A Systematic Study of Test Case Design in Software Testing

A.Nirmal Kumar[#1], Dr.B.G.Geetha[*2]

[#] Assistant Professor, Department of Computer Science and Engineering, Christian College of Engineering and Technology, Dindigul, Tamilnadu - 624619, India.
[1]sa.nirmalkumar@gmail.com

[*] Professor & Head, Department of Computer Science and Engineering , K S Rangasamy College of Technology, Thiruchengode, Tamilnadu - 637215, India.

Abstract— **In this information era, the softwares are very important in our daily life. This paper presents about how the softwares were made with high quality. The software development life cycle includes various phases, but in which only the software testing can ensure the software quality. So the importance should be given to the designing of test cases which is one of the most important steps in Test Life Cycle. In this research, how the test life cycle should be included with Software Development Life Cycle. The various designing techniques of test cases in software testing are also explained.**

Keywords— **Software Testing, Test Case, Unit Testing.**

I. Introduction

The software engineering consists of various phases like requirement analysis, designing the low level and high level documents, programming, testing, implementation and maintenance. The requirements of the project will be collected from the client. From these requirements, the low level design and high level design documents are prepared. The developers will program for that particular software project. Then the testers will check the quality of the software and ensures that the developed software will meet the client or users requirements. This paper explains about the detailed and systematic study of the test case design in software testing.

II. SOFTWARE TESTING

Due to the demand of high quality software, software developers and testers should be well – educated and trained. Software Testing is the process used for revealing the defects in software. Testing is the group of procedures carried out to evaluate some aspect of a piece of software. The participants in the testing are software customer, software user, software developer, software tester, etc.

A. *Software Testing principles*

Software testing principles are guiding the testers, how to test the software system and provide rules of conduct[1]. Testing is the process of exercising a software component using a selected set of test cases, with the intent of revealing defects and evaluating quality. The test results should be inspected carefully. Test case must contain the expected output and result. The test cases should be developed for both valid and invalid input conditions. Test must be repeatable and reusable. Testing should be planned. The testing activities should be integrated with software life cycle.

B. *Levels of Testing*

In unit testing, the single component is tested. the main goal is to detect the functional and structural defects in the software unit. In integration testing, several components are tested as a group and it is performed to test the component interaction. In system testing, the system as a whole is tested to evaluate the attributes such as usability, reliability and performance and also to check the specifications or requirements. In user acceptance testing, the software organization must show that the software meets all the users requirements.

III. TEST LIFE CYCLE

The test life cycle includes various phases. A tester requires extensive programming experience.

A. *Test planning*

The document that defines the overall testing approach is called Test Plan. It contains test objective, test strategy, test risk analysis, test schedule and resources, roles and responsibilities, communication approaches, test environment automated test tools.

B. *Test case design*

Test case design is the process of selecting the set of input data and then executes the software with the input data under a particular set of conditions [2]. Test case designing contains the various steps like test case id, test case description, test case procedure, test inputs or test data and expected results[5].

C. *Test execution*

In test execution, the actual result is compared with the expected result. The process of executing the test cases is called test execution [4].

D. Test log preparation

After the test execution, the pass or fail information of test cases will be kept in Test Log.

E. Defect tracking

All the failed defect executions or results will come under the defect tracking. It contains defect id, test case id, defect description, defect recovery procedure, defect status, defect severity, defect priority, defect identified by, etc.

F. Test report generation

Finally the test summary report will be prepared and recorded the entire testing activities.

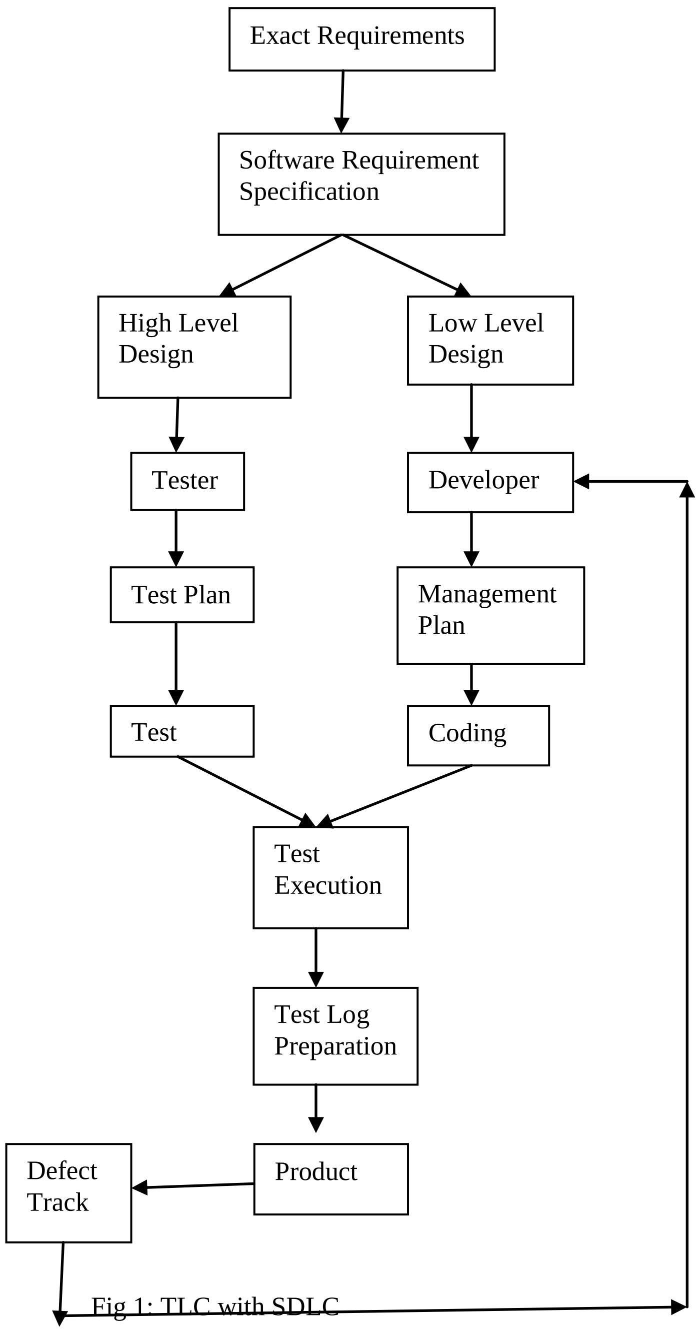


Fig 1: TLC with SDLC

IV. RELATED WORKS

The software testing can be classified into two types namely manual testing and automated testing [3].

Roberto et al. in the year 2010 discovered an architecture based approach for software reliability and testing time allocation [6].

Ammar Masood et al.during the year 2009 proposed their framework for scalable and effective test generation for role-based access control systems [7].

Sebastian et al. in 2009 developed an approach that is used for replaying the differential unit test cases [8].

Myra et al. in 2008 constructed an interaction test suites for highly configurable system by greedy approach [9].

Matthew et al. in 2008 evaluated the test suites and adequacy criteria for distributed systems [10].

Ingo et al. in 2007 describes a general and tool independent architecture for code generators [11].

Wes Masri et al. in 2007 reported the results of an empirical study of several test case filtering techniques [12].

Zheng et al. in 2007 proposed several search algorithms for regression test case prioritization [13].

V. AUTOMATION FRAMEWORK

Test automation framework is designed in such a way that all the basic screens are automated completely. The data to be fed in, for the scripts, is externalized in a spread sheet. The applications under test are completely analysed to create this framework. The application is defined with the static and dynamic screens based on the business requirements.

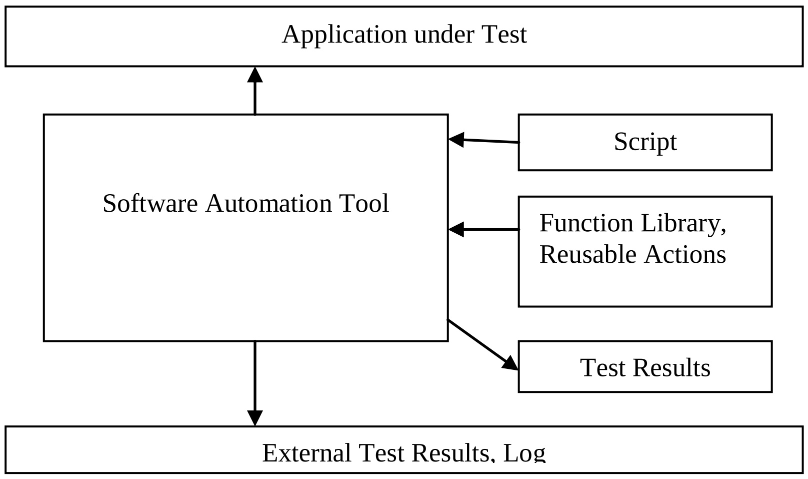


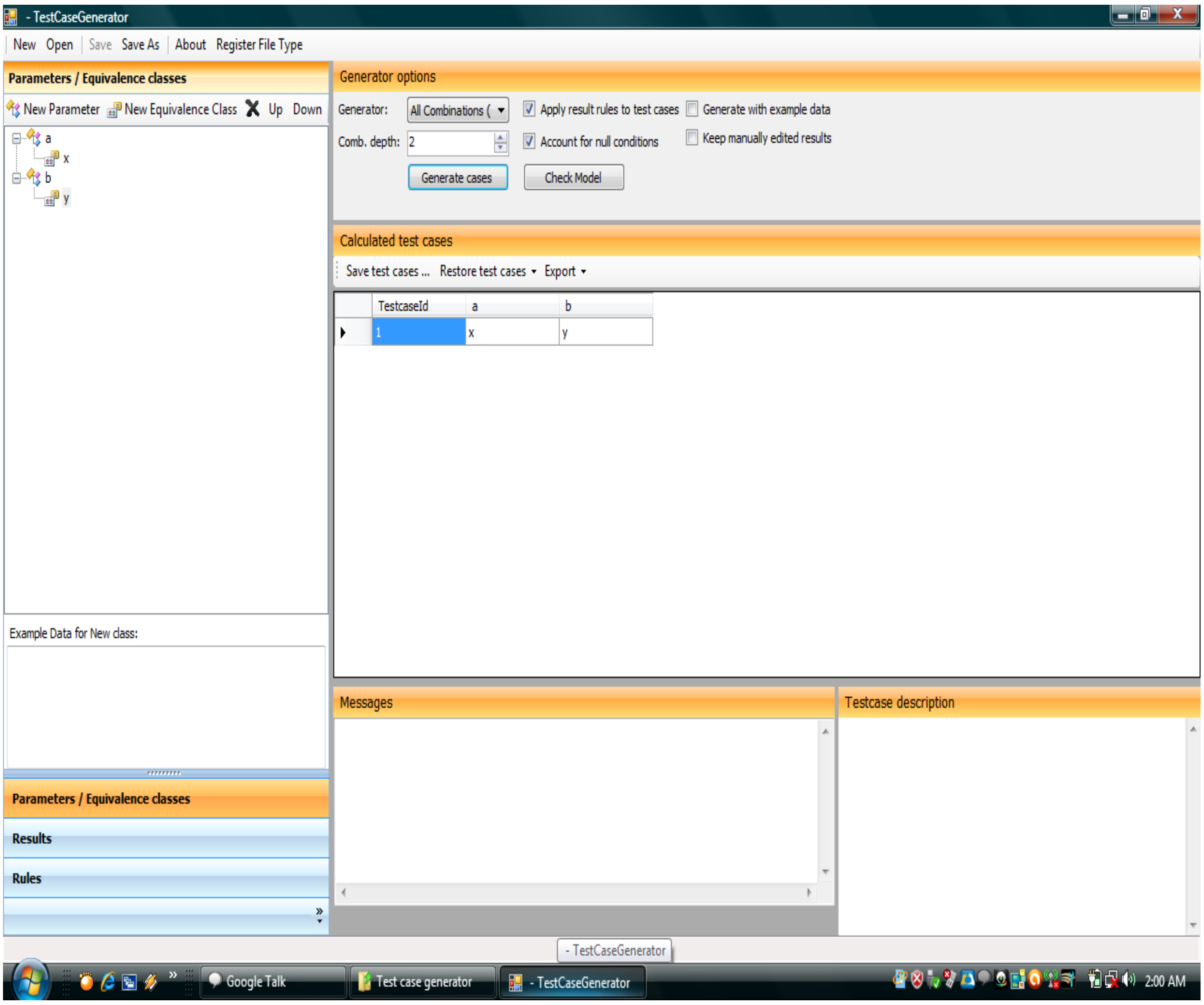
Fig 2: Automation Framework

- Static screens are the pop ups, pre defined error/warning messages etc., that does not change during run time.
- Dynamic screens change during run time depending on user inputs or application process.

The automation framework fundamentally follows the hybrid approach, which offers flexibility to the framework .

VI.RESULTS

The following results shows that the generation of test cases in Test Case generator



- TestCaseGenerator
New Open Save Save As About Register File Type
Parameters / Equivalence classes
New Parameter New Equivalence Class Up Down
Generator options
Generate cases
Check Model
Calculated test cases
Save test cases ... Restore test cases Export
Example Data for New class:
Messages
Testcase description
Results
Rules
Google Talk
Test case generator
2:00 AM

Generator: All Combinations (
Comb. depth: 2
[✓] Apply result rules to test cases
[✓] Account for null conditions
[ ] Generate with example data
[ ] Keep manually edited results

Parameters: a (x), b (y)

TestcaseId	a	b
1	x	y

Fig 3: Test Case Generation

CONCLUSION

The criteria for test completion are that all the planned test that were developed have been executed and passed. Also check that all specified coverage goals have been met. Then the detection of a specific number of defects has been accomplished. Thus the testing activity should be integrated with software development life cycle.

ACKNOWLEDGMENT

Authors are very thankful to the Department of Computer Science and Engineering, KSR College of Technology, Tiruchengode, Tamilnadu, India for every support to carry out this research work.

REFERENCES

1. Poulding, S.; Clark, J.A., "Efficient Software Verification:Statistical Testing Using Automated Search", IEEE Transactions on Software Engineering, Volume: 36 , Issue: 6 , Page(s): 763 – 777, 2010.

2. Ilene Burnstein, "Practical Software Testing", Springer International Edition , 2010.

3. ManjitKaur, "Comparative Study of Automated Testing Tools", International Journal of Computer Applications (0975 – 8887), Volume 24– No.1, June 2011.

4. Mesbah, A.; van Deursen, A.; Roest, D., "Invariant-Based Automatic Testing of Modern Web Applications", IEEE Transactions on Software Engineering, Volume: 38 , Issue: 1 , Page(s): 35 – 53, 2012.

5. Hong Mei; Dan Hao; Lingming Zhang; Lu Zhang; Ji Zhou; Rothermel, G., "A Static Approach to Prioritizing JUnit Test Cases", IEEE Transactions on Software Engineering, Volume: 38 , Issue: 6 , Page(s): 1258 – 1275, 2012.

6. Roberto Pietrantuo, Stefano Russo, Kishore Trivedi, "Software Reliability and Testing Time Allocation: An Architecture-Based Approach", IEEE Transactions on Software Engineering, Volume: 36 , Issue: 3 , Page(s): 323 – 337, 2010.

7. Ammar Masood, Rafae Bhatti, Arif Ghafoor, Aditya Mathur, "Scalable and effective Test generation for role based access control systems", IEEE Transactions on Software Engineering, Volume: 35 , Issue: 5 , Page(s): 654 – 668, 2009.

8. Sebastian Elbaum, Hui Nee Chin, Matthew B. Dwyer, Matthew Jorde, "Carving and Replaying Differential Unit Test Cases from System Test Cases", IEEE Transactions on Software Engineering, Volume: 35 , Issue: 1 , Page(s): 29 – 45, 2009.

9. Myra B. Cohen, Matthew B. Dwyer, Jiangfan Shi, "Constructing Interaction Test Suites for Highly-Configurable Systems in the Presence of Constraints: A Greedy Approach", IEEE Transactions on Software Engineering, Volume: 34 , Issue: 5 , Page(s): 633 – 650, 2008.

10. Matthew J. Rutherford, Antonio Carzaniga, Alexander L. Wolf, "Evaluating Test Suites and Adequacy CriteriaUsing Simulation-Based Models of Distributed Systems", IEEE Transactions on Software Engineering, Volume: 34 , Issue: 4 , Page(s): 652 – 470, 2008.

11. Ingo Sturmer, Mirko Conrad, Heiko Do¨ rr, and Peter Pepper, "Systematic Testing of Model-Based CodeGenerators", IEEE Transactions on Software Engineering, Volume: 33 , Issue: 9 , Page(s): 622 – 634, 2007.

12. Wes Masri, Andy Podgurski, David Leon, "An Empirical Study of Test CaseFiltering Techniques Based on Exercising Information Flows", IEEE Transactions on Software Engineering, Volume: 33 , Issue: 7 , Page(s): 454 – 477, 2007.

13. Zheng Li, Mark Harman, and Robert M. Hierons, "Search Algorithms for Regression Test Case Prioritization", IEEE Transactions on Software Engineering, Volume: 33 , Issue: 4 , Page(s): 225 – 237, 2007.