ISSN(Online) : 2456-8910

International Journal of Innovative Research in Applied Sciences and Engineering (IJIRASE)
Volume 1, Issue 2, DOI: 10.29027/IJIRASE.v1.i2.2017.44-50, August 2017

New Frameworks for Automated Test and Retest (ATRT) Test Case Requirement Traceability Matrix

R.Padmavathi
Research Scholar D.B.Jain College
Autonomous Chennai-97
elangotesting@gmail.com

P.Saravanan
Asst Professor Department of Computer Science,
D.B.Jain College Autonomous Chennai-97
npsindian@yahoo.co.in

ABSTRACT- In the software testing domain the research paper implementing automated test and retest (ATRT) is one of the an Innovative technical solution provider that solves the uniqueness testing problem the ATRT is one of the numerous automated testing tools. The ATRT is a one of the way that handle are related test cases and test management activities, the test case processing. The processing is a hierarchical break down of test cases into test scenario and test steps and test sub steps the user to import their requirement and map test cases to requirements. In the research Framework successfully built by reusing the test cases in the research paper implementing continuous improvements and continuous integration. The integration merging and testing, test case requirements through the frequent intervals applying in this method prevent integration problem and increasing quality control and improved technical performance, decrease cost and increasing speed. The automated testing using matrix technology accessing and improving customer requirements the automated testing process. The process and tracking its status and matrix is a standard of measurement, a metrics can be performed at as a measure can utilized and implementing to display past and present performance and predicting future performance.

Keywords: Software testing, ATRT, mapping test cases, Requirements, Test coverage

1. Introduction

ATRT automated testing and retesting is an automated tool the tool capability that can be applied across the entire system testing lifecycle, the testing result in broader test coverage, to increased efficiency and improved quality. It supports the functional, interface, longevity and performance testing of legacy systems, systems being modernized and new systems under development. This patented technology has transformed the

development of many complex systems; Regression testing is a type of software testing that involves retesting software after changes or enhancements have been introduced to a system. As a result, one of the goals of regression testing is to assess whether functionality previously delivered continues to operate as expected some changes. The most common technique used to meet regression testing. The testing goals are through assessing requirements previously verified. Automated regression testing applies an automated test strategy to the regression testing effort and efficiency. The automated test strategy was to apply Automated Test and Retest (ATRT) technology to support the project's need to expand their regression test coverage from approximately 15% to more than 92% while at the same time not increasing the time allocated for regression testing. The current manual test doing regression tests were conducted by stimulating the software by injecting signals into the system and then observing the operator screens to check for status and alarms. The change would be sent to the pressure gauge by lowering the pressure below a certain threshold. Then actual output values would be observed on the operator console to determine if the software had correctly detected the change and sent the necessary "low pressure" alarm. After reviewing the nature of the manual tests, IDT adopted that the application of ATRT: Test Manager would facilitate meeting the project's objectives. The Test Manager is designed to support large, complex systems of this type and provides the capability to automate operator inputs/actions and verify the responses.

1.1 Traceability Matrix

In the research focus of any testing engagement is crossed should be maximum test coverage to coverage, it simply means that we need to test everything there is to be tested. The aim of any testing project should be 100% test coverage. Requirements Traceability Matrix to begin with, establishes a way to make sure the project place checks on the coverage aspect. It helps in creating a snap shot to identify coverage gaps.

1.2 Test Automation

Automated testing tools are capable of executing tests, reporting outcomes and comparing results with earlier test runs. Tests carried out with these tools can be run continuously, at any time of day to day. The method or process being used to implement automation is called a test automation framework. Vastly

Increases In the test coverage Automated software testing Can increase the depth and scope of tests to help Improve software quality.

Lengthy tests that are often avoided during manual testing can be run unattended. They can even be run on multiple computers with different configurations.

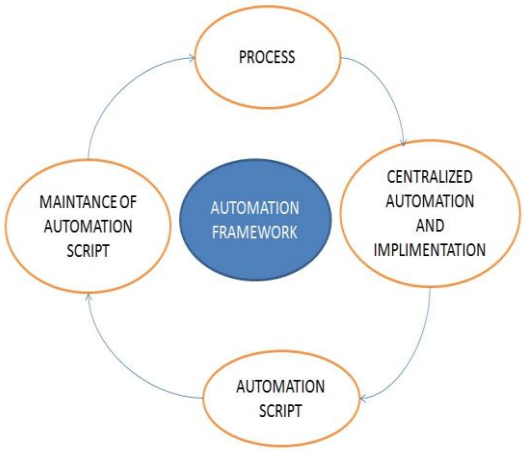

Fig-1 Automation Testing Framework

The test automation framework is defined as a set of assumptions concepts and practices that constitute a work plat form or support for automation testing

2. RELATED WORK

In the Author Blackwell, Barry Mark, et al. [1] describes in the paper The testing tool includes an Automated Multidimensional Traceability Matrix system for determining linkages between interrelated system components, a means for identifying a change in one or more of the interrelated system components, a means for applying the Automated Multidimensional Traceability Matrix, a means for executing all of or a subset of the test scenarios associated with the interrelated system components that may be affected by the change and a means for evaluating the results of the executed test scenarios.

The author discussed in the paper [2] Oliveto, Rocco, et al. The analysis is based on Principal Component Analysis and on the analysis of the overlap of the set of candidate links provided by each method. The studied techniques are the Jensen-Shannon (JS) method, Vector Space Model (VSM), Latent Semantic Indexing (LSI), and Latent Dirichlet Allocation (LDA). The results show that while JS, VSM, and LSI are almost equivalent, LDA is able to capture a dimension unique to the set of techniques which we considered.

In the author discussed in the paper [10] Ståhl, Daniel, Kristofer Hallén, and Jan Bosch.etl.. "Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework." *Empirical Software Engineering* (2017) This paper presents, investigates and discusses Eiffel, an industry developed solution designed to provide real time traceability in continuous integration and delivery. The traceability

needs of industry professionals are also investigated through interviews, providing context to that solution. It is then validated through further interviews, a comparison with previous traceability methods and a review of literature. It is found to address the identified traceability needs and found in some cases to reduce traceability data acquisition times from days to minutes, while at the same time alternatives offering comparable functionality are lacking. In this work, traceability is shown not only to be an important concern to engineers, but also regarded as a prerequisite to successful large scale continuous integration and delivery. At the same time, promising developments in technical infrastructure are documented and clear differences in traceability mindset between separate industry projects is revealed.

In the author Warfield, Robert W discussed in the paper [11]. "Automatic software testing tool." U.S. Patent No. 5,754,760. 19 May 1998. The software module has a number of possible states. A set of state machines is created which represent a definition for either a user interface or an application program interface (API) of the software module in terms of the possible states. From the state machines, a set of test cases is automatically generated, such that each test case consists of code for manipulating the user interface or API. A genetic algorithm creates populations of test scripts from the test cases, in which each test script includes a number of test cases. Each test script from each successive generation of test scripts is executed by applying the script as input to the software module. A code coverage analyzer provides a measure of code coverage to the genetic algorithm for each test script. The genetic algorithm uses the measure of code coverage as a fitness value in generating future populations of test scripts and in determining a best script.

AUTOMATION TESTING, TESTCASE REQUIREMENT, TRASABLITY MATRIX FRAMEWORK

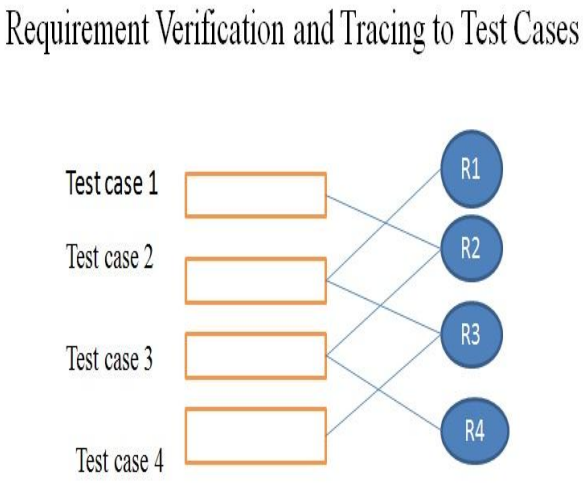

Fig-2 Traceability Matrix

The unexpected and unplanned changes as well as the fact the continuous maintained and growth the software Requirement

ISSN(Online) : 2456-8910

International Journal of Innovative Research in Applied Sciences and Engineering (IJIRASE)
Volume 1, Issue 2, DOI: 10.29027/IJIRASE.v1.i2.2017.44-50, August 2017

Management is to handle traceability the main purpose to implement Bi-directional trace between requirements and components

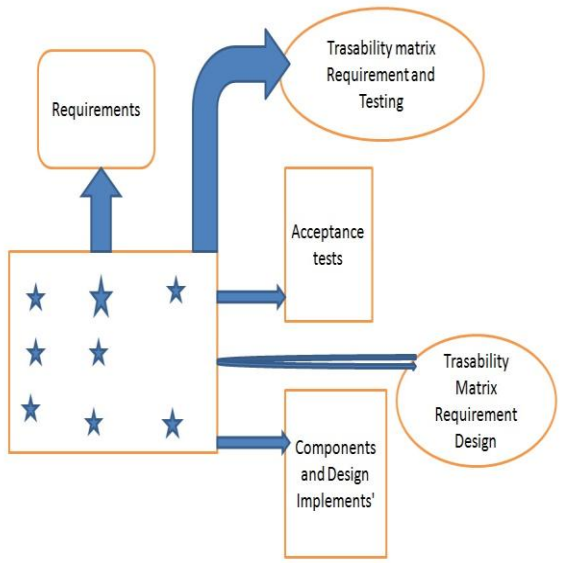

FIG-3

Example Trasability Matrix Test cases and Requirement Test cases

Reqs ID	Req s ID	Use case 1.1	Use case 1.2	Use case 1.3	Use case 1.4	Use case 1.5
Test cases	250	2	2	3	2	2
1.1	2	X		X		X
1.2	2		X	X		
1.3	2	X			X	
1.4	2		X			X
1.5	2			X	X	

FIG-4

The above Table like needed if the right tools and automated tests automated acceptance tests in traceability the requirement traceability tool have used in the above diagram this also positive effects on maintainability of our code acceptance tests and requirement specification

The research work using automation tool in test rail to get all the test cases of type Automation (Type_id =3) for that work

```
JArray response = (JArray)
api.SendGet("get_cases/" +
projectId + "/&type_id=3");
```

I then create an array with all the test case ids and then pass that to the create test run API call. ("add_run/").

```
        var data = new
Dictionary<string, object>
        {
            { "suite_id", 1 },
            { "name", "Test
Run From Framework" },
            { "include_all",
false },
            { "case_ids",
testCaseIds }
        };

        JObject
testRailResponse =
(JObject)api.SendPost("add_run
/" + projectId, data);
```

This will then create a test run with only automated tests in the run. The

test just goes through and updates each test with pass or fail

integrate an existing automation framework targeting an end-to-end integration

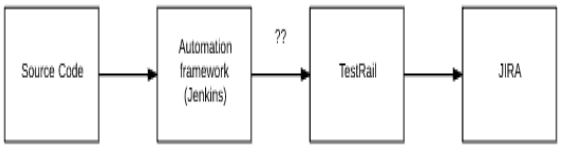

Fig-5

the test rail test cases since the automation test scripts contain the info if you do put steps, expected results in test rail, how do you keep them in-sync with the actual test scripts we were thinking of leaving test rail test cases essentially empty with a small description in an automation-driven workflow where you have both devs/QA writing automation scripts, does the Test rail is a one of the too the tool become solely a repository of results produces Currently, to performance good automation tests, the "test run" is constructed in Jenkins by inspection of the automation folder in source code, not Test rail

CONCLUSION

In this Research work successfully implementing ATRT test case requirement Traceability matrix in the paper Implementing Automated Software Testing. And retesting The Automation testing can reduce the time and cost the testing improve software quality, and improve software test programs in measurable performance and significant ways of improved software quality, among other major benefits. With manual software testing, it is difficult to repeat tests. The steps taken during the first run of a test will not be the exact steps followed during a second iteration. Without qualified, repeated tests it is difficult to produce quality measurements. Automated software testing allows for test optimization and quality metrics because automated tests can be easily repeated and results measured. The measuring test case analysis of qualified measurements supports efforts to optimize tests only when tests are repeatable. Automated Software Testing can support each phase of the software development lifecycle (SDLC). There are automated tools to assist the requirements definition phase and help produce test-ready requirements. These minimize the test effort and the cost of testing. The testing tools to support the design phase, coding, testing phases such as modelling tools, the record requirements within use test cases.

6 References

[1] Blackwell, Barry Mark, et al. "Testing tool comprising an automated multidimensional traceability matrix for implementing and validating complex software systems." U.S. Patent No. 7,490,319. 10 Feb. 2009.

[2] Oliveto, Rocco, et al. "On the equivalence of information retrieval methods for automated traceability link recovery." *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*. IEEE, 2010.

[3] Lormans, Marco, and Arie Van Deursen. "Can LSI help reconstructing requirements traceability in design and test?." *Software Maintenance and Reengineering, 2006. CSMR 2006. Proceedings of the 10th European Conference on*. IEEE, 2006.

[4] Cleland-Huang, Jane. "Toward improved traceability of non-functional requirements." *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*. ACM, 2005.

[5] Bouquet, Fabrice, et al. "Requirements traceability in automated test generation: application to smart card software validation." *ACM SIGSOFT Software Engineering Notes*. Vol. 30. No. 4. ACM, 2005.

[6] Yang, Meihong, et al. "An ontology based improved software requirement traceability matrix." *Knowledge Acquisition and Modeling, 2009. KAM'09. Second International Symposium on*. Vol. 1. IEEE, 2009.

[7] Hayes, Jane Huffman, Alex Dekhtyar, and Senthil Karthikeyan Sundaram. "Improving after-the-fact tracing and mapping: Supporting software quality predictions." *IEEE software* 22.6 (2005): 30-37.

[8] Hall, Ronald J., Richard A. Fournier, and Paul Rich.Introduction." *Hemispherical Photography in Forest Science: Theory, Methods, Applications*. Springer Netherlands, 2017. 1-13.

[9] Hoffman, Daniel, Paul Strooper, and Lee White. "Boundary values and automated component testing." *Software Testing, Verification and Reliability*9.1 (1999): 3-26.

[10] Ståhl, Daniel, Kristofer Hallén, and Jan Bosch. "Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework." *Empirical Software Engineering* (2017): 1-29.

[11] Warfield, Robert W. "Automatic software testing tool." U.S. Patent No. 5,754,760. 19 May 1998.