Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 12, December 2015, pg.224 – 234

AUTOMATED TEST FRAMEWORK FOR SOFTWARE QUALITY ASSURANCE

D. Nirmala[1], T. Latha Maheswari[2]

[1] PG Scholar, Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore, India
[2]Assistant Professor, Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Coimbatore, India
[1] Nirma.Savithri@Gmail.Com; [2] lathamaheswari@skcet.ac.in

Abstract: The overall endeavor of the software engineering is to ensure delivery of high quality software to the end user. To guarantee high quality software, it is required to test software. Testing is a crucial constituent of software engineering. In software testing there are number of underlying issues like effective generation of test cases, prioritisation of test cases which need to be tackled. This automated test framework mainly depends on these four aspects: test strategy, test case generation, test execution and test evaluation. Test strategy is a collection of procedures that determines the testing approach to be followed by the testing team. The test case generation refers to the generation of testcases based on the given application. The test execution briefs about the execution of those tests then comparing the expected result with actual result. The test evaluation investigates the test cases and helps us to generate test summary report and software quality assurance report automatically. The intention of producing this tool is to generate test cases automatically and to decrease the cost of testing in addition to accumulate the time of deriving test cases physically. Hence this system helps to improve overall quality of the software.

Keywords: Software Quality Assurance Report, Software Testing, Test Cases, Test Evaluation, Test Execution, Test Strategy, Test Summary Report.

I. INTRODUCTION

Software testing is the significant part in software development life cycle as well as it plays a decisive part in assuring software quality. Test automation makes use of particular software to control the execution of tests and to compare the actual result with predicted result. Deciding right time to go for automation, defining scope for automation and selecting the right tool for automation are the important decisions in which the testing team must formulate in the test plan. Specifying the exact details of the product for automation highly determines the victory of the automation. The effectiveness of this verification and validation method depends upon the number of bugs identified and fixed before releasing the system [1]. It depends upon the quality of test cases generated. The most significant issue in the field of software testing research is the generation of the test cases based on automation .To slice down the cost of manual testing and to increase consistency of the testing, it is essential to automate the test case generation [1]. Depending upon these testcases, the test summary report and software quality assurance report will also be automated. Test Summary Report is a significant deliverable which is prepared at the end of a testing, or rather after testing is completed. The main objective of this test summary report is to clarify different details and activities about the testing performed. Software quality assurance report helps to evaluate the quality of a product and accomplish adherence to software product standards and procedures. It is a sunshade activity that assures accordance to standards and procedures throughout the Software development life cycle of a software product.

II. PROPOSED SYSTEM

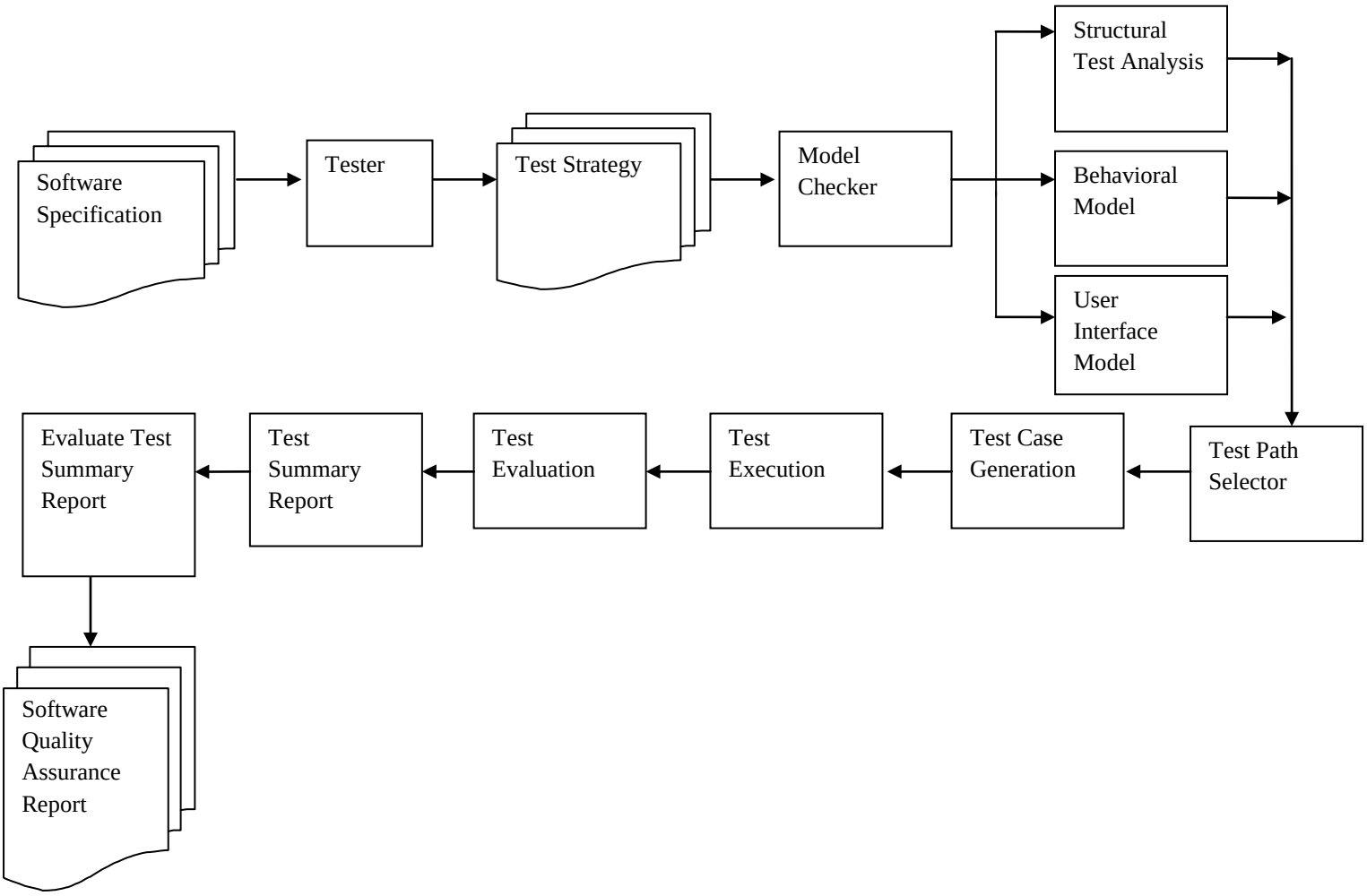
The automation test framework is an execution environment for computerized tests. It is an incorporated structure that sets the convention of automation for a particular product. A framework is a constructive blend of various strategies, programming standards, perceptions, methods, conventions, system hierarchies, modularity, coverage mechanism and test data injections. These factors act as tiny structural sections which need to be gathered to represent an industry process .This framework provides the customer with diverse benefits that helps them to develop, perform and testimony the automation test scripts efficiently. These automated tests can run quickly and periodically, which is worthwhile for software products with a extended service life. This system helps to organize the test suites and in turn helps to improve the efficiency of testing. A structured test framework helps in eliminating the duplication of test cases which is automated across the application.

A testing framework is always independent of application and it can be used with any application nevertheless of issues (like components, stack, structural design etc.) of application under test. Newly generated test cases are constantly added to existing automation in corresponding to the progress of the software development. It is important to be conscious that overall coverage of all tests by means of test automation is not viable. While deciding what tests needs to be automated foremost, hence cost and effort are required to be considered. Test cases which contain high cost and low effort should be automated early. Then test cases with common use, changes, and past errors in addition to test cases with low to moderate effort will be further automated. Test automation pacifies tester's annoyance and allows the test execution without user contact while guaranteeing repeatability and accuracy. Rather testers can now focus more on complicated test scenarios. Arbitrarily generated tests can find defects with high testability. This Test automation framework allows us to perform different types of testing efficiently and effectively.

This Testing framework is responsible for:

- Specifying the pattern in which to articulate expectations.
- Building a method to hook into or drive the application under test.
- Perform the tests.
- Testify the results.

Fig1: Test Framework for Software Quality Assurance- Architecture



III. TEST STRATEGY

A test strategy is a sketch that describes the testing approach for the software development life cycle. This document removes all ambiguities or blurred requirement statements with an obvious sketch for achieving the test objectives [2]. It includes testing objective, type of testing that has to be carried out, total time, resources required, features and functions to be tested, risk analysis, defect resolution, process improvement and test environment. This testing strategy provides the framework that is required

to implement our testing methodology. A distinct test strategy should be developed for each system based on the specific application. Test strategy is the procedure that describes how the testing objectives would be met efficiently with the assist of the available resources. Thus test strategy provides lucidity on the testing approach to the project stakeholders.

A detailed test strategy should cover up the following:

Scope

Restate the objective of the application and identify the possibility of testing. It contains the information like who will assess and endorse this document. Define the testing activities and stages to be carried out with schedule as for overall project timelines defined test plan.

Test Objectives

Application to be tested should be measured by its conformity to the specifications and the customer approval criteria. Therefore these specifications and approval criteria's have got to be generalized to particular test plans that verifies and measures the predictable outcomes for each test to be performed [5]. The objectives should be categorized based on their importance and their risk.

Features and Functions to be tested

All the features and functions must be scheduled for test enclosure or omission, along with a portrayal of the exceptions. Some features might not be testable due to the absence of hardware or lack of control [8]. The list should be classified based on the functional area to adjoin clarity.

Testing Approach

This approach describes the details essential to describe the stages and categories of testing. Describe why it should be performed along with details like when to start testing, test holder, tasks, testing approach and particulars of automation strategy and tool if it is pertinent.

Test Environment

Test environment setup must summarize the details about number of environments and essential setup for all environments. Test Environment comprises of components that sustain test execution with software, hardware and network configured. Test environment configuration must characterize the production environment in order to reveal any environment/configuration related issues.

Process Improvement

The entire testing process must be paying attention on process improvement. This test strategy has to catalog the ways to monitor progress and provide regular opinion. This opinion can assist to improve the process, deliverables and metrics used in testing.

Deliverables

Entire list of deliverables should be defined and their location must be specified. The deliverables are test cases, test data, test summary report and software quality assurance report.

Schedule

Each and every testing activity should be pooled into a solitary master testing schedule. This schedule must include estimated time for each task. Testing resources must be assigned for each task and quality gates must be listed to assure oversight to entire methods.

IV. TEST CASE GENERATION

In general there are several reasons for automated test case generation in software testing. Some of the most significant reasons are as follows.

Lowering the cost of software testing

Throughout the testing phase cost can amplify further than the expected price due to unfortunate test cases [1]. These unfortunate test cases can be the reason for the depletion of organizational assets as well as time. So it is required to minimize the cost for getting an acceptable product.

Lowering the human errors

So as to find out whether a test case is valid or not there is no specific method. On the whole it depends on the how the testers understand the requirements. To avoid human errors the tester's fundamental skill level has been taken into consideration [1]. It leads to the enclosure of bugs in the system subsequent to testing. To prevail over this difficulty, automatic test case generation part must be considered.

Enhancing software products quality

It is usually fixed that manual testing is becoming a hindrance and it is a common cause for project delays particularly for outsized programs [3]. As a result, these automatic test case generations have turned out to be a key to guarantee the quality of immense software products.

Downsizing the number of test cases

Generating effective test cases are the vital identification to simplify the test work and to enhance the test efficiency [1]. The test work is ineffective because of the huge quantity of the preliminary test cases, so some automation algorithms are required to upgrade the test cases.

Casing all the system requirements

This automated test case generation provides a way to guarantee that test cases have been derived in a reliable and objective style so that all system requirements have been enclosed [2].

V. PROPOSED AUTOMATED TEST FRAMEWORK MODEL

This proposed automated test framework is based on these three models are:

Structural Test Analysis Model

It is a method of testing the system at the intensity of the source program and used to fabricate testcases accordingly. This model is used to test software's interior structures or mechanism of an application, as divergent to its functionality. It observes the programming structure and derives test data from the source code [8]. These automated test cases are based on the coverage's of statements, path, condition and functions**.**

Behavioral Model

This model mainly focuses on the requirements of the system. It is a technique that scrutinizes the functionality of a system based on these specifications [2]. This model is used to test the application contingent on the requirements specification and used to generate the testcases involuntarily. It is used to address the stated requirements as well as the implied requirements [5]. It encompasses the end user perspectives.

User Interface Model

This model is mainly designed to perform tests on interfaces. It is performed to assess whether systems or components pass data and control properly to one another. It is to make sure if all the connections among these components are functioning appropriately and inaccuracies are handled properly [4]. Based on the user interface of the application it is used to carry out the testing process and then testcases are automated consequently. These models help in producing efficient automated testcases when compared with testcases which are created manually.

Test Generation Technique Algorithms

Model Generation Algorithm

Input: Group of Model Segments S, Meta-Model MM
Output: Group of Models L matched to MM

1. If there are unmasked Model Segments in S do
2. { construct an vacant model M
3. If the model size boundary is not extended (1) and M still can
4. amplify do
5. { select an unmasked model segment MS in S
6. for all object segment OS in MS do
7. { search an object O which is occurrence of the class partly specified by OS (3)
8. for each condition CT defined in OF on the attribute A do
9. { if A is an attribute (value division case) then

10. select a value and place it to P in O (5)
11. else (multiplicity division case)
12. { select a cardinality N following to CT (5)
13. if the category of A is a class then
14. discover N objects with a P category and put them to A in O (3)
15. else uncover N values in the division of A and locate them to A in O (5)
16. }}
17. append O to M
18. end of M until it is conformed to MM (2, 4)
19. }}
20. spot MS as enclosed
21. append M to L}

Test Generation Algorithm

Input: Test Generation group of Test Case Segments TS, categories C, Managed Meta-Model MMM

Output: Group of Models L matched to MM

1. If there are unmasked Model Segments in S do
2. { construct an vacant model M
3. If the model size boundary is not extended (1) and M still can
4. amplify do
5. { select an unmasked model segment MS in S
6. for all object segment OS in MS do
7. { search an object O which is occurrence of the class partly specified by OS (3)
8. for each condition CT defined in OF on the attribute A do
9. { if A is an attribute (value division case) then
10. select a value and place it to P in O (5)
11. else (multiplicity division case)
12. { select a cardinality N following to CT (5)
13. if the category of A is a class then
14. discover N objects with a P category and put them to A in O (3)
15. else uncover N values in the division of A and locate them to A in O (5)
16. add test generation based on category C,
17. Add Test generation set items , TS
18. Generate report.
19.}}}}}

Pre-Requisites

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace Jewellary_shop_systemDemo_Product
{
   public partial class AddNewSchemeDetails : Form
   {
        SqlConnection con = new SqlConnection(@"Data Source=ADMIN-PC;Initial Catalog=Test_Case;Integrated Security=True");
```

Algorithm For Generating Test Summary

```
1. string comma = ",";
2. string desc1 = "To Test the ";      //Get the description
3. string field1 = label7.Text;        //Assign the description
4. string desc3 = " correct or not";   //Get the description
5. string finaldesc = desc1 + field1 + comma + field2 + comma + field3 + comma + field4 +          comma  +  field5  +  comma  +  field6 + desc3;
6. string tdata1 = "The given values ";      //Get the test data
7. string data1 = textBox1.Text;             //Assign the test data
8. string tdata2 = " are checked correct or not";     //Get the test data
9. string funaltdata = tdata1 + data1 + comma + daat2 + comma + daat3 + comma + daat4 +          comma + daat5 + comma + daat6 + tdata2;
10. Random r = new Random();       //To generate the test case ID
11. int n = r.Next(1, 3);          //Assign the test case ID
12. if (n == 1)                    // To check the test case ID
13. {
14. con.Open();                                  //Open the database connection
```



```
15. SqlCommand cmd1 = new SqlCommand("insert into UnitTest values('1','107','New Scheme','" + finaldesc + "','" + funaltdata + "','The given scheme values are stored into database','The given scheme values are stored into database','None','None','High','Pass')", con);
16. cmd1.ExecuteNonQuery();                    // Execute Insert Query
17. con.Close();                               //Close the database connection
18. }
```

VI. TEST EXECUTION

In this paper test execution is the method of executing the system and to examine the expected and actual results. Throughout this stage the testing group will perform the testing based on the test strategy and test cases which have been automated. If the expected outcome is not met, it then it is a defect. Defects will be revealed back to the development group for amendment and retesting will be done [7]. Test execution focuses on, the test steps for the application under test, provides the test data and monitors the behavior of the application under test to verify whether it satisfies the expected outcome or not [3]. In this paper efficiency of testing is increased by using this framework.

Following factors are to be considered for a test execution process:

- Carry out the tests as per the test strategy.
- Depending on the risk, opt for a detachment of test suite to be executed for this sequence.
- Perform tests, describe the bugs and capture test status constantly.
- Determine the blocking issues if it has been occurred.
- Dispatch the test cycle results and status.
- Trace the defects to termination.

The generated test case report must contain the following:

Test Suite ID	The identification number of the test suite to which this test case belongs.
Test Case ID	The identification number of the test case.
Test Case Summary	The outline / goal of the test case.
Related Requirement	The credentials of the specification this test case relates/traces to.
Prerequisites	Any rudiments or preconditions that should be satisfied earlier to executing the test.
Test Procedure	Step-by-step process to perform the test.
Test Data	The test data, or associations to the test data, that are used while performing the test.
Expected Result	The predictable outcome of the test.
Actual Result	The tangible outcome of the test to be filled subsequent to executing the test.
Status	Pass or Fail. Other statuses can be ‘Un Executed’ if testing is not performed and ‘Blocked’ if testing is obstructed.

Remarks	Several comments on the test case or test execution.
Created By	The name of the creator of the test case.
Date of Creation	The date of formation of the test case.
Executed By	The name of the person who performed the test.
Date of Execution	The date of carrying out the test.
Test Environment	The environment (Equipments/Groupware/Structure) into which the test was executed.

VII. TEST EVALUATION

The Test evaluation collects, organizes, and presents the test results and key measures of test to facilitate intent quality evaluation and assessment. It also presents an intervening evaluation from the testing group, representing their consideration of the software against the evaluation assignment and analogous recommendations for the next testing efforts required [6]. Besides, this test evaluation it might enclose a common testimonial of virtual quality and provides recommendations for upcoming test effort. This evaluation is done based on the test coverage on requirements, code, risks, security, privacy and compliance. In this paper it is evaluated using the automated test cases results. Based on this test evaluation, the test summary report and software quality assurance report will also be generated automatically.

Test Summary Report

Test summary report contains the overall status of testing process. This report summarizes about the number of modules tested, number of test cases executed, number failed and passed testcases, number of bugs, status of those bugs, number of resources, types of testing carried out. This report helps us to identify and rectify the bugs to increase the quality of a product.

Software Quality Assurance Report

Quality Assurance encompasses the complete software development process, which incorporates processes such as software design, programming, source program control, code reviews, configuration and release management. While software quality control is a mechanism to control the products, software quality assurance is a mechanism to control the processes. This report is used to ensure the quality in these four aspects they are:

Requirements Quality

The precision, comprehensiveness and stability of the requirements model will encompass a sturdy influence on the quality of all work products.

Design Quality

All the components of design must be examined by the software team to make sure that it exhibits high quality as well as the design itself complies with the specifications.

Code Quality

Source program with associated work products (e.g., other descriptive information) should conform to confined programming standards and reveal the uniqueness that will smooth the progress of maintainability.

Quality Control Effectiveness

A software team must apply narrow resources in a way to facilitate the utmost possibility of achieving a high quality product.

VIII. CONCLUSION AND FUTURE WORK

In this paper the automated test framework generates the testcases automatically and it evaluates those testcases and produces the test summary report and software quality assurance report based on the automation**.** This framework can be applied to perform both functional and non functional testing but all types of non functional testing cannot be performed. So in future we need to concentrate on those non functional testing types (like load testing, stress testing, scalability testing) which have not been implemented in present.

REFERENCES

[1] Mohammad Reza Keyvanpour, Hajar Homayouni and Hossein Shirazee "Automatic Software Test Case Generation: An Analytical Classification Framework" at International Journal of Software Engineering and Its Applications Vol. 6, No. 4, October, 2012.

[2] Rosziati Ibrahim, Mohd Zainuri Saringat, Noraini Ibrahim and Noraida Ismail "An Automatic Tool for Generating Test Cases from the System's Requirements" at Seventh International Conference on Computer and Information Technology.

[3] Mr. Navnath Shete, Mr. Avinash Jadhav "An Empirical Study of Test Cases in Software Testing" at ICICES2014 - S.A.Engineering College, Chennai, Tamil Nadu, India.

[4] Isabella And Emi Retna "Study Paper On Test Case Generation For GUI Based Testing" at International Journal Of Software Engineering & Applications, Vol.3, No.1, January 2012.

[5] Nicha Kosindrdecha and Jirapun Daengdej "A Test Case Generation Technique And Process".

[6] Itti Hooda and Rajender Chhillar "A Review: Study of Test Case Generation Techniques" at International Journal of Computer Applications (0975 – 8887) Volume 107 – No. 16, December 2014.

[7] Karambir and Kuldeep Kaur "Survey of Software Test Case Generation Techniques" at International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 6, June 2013.

[8] Steven P. Reiss "Towards Creating Test Cases Using Code Search" at IEEE International Conference on Software Maintenance and Evolution 2014.

