Testing UML Designs using Class, Sequence and Activity Diagrams

Bhavnesh Kumar
M. Tech Student
Department of Computer Science & Engineering
U.I.E.T, Kurukshetra University Kurukshetra, Haryana, India

Kulvinder Singh
Faculty
Department of Computer Science & Engineering
U.I.E.T, Kurukshetra University Kurukshetra, Haryana, India

Abstract

Testing is performed by different types of strategies. Generally testing is performed on code, but if the software can be tested in the earlier phases then most of the errors can be eliminated and can be stopped from propagating to next phase. The proposed work presents a novel design based testing approach that can fix errors in initial phase. To perform design based testing, we need a language that can deal with the design ably i.e. Unified Modeling Language (UML). UML, which supports object-oriented technology, is widely used to describe the analysis and design specifications of software development. UML models are an important source of information for test case design. UML activity diagrams describe the realization of the operation in design phase and also support description of parallel activities and synchronization aspects involved in different activities perfectly. However UML Sequence diagram describes the way in which different objects interacts with each other, sequence of message passing between different objects. And Class diagram identifies the different classes, its attributes and operations respectively. We propose a method to generate test cases using UML activity diagram. Next, we propose a novel approach to generate test cases from test scenarios using UML activity, sequence and class diagram. First we generate xml from UML diagram. UML consists of different designs that are used to specify the static and dynamic behavior of the software. Proposed system focuses on three diagrams viz. class, sequence and activity diagrams. The class diagrams help us to generate static test cases and the sequence and activity diagrams give dynamic test cases and integrity test cases.

Keywords: UML, Class Diagrams, Activity Diagrams, Sequence Diagrams

I. Introduction

On behalf of the waterfall software development life cycle (SDLC) model; a software life cycle is consist of five phases which are (I) requirements (II) design (III) implementation (IV) software testing (V) maintenance. Execution of a program is done in software testing phase and the system works with an intention of finding the errors. With the increase in the complexity and size of the software system; there will be a requirement of more and more time and manpower for testing. Testing process is done in three parts: test case generation, test execution and test evaluation. Test case generation is the most difficult in all the three parts. Now we will discuss about UML testing.

A. Unified Modeling Language

UML is a standard visual modeling language which is designed for specifying, visualizing, constructing and documenting of the artifacts of software systems. A number of diagrams for describing particular aspects of software artifacts are provided by UML. These diagrams are classified on the behalf whether they are used for describing structural or behavior aspects of the systems. The diagrams under consideration for this particular task are:

B. Class Diagram

The class diagram represents the static view of an object oriented application. It describes the attributes and operations of a class and also the constraints imposed on these members. It is not only used for visualizing, describing and documenting different aspects of a system but it also can be convert to executable code of the software application.

The class diagrams are most important component used in the modelling of object oriented systems because they are the only UML diagrams which can be directly associated with object oriented languages and hence help in automatic code generation.

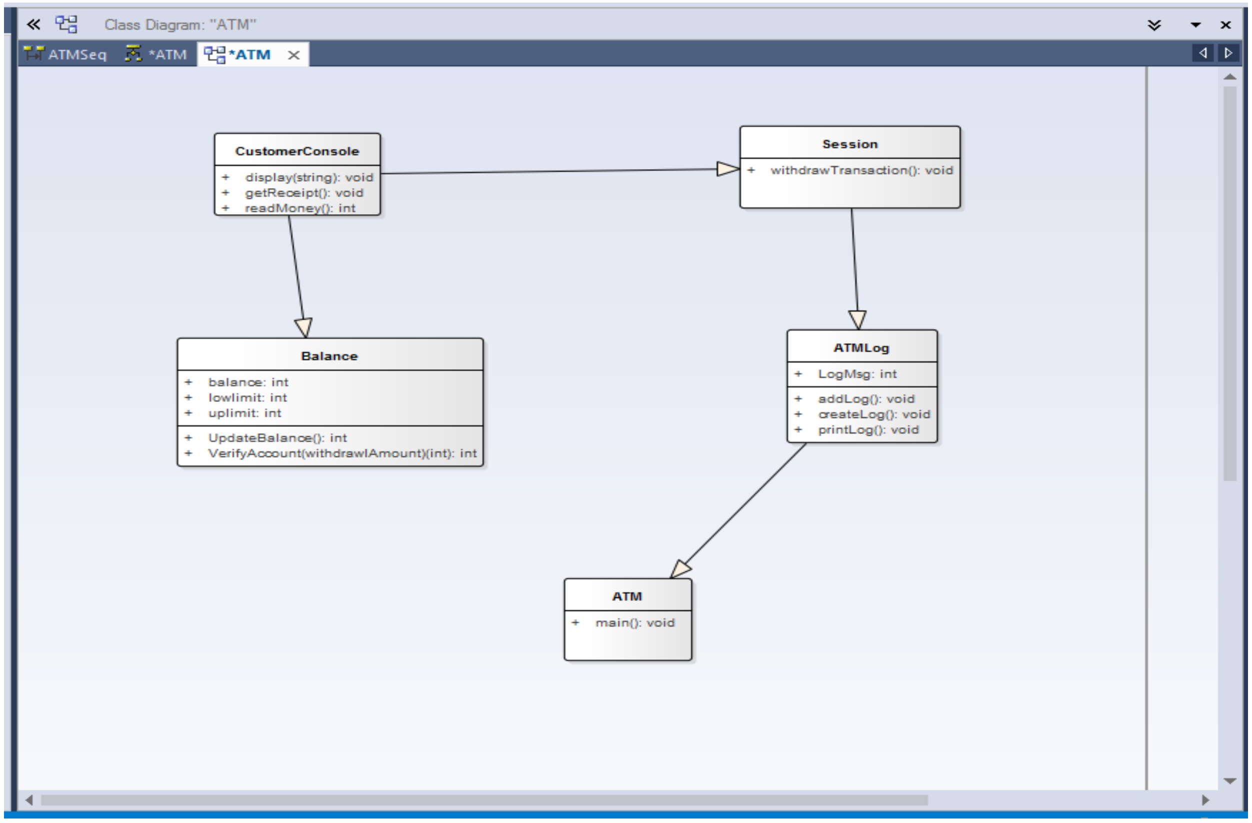

Fig. 1: Class Diagram for Banking system

C. Exported XML

```
<UML:Class name="ATMLog" xmi.id="EAID_4D98C62C_A194_425f_B661_C33E9B676805" visibility="public"
namespace="EAPK_2183C10C_E8BA_4379_BE33_26D779DAFD38" isRoot="false" isLeaf="false" isAbstract="false"
isActive="false">
<UML:Classifier.feature>
   <UML:Attribute name="LogMsg" changeable="none" visibility="public" ownerScope="instance" targetScope="instance">
   <UML:Attribute.initialValue>
   <UML:Expression/>
   </UML:Attribute.initialValue>
   <UML:StructuralFeature.type>
   <UML:Classifier xmi.idref="eaxmiid1"/>
   </UML:StructuralFeature.type>
   <UML:ModelElement.taggedValue>
            <UML:TaggedValue tag="type" value="int"/>
   <UML:TaggedValue tag="containment" value="Not Specified"/>
   <UML:TaggedValue tag="ordered" value="0"/>
   <UML:TaggedValue tag="collection" value="false"/>
   <UML:TaggedValue tag="position" value="0"/>
   <UML:TaggedValue tag="lowerBound" value="1"/>
   <UML:TaggedValue tag="upperBound" value="1"/>
   <UML:TaggedValue tag="duplicates" value="0"/>
   <UML:TaggedValue tag="ea_guid" value="{331E56D5-EE9A-448a-AA8B-360E46381E2D}"/>
   <UML:TaggedValue tag="ea_localid" value="4"/>
   <UML:TaggedValue tag="styleex" value="volatile=0;"/>
           </UML:ModelElement.taggedValue>
   </UML:Attribute>
   <UML:Operation name="addLog" visibility="public" ownerScope="instance" isQuery="false"    concurrency="sequential">
   <UML:ModelElement.taggedValue>
```

D. Sequence Diagram

Sequence diagram is one of the common interaction diagrams used in UML, which focuses on the message interchange between a number of object lifelines. Sequence diagram describes an interaction by describing the sequence of messages that are exchanged, and also explains their corresponding occurrence specifications on the lifelines.

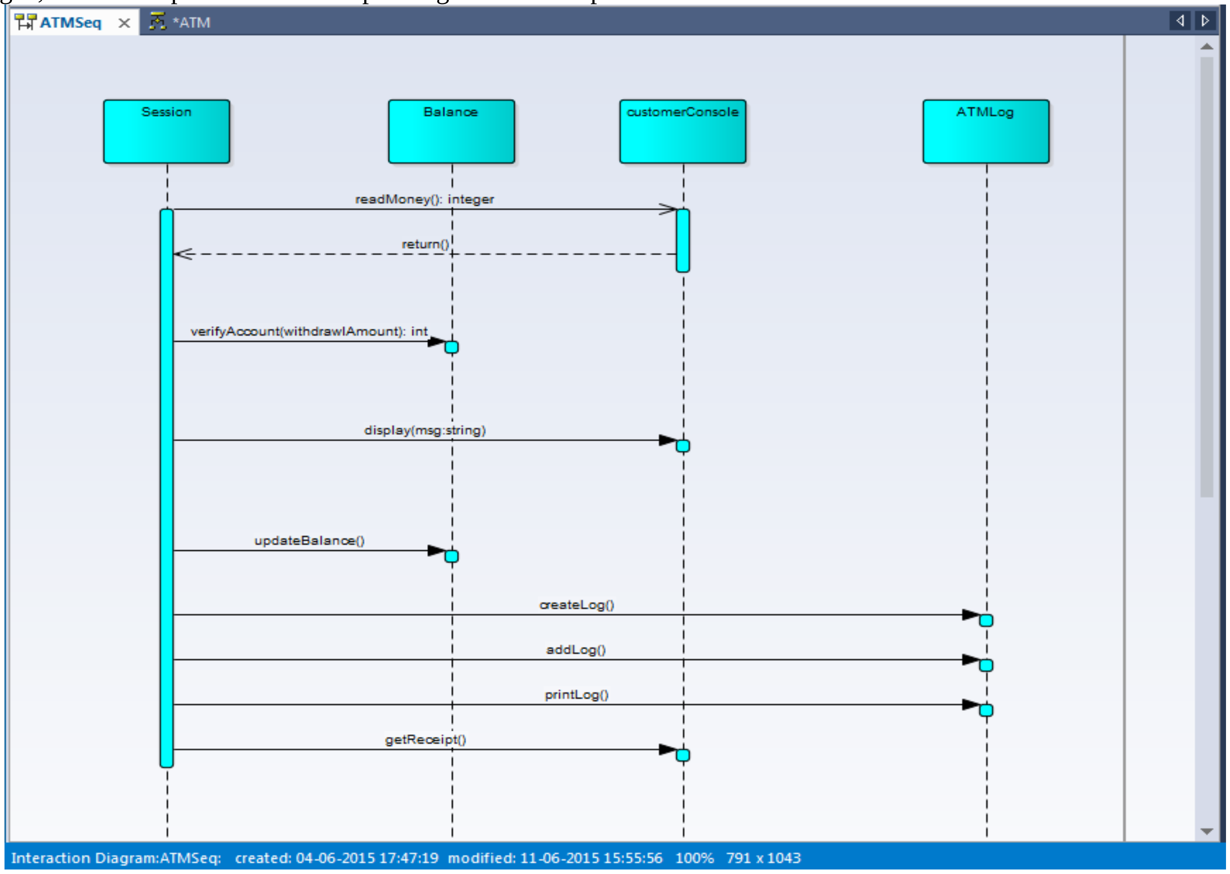

Fig. 2: A Sequence Diagram Showing Various Notations [10]

E. Exported XML

```
<UML:Collaboration xmi.id="EAID_2183C10C_E8BA_4379_BE33_26D779DAFD38_Collaboration"
name="Collaborations">
   <UML:Namespace.ownedElement>
   <UML:ClassifierRole name="ATMLog" xmi.id="EAID_A81BB782_EB8D_424a_93EE_CBB638883F80"
visibility="public" base="EAID_11111111_5487_4080_A7F4_41526CB0AA00">
<UML:ClassifierRole name="Session" xmi.id="EAID_B00D787C_8974_48dc_BCFC_9A9E76B2FD87" visibility="public"
base="EAID_11111111_5487_4080_A7F4_41526CB0AA00">
<UML:Collaboration.interaction>
<UML:Interaction xmi.id="EAID_2183C10C_E8BA_4379_BE33_26D779DAFD38_INT"
name="EAID_2183C10C_E8BA_4379_BE33_26D779DAFD38_INT">
<UML:Interaction.message>
<UML:Message name="printLogg" xmi.id="EAID_4CB0994A_BFDE_4dc3_8F6A_C080BA9F9913" visibility="public"
sender="EAID_B00D787C_8974_48dc_BCFC_9A9E76B2FD87"
receiver="EAID_A81BB782_EB8D_424a_93EE_CBB638883F80">
   <UML:ModelElement.taggedValue>
   <UML:TaggedValue tag="style" value="1"/>
   <UML:TaggedValue tag="ea_type" value="Sequence"/>
   <UML:TaggedValue tag="direction" value="Source -&gt; Destination"/>

   <UML:TaggedValue tag="seqno" value="8"/>
   <UML:TaggedValue tag="ea_sourceName" value="Session"/>
   <UML:TaggedValue tag="ea_targetName" value="ATMLog"/>
   <UML:TaggedValue tag="ea_sourceType" value="Sequence"/>
   <UML:TaggedValue tag="ea_targetType" value="Sequence"/>
```

```
    <UML:TaggedValue tag="ea_sourceID" value="16"/>
    <UML:TaggedValue tag="ea_targetID" value="32"/>
    <UML:TaggedValue tag="src_visibility" value="Public"/>
    <UML:TaggedValue tag="src_isOrdered" value="false"/>
    <UML:TaggedValue tag="src_targetScope" value="instance"/>
    <UML:TaggedValue tag="dst_style" value="Union=0;Derived=0;AllowDuplicates=0;Owned=0;Navigable=Navigable;"/>
    <UML:TaggedValue                                                          tag="privatedata5"
value="SX=0;SY=0;EX=0;EY=0;$LLB=;LLT=;LMT=CX=45:CY=13:OX=0:OY=0:HDN=0:BLD=0:ITA=0:UND=0:CLR=-
1:ALN=1:DIR=0:ROT=0;LMB=;LRT=;LRB=;IRHS=;ILHS=;"/>
    <UML:TaggedValue tag="sequence_points" value="PtStartX=117;PtStartY=-521;PtEndX=688;PtEndY=-521;"/>
    <UML:TaggedValue tag="virtualInheritance" value="0"/>
    <UML:TaggedValue tag="diagram" value="EAID_C8BC141D_A238_476f_8963_3E2CD9529D70"/>
    <UML:TaggedValue tag="mt" value="printLog()"/>
    </UML:ModelElement.taggedValue>
    </UML:Message>
    </UML:Interaction.message>
    </UML:Interaction>
    </UML:Collaboration.interaction>
</UML:Collaboration>
```

F. Activity Diagram

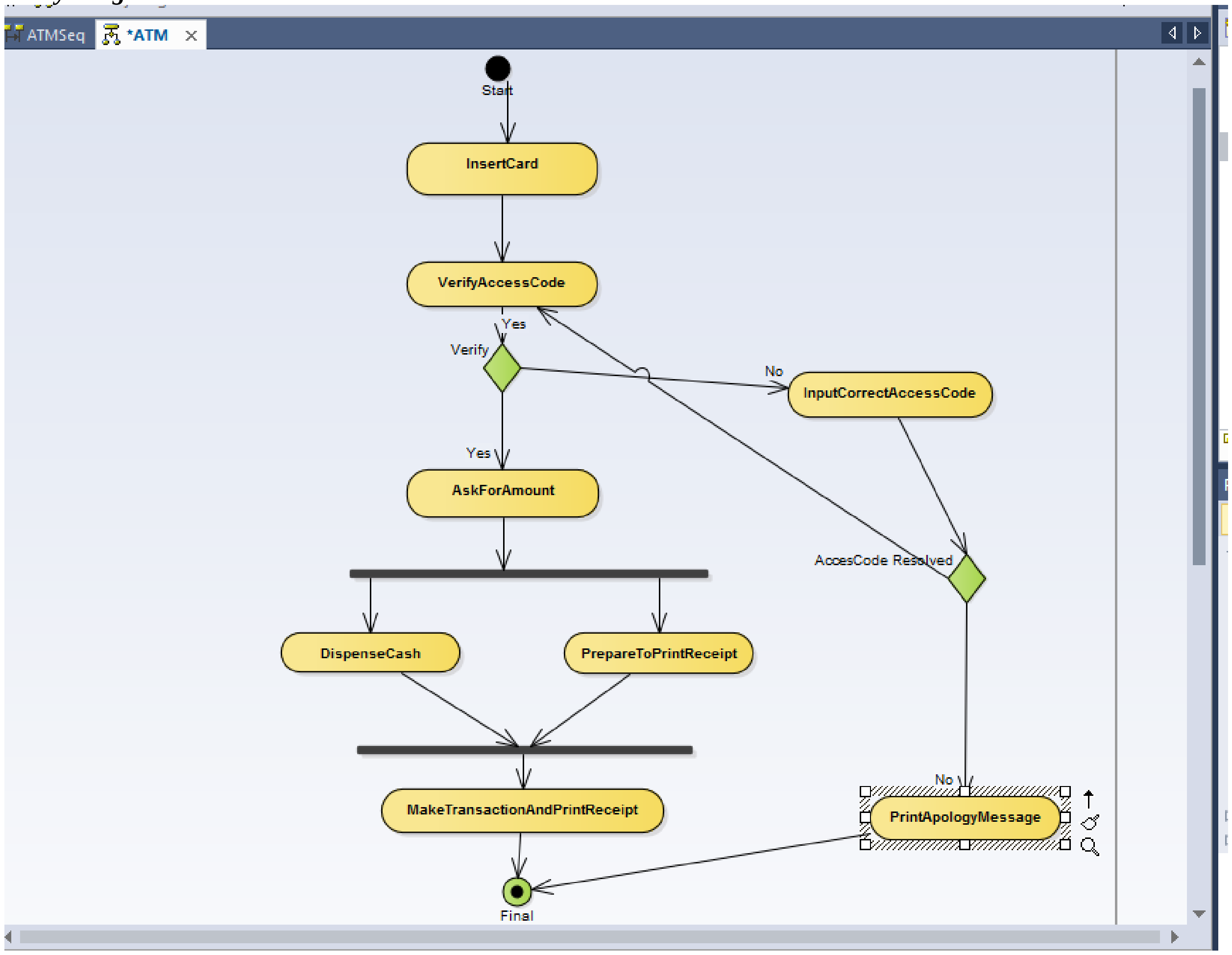

Fig. 3: Activity diagram for Banking system

The UML activity diagrams describe the sequential or concurrent control flow of activities. They are used for modeling the dynamic aspects of an object oriented system. The UML activity diagrams are also used as the models for the driven test case generation.

UML activity diagram portraits all the important steps carried out during the execution of a single use case. Test cases generated from the UML activity diagrams do not represent any information of the system. They only depict the dynamic behavior of system.

G. Exported XML

```
<UML:ActivityModel xmi.id="EAID_2183C10C_E8BA_4379_BE33_26D779DAFD38_ActivityModel"
context="EAPK_2183C10C_E8BA_4379_BE33_26D779DAFD38" name="ActivityModel" visibility="public">
<UML:StateMachine.transitions>
<UML:Transition xmi.id="EAID_3A301AB6_0209_414f_95EE_C0487D3EA00E" visibility="public"
source="EAID_0D379B52_E56B_484e_8508_DBE1AEE001B5"
target="EAID_9E752270_2309_46c9_8DCA_70B819E2D521">
  <UML:ModelElement.taggedValue>
    <UML:TaggedValue tag="style" value="3"/>
    <UML:TaggedValue tag="ea_type" value="ControlFlow"/>
    <UML:TaggedValue tag="direction" value="Source -&gt; Destination"/>
    <UML:TaggedValue tag="linemode" value="3"/>
    <UML:TaggedValue tag="linecolor" value="-1"/>
    <UML:TaggedValue tag="linewidth" value="0"/>
    <UML:TaggedValue tag="seqno" value="5_1"/>
    <UML:TaggedValue tag="headStyle" value="0"/>
    <UML:TaggedValue tag="lineStyle" value="0"/>
    <UML:TaggedValue tag="ea_localid" value="68"/>                    <UML:TaggedValue tag="ea_sourceName"
value="AskForAmount"/>
    <UML:TaggedValue tag="ea_sourceType" value="Activity"/>
    <UML:TaggedValue tag="ea_targetType" value="Synchronization"/>
    <UML:TaggedValue tag="ea_sourceID" value="37"/>
    <UML:TaggedValue tag="ea_targetID" value="45"/>
    <UML:TaggedValue tag="src_visibility" value="Public"/>
    <UML:TaggedValue tag="src_aggregation" value="0"/>
    <UML:TaggedValue tag="dst_style" value="Union=0;Derived=0;AllowDuplicates=0;"/>
    <UML:TaggedValue tag="virtualInheritance" value="0"/>
    </UML:ModelElement.taggedValue>
    </UML:Transition>
```

II. RELATED WORK

In 2013 Parampreet Kaur *et al.* [35] stated several heuristic means to measure the quality of test suites, e.g. fault detection, mutation analysis, or coverage criteria. These means of quality measurement could also be used to decide when to stop testing. This paper was centered upon coverage criteria. There are many different kinds of coverage criteria, e.g. focused on data flow, control flow, transition sequences, or boundary values. The authors presented new approaches, e.g. to combine coverage criteria and generation of test paths manually as well as automatically using tools based on Chinese postman and prefix based algorithms. In 2014 Amitashree *et al.* [41] focused an approach to generate test-cases from UML sequence diagram for detecting deadlocks during the design phase. This reduced the effort and cost involved to fix deadlocks at a later stage. This work began with design of sequence diagram for the system, then converting it to intermediate graph where deadlock points were marked and then traversed to get test-cases. The test-cases thus generated were suitable for detecting deadlocks. The proposed work mainly consisted of 3 steps model UML sequence diagram, representing the sequence diagram graph and traversing it to generate test-cases that would be able to detect deadlocks. The used example was a generalized model that showed the test could exercise all paths and had the ability to detect deadlocks. S. Shanmuga Priya *et al.* [3] proposed idea about test path generation using UML sequence diagram. In this authors told that Software testing played a major role in deciding the delivery of the product as well as in the quality of the product. Testing should be performed in the initial phases in SDLC so that most of the errors could be eliminated. That's why; testing shouldn't be isolated to a single phase in SDLC. Test case generation was the challenging part in software testing process. The proposed work presented a model based testing approach from which the test paths were automated. Unified Modeling Language (UML) Sequence Diagram is considered for designing and a case study of Medical Consultation System is taken for the proposed work by the authors.

Saru Dhir [5] described an idea about the impact of UML techniques in test case generation. He told in his paper that UML is a standard language which is used in business modeling for specifying, visualizing and constructing the software artifacts. Different UML strategies and techniques were implemented in the past. This paper explained the UML2.0 testing for test case generation. In this paper, he focused on effective use of UML techniques and test-case generation methods.

Maicon B. da Silveira *et al.* [6] proposed an idea about generation of scripts for performance testing based on UML Models. The cost of software testing process is very high in respect of the other stages of SDLC. MBT is a technique used for automatic generation of testing artifacts. In this paper authors described a case study which shows the procedure how to implement the MBT process. Results generated automatically by a Software Product Line (SPL) in our method.

III. PROBLEM FORMULATED

All major testing techniques perform testing once the coding has been done. But the most important part of SDLC is software design because if the design is not perfect, then the coder would never get the correct platform to write his code on. So, there must be a technique which is able to test the design of a object oriented software. UML being the most widely used standard for software design, we here present a technique to test the design created. The class diagrams only tell about the contents and associations of a class. So they can be used to generate static test cases. But sequence and activity diagrams give us the dynamic behavior of class, so they give us the dynamic test cases. And to check whether sequence and activity diagrams are as per class diagram specifications, we create the integrity test cases.

IV. PROPOSED METHODOLOGY

The software design created in UML can be exported to formats like mdl, csv, xls or xml formats. Once exported, we can use language like java, matlab or C# to study the details programmatically and then generate test cases. In proposed method, we convert the UML design into XML format and then generate the static, dynamic and integrity test cases. This methodology can be further used to check the compatibility of one particular module design with a previous existing module. The steps to perform the implementation can be summarized as follows:

- Create a modeling project in Enterprise Architect 12.
- Create class, sequence and activity diagrams for the model.
- Export this UML design to XML format.
- Create a C# project.
- Include the XML created in this project.
- Parse the XML code using C# language coding to explore the different diagrams and the associations among the diagrams.
- Generate static test cases from class diagrams by analyzing their associations.
- Parse the sequence and activity diagrams and generate the dynamic test cases.
- Alternatively include the xml for sequence and activity diagram for another module and check its compatibility with previously existing design. This is integrity test case generation.

Tools to be used:

- Visual studio 2012.
- Enterprise Architect 12.

A. Types of test cases

- Static test cases:- These Test Cases Test the UML Design class and sequence diagram for their compatibility test .its check if all life lines exists in sequence diagram exist as classes in class Diagram or not.
- Integrity Test Cases for class and sequence Diagram:-These testing test the classes and Sequence Diagram in the UML Design for their Dynamic behavior i.e. this test checks if all messages call destined towards of life line exist in corresponding class in form of function or not if the messages feel to exist in destination class then the integrity test fails so the sequence diagram is not valid.
- Deadlock Testing:- there is another test for dynamic Behavior of a UML Sequence Diagram wherein we check for existence of any circular reference exists then there is always a probability of a Dead lock happening.
- Test Path Generation from Activity Diagram:- This Particular test cases checks the Activity Diagram for its Function Sequence and Create all Possible test Cases from the activity Diagram their test cases verifies the logic and Structure of Activity Diagram from the UML Designer.
- Summary:- In OOP the Direction of Development of entire Project is Decided by the UML Design any shortcoming in the design can lead to total or particular failure of code and because loss of time, money and resources . The above maintain test cases enable us to perform to complete checks of their major UML Diagram that is class, Sequence and Activity.

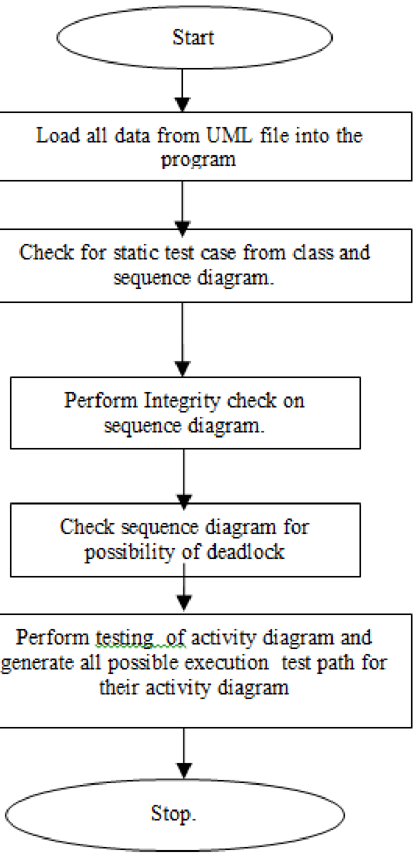
Start
Load all data from UML file into the program
Check for static test case from class and sequence diagram.
Perform Integrity check on sequence diagram.
Check sequence diagram for possibility of deadlock
Perform testing of activity diagram and generate all possible execution test path for their activity diagram
Stop.

Fig. 3: Flowchart for main menu

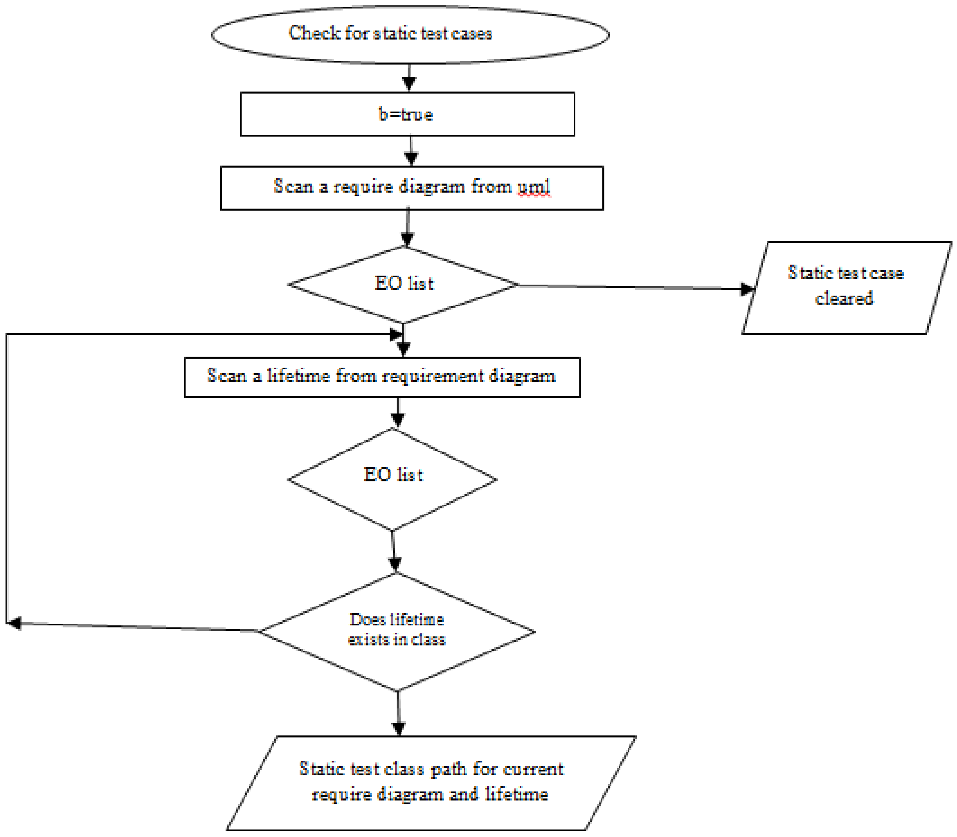
Check for static test cases
b=true
Scan a require diagram from uml
EO list
Static test case cleared
Scan a lifetime from requirement diagram
EO list
Does lifetime exists in class
Static test class path for current require diagram and lifetime

Fig. 4: Static test case generation for sequence and class diagram

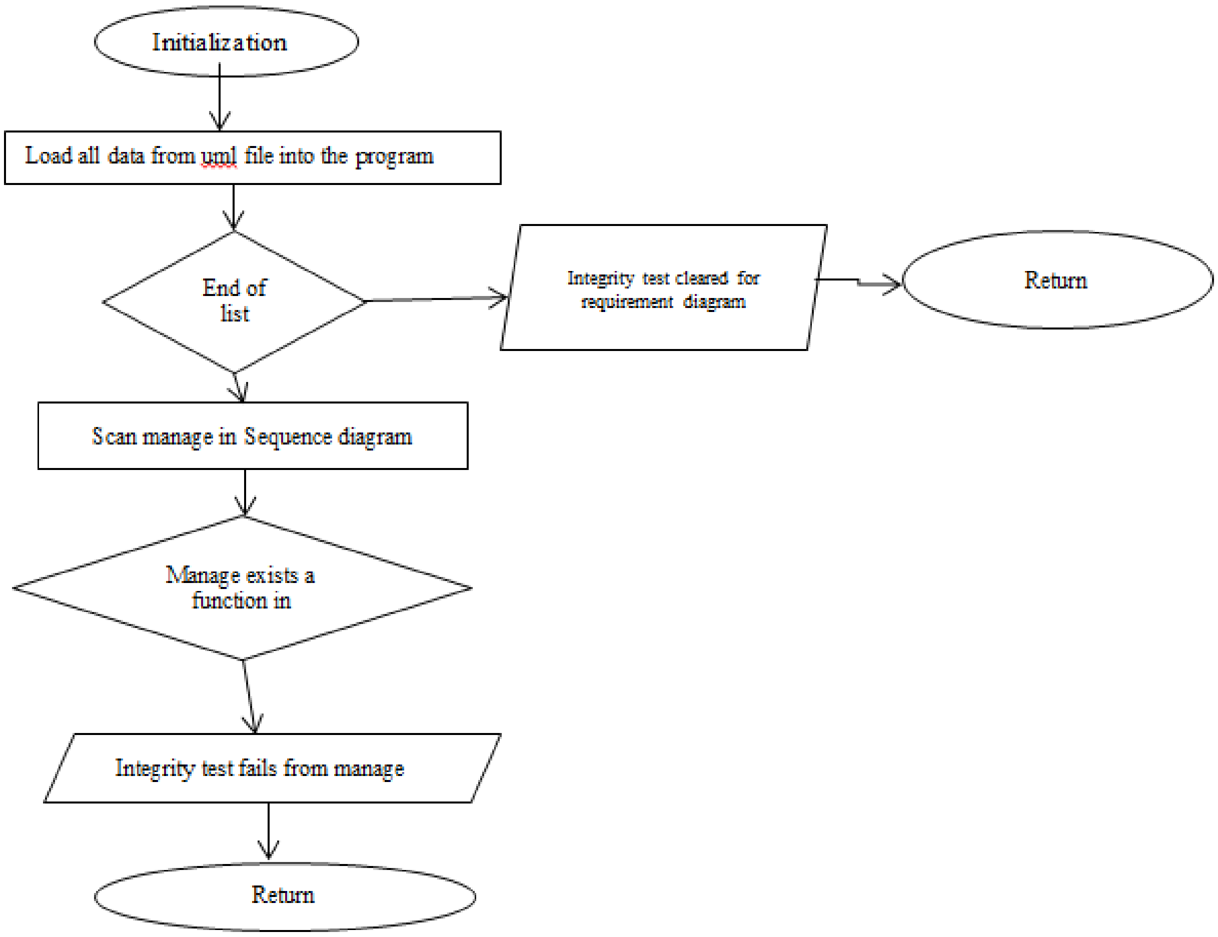
Initialization
Load all data from uml file into the program
End of list
Integrity test cleared for requirement diagram
Return
Scan manage in Sequence diagram
Manage exists a function in
Integrity test fails from manage
Return

Fig. 5: Integrity test case generation for class and sequence diagram

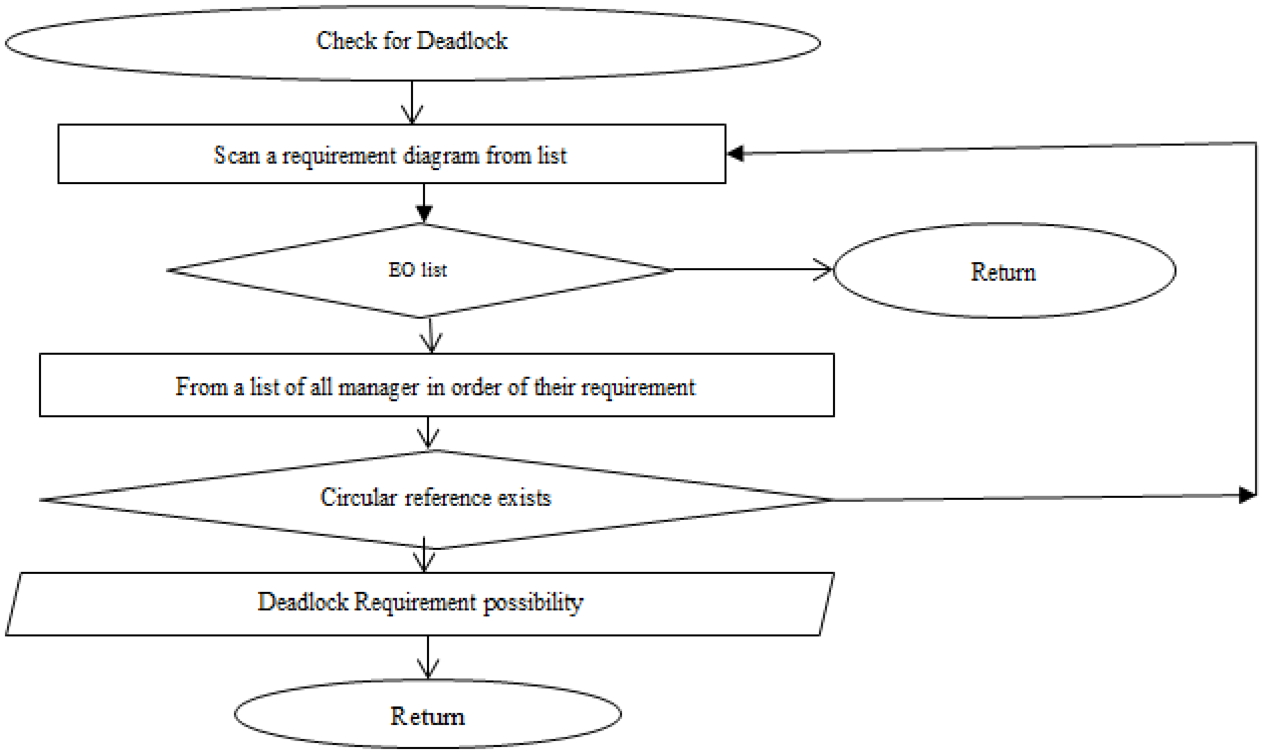
Check for Deadlock
Scan a requirement diagram from list
EO list
Return
From a list of all manager in order of their requirement
Circular reference exists
Deadlock Requirement possibility
Return

Fig. 6: Deadlock test in sequence diagram

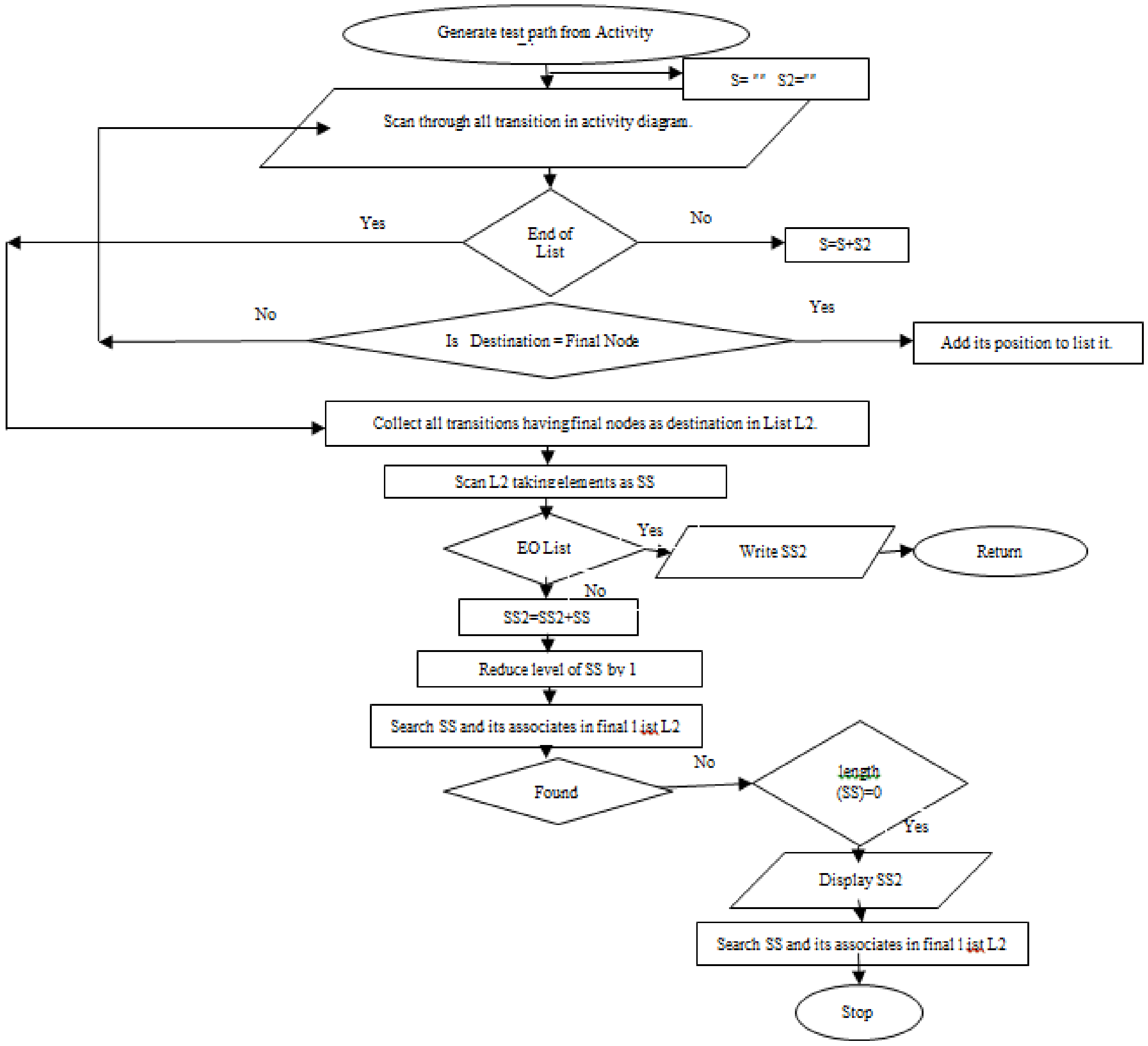
Generate test path from Activity
S="" S2=""
Scan through all transition in activity diagram.
End of List
Yes
No
S=S+S2
Is Destination = Final Node
No
Yes
Add its position to list it.
Collect all transitions having final nodes as destination in List L2.
Scan L2 taking elements as SS
EO List
Yes
Write SS2
Return
No
SS2=SS2+SS
Reduce level of SS by 1
Search SS and its associates in final list L2
Found
No
length (SS)=0
Yes
Display SS2
Search SS and its associates in final list L2
Stop

Fig. 7: Generate test path for activity diagram

The proposed method successfully generates the static, dynamic and integrity test cases for an UML design. The method also checks if any new module which is being added to an existing module in an OO Project, is compatible with existing module or not. Collaboration of XML code of sequence diagram, activity diagram and design based code of class diagram generated through Enterprise Architect, has been used to generate test cases automatically for dynamic behavior. The proposed tool quite successfully checks for test cases via class, activity and sequence diagrams.

The proposed method worked with Class diagrams, activity and Sequence diagrams to produce static, dynamic and integrity test cases. Moreover the proposed method fails to work with any other format except XML format. The future researchers can extend this research to work with other formats which UML can be exported to. Also if the discrepancies can be demonstrated graphically, that would help the system designers to create more robust designs. Further work can be explored to use formal methods to make system suitable for real and large systems. This technique further can be extended to include all types of software artifacts to predict change for regression testing.

ACKNOWLEDGMENT

I owe this work done by me to my guide Mr. Dr. Kulvinder Singh, without whose support and guidance, this work would never have been possible. I also thank my parents and family for their support and patience.

REFERENCES

[1] M. Auer, T. Tschurtschenthaler and S. Biffl, "A Flyweight UML Modelling Tool Software Development in Heterogeneous Environments," Proceedings of the 29thEUROMICRO Conference New Waves in System Architecture, Vol.3, pp. 267- 272,IEEE,2003.
[2] Bin Zheng, Ye Sho Chen,"Enhancing UML Conceptual Modelling Through The Use Of Virtual Reality," Proceedings of the 38th Hawaii International Conference on Sciences, pp. 11b,IEEE, 2005.
[3] JinShan Yu, Tun Li, QingPing Tan, "The Use of UML Sequence Diagram for System-on-Chip System Level Transaction-based Functional Verification," Proceedings of the 6th World Congress on Intelligent Control and Automation, pp. 6173-6177, IEEE, 2006.
[4] Yoshi Yuki Shinkawa,"Inter-Model Consistency in UML Based on CP Formalism," ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE, IEEE, 2006.
[5] Chen Mingsong, Qiu Xiaokang, and Li Xuandong, "Automatic Test Case Generation for UML Activity Diagrams", proceedings of ACM, pp. 2-8, ACM, 2006.
[6] Huo Yan Chen, Chuang Li and T.H. Tse, "Transformation of UML Interaction Diagrams Into Contract Specifications for Object-Oriented Testing," IEEE International Conference on Systems, Man and Cybernetics, pp. 1298-1303, IEEE, 2007.
[7] M. Nikolaidou, N. Alexopoulou, A. Tsadimas, A. Dais and D. Anagnostopoulos, "Accommodating EIS UML 2.0 Profile using a standard UML modeling tool,"International Conference on Software Engineering Advances, pp. 26, IEEE, 2007.
[8] Weikun Zheng, Gary Bundell, "Model based software component testing: A UML based Approach" 6th International Conference on Computer and Information Science, pp. 891-899, Vol.4, IEEE, 2007.
[9] Jung Ho Bae and Heung Seok Chae, "UML Slicer: A Tool for Modularizing the UML Meta model using Slicing," International Conference on Computer and Information Technology, pp. 772-777, IEEE 2008.
[10] Haifeng Shen, Siyuan Liu, Steven Xia, and Chengzheng Sun, "Distributed Constraints Maintenance in Collaborative UML Modeling Environments," International Conference on Automated Software Engineering, pp. 367-370, IEEE, 2008.
[11] Wenjung Deng and Yiwen Liang, "Reason on UML Diagrams with Answer Set Programming," International Conference on Automated Software Engineering, pp. 205-209 IEEE, 2008.
[12] Weiqun Zheng and Gary Bundell, "Test by Contract for UML-Based Software Component Testing," International Symposium on Computer Science and its Application", pp. 377-388, IEEE, 2008.
[13] Philip Samuel and Rajib Mall, "A Novel Test Case Design Technique Using Dynamic Slicing of UML Sequence Diagrams", in e-Informatica Software Engineering Journal, Volume 2, Issue 1, pp. 71-91, IEEE, 2008.
[14] John C Zubeck(PhD.), "Paring UML And DODAF Down To The Project-Vital Set Of Diagrams," 3rd Annual System Conference, pp. 18-22, IEEE, 2009.
[15] Xin Fan, Jian Shu, LinLan Liu and QiJun Liang, "Test Case Generation from UML Sub activity and Activity Diagram," Second International Symposium on Electronic Commerce and Security, pp. 244-248, IEEE, 2009.
[16] Ye Peilei, "An object-oriented development process and UML modeling tools," International Conference on Services Science, Management and Engineering, pp. 225-228, IEEE, 2009.
[17] Tibor Farkas, Carsten Neumann and Andreas Hinnerichs, "An Integrative Approach for Embedded Software Design with UML and Simulink," Annual IEEE International Computer Software and Applications Conference, Pp. 516-520, IEEE, 2009.
[18] Patrick Konemann, "Integrating Decision Management with UML Modelling Concepts and Tools," European Conference on Software Architecture, Pp. 297-300, IEEE, 2009.
[19] Debasish Kundu, Debasis Samanta: "A Novel Approach to Generate Test Cases from UML Activity Diagrams", In Journal of Object Technology, vol. 8, no. 3, pp. 65-83, JOT, 2009.
[20] Santosh Kumar, Swai Durga Prasad, Mohapatra Rajib Mall, "Test Case Generation Based on State and Activity Models", In Journal of Object Technology, pp. 1-27, JOT, 2010.
[21] Fabian Mischkalla, Da He and Wolfgang Mueller, "Closing the Gap between UML- based Modeling, Simulation and Synthesis of Combined HW/SW Systems," Design, Automation & Test in Europe Conference & Exhibition, pp. 1201-1206, IEEE, 2010.
[22] Sungwon Kang, Hyunho Kim, Jongmoon Baik, Hojin Choi and Changsup Keum, "Transformation Rules for Synthesis of UML Activity Diagram from Scenario-based Specification," 34th Annual Computer Software and Applications Conference, pp. 431- 436, IEEE, 2010.
[23] Fanchao Meng, Dianhui Chu and Dechen Zhan, "Transformation from Data Flow Diagram to UML2.0 Activity Diagram," International Conference on Progress in Informatics and Computing, Pp. 1010-1014, IEEE, 2010.
[24] Nicha Kosendrdecha and Jirapung Daengdej, "A Test Case Generation and Process Technique," Journal of software engineering, Pp. 265-287, JOT, 2010.
[25] Mohammed Misbhauddin and Mohammad Alshayeb, "Extending the UML Meta model For Sequence Diagram to Enhance Model Traceability," Fifth International Conference On Software Engineering Advances, Pp. 129-134, IEEE, 2010.
[26] Qaisar A. Malik, Dragos¸ Trus¸can and Johan Lilius," Using UML Models and Formal Verification in Model-Based Testing," 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, pp. 50-56, IEEE, 2010.
[27] Pakinam N. Boghdady, Nagwa L. Badr, Mohamed Hashem and Mohamed F.Tolba, "A Proposed Test Case Generation Technique Based on Activity Diagrams," International Journal of Engineering & Technology, pp. 35-52, IJET, 2011.
[28] Kanjanee Pechtanun and Supaporm kansonkeat, "Generation Test Case from UML Activity Diagram Based on AC Grammar," In Proceeding of the International Conference On Computer & Information Science, pp. 895-899, IEEE, 2012.
[29] Saru Dhir, "Impact of UML Techniques in Test Case Generation," In International Journal of Engineering Science & Advanced Technology, Volume-2, Issue-2, pp. 214- 217, IJESAT, 2012.
[30] Vibhash yadav and Raghuraj Singh, "Predicting design quality of Object oriented software Using UML diagrams," 3rd International Advance Computing Conference, pp. 1462-1467, IEEE, 2012.
[31] Suman Kumar Mishra, Dr. Harsh dev and Ajay Partap, "UML modeling for banking System using Object oriented databases," International Journal of Computer Engineering and Technology , pp. 630-639, IJCET, 2012.
[32] S. Shanmuga Priya, P. D. Sheba Kezia Malarchelvi, "Test Path Generation Using Uml Sequence Diagram," In International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, pp. 1069-1076, IJCET, 2013.
[33] Sunitha E.V and Philip Samuel, "Enhancing UML activity diagrams using OCL," International Conference on Computational Intelligence and Computing Research, pp. 1- 6, IEEE, 2013.
[34] Mrs. Sulakshana Nagpurkar and Mr. Y.B.Gurav, "A Survey on Test Case Generation From UML Based Requirement Analysis Model," International Conference on Information Systems and Computing, pp. 449-459 Vol. 2, Issue 5, IJART, 2013.
[35] Parampreet Kaur and Gaurav Gupta, "Automated Model-Based Test Path Generation From UML Diagrams via Graph Coverage Techniques," International Journal of Computer Science and Mobile Computing, Vol.2, Issue 7, pp. 302-311, IJCSMC, July 2013.
[36] M. Sijtema, A. Belinfante, M.I.A. Stoelinga and L. Marinelli, "Experiences with Formal Engineering: Model-Based Specification, Implementation and Testing of a Software us at Neopost," ELSEVIER, April 24 2013.
[37] Fozia Mehboob, Atif Aftab, Ahmed Jilani and Dr M.Abbass, "State Based Testing Using Swarm Intelligence," In Proceedings of the Science and Information Conference, pp.630-635, IEEE, 2013

[38] Regina Moraes, Helene Waeselynck and Jeremie Guiochet, "UML-Based Modeling of Robustness Testing" 15th International Symposium on High-Assurance Systems Engineering, pp. 168-175, IEEE, 2014.
[39] Shaukat Ali and Hadi Hemmati, "Model-based Testing of Video Conferencing Systems: Challenges, Lessons Learnt, and Results," Seventh International Conference on Software Testing, Verification and Validation, pp. 353-362, IEEE, 2014.
[40] Nina Elisabeth Holt, Lionel C. Briand and Richard Torkar, "Empirical evaluations on the Cost-effectiveness of state-based testing: An industrial case study," pp. 890-910, Elsevier, 2014.
[41] Amitashree Mallick, Namita Panda and Arup Abhinna Acharya, "Generation of Test Cases from UML Sequence Diagram and Detecting Deadlocks using Loop Detection Algorithm", International Journal of Computer Science and Engineering, Volume-2, Issue-3. Pp. 199-203, JCSE, 2014.
[42] Kerstin Duschl, Martin Obermeier, Birgit Vogel-Heuser, "An Experimental Study on UML Modelling Errors and their Causes in the Education of Model Driven PLC Programming," Global Engineering Education Conference, pp. 119-128, IEEE, 2014.
[43] http://tutorialspoint.com/UML