IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-9, Sept 2019 E-ISSN: 2347-2693

Slicing based on UML Diagram & Test Case Generation

Venus Grover[1*], **Jitender Kumar**[2]

[1,2]Dept. of Computer Science and Engineering, N.C. College of Engineering, Israna Panipat, India

**Corresponding Author: venusgrover27@gmail.com*

DOI: https://doi.org/10.26438/ijcse/v7i9.95101 | Available online at: www.ijcseonline.org

Accepted: 10/Sept/2019, Published: 30/Sept/2019

***Abstract*:** Software testing issued to evaluate a trait or potential of system and conclude that whether it meets necessary prospects. The most reasonably demanding part of testing is to plan of test cases. These days, UML has been broadly used for object oriented modeling and design. UML matamodel is used to describe structural and behavioural aspects of an architecture. However to recognize this performance is still hard, because the size of automatically generated model diagrams tends to be huge. To overcome this problem Software visualization model based slicing procedure has been developed. Model based slicing is a coherent advance to extract and recognize appropriate model parts or associated elements across diverse model views. On the basis of slicing criteria an original procedure has proposed to extort the sub-model from a big model diagrams. The planned methodology use the concept of model based slicing to slice the sequence diagram to extract the desired hunk. In the presented approach UML, conversion of UML into XML, Java DOM API for parsing and slicing has been used. Then Extracted Sequence Diagram has been generated by using the Editor. After that test case generation is performed.

***Keywords*-** Model Based Slicing, Sequence Diagram, Parsing, Slicing, UML.

I. INTRODUCTION

Increase in size and complexity the implication of architectural design has been increased software products. The architecture of software system defines its design structure and allows designer to find about several properties of the organization which are at the sophisticated level of Abstraction. For doing this, Unified Modeling Language (UML) is the best selection and with the help of its various model diagrams of software system, it is used to signify and construct the architecture. UML diagrams tell us about the behavioural and structural aspects of architecture [1, 2]. Several relations among objects, such as Combination, association, configuration and simplification/specialism etc. can be distinct by Basic models (e.g., class diagrams, object diagrams, component diagrams). On the other hand, sequence of actions, states and their communication, through which a use case is comprehended, can be defined through the behavioural models (e.g. communication and sequence diagrams, activity diagram, state diagrams). Evaluating UML Models is perplexing task since the information about the system can be spread across numerous model views. The concept of model based slicing came into existence to overcome this problem. To extract and identify relevant model parser related an element a decomposition technique is used and that is Model Based slicing. It takes the user defined slicing criteria as input and slices the architecture, as a view of interest [3]. Slicing is helpful in reengineering, software maintenance, testing, program comprehension, decomposition, integration, recompilation, and debugging. The goal of software testing is to substantiate quality. Highly reliable systems are produce by Software Testing, because static verification practices vacillate from several difficulties in detecting all software mistakes [15]. The most exciting part of testing is the aim of test cases. With the help of program source code Test Cases are generated. Supplementary approach is to generate test cases developed using formalisms such as UML models. Without affecting their core structure and functionality, structure can be decomposed into sub models and best technique for this is slicing. According to their requirement it helps the developer to put on the perfect view of software.

II. RELATED WORK

In early stage of development model based slicing has been applied to state machines [4] where related profits as those recorded above have been demanded. State machine slicing is that when applying slicing to a model of a system rather than to the system implementation. However, system models which are represented in terms of the UML-family of languages are more complex than state machines (and contain state machine sub-languages). Some efforts and approaches have been made for slicing UML diagrams. The approach in [5] define context-free slicing of UML class models where the matter of context has been defined to be object setting, which is a dynamic property of the situation therefore for the structural model context free slicing is a static slice. As described in [5] the procedure used for slicing a program or state machine is not much complex than that of slicing a model since there are more types of elements and relationships in program slicing or state machine. OCL (object constraint language) should be used to express the slicing criteria. A similar approach has been used to modularize the UML meta-model into groups of constituents that are related to the different UML diagram types in [6] although the predicate that are used to find out the slicing criteria has been secure in terms of traversing the meta-model elements starting with a collection of supplied classes. Class Models has been sliced in conjunction with OCL invariants, in [7] thereby decreasing the state-space explosion that would otherwise occur after using a model-checker to authenticate a class-model. UML state-charts can be sliced as described in [8] [9] [10] although these methods do not simplify the results to include other parts of the UML language family. Both static and dynamic aspects of UML can be combined and sliced as described in [11] [12] where class and sequence diagrams are merged into a single representation (a model dependency graph MDG) that can be consequently sliced to show partial dynamic and structural information resulting from criteria containing both structural and dynamic constraints. In order to generate test cases Slicing UML sequence diagrams has been described in [13] [14]. UML sequence diagrams (or scenarios) are basically an integral part of implementations of a program. It shows the objects and classes involved in the situation and the sequence of messages exchanged between the objects. Sequence diagrams are usually associated with use case realizations in the Logical View of the system under progress. It has been methodically analysed that for the procedure of slicing sequence diagram no consolidate technique have been developed to extract the point of interest from architecture of software to ease the software visualization that uses conditional predicate for finding out a relative slice. Generic view of functional behaviour of software models in the form of sequence diagram can be shown by Figure 1. Rectangular box states the objects within model diagram that are communicating with each other. Doted lines denote the life line of the objects on which instances of the objects has been created. Arrow tells about the particular action (in the form of messages) of objects and their direction.

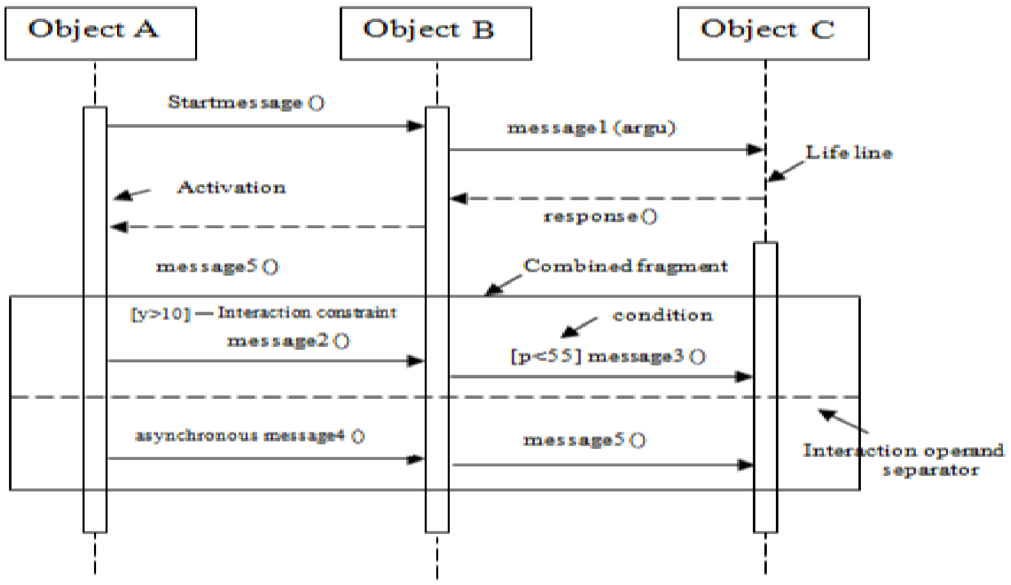


Figure 1: Generic view of Sequence diagram [1]

III. PROPOSED METHODOLOGY AND IMPLEMENTATION

The designed work addresses the slicing of sequence diagram to relieve the software visualization by using conditional predicate for finding applicable slices.
In the projected methodology, consequent steps have been followed:

1. From a particular constraint specification UML (Sequence) diagram has been generated.
   1.1. Rational rose, Visual paradigm for UML and Magic-draw, etc can be used to make the UML diagrams.

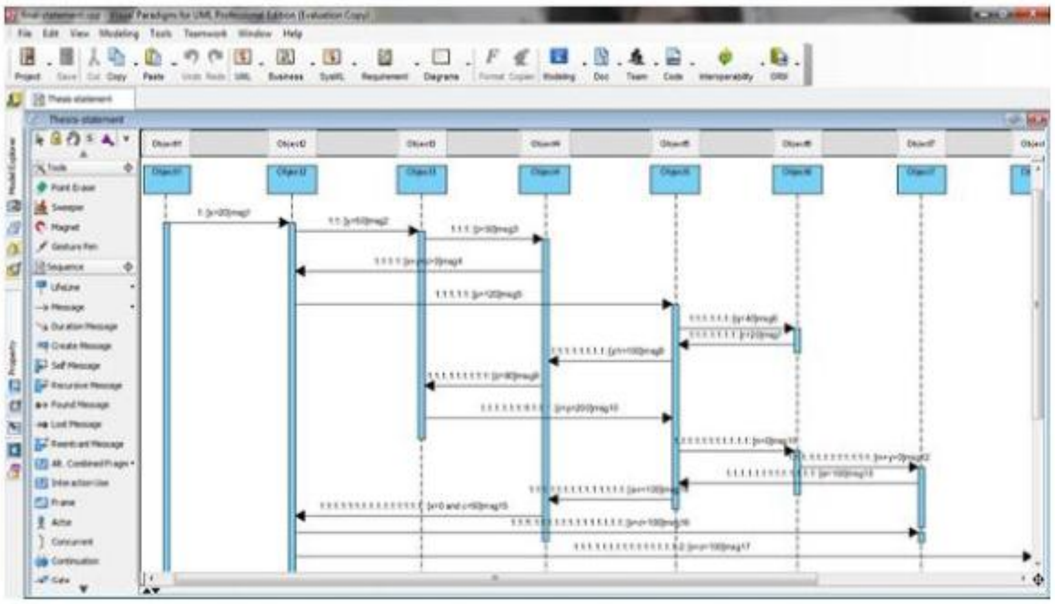
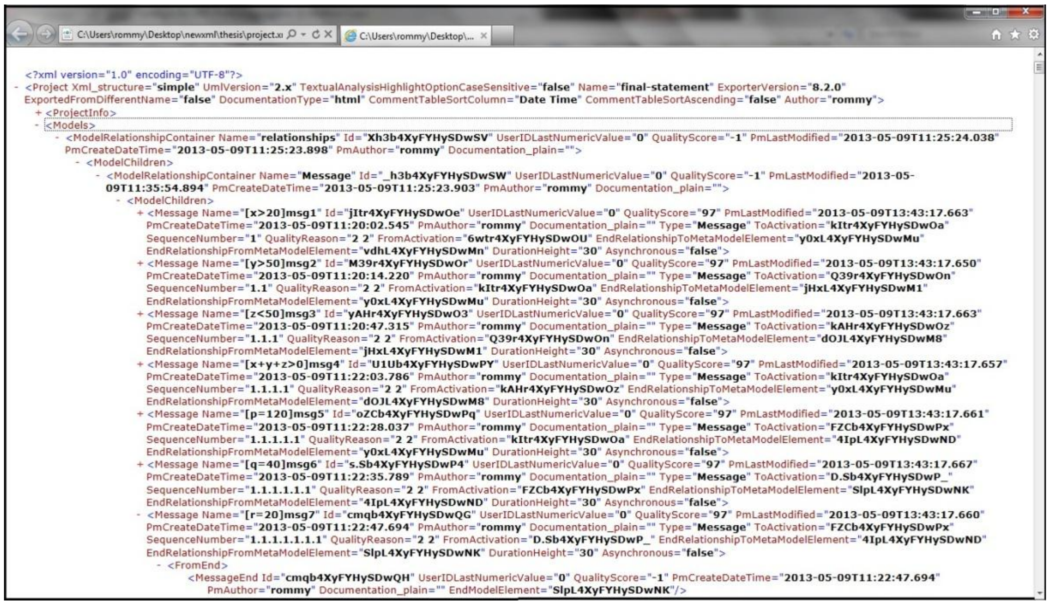


Figure 2: Designing sequence diagram using visual paradigm

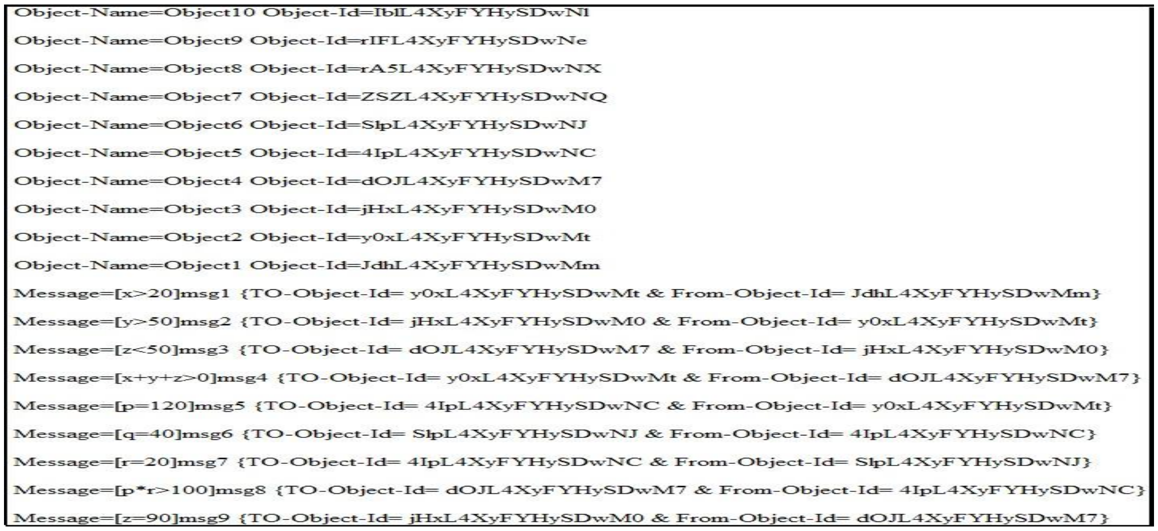
2. From the specified UML diagram (Sequence diagram) generate XML.
   2.1. To import the diagrams into XML format Visual paradigm for UML 10.0 version provides the in-built functionality.


C:\Users\rommy\Desktop\newxml\thesis\project.xı
C:\Users\rommy\Desktop\...

```
<?xml version="1.0" encoding="UTF-8"?>
- <Project Xml_structure="simple" UmlVersion="2.x" TextualAnalysisHighlightOptionCaseSensitive="false" Name="final-statement" ExporterVersion="8.2.0"
  ExportedFromDifferentName="false" DocumentationType="html" CommentTableSortColumn="Date Time" CommentTableSortAscending="false" Author="rommy">
  + <ProjectInfo>
  - <Models>
    - <ModelRelationshipContainer Name="relationships" Id="Xh3b4XyFYHySDwSV" UserIDLastNumericValue="0" QualityScore="-1" PmLastModified="2013-05-09T11:25:24.038"
      PmCreateDateTime="2013-05-09T11:25:23.898" PmAuthor="rommy" Documentation_plain="">
      - <ModelChildren>
        - <ModelRelationshipContainer Name="Message" Id="_h3b4XyFYHySDwSW" UserIDLastNumericValue="0" QualityScore="-1" PmLastModified="2013-05-
          09T11:35:54.894" PmCreateDateTime="2013-05-09T11:25:23.903" PmAuthor="rommy" Documentation_plain="">
          - <ModelChildren>
            + <Message Name="[x>20]msg1" Id="jItr4XyFYHySDwOe" UserIDLastNumericValue="0" QualityScore="97" PmLastModified="2013-05-09T13:43:17.663"
              PmCreateDateTime="2013-05-09T11:20:02.545" PmAuthor="rommy" Documentation_plain="" Type="Message" ToActivation="kItr4XyFYHySDwOa"
              SequenceNumber="1" QualityReason="2 2" FromActivation="6wtr4XyFYHySDwOU" EndRelationshipToMetaModelElement="y0xL4XyFYHySDwMu"
              EndRelationshipFromMetaModelElement="vdhL4XyFYHySDwMn" DurationHeight="30" Asynchronous="false">
            + <Message Name="[y>50]msg2" Id="M39r4XyFYHySDwOr" UserIDLastNumericValue="0" QualityScore="97" PmLastModified="2013-05-09T13:43:17.650"
              PmCreateDateTime="2013-05-09T11:20:14.220" PmAuthor="rommy" Documentation_plain="" Type="Message" ToActivation="Q39r4XyFYHySDwOn"
              SequenceNumber="1.1" QualityReason="2 2" FromActivation="kItr4XyFYHySDwOa" EndRelationshipToMetaModelElement="jHxL4XyFYHySDwM1"
              EndRelationshipFromMetaModelElement="y0xL4XyFYHySDwMu" DurationHeight="30" Asynchronous="false">
            + <Message Name="[z<50]msg3" Id="yAHr4XyFYHySDwO3" UserIDLastNumericValue="0" QualityScore="97" PmLastModified="2013-05-09T13:43:17.663"
              PmCreateDateTime="2013-05-09T11:20:47.315" PmAuthor="rommy" Documentation_plain="" Type="Message" ToActivation="kAHr4XyFYHySDwOz"
              SequenceNumber="1.1.1" QualityReason="2 2" FromActivation="Q39r4XyFYHySDwOn" EndRelationshipToMetaModelElement="dOJL4XyFYHySDwM8"
              EndRelationshipFromMetaModelElement="jHxL4XyFYHySDwM1" DurationHeight="30" Asynchronous="false">
            + <Message Name="[x+y+z>0]msg4" Id="U1Ub4XyFYHySDwPY" UserIDLastNumericValue="0" QualityScore="97" PmLastModified="2013-05-09T13:43:17.657"
              PmCreateDateTime="2013-05-09T11:22:03.786" PmAuthor="rommy" Documentation_plain="" Type="Message" ToActivation="kItr4XyFYHySDwOa"
              SequenceNumber="1.1.1.1" QualityReason="2 2" FromActivation="kAHr4XyFYHySDwOz" EndRelationshipToMetaModelElement="y0xL4XyFYHySDwMu"
              EndRelationshipFromMetaModelElement="dOJL4XyFYHySDwM8" DurationHeight="30" Asynchronous="false">
            + <Message Name="[p=120]msg5" Id="oZCb4XyFYHySDwPq" UserIDLastNumericValue="0" QualityScore="97" PmLastModified="2013-05-09T13:43:17.661"
              PmCreateDateTime="2013-05-09T11:22:28.037" PmAuthor="rommy" Documentation_plain="" Type="Message" ToActivation="FZCb4XyFYHySDwPx"
              SequenceNumber="1.1.1.1.1" QualityReason="2 2" FromActivation="kItr4XyFYHySDwOa" EndRelationshipToMetaModelElement="4IpL4XyFYHySDwND"
              EndRelationshipFromMetaModelElement="y0xL4XyFYHySDwMu" DurationHeight="30" Asynchronous="false">
            + <Message Name="[q=40]msg6" Id="s.Sb4XyFYHySDwP4" UserIDLastNumericValue="0" QualityScore="97" PmLastModified="2013-05-09T13:43:17.667"
              PmCreateDateTime="2013-05-09T11:22:35.789" PmAuthor="rommy" Documentation_plain="" Type="Message" ToActivation="D.Sb4XyFYHySDwP_"
              SequenceNumber="1.1.1.1.1.1" QualityReason="2 2" FromActivation="FZCb4XyFYHySDwPx" EndRelationshipToMetaModelElement="SlpL4XyFYHySDwNK"
              EndRelationshipFromMetaModelElement="4IpL4XyFYHySDwND" DurationHeight="30" Asynchronous="false">
            - <Message Name="[r=20]msg7" Id="cmqb4XyFYHySDwQG" UserIDLastNumericValue="0" QualityScore="97" PmLastModified="2013-05-09T13:43:17.660"
              PmCreateDateTime="2013-05-09T11:22:47.694" PmAuthor="rommy" Documentation_plain="" Type="Message" ToActivation="FZCb4XyFYHySDwPx"
              SequenceNumber="1.1.1.1.1.1.1" QualityReason="2 2" FromActivation="D.Sb4XyFYHySDwP_" EndRelationshipToMetaModelElement="4IpL4XyFYHySDwND"
              EndRelationshipFromMetaModelElement="SlpL4XyFYHySDwNK" DurationHeight="30" Asynchronous="false">
              - <FromEnd>
                  <MessageEnd Id="cmqb4XyFYHySDwQH" UserIDLastNumericValue="0" QualityScore="-1" PmCreateDateTime="2013-05-09T11:22:47.694"
                    PmAuthor="rommy" Documentation_plain="" EndModelElement="SlpL4XyFYHySDwNK"/>
```

Figure 3: XML file of Sequence diagram

3. Document Object Model (DOM) parser for parsing XML code and create an output file (with .txt extension) having Object name, identifier, message name, message to & fro information.
   3.1. Java API DOM is used to parse the XML code file generated in step 2.
   3.2. DOM parser uses the function Document-Builder-Factory ( ) to create the instance of the class to parse the file.
   3.3. DOM parser will generate a txt file having information regarding object name and its identifier. This file also contains the information related to all the messages and the objects among which the message is floating.
   3.4. All the information generated by parser will be stored in separate .txt file.



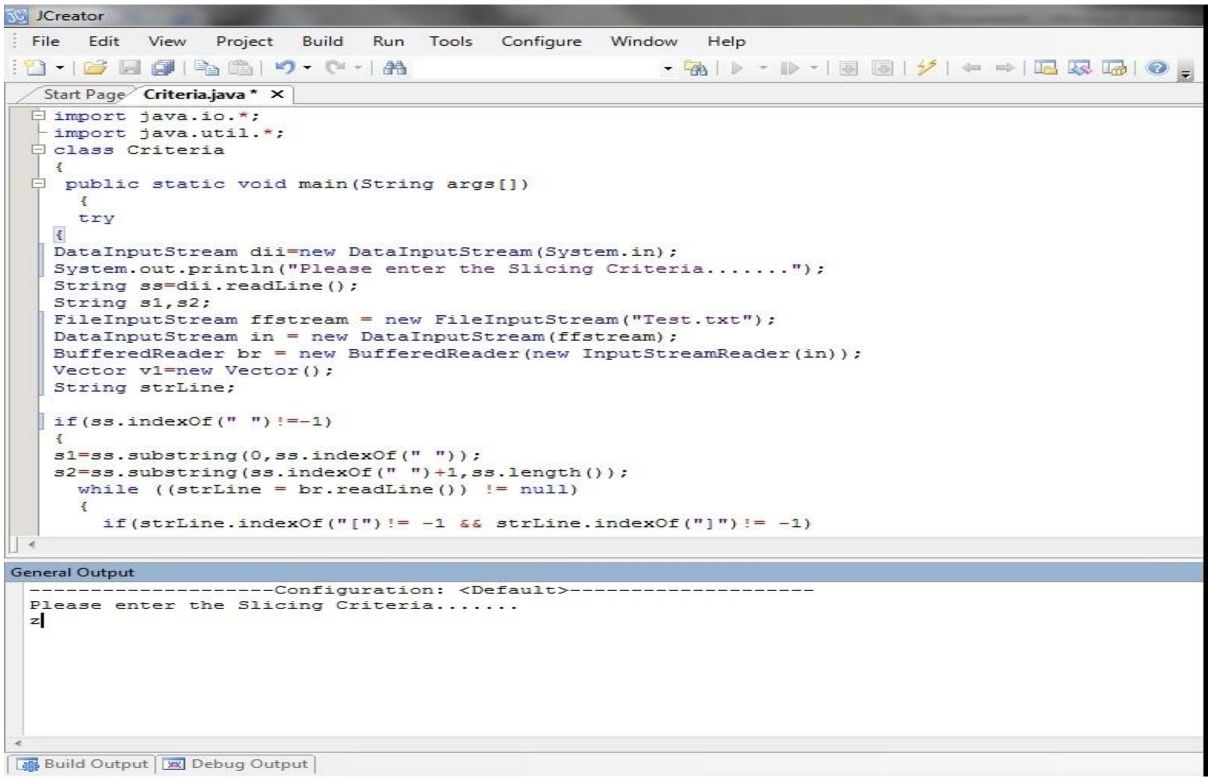
```
Object-Name=Object10 Object-Id=IblL4XyFYHySDwNl
Object-Name=Object9 Object-Id=rIFL4XyFYHySDwNe
Object-Name=Object8 Object-Id=rA5L4XyFYHySDwNX
Object-Name=Object7 Object-Id=ZSZL4XyFYHySDwNQ
Object-Name=Object6 Object-Id=SlpL4XyFYHySDwNJ
Object-Name=Object5 Object-Id=4IpL4XyFYHySDwNC
Object-Name=Object4 Object-Id=dOJL4XyFYHySDwM7
Object-Name=Object3 Object-Id=jHxL4XyFYHySDwM0
Object-Name=Object2 Object-Id=y0xL4XyFYHySDwMt
Object-Name=Object1 Object-Id=JdhL4XyFYHySDwMm
Message=[x>20]msg1 {TO-Object-Id= y0xL4XyFYHySDwMt & From-Object-Id= JdhL4XyFYHySDwMm}
Message=[y>50]msg2 {TO-Object-Id= jHxL4XyFYHySDwM0 & From-Object-Id= y0xL4XyFYHySDwMt}
Message=[z<50]msg3 {TO-Object-Id= dOJL4XyFYHySDwM7 & From-Object-Id= jHxL4XyFYHySDwM0}
Message=[x+y+z>0]msg4 {TO-Object-Id= y0xL4XyFYHySDwMt & From-Object-Id= dOJL4XyFYHySDwM7}
Message=[p=120]msg5 {TO-Object-Id= 4IpL4XyFYHySDwNC & From-Object-Id= y0xL4XyFYHySDwMt}
Message=[q=40]msg6 {TO-Object-Id= SlpL4XyFYHySDwNJ & From-Object-Id= 4IpL4XyFYHySDwNC}
Message=[r=20]msg7 {TO-Object-Id= 4IpL4XyFYHySDwNC & From-Object-Id= SlpL4XyFYHySDwNJ}
Message=[p*r>100]msg8 {TO-Object-Id= dOJL4XyFYHySDwM7 & From-Object-Id= 4IpL4XyFYHySDwNC}
Message=[z=90]msg9 {TO-Object-Id= jHxL4XyFYHySDwM0 & From-Object-Id= dOJL4XyFYHySDwM7}
```

Figure 4: Output-file generated by parser

4. Passing file obtained from step 3 and slicing criteria to a .java program (which act as slicer) for getting the relative/required chunk of information in a separate .txt file.

4.1. Generated .txt file in step 3 as input taken by the Slicer.

4.2. Slicer will ask user to tell about the slicing criteria at run time to generate the slice as per requirements.



```
import java.io.*;
import java.util.*;
class Criteria
{
 public static void main(String args[])
  {
  try
{
DataInputStream dii=new DataInputStream(System.in);
System.out.println("Please enter the Slicing Criteria.......");
String ss=dii.readLine();
String s1,s2;
FileInputStream ffstream = new FileInputStream("Test.txt");
DataInputStream in = new DataInputStream(ffstream);
BufferedReader br = new BufferedReader(new InputStreamReader(in));
Vector v1=new Vector();
String strLine;

if(ss.indexOf(" ")!=-1)
{
s1=ss.substring(0,ss.indexOf(" "));
s2=ss.substring(ss.indexOf(" ")+1,ss.length());
  while ((strLine = br.readLine()) != null)
  {
    if(strLine.indexOf("[")!= -1 && strLine.indexOf("]")!= -1)
```

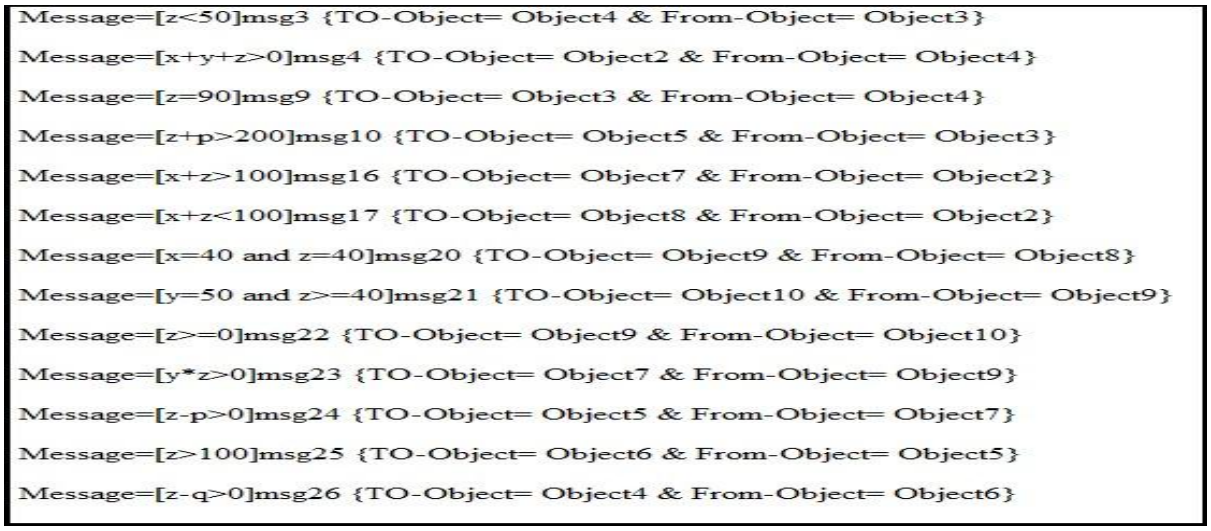
```
--------------------Configuration: <Default>--------------------
Please enter the Slicing Criteria.......
z
```

Figure 5: Java program for finding out the specified chunk

4.3. Computed slices will be store in separate .txt file which holds the information of messages, their guard condition and objects id's among which messages are being passed.

5. Changing object id with relative object name among which message is passing so that information can be retrieved easily (this step will only deal with sliced part).

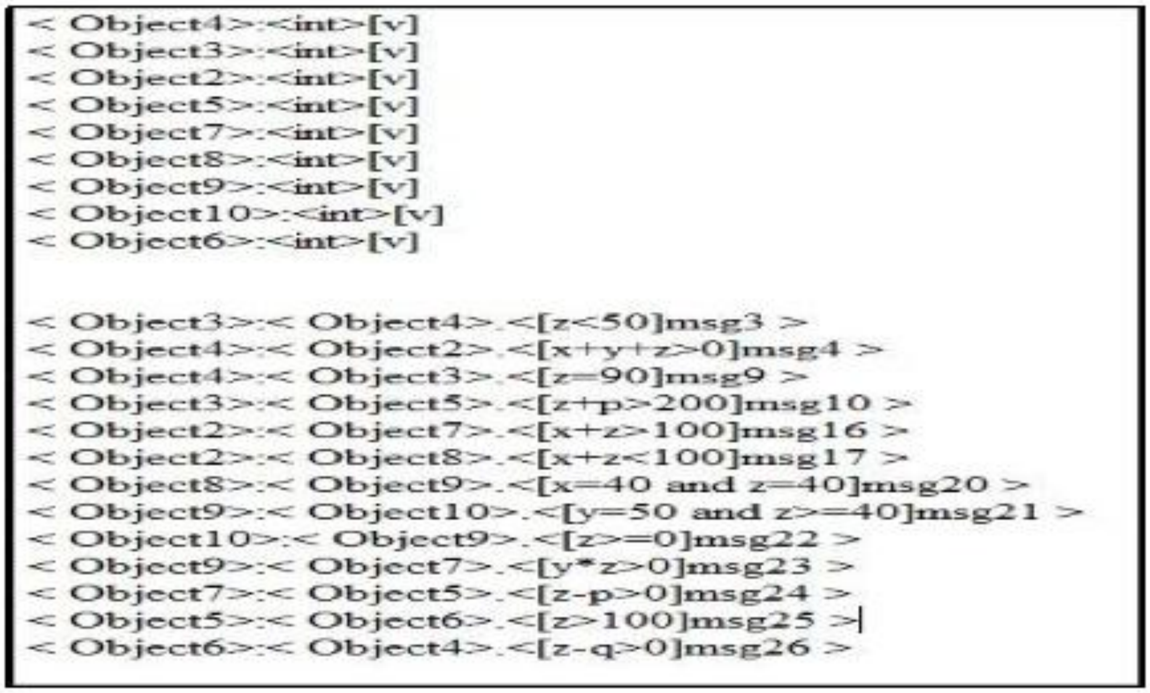
5.1. To ease the retrieved of information objects id's will replaced by their corresponding object name (in the file retrieved from step 4.3).
5.2. All the information will store in separate .txt file which holds the information of messages and the objects name (among which they are communicating relative to user defined slicing criteria).



```
Message=[z<50]msg3 {TO-Object= Object4 & From-Object= Object3}
Message=[x+y+z>0]msg4 {TO-Object= Object2 & From-Object= Object4}
Message=[z=90]msg9 {TO-Object= Object3 & From-Object= Object4}
Message=[z+p>200]msg10 {TO-Object= Object5 & From-Object= Object3}
Message=[x+z>100]msg16 {TO-Object= Object7 & From-Object= Object2}
Message=[x+z<100]msg17 {TO-Object= Object8 & From-Object= Object2}
Message=[x=40 and z=40]msg20 {TO-Object= Object9 & From-Object= Object8}
Message=[y=50 and z>=40]msg21 {TO-Object= Object10 & From-Object= Object9}
Message=[z>=0]msg22 {TO-Object= Object9 & From-Object= Object10}
Message=[y*z>0]msg23 {TO-Object= Object7 & From-Object= Object9}
Message=[z-p>0]msg24 {TO-Object= Object5 & From-Object= Object7}
Message=[z>100]msg25 {TO-Object= Object6 & From-Object= Object5}
Message=[z-q>0]msg26 {TO-Object= Object4 & From-Object= Object6}
```

Figure 6: computed slice after the conversion of object-id to object-name

6. Passing txt file as obtained from step 5, to a .java program so that it can be converted into input file format for Quick Sequence Diagram Editor.



```
< Object4>:<int>[v]
< Object3>:<int>[v]
< Object2>:<int>[v]
< Object5>:<int>[v]
< Object7>:<int>[v]
< Object8>:<int>[v]
< Object9>:<int>[v]
< Object10>:<int>[v]
< Object6>:<int>[v]


< Object3>:< Object4>.<[z<50]msg3 >
< Object4>:< Object2>.<[x+y+z>0]msg4 >
< Object4>:< Object3>.<[z=90]msg9 >
< Object3>:< Object5>.<[z+p>200]msg10 >
< Object2>:< Object7>.<[x+z>100]msg16 >
< Object2>:< Object8>.<[x+z<100]msg17 >
< Object8>:< Object9>.<[x=40 and z=40]msg20 >
< Object9>:< Object10>.<[y=50 and z>=40]msg21 >
< Object10>:< Object9>.<[z>=0]msg22 >
< Object9>:< Object7>.<[y*z>0]msg23 >
< Object7>:< Object5>.<[z-p>0]msg24 >
< Object5>:< Object6>.<[z>100]msg25 >
< Object6>:< Object4>.<[z-q>0]msg26 >
```

Figure 7: Input file for quick sequence diagram editor

7. Tool will generate the final and relatively small sequence diagram.
7.1. Tool will take the input format defined at step 6 as input to convert into its equivalent diagram.

7.2. Refined slice (small sequence diagram) will be generated as final output according to slicing criteria as per requirement to ease the software visualization.

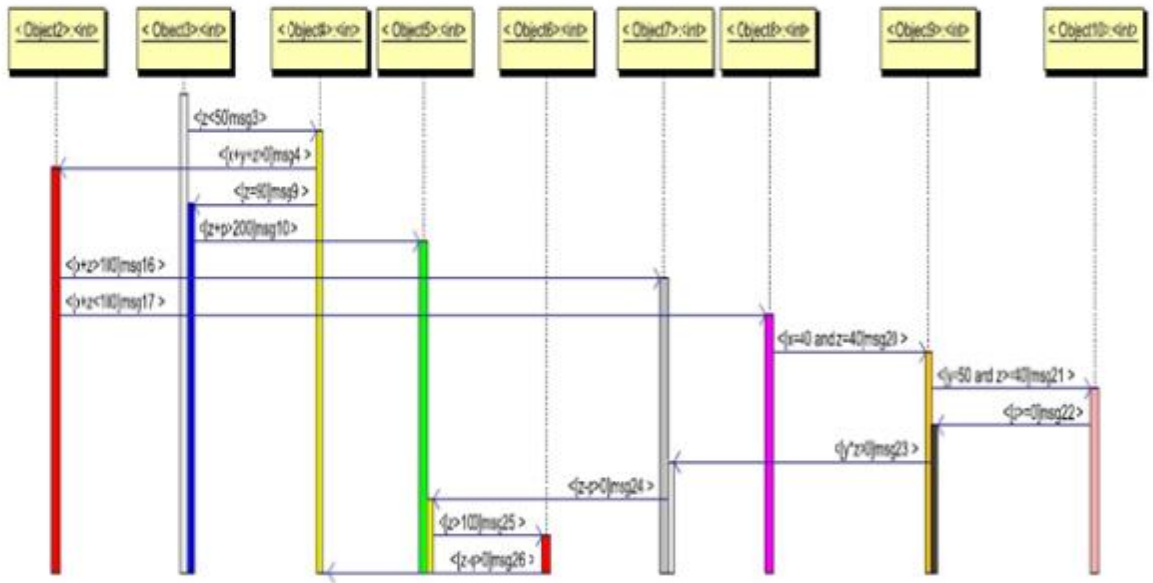


Figure 8: Computed Sliced Sequence diagram

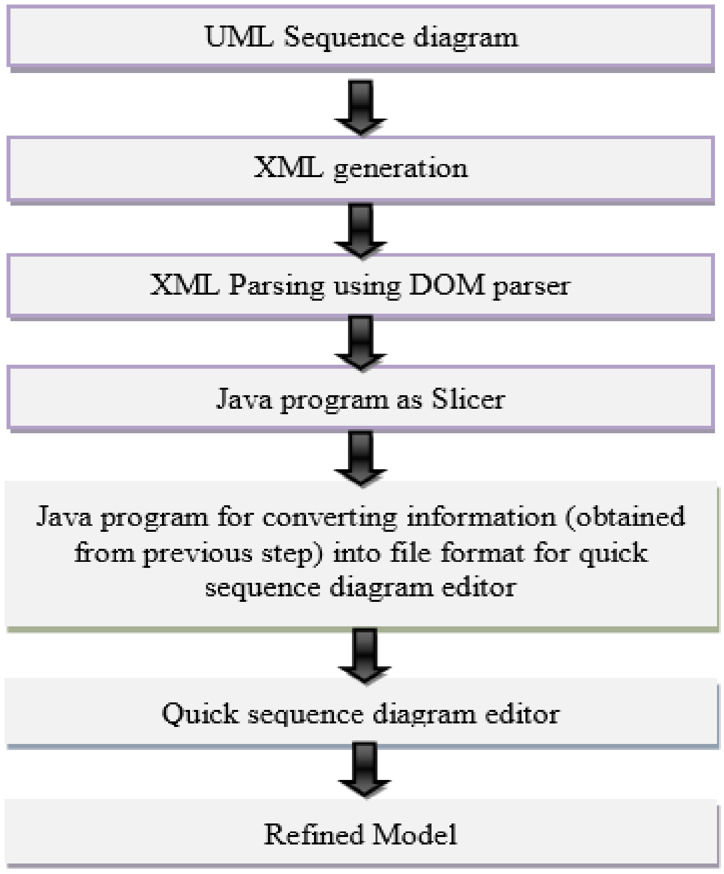

UML Sequence diagram
XML generation
XML Parsing using DOM parser
Java program as Slicer
Java program for converting information (obtained from previous step) into file format for quick sequence diagram editor
Quick sequence diagram editor
Refined Model

Figure 9: Overview of Methodology

Test Case Generation: Finally, the test cases are generated by combining all the conditions of messages from each of the case in a Sequence Diagram. In other words, we need to go from sequence diagram to Sequence variant, using decision tables generated from sequence diagram. Test cases should cover all variants, at least once. So, test cases are generated by taking the predicates as input and Message to Actor (MtA) as output from the decision table. The columns model the initial conditions in which test cases must be run, the actions that are taken as a result of running the test cases.

Test Case1 (Title.isbn=read.java, Output: test.txt file created)
Test Case2 (Title.isbn ≠ read.java, Output: Title does not exist)
Test case 3 (Title.isbn = criteria.java, Output: Display in abc.txt file)
Test case 4 (Title.isbn = after program sliced, message= [x>20] msg1 [To-Class-Id= pY2VUZyFYHySawNz & From- Class-Id= 7RWVUZyFHySawNs] Output: Id's converted into respective names.
Test case 5 (Title.isbn = Fileread.java Output: Output will generate in out.txt, Sliced program generated)

IV. CONCLUSION

To create the refined model slices related to slicing criteria using conditional predicate in sequence diagram is the key contribution of the technique. To ease the software visualization practical implementation of technique that will extract the sub-model from architecture of software has been discussed. The foundation of the proposed technique is 'Slicing' and 'UML'. With this, the problem of visualization of large and complex software can be handled efficiently. The projected technique has focused on the generation of chunk using model based slicing but still there are the few points that can be explored further like model reduction in synchronized and dispersed software design.

REFERENCES

[1] Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide," 2nd Edition, May 2005, Publisher. Addison Wesley.

[2] Jianjun Zhao, "Slicing Software Architecture," Technical Report 97-SE-117, pp.85-92, Information Processing Society of Japan, Nov 2007.

[3] Rupinder Singh and VinayArora, "Literature Analysis on Model based Slicing," International Journal of Computer Applications, vol. 70(16), pp: 45-51, May 2016. Published by Foundation of Computer Science, New York, USA.

[4] K. Androutsopoulos, D. Clark, M. Harman, Z. Li, and L. Tratt. Control dependence for extended finite state machines. Fundamental Approaches to Software Engineering, pp. 216–230, 2017.

[5] H. Kagdi, J.I. Maletic, and A. Sutton, "Context-Free Slicing of UML Class Models," Proc. 21st IEEE Int"l Conf. Software Maintenance, pp. 635-638, 2018.

[6] J.H. Bae, K.M. Lee, and H.S. Chae. Modularization of the UML metamodel using model slicing. In Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on, pages 1253–1254. IEEE, 2018.

[7] A. Shaikh, R. Clarisó, U.K. Wiil, and N. Memon, "Verification-driven slicing of UML/OCL models," In Proceedings of the IEEE/ACM international conference on Automated software engineering, pages 185–194. ACM, 2019.

[8] KevinLano Crest, "Slicing of UML State Machines," Proceedings of the 9th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (AIC '09), 2010.

[9] V. Ojala, "A slicer for UML state machines," Helsinki University of Technology, 2012.

[10] S. Van Langenhove, "Towards the Correctness of Software Behavior in UML: A Model Checking Approach Based on Slicing," Dissertation, Department of Mathematics, Ghent University, 2014.

[11] J.T. Lallchandani and R. Mall, "Slicing UML architectural models," ACM SIGSOFT Software Engineering Notes, vol.33, no.3, pp. 1–9, 2018.

[12] J.T. Lallchandani and R. Mall, "Integrated state-based dynamic slicing technique for UML models," Software, IET, vol. 4, no. 1, pp. 55–78, 2010.

[13] P. Samuel and R. Mall. A Novel Test Case Design Technique Using Dynamic Slicing of UML Sequence Diagrams.e-Informatica Software Engineering Journal Selected full texts, vol. 2, no. 1, pp. 61–77, 2008.

[14] P. Samuel, R. Mall, and S. Sahoo, "UML Sequence Diagram Based Testing Using Slicing," IEEE Indicon 2005 Conference, pages 176–178, IEEE, 2016.

[15] R. V. Binder, "Testing object-oriented software: a survey," Software Testing Verification and Reliability, vol. 6(3/4), pp: 125 – 252, 2017.

Authors Profile

Venus Grover had completed B.tech from N C College of Engineering, Israna affiliated from Kurukshetra University and pursuing M.Tech from N.C.College of Engineering Israna Panipat.