A Validation Technique for UML Activity Model

Sudhir Kumar Singh
Deptt. Of Computer Science & Engg.,Bhagwant University
Ajmer,Rajsthan, India, 205004

Taskeen Zaidi
Deptt. Of Computer Science
B.B. Ambedkar University
Lucknow, (U.P.) India, 226025

Vipin Saxena
Deptt. Of Computer Science
B.B. Ambedkar University
Lucknow, (U.P.) India, 226025

ABSTRACT
In the current scenario of modeling, object-oriented modeling has completely replaced the structured modeling approach. Software industries are slowly-slowly shifting their old structured based softwares into the object-oriented based softwares, for e.g. Foxpro has been changed into the Visual Foxpro. From the literature, it is observed that various researchers are proposing the software models based on the object-oriented technology. It is a big challenge whether the proposed design is correct or reliable for a long time. For solution of this problem, the present work deals with a proposal of Unified Modeling Language (UML) model for a real case study of Mobile Bill Deposit System (MBDS) By the use of UML, class and activity models are designed for static and dynamic representation of the problem. For validation purpose the activity model is validated by the use of Finite State Machine (FSM) technique and results are presented in the form of test cases. When the size of the model becomes complex then presented technique shall help for validation of the complex model.

Keywords
Object-oriented modeling, UML, Class, Activity, Test Cases, Validation.

1. INTRODUCTION
UML is most popular multi-purpose platform independent language used to model for software and hardware architecture problems. One can develop the code in any object-oriented programming language from the UML diagrams. It supports static and dynamic representation of the problem. Let us first explain some of the important references related to the UML. Parade et al. [1] have described the UML diagrams and designed java codes which automatically generate structural and behavioral codes from class and sequence diagrams, respectively. Hu et al. [2] have explained the UML class diagram layout for important parts for software visualization and presented the graphical representation of any software model. Alanazi et al. [3] have presented basic diagrams for Unified Modeling Language which is used for object-oriented modeling. Salleh et al. [4] have generated a tool for analysing the Unified Modeling Language class diagram by the use of step by step design approach and clearly explained the relation between class and operation. Deng and liang [5] have described Unified Modeling Language for setting the answering system. Ali et al. [6] developed an assessment system for the Unified Modeling Language and by the use of developed system; one can make analysis of UML diagrams. Alsaadi [7] observed the data integrity for the database system by the use of UML. Lee et al. [8] have described an approach for classification of database through distributed approach. By the use of this approach a large database can be easily distributed on the different machines. Alhajj et al. [9] have described the object-oriented data model from the relational database model which converts the structured database into the object-oriented database. Fraser [10] has presented a approach for finding the length of the test cases generated from the software code. Welte [11] has presented state diagrams approach for the work of maintenance modeling and maintenance strategies. Kulakowski et al. [12] have also explained UML state diagrams which are used as a business rules for formulation and visual modeling. UML state diagrams generally minimize the complexity of the research problems. Alvarez et al.[13] have described finite state machines which can be used to model the temporal evolution of this type of phenomenon. Chaurasia and Saxena [14] have described UML is a unique and most popular modeling language and present time slowly-slowly Software Industries are changing by their old structured design models in the form of object oriented design models by the use of UML. Sexena and Kumar[15] have explained the UML is a powerful and graphical modeling language present time software developer are used in Object-Oriented database and designing, Modeling purpose used UML.

The present work deals with the design of UML class and activity diagram for showing the static and dynamic behavior of Mobile Bill Deposit System (MBDS). Activity diagram is converted into the finite state machine for designing the transition table. By the use of transition table, test cases are generated for the validation of the dynamic behavior of the problem. The entire procedure is demonstrated on the Mobile Bill Deposit System (MBDS). When the problem becomes more complex, then the presented technique shall be helpful for validation of complex model of research.

2. OBJECT-ORIENTED DATABASE
Object-oriented database is a better approach than the relational database. Object-Oriented database has major advantages like reducing the paging for better concurrency control. Object-oriented database stores the data in integer number, real number or string forms. The following code creates a table for the object-oriented database which is used in the UML class diagram:

create database database_name;

create table database_name.table_name;

create database Mobilebill;

create table Mobilebill.p1 (Cust_id INT NOT NULL, Mobile_no INT , Bill_amount INT);

insert into Mobilebill.p1 values(Cust_id, Mobile_no, Bill_amount)

values
(1001, 9452184193*,3500);*

insert into Mobilebill.p1 values(Cust_id, Mobile_no, Bill_amount)

values
(1002, 8004922219, 2300);

insert into Mobilebill.p1 values(Cust_id, Mobile_no, Bill_amount)

values
(1003, 9027374863, 1800);

On the basis of above code, an object-oriented database table is designed for the three major fields like Cust_id, Mobile_no and Bill_amount. The three fields are selected since these can be easily depicted in the form of data cubes and customer can easily trace the desired information when the object-oriented database becomes too complex. A sample of object-oriented database is shows in table1.

Table 1: An Object-Oriented Database for MBDS

Cust_id	Mobile_no	Bill_amount (In Indian Rupees)
1001	9452184193	3500
1002	8004922219	2300
1003	9027374863	1800
1004	9956098198	4500
1005	9795777512	5000
1006	9454272113	2345
1007	7499047254	7890
1008	9838449043	4567
1009	9839841701	4329
1010	8004922214	3789
1011	9452372550	4213
1012	9984111136	3123
1013	9958180278	3450
1014	9956123560	2341
1015	8739236012	4512

3. UML MODELING FOR MBDS

3.1 UML Class Diagram

UML class diagram is a type of static structure which represents attributes and operations. In the pictorial form, it is represented as a rectangle with three parts. First part shows the name of class, second part shows attributes while third part shows the operations. The object is an instance of class which follows object property. For the MBDS , a UML class model is designed for creation of database and a large size of database is converted into the different clusters and thereafter fuzzy clustering approach is used for finding the desired record. Let us first describe the UML class diagram. Customer is a class and Cust_id, Cust_add, and Cust_name are attributes. Exchange_Office is a class and Branch_id, Branch_name, Branch_add, and Branch_city are attributes. Mobile_Bill is a class and Bill_no, Bill_amount and Bill_date are attributes. Mobile is a class and Iemi_no Mobile_no, Model_no and Mobile_make are attributes. Customer makes enquiry from Exchange_Office and Customer's Mobile_no is registered with Exchange_Office who is responsible to generate the Mobile_Bill. Then amount of bills controlled by Bill_amount is sent on the registered Mobile_no of the customer. The different associations are also depicted in following figure 1.

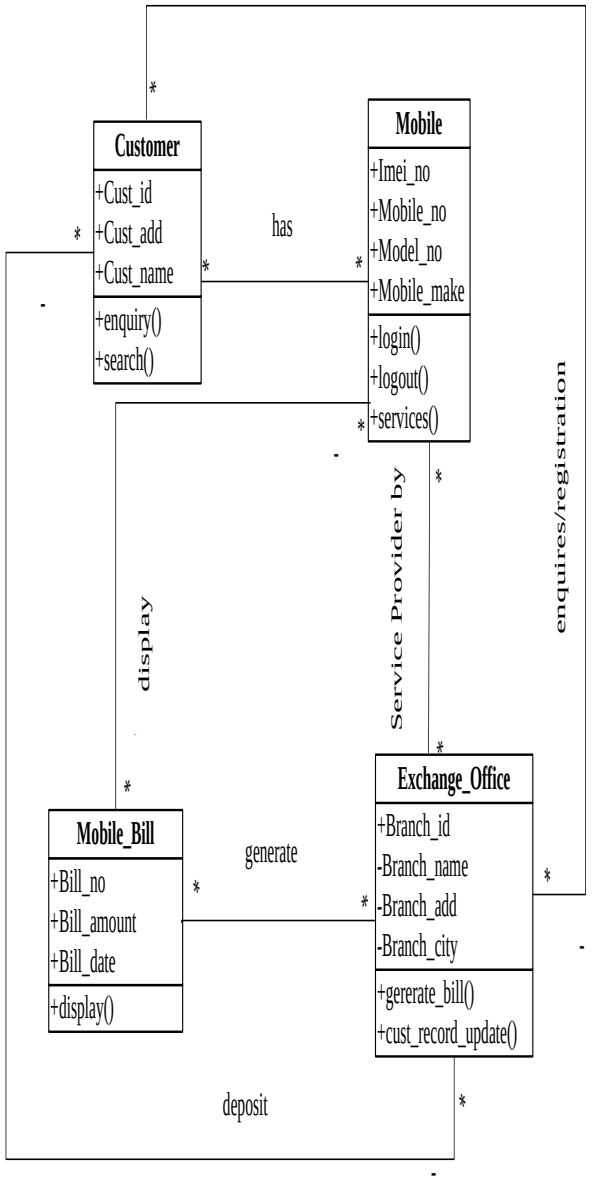

Fig 1: UML Class Diagram for MBDS

3.2 UML Activity Diagram

UML activity diagram is most important part of UML It is used to describe dynamic behavior of the system. Activity diagram basically works similar to flow chart. In the present work, activity diagram is prepared for identification of some of the important elements. In the problem, customer has the mobile number and this activity is represented as ***"Customer has Mobile No"*** which is connected to the next activity like ***"Exchange Office Search Requested Bills"*** by an event "***Search for Mobile Bills***". In this manner different activities are joined together with events and represented in the following table 2. The activity ***"Customer Deposit Bills"*** shows that the customer has deposited the bill and a receipt is received by the customer. It is done after searching the record by the Mobile Bill Deposit System (MBDS). The complete activity diagram is represented in the following figure 2.

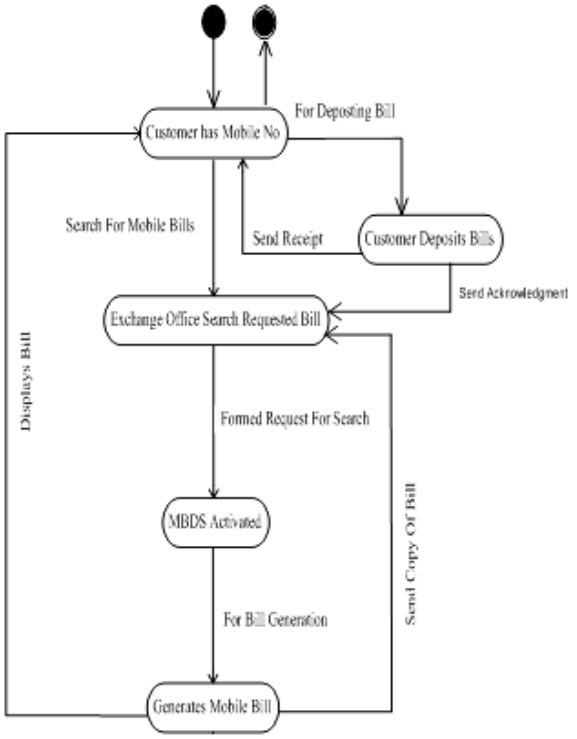

Fig 2: UML Activity Diagram for MBDS

4. FINITE STATE MACHINE

Finite State Machine (FSM) has a limited or finite number of possible states. A finite state machine can be used both as a development tool for approaching and solving problems and as a formal way of describing the solution so there later developers and system maintainers**.** Finite State Machine the number of possible states and the number of inputs both are finite and the change of the state is totally governed by the inputs. The following symbols are used in the FSM. The different eight events are shown below in the table 2.

$\sum$ is the input symbol.

q_0 is the initial state.

q_f is the final state

s is the transaction function.

Table 2: Representation of Events from Activity Diagram

a	Search for Mobile Bills
b	Formed Request For Search
c	For Bill Generation
d	Send Acknowledgment
e	Send Receipt
f	lliB gnitsopeD roF
g	lliB syalpsiD
h	lliB fO ypoC dneS

On the basis of above definition, the different states are taken from UML activity diagram and represented in the following table 3.

Table 3: Representation of State from Activity Diagram

Symbol	Name of State
q_0	Customer has Mobile No
q_1	Exchange Office Search Requested Bill
q_2	MBDS Activated
q_3	Generates Mobile Bill
q_4	Customer Deposits Bill

Table 4: A Transition Table

δ/∑	a	b	c	d	e	f	g	h
q_0	q_1	Ø	Ø	Ø	Ø	q_4	Ø	Ø
q_1	Ø	q_2	Ø	Ø	Ø	Ø	Ø	Ø
q_2	Ø	Ø	q_3	Ø	Ø	Ø	Ø	Ø
q_3	Ø	Ø	Ø	Ø	Ø	Ø	q_0	q_1
q_4	Ø	Ø	Ø	q_1	q_0	q_0	Ø	Ø

When events are operated on the different activities then different transitions take place. From the states and activities a transition table is explained above after generating the grammar given below from the finite state machine. A finite state machine from the activity diagram is shown below in figure 3.

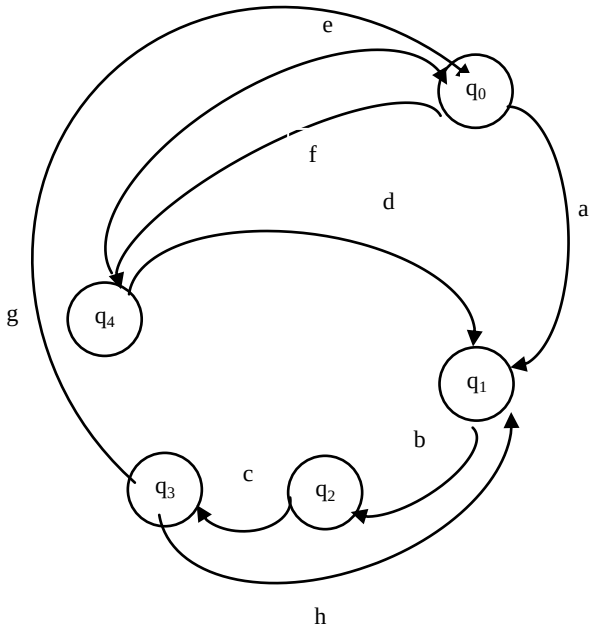

Fig. 3: Representation of FSM from Activity Diagram

5. TEST CASES GENERATION

Test cases are used to check the program or model and they are finite and selected either from the model or from the problem. It check the validity of the proposed models. In the present work test cases are generated from the proposed UML activity model which has been converted into FSM. From figure 3 the relationships between activities and events are represented in the form of following equivalent grammar:

$\delta(q_0,a)= q_1 \Rightarrow q_0 \rightarrow aq_1$
$\delta(q_1,b) = q_2 \Rightarrow q_1 \rightarrow bq_2$
$\delta(q_2,c)= q_3 \Rightarrow q_2 \rightarrow cq_3$
$\delta(q_3,h)= q_1 \Rightarrow q_3 \rightarrow hq_1$
$\delta(q_4,d) = q_1 \Rightarrow q_4 \rightarrow dq_1$
$\delta(q_4,e)= q_0 \Rightarrow q_4 \rightarrow eq_0$
$\delta(q_4,f)=q_0 \Rightarrow q_0 \rightarrow fq_4$
$\delta(q_3,g) = q_0 \Rightarrow q_3 \rightarrow gq_0$

Test Case 1: Customer gets a copy of bill

The equivalent grammar from the above production rule is

$q_0 \rightarrow aq_1$
$q_1 \rightarrow bq_2$
$q_2 \rightarrow cq_3$
$q_3 \rightarrow gq_0$
$q_0 \rightarrow Ø$

By replacing the non terminals on RHS of above production rules ,we can get

$q_0 \rightarrow abcg$

which represents from the activity diagram that customer gets a copy of bill.

Test Case 2: Customer gets a copy of receipt

The equivalent grammar from the above production rule is

$q_0 \rightarrow aq_1$
$q_1 \rightarrow bq_2$
$q_2 \rightarrow cq_3$
$q_3 \rightarrow gq_0$
$q_0 \rightarrow fq_4$
$q_4 \rightarrow eq_0$
$q_0 \rightarrow Ø$

By replacing non terminals on RHS of production rules, we can get

$q_0 \rightarrow abcgfe$

This represents from activity diagram that after depositing bill, customer gets receipt.

6. CONCLUSIONS

From the above work, it is concluded that any software and hardware research problem can be modelled in the pictorial form and for this purpose, software developers are using the platform independent Unified Modeling Language as many of the softwares have been converted into the object oriented style. The coders can easily convert the models by using any object-oreinted programming language but before moving towards development of code, validation of model is a necessary task. In the present work, a FSM techniques is used for validation of the proposed model and observed that the proposed model is an efficient which will further save the cost of the development. The proposed technique can be applied for other software and hardware problems and even real life problems can also be easily solved.

7. REFERENCES

[1] Parada, G. A., Siegert, F., Brisolara, L., "Generating Java Code from UML Class and Sequence Diagrams", Presented in Brazilian Symposium on Computing System Engineering, Page 99-101 Date 7 Nov 2011.

[2] Hu , H., Fang, J., Lu, Z. ,Zhao, F., Qin, Z., " Rank-Directed Layout of UML Class Diagrams", In Proceedings of the First International Workshop on Software Mining, Pages 25-31,Date 7 Nov 2011.

[3] Alanazi , N, M., "Basic Rules to Build Correct UML Diagrams", Presented in International Conference on New Trends in Information and Service Science , June 30-July 02,

[4] Salleh, M. F., Ibrahim, N., Ling, Y.L., " Design of Tool or Generating UML Analysis Class Diagram", Presented in International Conferences on Computational Intelligence for Modelling, Control and Automation, Intelligent Agents, Web Technologies and Internet Commerce and Innovation in Software Engineering, Date 10-12 Dec,2008 .

[5] Deng, W. and Liang, Y., "Reason on UML Diagrams with Answer Set Programming", In Proceeding of International Conference on Computer Science and Software Engineering 2008, Date 12-14 Dec, 2008.

[6] Ali, H. N., Shukur, Z. and Idris, S., "A Design of an Assessment System for UML Class Diagram", In Proceeding of International Conference on Computational Science and its Applications (ICCSA 2007), Page 539-546 ,Date 26-29 Aug 2007,

[7] Alsaadi, A., "Checking Data Integrity via the UML Class Diagram", Presented in International Conference on Software Engineering Advances (ICSEA'06), Page 37, Nov 2006.

[8] Lee, M.S., Yan Ha, Y., Park, S. H., "Allocation of Classes in Distributed Object-Oriented Databases", Presented in 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, Page 237-242, Date 27-29 2009.

[9] Alhajj, R., and Polat, F., "Reengineering Relational Databases to Object-Oriented: Constructing the Class Hierarchy and Migrating the Data", Presenting in Eight Working Conference on Reverse Engineering (WCRE 2001),Date 2-5Oct 2001. ,

[10] Fraser, G. , "Experiments on the Test Case Length In Specification Based Test Case Generation", Published in: Automation of Software Test, 2009. AST '09. ICSE Workshop on, page 18-26, Date 18-19 May 2009.

[11] Welte, T.M. ,"Using State Diagrams for Modeling Maintenance of Deteriorating Systems", Published in: Power Systems, IEEE Transactions on (Volume:24, Issue: 1),Page 58-66,Date 9 Dec 2008.

[12] Kulakowski, K. , and Nalepa, G. J. ," Using UML State Diagrams for Visual Modeling of Business Rules", Published in Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference, Page 189-194, Date 20-22 Oct 2008.

[13] Alvarez, A, A., Trivino, G. , Cordon, O., "Human Gait Modeling Using a Genetic Fuzzy Finite State Machine" , Published in Fuzzy Systems, IEEE Transactions on (Volume:20 , Issue: 2) Biometrics Compendium, IEEE, Page 205 – 223, Date of Publication 19 October 2011.

[14] Chaurasia, P., K and Saxena, V., "Mobile Based Electricity Bill Deposit System through UML", Published in Journal of Software Engineering and Applications, Volume 4, Page 187-190, Date 12 March 2011. doi:10.4236/jsea.2011.43021.

[15] Saxena, V. and Kumar , S., "Object-Oriented Database Connectivity for Hand Held Devices", Published in Journal of Software Engineering and Applications, Volume 5, Page 314-320, Date 5 May 2011.