Measuring Hit ratio of Software Systems using UML Sequence Diagram

Durga Prasad Mohapatra[1], Sangharatna Godbley[1] and Arpita Dutta[1]
[1]DOS Lab, Dept. of CSE NIT Rourkela, Odisha, India.
Email: Durga@nitrkl.ac.in, sanghu1790@gmail.com, arpitad10j@gmail.com,

***Abstract*— In this proposed work, we discuss how to generate test cases automatically from UML sequence diagram to compute Hit Ratio percentage of software systems. First, we construct the sequence diagram of the given software using ArgoUML tool. This sequence diagram is supplied to code converter to get XML code of the designed software. Then, we supply this XML code to Tcases tool to generate test cases according to black-box testing technique. These test cases are supplied to Hit Ratio Calculator to compute Hit Ratio percentage of the software system for five case studies comprise of twenty three sequence diagrams, on an average, we achieved 69.69 % Hit Ratio percentage.**

Keywords: Sequence Diagram; Black-Box testing; TCases; Hit Ratio.

I. INTRODUCTION

There are many software development life cycle models available for developing software products. All the life cycle models are following the five phases: Requirement gathering and analysis, Designing, Coding, Testing and Maintenance. According to these phases, testing should be done only after the code is available. In this paper, we have proposed a novel approach of software testing, which requires only the design documentation of the software. This proposed approach enables software testing engineers to generate test cases efficiently using design documents.

Previously Data Flow Diagrams (DFD) were used for the modeling of software at the design phase, but DFDs were-unable to present much information of the system e.g. Control aspects, decomposition, etc. To remove those limitations, UML diagrams are coming into force. UML Sequence diagrams show all the conditions, predicates and interactions among all the objects present in the system. So, we have chosen this diagram to generate test cases for a given software system.
Software testing is classified into two categories. 1) White-Box testing 2) Black-Box testing. White Box testing tests the internal structure of the program. It should be performed only when the complete code of the program is available. Instead of this, in black box testing we need to know only the functional requirements of the system to generate test cases. We have taken Black-box testing to generate the test cases using the input sequence diagram.

In this paper we have proposed a novel technique to calculate Hit Ratio using sequence diagram. To automate the approach, we have developed a tool called HRGSD (Hit Ratio Generation Using Sequence Diagram). It is the integration of

two open source tools namely ARGOUML (for designing sequence diagrams) and Tcases (to generate test cases) and two more tools developed by us, namely "XMI to XML convertor" (for converting XMI file to XML file and to make it compatible with the Tcases tool) and "Hit Ratio Calculator" (to calculate the hit ratio).

The rest of the article is organized as follows: Section II presents the Basic Concepts. Section III describes our Proposed Work. Section IV presents the Experimental Results. Section V deals with some Related Work. Section VI concludes the proposed work and suggests some important possible future work.

II. BASIC CONCEPTS

In this section, we discuss some basic concepts to understand our proposed work:

a) ***UML Sequence Diagram:*** It is used to show the interaction of a given set of objects. A sequence diagram (SD) comprises of a sequence of messages "M" and a set of objects "O". Each message m that belongs to M depicts a sequential control flow from one object (o1) of the system to another object (o2) of the system.

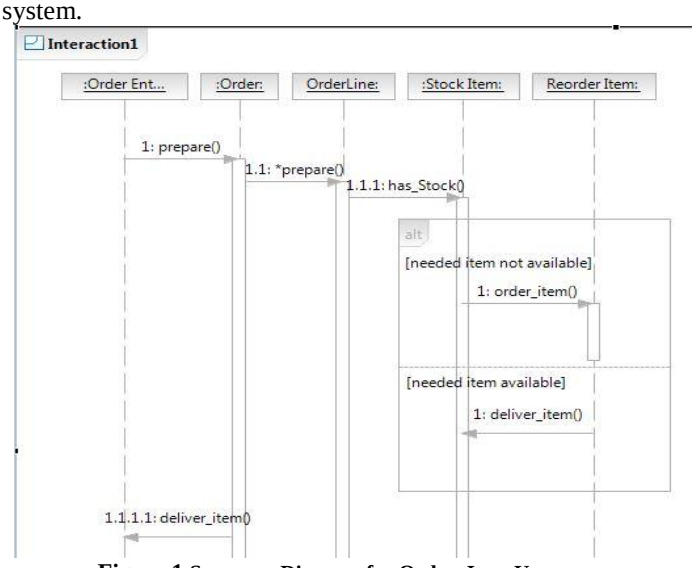

Figure 1 Sequence Diagram for Order_Item Usecase

b) ***Black-Box testing:*** In this testing technique, there is no requirement of the source code. It is also known as functional testing. It requires only formal inputs and expected outputs. Test cases can be generated as soon as functional requirements are available.

c) ***Hit Ratio percentage:*** It defines the percentage of number of valid test cases generated among the total generated test cases from Tcases tool.

III. PROPOSED WORK: HRGSD

In this section, we discuss our proposed approach. First, we present the overall flow of the proposed approach, and then show the schematic representation of our proposed approach followed by it's the algorithmic description.

A. Overview

Figure 2 shows the schematic representation of our proposed work Hit Ratio Generation Using Sequence Diagram (HRGSD). HRGSD consists of four modules, which are as follows: ArgoUML, XMI to XML convertor, Tcases Tool, and Hit Ratio calculator. ArgoUML and Tcases are the open source modules, whereas the remaining two XMI to XML convertor and Hit Ratio calculator are developed by us. ArgoUML is used to design a sequence diagram and it generates an .xmi file for that diagram. Then, this .xmi file is passed to the "XMI to XML generator" which will generates the .xml presentation of the .xmi file and creates the input space for that .xml file again in .xml format and gives it as an output. After that the .xml file is passed into the Tcases tool to generate the test cases. Tcases is a tool used for designing the test cases. It is used for all levels of systems (full system, subsystem, and unit). In this tool, we have to provide input space of the system under test. Based upon the input space, it generates all possible test cases and it also provides the information that whether the test case will lead to a valid system state or not. It is primarily a black-box testing tool. According to Tcases, the concept of "coverage" is slightly different from the structural testing (e.g. statement coverage, branch coverage, etc.). Instead, Tcases is guided by the coverage of the input space of our system. Finally, the test cases are supplied into Hit Ratio calculator to generate the Hit Ratio%.

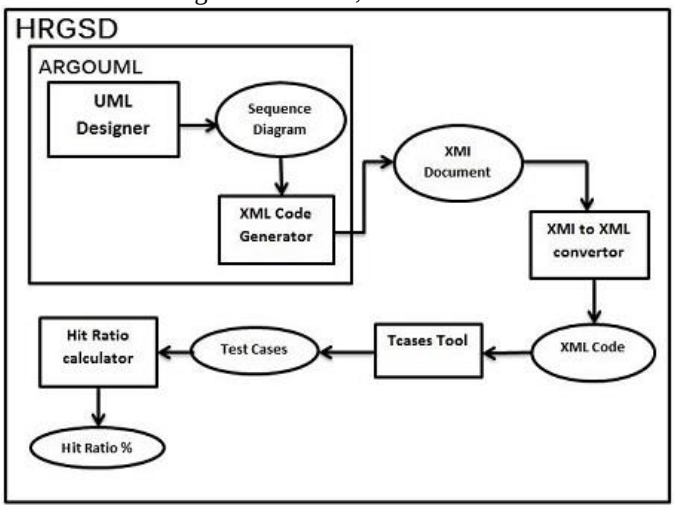

Figure 2: Schematic representation of HRGSD

B. Description of proposed work with example

In this section, we explain our proposed approach with the help of an example. We start with designing of sequence diagram with the help of ArgoUML. We have taken Order_Item use case for creating an example sequence diagram which is shown in Figure 1. ArgoUML generates .xmi code for the sequence diagram. Then the output .xmi document is passed to the "XMI to XML convertor". This convertor converts the .xmi file into .xml format and it also creates the input space definition for that .xml file. XMI is an application of XML. So, in order to convert, the convertor has to change only syntax in the original .xmi document. Then the converter generates all possible combinations of the input values for that file from its input space. The .xml file generate for the example Order_Item sequence diagram is shown in Figure 3. After that, the .xml file is passed to the Tcases tool to generate the test cases. We can also input the generator definition for the .xml file which will define the coverage that we require for each function. But we can also skip this step. Tcases can be run through directly from the command prompt or also we can install maven plugin for eclipse to do this. Tcases also generates a test report for the generated test cases to show the count of valid and invalid test cases. The test cases and test report generated by the Tcases are shown in Figure 4 and 5 respectively. From, Figure 5 we can see that five valid test cases and four invalid test cases are present. Therefore, by using the Hit Ratio calculator, we have calculated 55.56% as Hit Ratio percentage using the formula given in Eq. (1)

$$Hit\ Ratio\% = \frac{Valid\ Test\ Cases}{Total\ Number\ of\ Test\ cases} * 100 \qquad (1)$$

C. Algorithmic Description

Algorithm 1 HRGSD

Input: UML Sequence Diagram
Output: Hit Ratio%

1. Design a Sequence diagram using ArgoUML.
2. Generate .xmi file using ArgoUML for the sequence diagram.
3. Execute XMI to XML convertor to produce .xml file.
4. Execute Tcases for the .xml file to generate test cases.
5. Compute Hit Ratio% by Hit Ratio calculator using test cases generated from Tcases.

```
<diagram xmlns="x-schema:sequenceDiagram.xsd">
 <class-name name="OrderEntryWindow">
  <class-operator condition="true" multiplicity="one">
   <method-name>prepare</method-name>
   <class-name name="Order">
    <class-operator multiplicity="many" condition="true">
     <method-name>prepare</method-name>
     <class-name name="OrderLine">
      <class-operator condition="true" multiplicity="one">
       <method-name>checkStock</method-name>
       <args-type>retval=hasStock</args-type>
       <class-name name="StockItem" />
      </class-operator>
      <class-operator condition="hasStock" multiplicity="one">
       <method-name>remove</method-name>
       <class-name name="StockItem">
        <class-operator condition="true" multiplicity="one">
         <method-name>needToReorder</method-name>
         <args-type>retval=needsReorder</args-type>
         <class-name name="StockItem" />
        </class-operator>
        <class-operator condition="needsReorder" multiplicity="one">
         <method-name>new</method-name>
         <class-name name="ReorderItem" />
        </class-operator>
       </class-name>
      </class-operator>
      <class-operator condition="hasStock" multiplicity="one">
       <method-name>new</method-name>
       <class-name name="DeliveryItem" />
      </class-operator>
     </class-name>
    </class-operator>
   </class-name>
  </class-operator>
 </class-name>
</diagram>
```

Figure 1 XML document generated by the converter

Figure 2: Test cases generated by Tcases tool

```
19:34:36.523 INFO  org.cornutum.tcases.Tcases - Tcases 1.5.4 (2016-02-18)
19:34:36.538 INFO  org.cornutum.tcases.Tcases - Reading system input definition=null
19:34:36.647 INFO  o.c.t.generator.TupleGenerator - FunctionInputDef[placeOrder]: Preparing constraint info
19:34:36.694 INFO  o.c.t.generator.TupleGenerator - FunctionInputDef[placeOrder]: Generating test cases
19:34:36.694 INFO  o.c.t.generator.TupleGenerator - FunctionInputDef[placeOrder]: Extended 0 valid base test cases
19:34:36.788 INFO  o.c.t.generator.TupleGenerator - FunctionInputDef[placeOrder]: Created 5 valid test cases
19:34:36.788 INFO  o.c.t.generator.TupleGenerator - FunctionInputDef[placeOrder]: Extended 0 base failure test cases
19:34:36.850 INFO  o.c.t.generator.TupleGenerator - FunctionInputDef[placeOrder]: Created 4 failure test cases
19:34:36.866 INFO  o.c.t.generator.TupleGenerator - FunctionInputDef[placeOrder]: Completed 9 test cases
19:34:36.866 INFO  org.cornutum.tcases.Tcases - Updating test definition file=null
```

Figure 3: Test report generated by Tcases

Algorithm 1 deals with the functioning of HRGSD. Step 1 deals with designing of sequence diagram using ArgoUML. Step 2 shows .xmi code-generation of the constructed Sequence Diagram. Step3 shows the code conversion from XMI to XML using code Convertor. Step 4 shows the use of Tcases to accept the .xml code in order to generate test cases. Step 5 deals with Hit Ratio calculator execution to calculate the Hit ratio%.

IV. EXPERIMENTAL RESULTS

In this section we discuss the experimental results of our proposed work.

A. Setup

Our Computer system is configured using dual core processor of i3 version with 4 GB RAM. We have used Windows 7 professional operating system of the experimental work. We have also used two open source tools and two developed tools. The open source tools are ArgoUML and Tcases, and our developed tools are XMI to XML convertor and Hit ratio% calculator.

B. Results

Table 1 Characteristics of case study projects

Sl. No.	Case Study project	Sequence Diagram considered	# of Objects Involved	# of Actors Involved	# Conditi Present	# Alterna Present	# Asynch us Mes present	# Sync Mess prese
1	LMS	User Login	4	2	2	1	0	4
2	LMS	Issue Book	3	1	2	1	0	2
3	LMS	Search Book	3	1	2	1	0	2
4	LMS	Book Reserve	3	2	3	2	1	4
5	LMS	Book Return	4	2	3	3	1	4
6	PLMS	PLMS	4	2	2	1	2	4
7	WBAMS	User Login	3	1	3	2	1	5
8	WBAMS	Place Order	4	2	0	0	1	5
9	WBAMS	Recv Order	4	2	2	1	3	2
10	WBAMS	Manage Product	3	1	0	0	1	5
11	WBAMS	Manage Employee	3	2	0	0	3	8
12	WBAMS	Supplier Management	3	1	0	0	1	3
13	RRTS	Controller	4	2	2	1	1	3
14	RRTS	Machine	5	3	3	2	3	6
15	RRTS	RRTS Clerk	4	2	2	1	1	2
16	RRTS	RRTS Repair	5	3	2	1	2	3
17	RRTS	Database	3	2	2	1	1	2
18	RAS	User Login	4	2	2	1	0	4
19	RAS	Menu Card	6	2	2	1	1	7
20	RAS	Purchase Order	6	3	4	2	2	14
21	RAS	Issue Ingredients	5	2	2	1	1	5
22	RAS	View Report	3	2	7	3	0	5

Table 2 Experimental Results

Sl. No.	Case Study project	Sequence Diagram considered	Test Cases	Valid Cases	Invalid Cases	Hit ratio %
1	LMS	User Login	8	5	3	62.5%
2	LMS	Issue Book	5	3	2	60%
3	LMS	Search Book	5	4	1	80%
4	LMS	Book Reserve	7	4	3	57%
5	LMS	Book Return	8	6	2	75%
6	PLMS	PLMS	12	8	4	66.67%
7	WBAMS	User Login	9	6	3	66.67%
8	WBAMS	Place Order	9	5	4	55.56%
9	WBAMS	Recv Order	6	5	1	83.33%
10	WBAMS	Manage Product	11	7	4	63.64%
11	WBAMS	Manage Employee	11	8	3	72.72%
12	WBAMS	Supplier Managemen	13	8	5	61.53%
13	RRTS	Controller	10	7	3	70%
14	RRTS	Machine	11	8	3	72.72%
15	RRTS	RRTS Clerk	6	4	2	66.67%
16	RRTS	RRTS Repair	8	4	4	50%
17	RRTS	Database	7	5	2	71.4%
18	RAS	User Login	6	5	1	83.33%
19	RAS	Menu Card	13	10	3	76.92%
20	RAS	Purchase Order	20	16	4	80%
21	RAS	Issue Ingredients	10	7	3	70%
22	RAS	View Report	8	7	1	87.5%

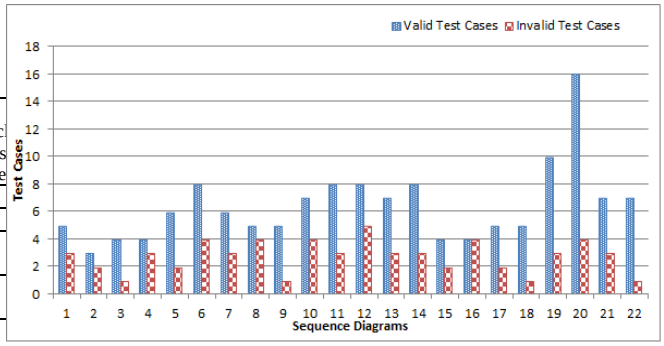

Figure 4 Comparison of valid vs. invalid Test cases

In Table 1 and Table 2, the Case Study Projects are LMS stands for Library Management System, PLMS stands for Personal Library Management System, WBAMS stands for Web Based Automated Manufacturing System, RRTS stands for Road Repair and transport System and RAS stands for Restaurant Automation System.

Table 1 shows the characteristics of the case study projects taken for our experiment. Column 2 and Column 3 shows the case studies and sequence diagrams related to various use cases. Column 4 and 5 shows the objects and the actors involved in that use case. Column 6 and 7 shows the total number of conditions present and total number of alternatives involved respectively. Column 8 and Column 9 deals with the number of asynchronous and synchronous messages present in the sequence diagram.

Table 2 shows the experimental results of our proposed technique. Column 3 shows the total generated test cases by Tcases tool. Column 4 and 5 shows the valid and invalid test cases in the total generated test cases. Column 6 shows the Hit Ratio % using Eq. (1).

Figure 6 shows the comparison between the total number of valid test cases and total number of invalid test cases. It is observed that for twenty two UML sequence diagrams, on an average we have achieved 69.69% of Hit Ratio.

V. COMPARISON WITH EXISTING WORKS

Swain et al. [1] developed a test case generation technique using UML use case and sequence diagram. Their proposed technique is used for system and integration testing. They have generated Use case dependency graph (UDG) using use-case diagram and using sequence diagram. They generated concurrent control flow graph (CCFG). By using these two graphs they have generated test cases. Their testing strategy was based upon predicate coverage. In our proposed approach, we have used sequence diagrams to generate test cases through Tcases for black-box testing.

Fraikin et al. [2] developed the concept for automated testing of OO (object oriented)-programs and also developed a tool called SeDiTeC. It uses sequence diagrams, which are complemented by test case data sets consisting of various parameters and return values for the method calls. It supports specification of various test case data sets for each and every sequence diagram. They have also introduced the concept of combined sequence diagram to reduce the number of sequence diagrams. We have also developed a tool named HRGSD (Hit

Ratio Generation using Sequence Diagram). It is used for test case generation for the black-box testing of the system.

Lei et al. [3] developed a test case generating tool using sequence diagram. Their tool was used to generate test cases for Java test classes of the JUnit framework. This tool automatically creates a test suite based on the test paths of the sequence diagram. Basically, it is based upon the dynamic behavior of the objects. In our proposed approach, we have converted the .xmi code generated through the sequence diagram to .xml format to generate the test cases.

Vadakkumcheril et al. [4] proposed a novel approach of .xml file generation from the UML sequence diagram. They have used BOUML tool (UML diagram design tool) to generate the .xml file. But the file generated by the BOUML tool is not in the exact required format suitable to generate Java program. So they have extracted the meta-elements from the XMI file and using that they have generated a transformed .xml file. The transformed .xml file is used to generate the equivalent Java program for the sequence diagram. In our technique, we have used ArgoUML to design the UML sequence diagram and to generate the .xmi file. Further, the .xml file is used to generate the test cases.

Dutta et at. [5] Proposed a technique to compute Hit Ratio metric for SOA (Service- oriented Architecture) using Tcases. They have developed SOA application using OPENESB tool. Then, they have used open source Jaxb plugin to generate .xml file for the developed SOA application. Finally, by using that .xmi file and Tcases tool, they have generated test cases and measured the hit ratio metric. In our proposed approach we have used sequence diagram to generate black-box test cases and measured the hit ratio metric.

Godboley et al. [7-11] have presented an automated approach to generate test data that helps to achieve an increase in MC/DC coverage of a program under test. Transformation techniques are used for transforming the input program. They have used program code transformation based on Quine-McClusky of sum of product simplification. They have also implement Exclusive-or based Boolean code optimization. This transformed program is inserted into the CREST TOOL to generate test suite and increase the MC/DC coverage. They have also analyzed the time and speed of test case generation.

VI. CONCLUSION AND FUTURE WORK

We have proposed a novel technique to generate Hit Ratio using Sequence diagram. We have experimented for five case study projects consisting of 22 sequence diagrams and on an average we achieved 69.69% of Hit Ratio.

In our future work, we will extend our approach for white box testing, and we will compute different metrics e.g. condition coverage, branch coverage, MCDC coverage etc. We will also try to generate test cases using other UML diagrams e.g. collaboration diagram, state machine diagram etc.

REFERENCES

[1] Swain, S.K., Mohapatra, D.P. and Mall, R., 2010. Test case generation based on use case and sequence diagram. *International Journal of Software Engineering*, 3(2), (pp.21-52).

[2] Fraikin, F. and Leonhardt, T., 2002. SeDiTeC-testing based on sequence diagrams. In the *Proceedings of the 17th IEEE international conference on Automated Software Engineering (ASE)* 2002. Washington, DC, USA, (pp. 261-266).

[3] Lei, Y.C. and Lin, N.W., Test Case Generation Based on Sequence Diagrams.

[4] Vadakkumcheril, T., Mythily, M. and Valarmathi, M.L., 2013. A Simple Implementation of UML Sequence Diagram to Java Code Generation through XML Code. *International Journal of Emerging Technology and Advanced Engineering*, 3, (pp.12).

[5] Dutta, A., Godboley, S., and Mohapatra, D.P., 2016. Measuring Hit Ratio metric for SOA based Application using Black-box testing. *International Conference on Computational Intelligence in data Mining*. (Registered)

[6] Godboley S., 2013. Improved Modified Condition/ Decision Coverage using Code Transformation Techniques. Thesis (M. Tech) NIT Rourkela.

[7] Godboley S., and Mohapatra DP., 2013. Time Analysis of Evaluating Coverage Percentage for C Program using Advanced Program Code Transformer. In Computer Society of India, *7th CSI International Conference on Software Engineering*, Chennai, India, Vol. 11, (pp. 91-97).

[8] Godboley S., Prashanth GS., Mohapatra DP, and Majhi B., 2013. Increase in Modified Condition Decision Coverage using program code transformer. In proceedings of *IEEE 3rd International Advance Computing Conference (IACC)*, Ghaziabad, India, (pp.1400-1407).

[9] Godboley S., Prashanth GS., Mohapatra DP, and Majhi B., 2013. Enhanced modified Condition decision coverage using exclusive-nor code transformer. In proceedings of *International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, Kottayam, Kerala, India, (pp. 524-531).

[10] Godboley S, Mohapatra D.P., Das A., and Mall R., 2016. An Improved Distributed Concolic Testing, *Software: Practices and Experiences*, DOI: 10.1002/spe.2405.