalization

The Generalization relationship indicates that one of the two related classes (the subclass) is considered to be a specialized form of the other (the super type) and superclass is considered as 'Generalization' of subclass. An exemplary tree of generalizations of this form is found in biological classification: human beings are a subclass of simian, which are a subclass of mammal, and so on.

E. *Inheritnce*

Inheritance is a mechanism by which more specific classes (called subclasses or derived classes) incorporate structure and behavior of more general classes (called super classes or base classes).

F. *Association*

An association represents a family of links. Binary associations (with two ends) are normally represented as a line. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.

G. *Aggregation*

Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship.

H. *Constarints*

A constraint is a packable element which represents one conditions, restriction and assertion related to some element or several elements. For example, operation can have pre-condition and/or post-condition constraints. Constraint could have an optional name, though usually it is anonymous. A constraint is shown as a text string in curly braces according to the following syntax:
Constraint:= '{' [*name* ':'] *Boolean-expression* '}'

I. *Packages*

A package in the Unified Modeling Language is used "to group elements, and to provide a namespace for the grouped elements". A package may contain other packages, thus providing for a hierarchical organization of packages.

III. STATE MODELLING

A State chart diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

A. *Events*

The term event refers to the type of occurrence rather than to any concrete instance of that occurrence. For example, Keystroke is an event for the keyboard, but each press of a key is not an event but a concrete instance of the Keystroke event. The different types of events are:

- *Signal event:* It is the event of sending or receiving signal usually we are more concerned about the receipt of signal because of the causes effects in the receiving objects.
- *Change event*: Change event is an event is cost by the satisfaction of Boolean expression content intent of a change is event that the expression is continuously. Whenever the expression change from true event happen.
- *Time event*: Time event is an event cost by the occurrence of an absolute time of the time interval.

B. *States*

A state captures the relevant aspects of the system's history very efficiently. For example, when you strike a key on a keyboard, the character code generated will be either an uppercase or a lowercase character, depending on whether the Caps Lock is active. Therefore, the keyboard's behavior can be divided into two states: the "default" state and the "caps locked" state.

C. *Transitions and Conditions*

In UML modeling you can add transitions to a state machine diagram to show how an object changes state .A trigger, a guard condition and an effect are the three optional parts of a transition .Add a trigger to a transition to show that an event must occur for a transition to initiate. Add an effect to a transition to show that an object performs a particular activity when a guard condition is satisfied.

D. *State and Behaviour*

State diagram behavior can be understood by following description:

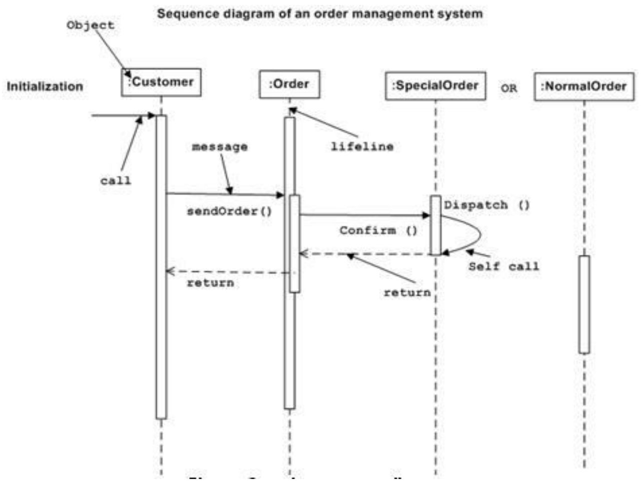
- *Activity effect:* An effect is a reference to a behavior that is executed in response to an event. For example, disconnected phone line might be an activity that is executed in response to an on Hook event.
- *Do-activities:* A do activity is an activity that continues for an extended time. By definition, a do activity can only occur within a state and cannot be attached to a transition. For example, the warning light may flash during the paper jam state for a copy machine.
- *Entry and Exit activity:* As an alternative to showing activities on transitions, one can build activities to entry or exit from a state.

E. *Concurrency*

The concurrent sections of the state diagram are places in which at any point, the given order is in two different states, one from each diagram. When the order leaves the concurrent states, it is in only a single state. We can see that an order starts off in both the Checking and Authorizing states. If the "check payment" activity of the Authorizing state completes successfully first, the order will be in the Checking and Authorized states. If the "cancel" event occurs, the order will be in only the Cancelled state.

IV. INTERACTION MODELLING

An interaction is a set of messages exchanged within collaboration. One uses interaction model to dynamic aspect of collaborations is presenting societies of object playing specific roles. Therefore every time there is collaboration between two objects there is an interaction.



A. Use Case Model

The use-cases will define the business services that the system must provide. Use-cases do not however define or describe how this will be achieved. The class diagram is the static view of the required system. The class diagram presents the building blocks (classes) and illustrates which classes interact but does not define how they interact. The interaction model defines the way in which services are provided by the objects within the system.

B. Sequence and Activities

The sequence diagram is having four objects (Customer, Order, Special Order and Normal Order).The following diagram has shown the message sequence for Special Order object and the same can be used in case of Normal Order object. Now it is important to understand the time sequence of message flows. The message flow is nothing but a method call of an object. The first call is send Order () which is a method of Order object. The next call is confirm () which is a method of Special Order object and the last call is Dispatch ( ) which is a method of Special Order object. So here the diagram is mainly describing the method calls from one object to another and this is also the actual scenario when the system is running.

V. REFERENCES

[1] Abello A., Samos J., & Saltor F., “A Multidimensional Conceptual Model Extending UML”, Journal of Information System, Vol.31Issue 6, PP. 541-567, 2006.

[2] Ghislain L., Valery B., “Estimating Software Size with UML Models”, In the Proceedings of Conference Montreal, Quebec,Canada, C3S2E; Vol. 290, 2008.

[3] RUMBAUGH, J. JACOBSON, I. and BOOCH, G. (1999): The Unified Modeling Language Reference Manual. Reading, Mass, Addison Wesley Longman Inc.

[4] Greedy B.,“Object-Oriented Analysis and Design with Application”, Second Edition. Addison-Wesley, 1994.

[5] Ņikiforova O., Sējāns J., Černičkins, “A Role of UML Class Diagram in Object-Oriented Software Development”, Applied computer systems. Vol.47, 2011, pp.65-74. ISSN 1407-7493

[6] Bansiya J., and Davis C.: “A Hierarchical Model for Object-Oriented Design Quality Assessment”, IEEE Transactions on Software Engineering, vol. 28, no. 1, pp. 4-17, 2002.

[7] E. Arisholm, L. Briand, S. Hove, and Y. Labiche. “The impact of uml documentation on software maintenance” An experimental evaluation. Software Engineering, IEEE Transactions on, 32(6):365–381, 2006.

[8] A. Fern´andez-S´aez, M. Genero, and M. Chaudron. “Does the level of detail of uml models affect the maintainability of source code?” In Proceedings of EESSMod 2011, 2011.

[9] M. Genero, J. Cruz-Lemus, D. Caivano, S. Abrah˜ao, E. Insfran, and J. Cars´ı. “Assessing the influence of stereotypes on the comprehension of uml sequence diagrams” A controlled experiment. Model Driven Engineering Languages and Systems, pp 280–294, 2008.

[10] Hoeben, F., “Using UML Models for Performance Calculation”, In Proceeding of WOSP 2000, p.p. 77-82, 2000

Singh Daljeet received his B-Tech degree in Computer Science and Engineering from Punjab Technical University, Jalandhar College, B.C.E.T, Ludhiana, Punjab, India, in 2008, the M-Tech, degree in Computer Science and Engineering from Punjab Technical University, Jalandhar, College, Guru Nanak Dev Engineering College, Ludhian, Punjab, India in year 2012. He is a Assistant Professor at present, with Department of Computer Science and Engineering, in Guru Nanak Dev engineering College. His research interests include Software Engineering, Software Metrics, UML, Object Oriented Paradigm, Object Oriented Metrics. At present, He is engaged in Research of UML diagrams simplifications.

Kumar Vivek, received his graduate degree as B.C.A and currently doing MCA from Guru Nanak Dev Engineering College, Ludhian, Punjab, India

Mahajan Palak, received his graduate degree as B.C.A and currently doing MCA from Guru Nanak Dev Engineering College, Ludhian, Punjab, India