International Journal of Computer Applications (0975 – 8887)
Volume 57– No.23, November 2012

Test Case Generation based on Activity Diagram for Mobile Application

Chanda Chouhan
Dept. of Information Technology
InstituteofTechnology&Management

Vivek Shrivastava
Dept. of Information Technology
Institute of Technology & Management

Parminder S Sodhi
Graphics Engineer

ABSTRACT

Testing impacts the popularity of any software and hardware products. If proper testing of any product is done it will increase the cost and sale of the product. Mobiles are widely used electronic device there are several mobile companies established which provide variety of applications and features. It is obvious that people attract on the model that provides new features and comparatively less cost than others. If any problem arises in mobile application, it will affect the selling and impacts bad result. This problem arises when proper testing is not done. To test any product, test cases are used. It shows all possible paths it needs to cover by the software. This paper proposed a TCBAD model for mobile application. TCBAD model generates test cases on the basis of the activity diagram where activity diagrams are used in representing the workflows of stepwise activity and actions with support for choice, iteration and concurrently the complexity will be calculated using Cyclomatic complexity.

General Terms

Unified Modeling Language, Test case Generation, and Cyclomatic Complexity**.**

1. INTRODUCTION

Navigational mobile applications are widely used software. In any mobile device to perform any task user have to switch on various pages or screens, these pages are interconnected and provide functions for end users. The development of mobile application is based on software development life cycle (SDLC). SDLC have six stages: Feasibility Study, Requirement Analysis, Design Phase, Coding, Testing, and Maintenance. Testing is the imperative phase for mobile application for bug detection and test case generation. Client provides information regarding mobile application to Programmer then he generates design document on the basis of pre-described information. This model proposed an approach to check the validity and the performance of activities. ADG (Activity Dependency Graph) is the conceptual diagram form of ADT (Activity Dependency Table), basically on the on the basis of ADG we can find all possible test paths. This work uses Model Based Testing (MBT) which depends on extracting test cases from different models, and quality of test cases depends on how far they would cover all the functionalities in the system under test. The generated test case covers all the branches in the activity diagram, so it uses as branch coverage criteria. Section II all appears on test case generation and activity diagrams in detail. Section III illustrates the proposed model TCBAD and TCBAD Algorithm. Section IV shows experimental evaluation and results. Section V concludes the research work.

2. BACKGROUND WORK

Mobile phone systems have become an avoidable necessity now days. As the requirement of mobile phone system increases the requirement of manufacturing and development of software systems for it also increases. The main aspect of any software is working of every function properly.In the case of mobile system many pages are linked with each other and have some predefined sequence of occurrences. Testing is one of the phases of software development life cycle and contains several phases to accomplish the testing task. Figure 1 shows STLC.

2.1 MODEL BASED TESTING

There are three types of models: Requirement model, usage model and model constructed from source code. Requirement models are specification based models and black box testing are applied on these types of models. The model constructed from source code contains program based testing, also known as white box testing.Model Based Testing (MBT) is a special type of testing strategy that depends upon extracting test cases from different models (Requirement model, usage model and model constructed from source code). MBT have three main key technologies:

- Notation used for the model
- The test generation Diagram
- The tools that generate supporting infrastructure for the tests.

The test cases derived from the behavior model are functional. Test on the same level of abstraction as the model. Many types of model are used to derive test cases, UML model are one of the highly ranked type of model used.UML is the most dominant standard language used in modeling the requirements [2,14] and is considered as an important source of information for test case design. Therefore if it is satisfactorily exploited, it will reduce testing cost and effort and at the same time improve the software quality [2].

2.2 SOFTWARE TESTING LIFE CYCLE

Software testing is one of the phase of SDLC which is used for development of software, all phases of SDLC generates a good quality software, testing is a essential phase of SDLC.The research results on SDLC shows that testing can cover 50%

efforts and also takes credit for popularity and cost. STLC is also accomplished in several phases.

Requirement Study: Testing cycle starts with the study of clients requirements. As requirement is already taken in SDLC phase but to test software, software testing engineer also needs to study the requirements of client. SRS (software requirement specification document) is the output of requirement analysis phase of SDLC; STLC uses this document for requirement study.

Analysis and Planning: After completion of requirement analysis testing objective and coverage is decided, overall scheduling, resources required for process is decided and role and responsibilities are assigned.

Test case design and development: Test case design and development phase covers component identification, test specification design, test specification review.

Test Execution: In this phase performance of testing is evaluated on the basis of code review.

Test Closure: In test closure phase test summary reported project documentation is created. Generated documents are useful for new developers as well as sort out the drawback and errors of the testing and code.

Test Process Analysis: Analysis done on the reports and improving the application's performance by implementing new technology and add features [5].

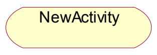
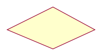
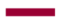
2.3 ACTIVITY DIAGRAMS

Activity diagrams belong from Unified Modeling Language (UML). UML is a standard language which helps customers and developers to develop software and provide various types of diagram for the purpose of specifying the task, visualizing results or models, constructing, documentation, business modeling and communication.UML basically provide notations in the form of diagrams and is capable to show a prototype of model without implementation or coding. The UML diagrams are used in analysis and design phases of SDLC. The selection of UML diagram is based on user's perspective. For clients, Use case diagrams are useful. Designers work with class diagram, analyzers work with activity or state chart diagram, which is a scenario-based approach. Activity Diagrams are used to describe the workflow behavior of the system. In the case of mobile application activity diagram shows behavior and stepwise activities and actions. Notations for the activity diagram are shown in table 1[2].

2.4 TEST CASE GENERATION

Software testing plays an important role to improve the quality of software. The purpose of software testing is not concern only with proper working of software it is also used for quality assurance, reliability estimation. If the software size and complexity of development and logic behind it increases, it takes more time and efforts for testing.

Table 1:Notations for Activity Diagram

SYMBOL	REPRESENTS
	Activity
	Decision
	Initial State
	Final State
	Joint

Test case generation is the most important issue in the research field of software testing. Test case generation can be divided in two main phases, test case design and test case execution, both phases are time consuming. So, automatic generation of test case is widely discussed issue in research field. Test cases show the flow of actions or all possible paths of the system. Test cases are used to describe an input, action or events and an expected response to determine if a feature of an application is working correctly. There are two main approaches to generate test cases automatically [5].

Design test cases from requirement and design specification: Test cases generated from requirement and design specification are basically generated in requirement analysis and specification phase and design phase. Test cases generated in this phase are designed after understanding userrequirement; the designer generates test cases in the absence of a working system, so it is a challenging job and requires lots of imagination and creativity. A good software design must show a series of steps and iterations[6,7].

Test cases generated from Code: Test cases generated from code are an easy task then test case generated from requirement and design specification. Coding is one of thephases of SDLC which comes after feasibility study, Requirement analysis and Design phase. If a software follows the same path as shown in test cases then it is considered that software is developed properly as per users requirement and its costalso increases.Test cases are useful in cost estimation, effort estimation, size estimation and scheduling of software. Coverage criteria are one of the metrics established to check the quality of test cases which are generated from behavior models and requirement and design specification.

2.5 COVERAGE CRITERIA

To check the quality of test cases which are generated by behavior models coverage criteria is used. There are various types of coverage criteria to check the quality of test cases. For example: Branch coverage criteria applied on control flow graphs, where control flow graphs are used to show the overall flow of the graph.It shows the number of branches which are covered by the graph. The second type of criteria is full predicate coverage which is applied on test case generation method by using state chart or communication diagram based techniques. Activity diagrams are used for basic path coverage criteria.

This paper proposed a model TCBAD based on hybrid coverage criteria. Hybrid coverage criteria are the combination of branch coverage criteria, full predicate coverage criteria, basic path coverage criteria and also Cyclomatic complexity criterion. Hybrid criterion can be defined as "a set of activity path P satisfies the hybrid coverage criterion if and only if P contains all start-to-end activity paths in an activity diagram, all predicates or conditions in an activity diagram and all test cases produced by Cyclomatic complexity technique.

a. Cyclomatic Complexity:

CyclomaticComplexity (CC) is used to calculate the minimum number of test cases that should be covered for each activity diagram. Cyclomatic complexity can be computed via using two notations number of edges (E) and number of nodes (N).Figure 1 shows the examples of calculate the Cyclomatic complexity and equation 1 shows the formula to calculate CC [2].

$$\boldsymbol{CC=E-N+2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(1)}$$

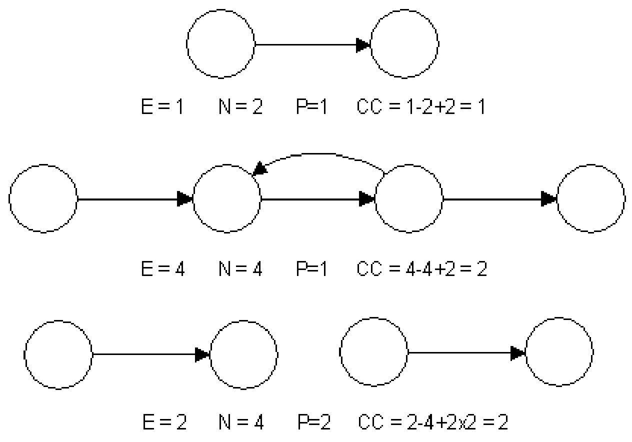

E = 1 N = 2 P=1 CC = 1-2+2 = 1
E = 4 N = 4 P=1 CC = 4-4+2 = 2
E = 2 N = 4 P=2 CC = 2-4+2x2 = 2

Figure 1:Cyclomatic Complexity Calculation Example

3 PROPOSED MODEL

The proposed model studies the activity diagrams as a element in initiation the automated algorithm of generating test cases for navigation mobile application. This model constructs an intermediate table called the Activity Dependency Table (ADT) which contains columns: Activity Name, dependency nodes, in degree value, dependent nodes, out degree values. The produced ADT table automatically generates a directed graph called Activity Dependency Graph (ADG). The ADG is then examined using the Depth First Search (DFS) in order to extract all the possible test cases for mobile applications. The ADT's form makes the ADG cover all the functionalities in the activity diagram. The generated test cases should go through all the branches in the activity diagram. Each activity diagram will be utilized to automatically generate its ADT which is specially designed to contain all necessary details that enable the model to examine all the activity diagram's functionalities and capabilities. The ADT will then be used to automatically generate the ADG. The ADG will be accessed using the DFS to extract all the possible test paths. Therefore, all the details are added to each test path using the ADT to have the final test cases [2]. Each activity diagram should pass through all the four modules to generate at the end a set of highly efficient test cases meeting the hybrid coverage criterion for the mobile application system. Figure 2 shows architecture of TCBAD.Figure 3 shows algorithm for TCBAD.This algorithm is the extended form of Algorithm Generating Test cases Suite. TCBAD proposed to add extra nodes to satisfy the condition for root node and end node, which state that for root node the value of in order is null as well as for end node value of out node is null. This condition can be used for automatic decision of root node and end node.

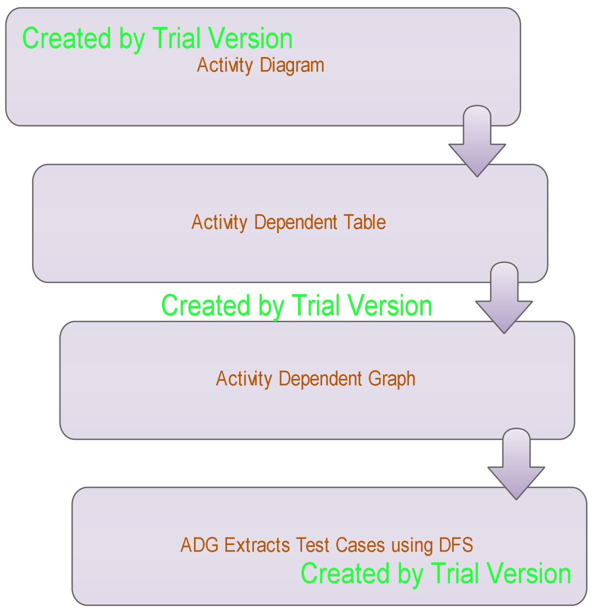


Figure 2: Architecture of TCBAD

Algorithm for TCBAD(Test case generation based on Activity Diagram.)

Input: ADT, ADG.

Output: Test Cases, Test case Table which contains columns test case no., Test Path Node, Node Input, Node expected Output, Test case Input, and Test case Expected Output.

//This algorithm is used to generate automatic test cases using ADG and ADT .Root node is a node whose in degree value is null and End node shows out degree value is null.

STEP 1: Select Root node and End node.//Root node and End node decided on the Basis of in order and out order values.

STEP 2: Mark Root node as visited, and push this node onto stack.

STEP 3: Scan the whole graph and push every adjacent node of root node into the stack.

STEP 4: Repeat STEP 1 to STEP 3 till all paths are covered.

Figure 3:Algorithm of TCBAD

4 EXPERIMENTAL EVALUATION

TCBAD is a model to automatic generates test cases using activity Diagram as well as Activity Dependency Table. In section III, the architecture of TCBAD is shown Figure 3 shows the algorithm for TCBAD. This section shows the experimental analysis of proposed algorithm.

*Activity Diagram:*Figure 4 shows activity diagram for contact field for mobile application. As discussed in section II activity diagrams covers all basic paths for any situation. Figure 4 shows all possible options and paths for user to access and use contact field in any mobile system. The diagram are generated in Rational rose enterprise edition.

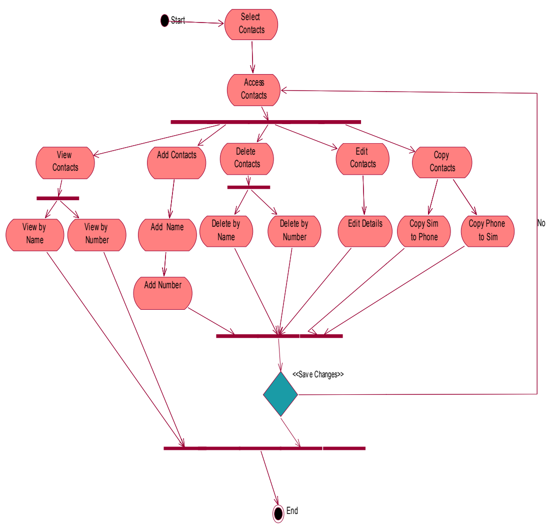


Figure 4: Activity Diagram for Contacts Field

*Activity Dependency Table:*Activity dependency table is created by activity diagram Table 2 shows activity dependency table which contains columns Activity Name, dependency nodes, in degree value, dependent nodes, out degree values.

Activity Dependency Graph: Figure 5 shows ADG (Activity Dependency Graph) which is generated by ADT.

Table 2: Activity Dependency Table

ADT (Activity Dependency Table)						
S NO.	VERTEX	ACTIVITY NAME	DEPENDENCY NODES	IN DEGREE	DEPENDENT NODES	OUT DEGREE
1	R.	Select Contacts	-	-	A	1
2	A.	Access Contacts	R	2	B,C,D,E,F,Q	6
3	B.	View Contacts	P	1	G,H	2
4	C.	Add Contacts	A	1	I	1
5	D.	Delete Contacts	A	1	J,K	2
6	E.	Edit Contacts	A	1	L	1
7	F.	Copy Contacts	A	1	M,N	2
8	G.	View by name	B	1	Q	1
9	H.	View by number	B	1	Q	1
10	I.	Add Name	C	1	O	1
11	J.	Delete by Name	D	1	P	1
12	K.	Delete by Number	D	1	P	1
13	L.	Edit Details	E	1	P	1
14	M.	Copy Sim to Phone	F	1	P	1
15	N.	Copy Phone to Sim	F	1	P	1
16	O.	Add Number	I	1	P	1
17	P.	Save Changes	Q,J,K,L,M,N	6	Q,A	2
18	Q.	End/exit Node	A,G,H,P	4	-	-

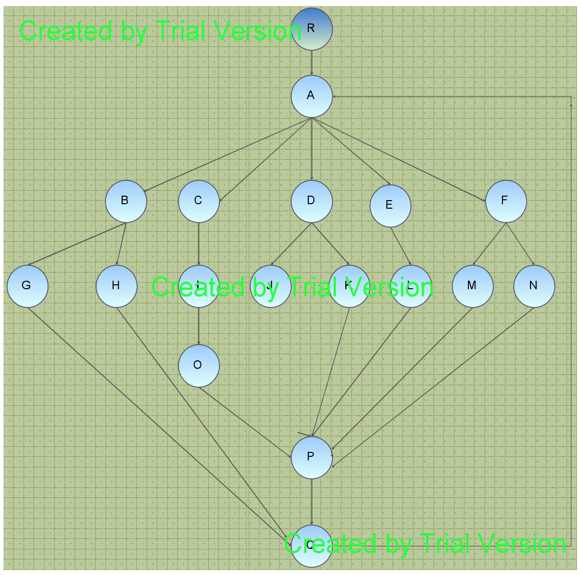


Figure 5: Activity Dependency Graph

Test Path Generation: Table 3 shows automated generated test paths based on TCBAD algorithm which is applied on ADT and ADG.

Table 3: Test Paths from ADT and ADG

TEST PATH NO.	TEST PATHS
1.	R-A-Q
2.	R-A-B-G-Q
3.	R-A-B-H-Q
4.	R-A-C-I-O-P-Q
5.	R-A-C-I-O-P-A-Q
6.	R-A-D-J-P-Q
7.	R-A-D-K-P-A-Q
8.	R-A-E-L-P-Q
9.	R-A-E-L-P-A-Q
10.	R-A-F-M-P-Q
11.	R-A-F-M-P-A-Q
12.	R-A-F-N-P-Q
13.	R-A-F-N-P-A-Q

Cyclomatic Complexity Calculation: Cyclomatic Complexity (CC) is used to calculate the minimum number of test cases that should be covered for each activity diagram. CC can be calculated using equation (1). Table 4 shows the value of CC.

CC=E-N+2

Where E=27

N= 18

Then;

CC=24-18+2 =8

We also know that test cases should also satisfy the condition

Branch Coverage ≤CC≤ Number of Paths

Then,

Branch Coverage= 7

CC=8

Number of Paths=13

7≤8≤ 13

Test Path Nodes	Cyclomatic complexity	Total Test Cases
80	8	72

5. CONCLUSION

Now a day's several mobile companies are established and trying to provide better facilities than others. This type of facilities increases the functionality of mobile application along with complexity. As mobile applications have become an important part of human life, if any application is not running in navigational mobile application, it will decrease the popularity of that model. So, the testing phase in SDLC becomes a crucial part of mobile application designing process.

This work proposes a model for test case generation for navigation mobile application based on activity diagram and complexity will be calculated by Cyclomatic complexity. Activity diagram is one of the famed UML diagram. The proposed model introduces an algorithm that automaticallycreates a table called Activity Dependency Table (ADT) and then uses it to create a directed graph called Activity Dependency Graph (ADG).The ADT isconstructed in a detailed form that makes the generated ADG cover all the functionalities in the activity diagram. Finally the ADG with the ADT is used to generate the final test cases. The proposed model includes validation of the generated test cases during the generation process to ensure their coverage and efficiency. The generated test cases meet a hybrid coverage criterion in addition to their form which enables using them in system, regression as well as integration testing. The proposed model saves time and effort. Besides, it also increases the quality of generated test cases.

6. REFERENCES

[1] Ayan Nigam, Bhawna Nigam , Devendra Kumar Vatsa "Generating all Navigational Test Cases using Cyclomatic Complexity from Design Documents for Mobile Application" International Journal of Computer Applications (0975 – 8887) Volume 40– No.12, February 2012.

[2] 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 IJET-IJENS Vol: 11 No: 03, June 2011.

[3] Santosh Kumar Swain , Durga Prasad Mohapatra, and Rajib Mall "Test Case Generation Based on Use case and Sequence Diagram" Int.J. of Software Engineering, IJSE Vol.3 No.2 July 2010.

[4] Giuseppe Antonio Di Lucca, Anna Rita Fasolino, Francesco Faralli, Ugo De Carlini "Testing Web Applications" Proceedings of the International Conference on Software Maintenance (ICSM.02),Napoli,Italy,2002

[5] Paolo Tonella and Filippo Ricca, "Statistical testing of Web applications" Journal of Software Maintenance and Evolution: Research and Practice, Trento, Italy, 2004

[6] Zhongsheng Qian, Zhongsheng Qian, Hongwei Zeng, "A Practical Web Testing Model for Web Application Testing" Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, Shangai, China, 2005

[7] Martin Sheppard," A critique of cyclomatic complexity as software metric", Software Engineering Journal, England, March, 1988

[8] L. Luo. "Software Testing Techniques, Technology Maturation and Research Strategies", Class Report, Institute for Software Research International, Carnegie Mellon University, Pittsburgh, USA, 2009.

[9] 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.

[10] A.C. Dias-Neto, G.H. Travassos. "Model-based testing approaches selection for software projects", Journal of Information and Software Technology 51 (2009).

[11] S.K. Swain, D.P. Mohapatra, R. Mall. "Test Case Generation Based on Use case and Sequence Diagram", International Journal of Software Engineering, IJSE 3 (2010).