Test Case Generation Technique by using Collaboration UML Diagram

Urooj [*], Anil Pandey
Department Of Computer Science and Engineering, Invertis University, Bareilly, India

Article Info

Article history:
Received 2 January 2014
Received in revised form
10 January 2014
Accepted 20 January 2014
Available online 1 February 2014

Keywords
UML,
Cyclomatic Complexity,
Testing Coverage Criteria,
Collaboration Diagram

Abstract

Software testing is an important activity in the Software Development Life Cycle. To reduce the time and cost of manual testing and to maintain the reliability of the software, various tools and techniques for automation of software testing have been proposed. The automated test cases generation is viewed as a guarantee to carry out effective and maintainable software testing.UML is used to describe the design specification and generate test cases from gathered requirements. Our proposed method generates test cases using UML collaboration diagram. Test Coverage Criteria is also considered in this work. The proposed model detects faults, reduces software development time and effort besides, and increases the quality of generated test cases.

1. Introduction

Software testing is a crucial part of software development process .Testing is a process consists of designing a set of good test cases. Test cases are the set of test inputs, execution conditions, and expected results developed for a particular objective. Test cases generate after completion of design phase so that inconsistency and ambiguity in the design specification can be removed before the beginning of coding part.

Model driven approach have been used for developing the software. In the development of the model-driven software, UML is important source for test case design. It has various diagrams like activity diagram, state chart, sequence, collaboration diagram etc. for describing the dynamic behavior of object in a system. Our work presents an idea of generating test cases automatically from the designing specification by using Collaboration diagram and it covers various coverage criteria such as basis path coverage criteria, branch/decision coverage criteria, statement coverage criteria.UML collaboration diagram (interaction diagram) illustrates the relationship and interaction between software objects. They require use cases, system operation contracts. Collaboration diagrams are a technique for defining external object behavior. They include the same information as Sequence Diagrams (or message trace diagrams) but are better able to show asynchronous message passing. Collaboration diagrams show how objects collaborate by representing objects by icons and their message passing as labeled arrows. It illustrate message being sent between classes and objects (instance).We automatically generate test case from UML specification with the support of the EdrawMax .The related work done in this area is highlighted in the Section 2.Our proposed methodology is outlined in Section3.Section 4 illustrate test coverage criteria . In section 5 we conclude our paper by describing the contribution of the paper and future research direction.

Corresponding Author,
E-mail address: uroojlaiba@gmail.com

2. Related Work

Pakinam N.Boghdady and Nagwa L.Badr[1] proposed test case generation technique based on activity diagram .This technique use a model driven approach. Santosh ei al [3] proposed a system to generate automatic test cases based on sequence and activity diagram combined together. Their work cover three important fault message: sequence fault, operation consistency fault and activity synchronization fault. . Linzhang et al [2] has work on UML activity diagram and generate automatic test cases using Gray box testing approach. His work represents test cases that are generated from high level design model which represent expected structure and behavior of a software under testing [2]. MINGSONG CHEN considers [6] the random generation test cases for Java programs. Author proposed UML Activity diagram based on automatic test case generation for java programs. This approach can be used to check the consistency between the programs execution

ISBN 978-93-5156-328-0

traces and behavior of the activity diagram. Puneet E Patel et al [4] compare two approaches that are used to generate test cases with the help of UML activity diagram. First approach is a novel approach to generate test cases [Debasish kundu and Debasis Samanta 2009], second is Automatic test case generation from UML activity diagram [Yasir Dawood Salman 2010].

3. Proposed Scheme

Our proposed methodology involves the following steps:

1. Design collaboration diagram.
2. Maintain message flow table with the help of collaboration diagram which shows message flow among the objects.
3. Draw the message flow diagram by using message flow table and obtain collaboration graph.
4. Measure the Cyclomatic Complexity by Graph Matrix and find out the number of test cases available.
5. Generate all possible path of generated test cases.

This proposed method is applied on the developed softwares i.e Face Recognition System, Library management, Hospital Management. Proposed Algorithm is used to generate test cases for developed software .It includes validation of the generated test cases during the generation process to ensure their coverage and efficiency. This approach uses in integration as well as regression Testing.

Step: 1. Draw Collaboration Diagram

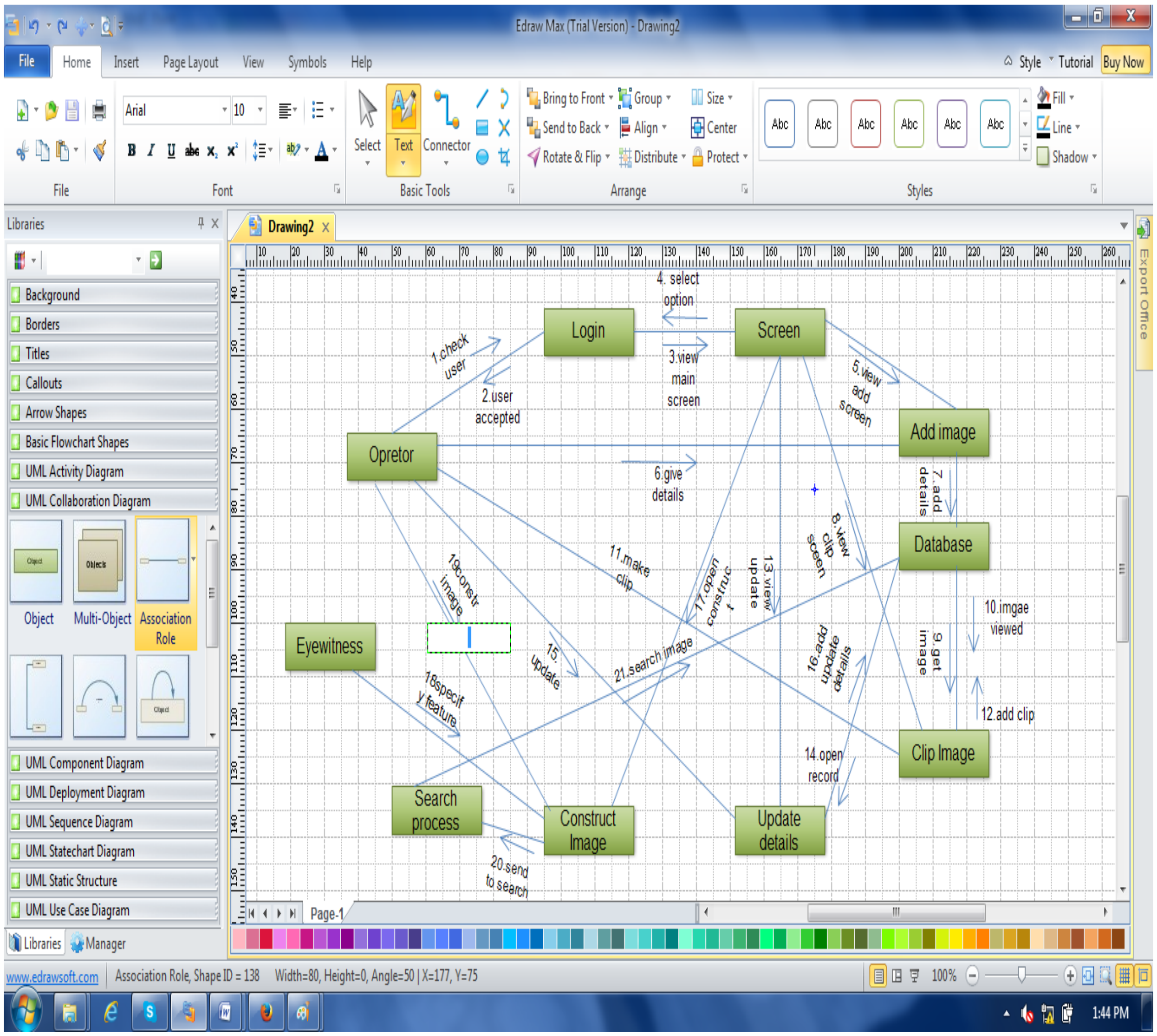

Fig: 1. Collaboration diagram for face recognition system using EdrawMax tool

Symbol	**Object**
A	Operatre
B	Login
C	Screen
D	Add image
E	Database
F	Clip image
G	Update details
H	Construct image
I	Search process
J	Eyewitness

Step: 2. Maintain message flow table with the help of collaboration diagram which shows message flow among the objects

Symbol	**Process Name**	**Controlling Object**
1	Operator	A
2	Check User	A,B,
3	User accepted	A,B
4	View main Screen and Select option	B,C
5	View add Screen	C,D
6	Give details	A,D
7	Add Details	D,E
8	View clip Screen	C,F
9	Get image and Image viewed	E,F
10	Make clips	A,F
11	Add clips to Database	E,F
12	View updates	C,G
13	Open record and Update	A,E,G
14	Add update details to database	E,G
15	Open construct Image	C,H
16	Give instruction(specify features	J,H
17	Construct image	A,H,I
18	Send to search	H ,I,E
19	Search image	I,E

In Message flow table each symbol represents the respective activities and these activities are controlling by the two or more objects. A, B, C, D, E, F, G, H, I, J are the objects.

Step: 3. Draw the message flow Diagram by using message flow table and obtain collaboration graph

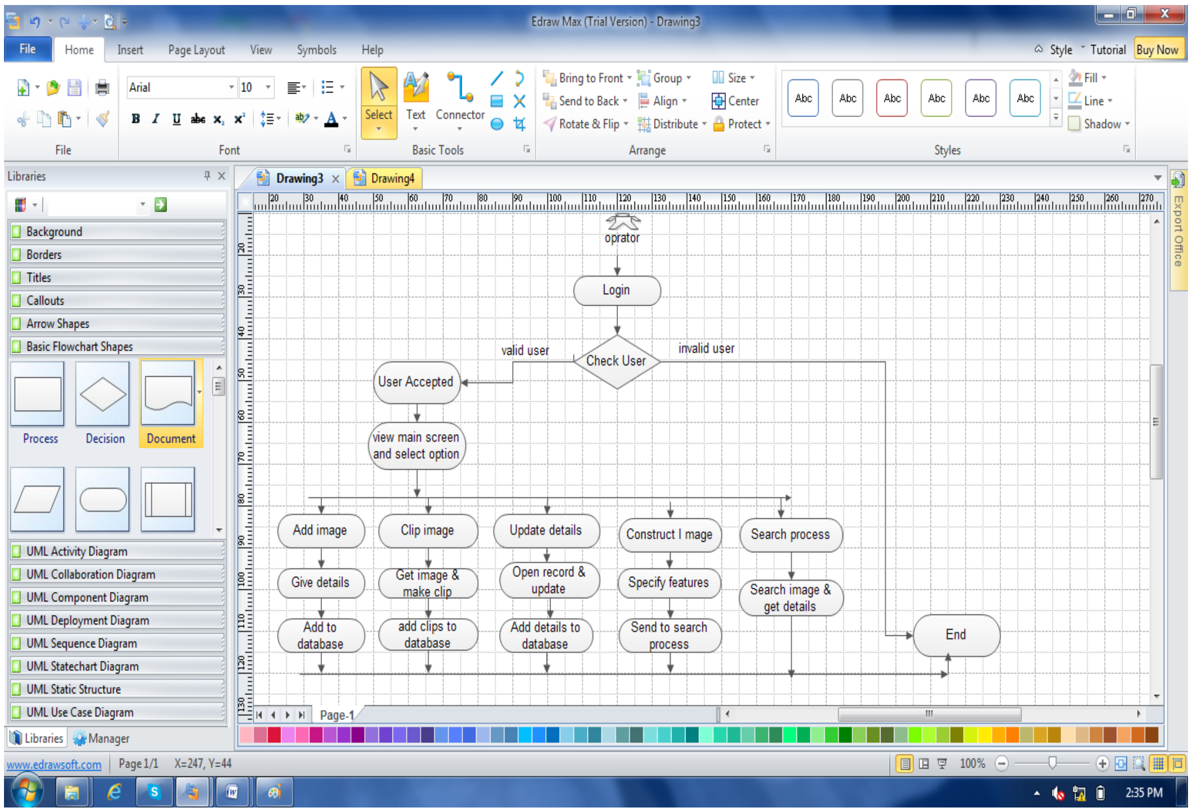

Fig: 2. Message flow Diagram using Edraw Max Tool

Message flow diagram presents flow of messages. It is used to describe the path of messages that will complete the process. This diagram has been drawn with the help of Edraw Max tool, is a tool which is used for drawing the UML diagram.

We can obtain Message Flow Graph or Collaboration Graph with the help of message flow diagram. We can calculate the Cyclomatic complexity of this graph and find number of test cases are generating. It covers various coverage criteria such as branch/decision coverage criteria, Basis path coverage criteria, statement coverage criteria. Branch/decision coverage criteria is one of the criteria in which every branch (decision) taken each way, true and false. It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application. To calculate Branch Coverage, find out the minimum number of paths which will ensure covering of all the edges. By following paths 1-2-3-5-6-7-8-4 maximum numbers of edges are covered but some edges are left. To cover these edges we can follow 1-2-4 path. Hence Branch coverage is 2. The aim is to cover all possible true/false decisions.

Basis path coverage criteria ensure covering of all the paths from start to end.

Flow Graph, Cyclomatic Complexity and Graph Metrics are used to arrive at basis path.

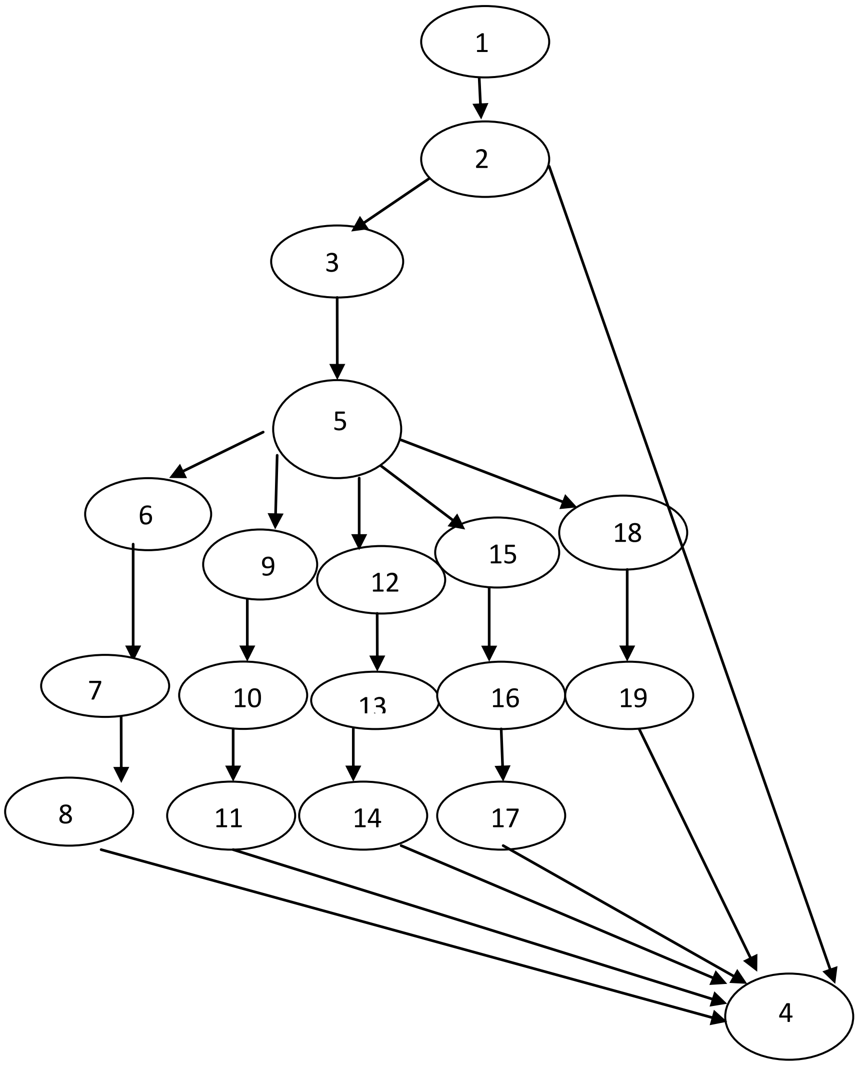

Fig: 3. Collaboration Graph obtain form Message flow Diagram

Step 4: Measure the Cyclomatic Complexity by Graph Matrix and find out the number of test cases available

ISBN 978-93-5156-328-0

Connected to node

Node

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1		1																	
2			1	1															
3					1														
4																			
5						1			1			1			1			1	
6							1												
7								1											
8					1														
9										1									
10											1								
11					1														
12													1						
13														1					
14					1														
15																1			
16																	1		
17																			
18																			1
19																			

1-1=0

2-1=1

0

5-1=4

Total=5

Find Cyclomatic Complexity using GRAPH MATRIX

Cyclomatic complexity= 5+1=6

Step 5: Generate all possible path of generated test cases.

The five test paths are shown as:

Test Path 1	:	**1 – 2 – 4**
Test Path 2	:	**1 - 2 - 3 - 5 -6 – 7- 8 - 4**
Test Path 3	:	**1- 2 – 3- 5 -9- 10 -11 -4**
Test Path 4	:	**1- 2- 3- 5- 12- 13 - 14 - 4**
Test Path 5	:	**1 - 2 - 3- 5- 15 -16- 17- 4**
Test Path 6	:	**1 - 2 – 3- 5 - 18 - 19 – 4**

These are final test cases generated from message flow graph.

4. Coverage Criteria

Test coverage indicates the extent to which a testing criterion such as path testing, branch testing or basis path testing is achieved[1]. A test coverage criterion [8] is a set of rules that guide to decide appropriate elements to be covered to make test case design adequate .We discuss the various test case coverage criteria which is followed by our proposed scheme. Basis path coverage criteria, Branch/decision coverage criteria, statement coverage criteria are followed by our proposed method.

4.1 Basis Path Coverage Criteria

In this the test case is executed in such a way that every path is executed at least once. All possible control paths taken, including all loop paths taken zero, once, and multiple (ideally, maximum) items in path coverage technique, the test cases are prepared based on the logical complexity measure of a procedural design. In this type of testing every statement in the program is guaranteed to be executed at least one time**. Flow Graph, Cyclomatic Complexity and Graph Metrics are used to arrive at basis path.**

4.2 Branch/ Decision Coverage Criteria

Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once. That is, every branch (decision) taken each way, true and false. It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application.

4.3 Statement Coverage Criteria

In this the test case is executed in such a way that every statement of the code is executed at least once. To calculate Statement Coverage, find out the shortest number of paths following

Which all the nodes will be covered. Here by traversing through path 1-2-3-5-6-7-8,1-2-3-5-9-10-11 ,1-2-3-5-12-13-14, 1-2-3-5-15-16-17, 1-2-3-5-18-19 all the nodes are covered. So by traveling through only 5 paths all the nodes are covered, so the Statement coverage in this case is 5.

5. Conclusion and Future work

In this research, we have focus on collaboration diagram of multiple use cases which are related to each other by various relationships such as generalization/specialization. Our approach is applied on Face recognition system, Library management, Hospital Management. Our proposed scheme is generating test cases and finding all possible test path that are essential for discovering faults and reducing software development time . In the future, the algorithm to generate test cases can be generalized so that it can accommodate various test coverage criteria within the same test derivation framework.

References

[1] Puneet E. Patel, Nitin N. Patil, 2013, Testcases Formation using UML Activity Diagram, IEEE 2013.191

[2] Paskinam N. Boghdady, Nagwa L. Badr, Mohamed Hashem, Mohamed F. Tolba, 2011, A Proposed Test CaseGeneration Technique Based on Activity Diagrams, International Journal of Engineering & Technology IJET-IJENS 11(03).

[3] Santosh Kumar Swain, Durga Prasad Mohapatra, Test case generation from Behavioral UML Models, International Journal of Computer Applications, 0975-8887, 6(8), 2010

[4] W. Linzhang, Y. Jiesong et al.:, Generating Test Cases from UML Activity Diagram based on Gray-Box Method, In Proceedings of the 11th Asia-Pacific Software Engineering Conférence (APSEC04), 284-291. IEEE, 2004

[5] M. Chen, X. Qiu, W. Xu, L. Wang, J. Zhao, X. Li, UML Activity Diagram-Based Automatic Test Case Generation for Java Programs, The Computer Journal 52 (2009) 545-556.

[6] D. Kundu, D. Samanta, A Novel Approach to Generate Test Cases from UML Activity Diagrams. Journal of Object Technology, 8(3), 65-83, 2009

[7] A.C. Dias-Neto, R. Subramanyan, M. Vieira, G.H. Travassos, A Survey on Model-based Testing Approaches: A Systematic Review, Proceedings of the 1st ACM international workshop on Empirical assessment of software engineering languages and technologies in conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE), New York, USA, 2007.