International Journal of Electronics Communications and Electrical Engineering
ISSN : 2277-7040 Volume 3 Issue 5 (May 2013)
http://www.ijecee.com/ https://sites.google.com/site/ijeceejournal/

Dynamic Test Case Design Scenario and Analysis of Module Testing Using Manual vs. Automated Technique

Prof. (Dr.) Vishwa Nath Maurya[1], Dr. Rajender Kumar Bathla [2],
Er. Avadhesh Kumar Maurya[3] and Diwinder Kaur Arora[4]

[1] Professor & Ex Principal,
Shekhawati Engineering College, Rajasthan Technical University, India
prof.drvnmaurya@gmail.com, prof_vnmaurya@yahoo.in

[2] Senior Assistant Professor,
Department of Computer Science & Engineering
Haryana Institute of Technology & Management, Kaithal-136027, Haryana
dr.bathla@gmail.com

[3]Assistant Professor,
Department of Electronics & Communication Engineering
Lucknow Institute of Technology, U.P. Technical University, Lucknow,
Lucknow-226002, U.P., India, avadheshmaurya09@gmail.com

[4] Inspector of Police,
Group Centre, Central Reserve Police Force, Lucknow-226002, U.P.
Ministry of Home Affairs, Govt. of India
hkdkarora@rediffmail.com, diwi.kaur1992@gmail.com

Abstract Present paper deals with problems of software testing. Software can be tested either manually or automatically. The two approaches are complementary: automated testing can perform a huge number of tests in short time or period, whereas manual testing uses the knowledge of the testing engineer to target testing to the parts of the system that are assumed to be more error-prone. Despite of this contemporary, tools for manual and automatic testing are usually different, leading to decreased productivity and reliability of the testing process. Auto test is a testing tool that provides a "best of both worlds" strategy: it integrates developers' test cases into an automated process of systematic contract-driven testing. This allows it to combine the benefits of both approaches while keeping a simple interface, and to treat the two types of tests in a unified fashion: evaluation of results is the same, coverage measures are added up, and both types of tests can be saved in the same format. In this paper, our objective is to discuss the importance of automation tool; associated to software testing techniques in software engineering. We provide introduction of software testing and describe the case tools along with solution of software testing

problem which leads to a new approach of software development known as software testing in the wide field of Information Technology.

1 Introduction

Software testing automation is the process of automating the steps of manual test cases using an automation tool or utility to shorten the testing life cycle with respect to time. Software testing is the process of executing a program with the intention of finding errors in the code. It is the process of exercising or evaluating a system or system component by manual automatic means to verify that it satisfies specified requirements or to identify differences between expected and actual results [12]. Software Testing should not be a distinct phase in System development but should be applicable throughout the design development and maintenance phases. 'Software Testing is often used in association with terms verification & validation 'Software testing is the process of executing software in a controlled manner, in order to answer the question: Does the software behave as specified. One way to ensure system's responsibility is to extensively test the system. Since software is a system component it requires a testing process also. Several noteworthy researchers [1, 5, 6, 9, 10, 14] contributed their devotion in this connection. Very recently Maurya and Bathla [10] focused their key attention for analytical study on manual vs. automated module testing using critique on overly simplistic cost model. In continuation to this, Maurya et. al. [9] explored some vital aspects pertaining to automated regression testing using software testing tool in a dynamic innovative scenario. The main contribution of this paper lies in the mechanisms that we provide to integrate the manual and automated testing strategies. This integration has the following advantages:

The overall testing process benefits from the strengths of both manual and automated testing; Support for regression testing: any automatically generated tests that uncover bugs can be saved in the same format as manual tests and stored in a regression testing database [2]. The measures of coverage (code, dataflow, specification) will be computed for the manual and automated tests as a whole; association with terms verification & validation 'Software testing is the process of executing software in a controlled manner, in order to answer the question: Does the software behave as specified. One way to ensure system's responsibility is to extensively test the system. Since software is a system component it requires a testing process also. The main contribution of this paper lies in the mechanisms that we provide to integrate the manual and automated testing strategies. This integration has the following advantages:

The overall testing process benefits from the strengths of both manual and automated testing; Support for regression testing: any automatically generated tests that uncover bugs can be saved in the same format as manual tests and stored in a regression testing database [2 &7].
The measures of coverage (code, dataflow, specification) will be computed for the manual and automated tests as a whole.

2 Testing Strategies

In this section we introduce the two strategies unified by our tool, manual testing and automated testing. After a brief introduction both of manual testing and automated testing, we will focus our attention to analyze the advantages and disadvantages of each and the rationale for integrating them.

2.1 Unit Testing

Unit testing is code-oriented testing. Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

2.2 Module Testing

A module is a collection of dependent components such as an object, class, an abstract data type or some loser collection of procedures and functions. A module encapsulates related components so it can be tested or checked without other system modules.

2.3 Sub-System Testing

This phase involves testing collections of modules, which have been integrated in to sub systems. It is a design-oriented testing and is also known as integration testing.

2.4 System Testing

The sub-systems are integrated to make up the entire system. It is also concerned with validating that the System meets its functional and non-functional requirements [1].

2.5 Acceptance Testing

This is the final stage in the testing process before the system is accepted for operational use. Acceptance testing may also reveal requirement problems where the system facilities do not really meet the user's needs [16] "Let us see there are many problems if we test to the above mentioned software testing techniques using manual testing rather automated tools".

3. Proposed Module Testing

During unit testing of C programs, a single C-level function is tested rigorously and in isolation from the rest of the application. Often unit testing is also called module testing. *Rigorous* means that the test cases are specially made for the unit in question and that they comprise of input data that may be unexpected by the unit under test. *Isolated* means that the test result does not depend on the behavior of the other units in the application. It can be achieved by directly calling the unit under test and replacing calls to other units by stub functions [15].

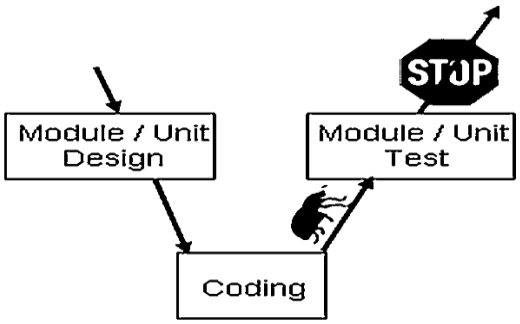

Figure 1: Module testing eliminates errors

4 Benefits of Module Testing

In this section, we will discuss about beneficial aspects of module testing.

4.1 Module Testing Reduces Complexity of Test Case Specification

Instead of trying to create test cases that test the whole set of interacting units, the test cases for unit testing are specific to the unit under test (*divide-and-conquer*). Test cases can easily comprise of input data that is unexpected by the unit under test, something which may be hard to achieve during system testing [16].

4.2 Easy Fault Isolation in Module Testing

If the unit under test is tested in isolation from the other units, detecting the cause of a failed test case is easy. The fault must be related to the unit under test, and not to a unit further down the calling hierarchy [15]

4.3 Early Errors Detection in Module Testing

Unit testing can be conducted as soon as the unit to be tested compiles successfully. Therefore errors inside the unit can be detected very early.

4.4 Module Testing Saves Money

It is generally accepted that errors detected late in a project are more expensive to correct than errors that are detected early. Hence unit testing saves money.

4.5 Confidence Feature in Module Testing

Unit testing gives confidence. After the unit testing, the application will be made up of single, fully tested units. A test for the whole application will then be more likely to pass. Module/Unit concentrates verification on the smallest element of the program – the module. Using the detailed design description important control paths are tested to establish errors within the bounds of the module. The tests that are performed as part of unit testing are shown in the figure below. The module interface is tested to ensure that information properly flows into and out of the program unit being tested. The local data structure is considered to ensure that data stored temporarily maintains its integrity for all stages in an algorithm's execution. Boundary conditions are tested to ensure that the modules perform correctly at boundaries created to limit or restrict processing. All independent paths through the control structure are exercised to ensure that all statements in been executed once. Finally, all error-handling paths are examined [6 & 13].

5 Module Testing Analysis

Module testing is code-oriented testing. Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components. A unit test is a piece of code written by a developer that exercises a very small, specific area of functionality in the code being tested. Usually a unit test exercises some particular method in a particular context. For example, you might add a large value to a sorted list, then confirm that this value appears at the end of the list [6 & 13]

Module Testing = Unit Testing

- Large programs cannot practically be tested all at once
- Break down programs into modules
- Test modules individually as first phase

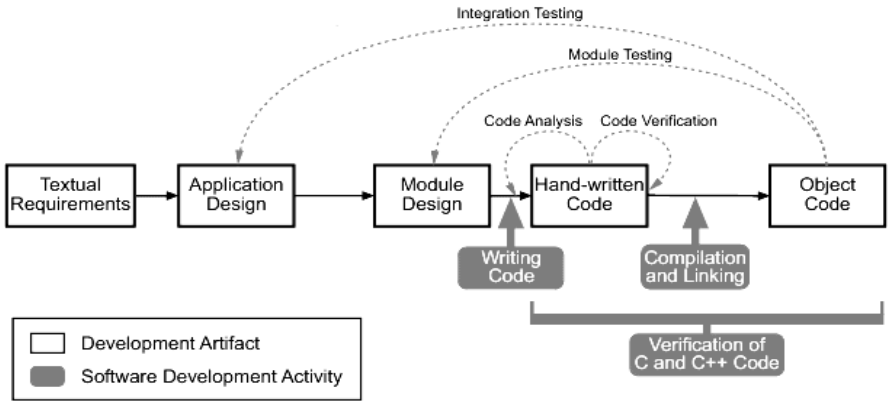

Figure 2: Structure of module testing

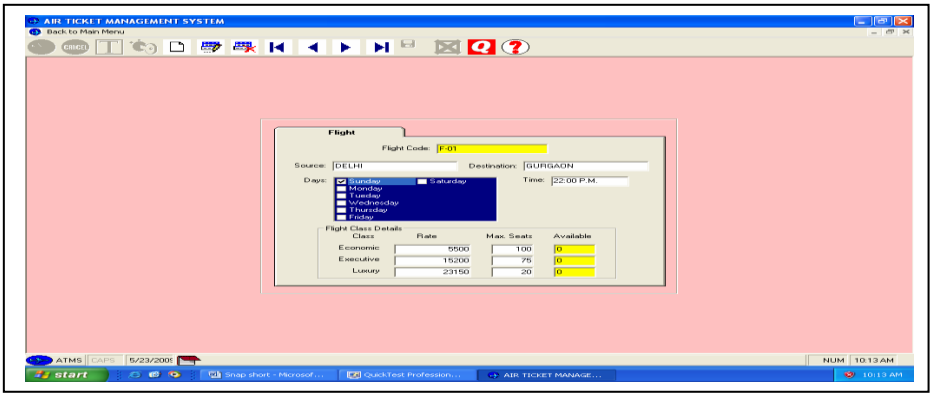

Figure 3: Air ticket management system

6 Module Test for Airlines System

6.1 Description: This is airlines ticket management system i.e. complete module. In which researcher categorized to the module part e.g. Airlines flight Unit, Airlines Reservation Unit system. By this module system, no doubt testing is done easily rather test to complete system. Because module tests are performed to prove that a piece of code does what the developer thinks it should be done. These module is compared by manually or Automated tool i.e. QTP.

6.2. Description: This module shows to the airline flight categories system. In this unit each flight class details are mentioned e.g. economic class, executive class, luxury class etc.

6.3. Description: This unit shows to the airline flight categories system. In this unit flight code is mentioned and validation and check point is given in the flight class details i.e. economic, executive, luxury e.g. economic class traveling rate under range 12000-18000, executive class rate is not less than 5000 or not more than 10000 rate, luxury rate 12000 to 18000 also.

7 Dynamic Test Case Design

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is functioning correctly. Test cases are often referred to as test scripts, particularly when written. Written test cases are usually collected into test suites [8]

7.1 Typical Written Test Case Format

A test case is usually a single step, or occasionally a sequence of steps, to test the correct behavior/functionalities, features of an application. An expected result or expected outcome is usually given.
Additional information that may be included:

- test case ID
- test case description
- test step or order of execution number
- related requirement(s)
- depth
- test category
- author
- Check boxes for whether the test is automatable and has been automated.

Additional fields that may be included and completed when the tests are executed:

- pass/fail remarks

7.2 Types of Test Case Design Techniques

There are two types of test case design techniques which are as following:

1. Equivalence class partition
2. Boundary value analysis

Equivalence class partition: here the test engineer writes the valid and invalid test cases i.e. positive test cases and negative test cases.
Boundary value analyses: if there is a range kind of input the technique used by the test engineer to develop the test Cases for that range are called as boundary value analyses.

7.3 Equivalence Class Partitioning

Equivalence partitioning is a method for deriving test cases. In this method, classes of input conditions called equivalence classes are identified such that each member of the class causes the same kind of processing and output to occur. In this method, the tester identifies various equivalence classes for partitioning. A class is a set of input conditions that are is likely to be handled the same way by the system. If the system were to handle one case in the class erroneously, it would handle all cases erroneously [17]. Designing test cases using equivalence partitioning, you will need to perform following two steps;

- Identify the equivalence classes
- Design test cases

Take each input condition described in the specification and derive at least two equivalence classes for it. One class represents the set of cases which satisfy the condition (the valid class) and one represents cases which do not (the invalid class) Following are some general guidelines for identifying equivalence classes:
If the requirements state that a numeric value is input to the system and must be within a range of values, identify one valid class inputs which are within the valid range and two invalid equivalence class's inputs which are too low and inputs which are too high.

7.4 Some Examples of Classes

In support of more understanding the equivalence partitioning, we present here a few examples of classes as following:

- One valid class: (QTY is greater than or equal to -9999 and is less than or equal to 9999). This is written as (- 9999 < = QTY < = 9999)
- The invalid class (QTY is less than -9999), also written as (QTY < -9999)
- The invalid class (QTY is greater than 9999), also written as (QTY >9999)

Moreover, if the requirements state that the number of items input by the system at some point must lie within a certain range, specify one valid class where the number of inputs is within the valid range, one invalid class where there are too few inputs and one invalid class where there are too many inputs.

7.5 Module with Boundary Value Study

It is software testing design technique in which tests are designed to include representatives of boundary values. The expected input and output values should be extracted from the component specification. The input and output values to the software component are then grouped into sets with identifiable boundaries. Each set, or partition, contains values that are expected to be processed by the component in the same way. Partitioning of test data ranges is explained in the equivalence partitioning test case design technique. It is important to consider both valid and invalid partitions when designing test cases [5]
For an example where the input values were months of the year expressed as integers, the input parameter 'month' might have the following partitions:

... -2 -1 0 1 ..... 12 13 14 15.....
---|---|---

invalid partition 1 valid partition invalid partition

The boundaries are the values on and around the beginning and end of a partition. If possible test cases should be created to generate inputs or outputs that will fall on and to either side of each boundary. This would result in three cases per boundary. The test cases on each side of a boundary should be in the smallest increment possible for the component under test. In the example above there are boundary values at 0,1,2 and 11,12,13. If the input values were defined as decimal data type with 2 decimal places then the smallest increment would be the 0.01. Where a boundary value falls within the invalid partition the test case is designed to ensure the software component handles the value in a controlled manner. Boundary value analysis can be used throughout the testing cycle and is equally applicable at all testing phases [3 & 4].
After determining the necessary test cases with equivalence partitioning and subsequent boundary value analysis, it is necessary to define the combinations of the test cases when there are multiple inputs to a software component.

Table: 1. Test cases of equivalence class partitioning

Test Case Name	Test Case Describe	Test Steps			Test Case Status (p/f)
		Steps	**Expected Result**	**Actual Result**	
Economic Rate	Economic rate should be within 5000-10000	1) <5000	Not Accepted	The input is accepted by the text box	Fail
		2) 5000-6000	Accepted	The input is accepted by the text box	Pass
		3) 6001-7000	Accepted	The input is accepted by the text box	Pass
		4) 7001-8000	Accepted	The input is accepted by the text box	Pass
		5) 8001-9000	Accepted	The input is accepted by the text box	Pass
		6) 9001-10000	Accepted	The input is accepted by the text box	Pass
		7) >10000	Not Accepted		Fail

Table: 2. Test case steps with status

Test Case Name	Test Case Describe	Test Steps			Test Case Status (p/f)
		Steps	**Expected Result**	**Actual Result**	
Economic Rate	Economic rate should be within 5000-10000	1) 4000 2) 5000 3) 10000 4) 11000	Not Accepted Accepted Accepted Not Accepted	The input is accepted by the text box The input is accepted by the text box The input is accepted by the text box The input is accepted by the text box	Fail Pass Pass Pass

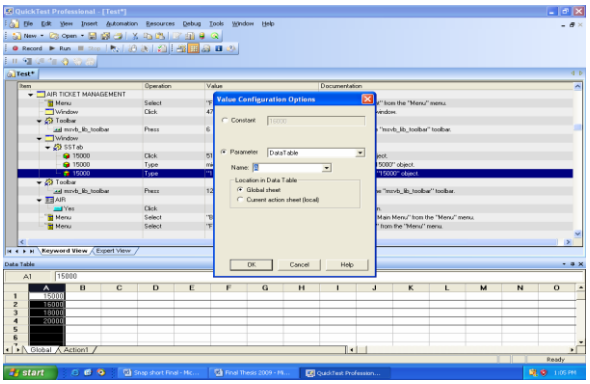

Figure 4: Parameterized testing for airline module

8 Airlines Module using with QTP Tool

8.1 Description: This window is running the conditioned Data table as mentioned 15000,16000,18000,20000 as we had implemented validation on flight class unit. Suppose if I take <10000 and >18000 value then it would show the failed result in the last rate value and first three values will be done.

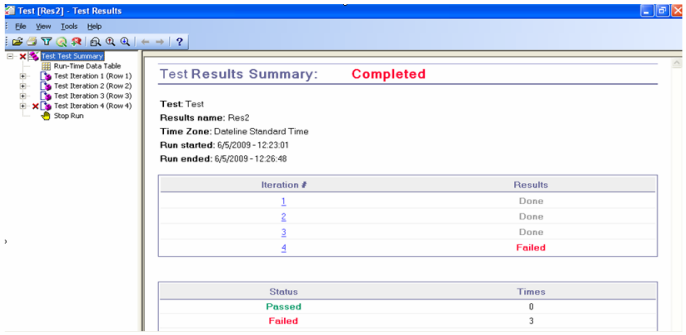

Figure 5: Testing results of airlines module

This test results summary is showing the actual result that is first three values are right e.g. 15000, 18000, 12000 that have tested and done and the last value is wrong that has failed e.g. 20000

9 Graphs of Manual Vs. Automated Testing

Release	Manual Test	Auto Test	Manual Test Cumulative
1	10	10	10
2	10	0	20
3	10	0	30
4	10	0	40
5	10	0	50

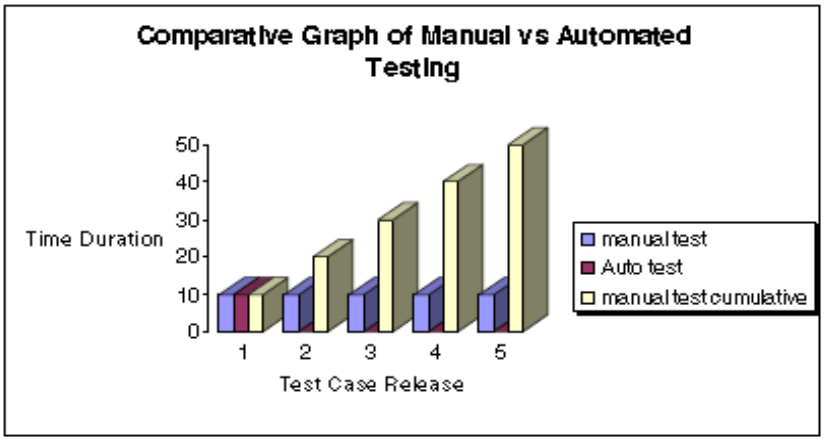

Figure 6: Graph of Manual Vs. Automated Testing

9.1 Description: The chart in figure 6 is showing the comparative results of manual vs. automated testing. Blue line is indicating to the manual testing and the red line is indicating the automated testing whereas yellow line shows the manual test cumulative. The time duration is mentioned 0 to 50 and total test cases release is 1 to 5. by this chart we can understand if one test case has be released and time in manual testing assigned i.e 10 minutes and same assigned in automated testing Suppose if again test case is to be release the manual testing will assume time 10 minute but in the case of Automated testing time will assume second the zero minutes

10 Comparative Analysis of Manual vs. Automated Testing

In this section, we summarize comparative features of manual and automated testing as following:

- Manual testing is time consuming.
- There is nothing new to learn when one tests manually.
- People tend to neglect running manual tests.
- None maintains a list of the tests required to be run if they are manual tests.
- Manual testing is not reusable.
- Tests have to be repeated by each stakeholder for e.g. Developer, Tech Lead, GM and Management.
- Manual testing ends up being an integration test.
- In a typical manual test, it is very difficult to test a single unit.
- Scripting facilities are not in manual testing, for more details, we refer [10]

Automated testing with Quick Test addresses these problems by dramatically speeding up the testing process. You can create tests that check all aspects of your application or web site, and then run these tests every time your site or application changes, for more details, we refer [3]. Automated testing with Quick Test possesses following advantageous features:

- **Faster Testing**: Quick test runs significantly faster than human user.
- **Reliable**: Automated testing with Quick Test performs precisely the same operations each time they are run, thereby eliminating human error.
- **Programmable**: One can program sophisticated tests that bring out hidden information.
- **Comprehensive**: One can build a suite of tests that covers every feature in a web site or application.
- **Reusable**: One can build a suite of tests that covers every feature in your website or application.

10.1 Cost Model Based Analysis

Building on the example from the previous section, we propose an alternative cost model drawing from linear optimization. The model uses the concept of opportunity cost to balance automated and manual testing. The opportunity cost incurred in automating a test case is estimated on basis of the lost benefit of not being able to run alternative manual test cases. Hence, in contrast to the simplified model presented in Section 2, which focuses on a single test case, our model takes all potential test cases of a project into consideration. Henceforth, it optimizes the investment in automated testing in a given project context by maximizing the benefit of testing rather than by minimizing the costs of testing [6]

10.2 Restriction of Fixed Budget

First of all, the restriction of a fixed budget has to be introduced to our model. This restriction corresponds to the production possibilities frontier described in the previous section. R1: na * Va + nm * Dm ≤ B na := number of automated test cases nm := number of manual test executions Va := expenditure for test automation Dm := expenditure for a manual test execution B := fixed budget Note that this restriction does not include any fixed expenditures (e.g., test case design and preparation) manual testing. Furthermore, with the intention of keeping the model simple, we assume that the effort for running an automated test case is zero or negligibly low for the present. This and other influence factors (e.g., the effort for maintaining and adapting automated tests) will be discussed in the next section. This simplification, however, reveals an important difference between automated and manual testing. While in automated testing the costs are mainly influenced by the number of test cases (*na*), manual testing costs are determined by the number of test executions (*nm*). Thus, in manual testing, it does not make a difference whether we execute the same test twice or whether we run two different tests. This is consistent with manual testing in practice – each manual test execution usually runs a variation of the same test case [15]

10.3 Benefits and Objectives of Automated and Manual Testing

In order to compare two alternatives based on opportunity costs, we have to evaluate the benefit of each alternative, i.e., automated test case or manual test execution. The benefit of executing a test case is usually determined by the information this test case provides. The typical information is the indication of a defect. Still, there are additional information objectives for a test case (e.g., to assess the conformance to the specification). All information objectives are relevant to support informed decision

making and risk mitigation. A comprehensive discussion about what factors constitute a good test case is given in [3].

10.4 Maximizing the Benefit

For the purpose of maximizing the overall benefit yielded by testing, the following target function has to be added to the model. T: Ra(na) + Rm(nm) □ max Maximizing the target function ensures that the combination of automated and manual testing will result in an optimal point on the production possibilities frontier defined by restriction *R1*. Thus, it makes sure the available budget is entirely and optimally utilized.

10.5 Real Example

To illustrate our approach we extend the example used in previous section. For this example the restriction *R1* is defined as follows. R1: $na * 1 + nm * 0.25 \leq 75$ To estimate benefit of automated testing based on the risk exposure of the tested object, we refer to the findings published by Boehm and Basili [5]: "Studies from different environments over many years have shown, with amazing consistency, that between 60 and 90 percent of the defects arise from 20 percent of the modules, with a median of about 80 percent. With equal consistency, nearly all defects cluster in about half the modules produced." Accordingly we categorize and prioritize the test cases into 20 percent highly beneficial, 30 percent medium beneficial, and 50 percent low beneficial and model following alternative restrictions to be used in alternative scenarios. R2.1: $na \geq 20$ R2.2: $na \geq 50$ To estimate the benefit of manual testing we propose, for this example, to maximize the test coverage. Thus, we assume an evenly distributed risk exposure over all test cases, but we calculate the benefit of manual testing based on the number of completely tested releases. Accordingly we categorize and prioritize the test executions into one and two or more completely tested releases. We model following alternative restrictions for alternative scenarios. R3.1: $nm \geq 100$ R3.2: $nm \geq 200$ Based on this example we illustrate three possible scenarios in balancing automated and manual testing. Figures 4a, 4b and 4c depict the example scenarios graphically.

- **Scenario A** – The testing objectives in this scenario are, on the one hand, to test at least one release completely and, on the other hand, to test the most critical 50 percent of the system for all releases. These objectives correspond to the restrictions *R3.1* and R2.2 in our example model. As shown in Figure 8, the optimal solution is point *S1* ($na = 50$, $nm = 100$) on the production possibilities frontier defined by *R1*. Thus, the 50 test cases

referring to the most critical 50 percent of the system should be automated and all test cases should be run manually once.

- **Scenario B** – The testing objectives in this scenario are, on the one hand, to test at least one release completely and, on the other hand, to test the most critical 20 percent of the system for all releases. These objectives correspond to the restrictions *R3.1* and *R2.1* in our example model. As shown in Figure 9 any point within the shaded area fulfills these restrictions. The target function, however, will make sure that the optimal solution will be a point between *S1* (*na* = 50, *nm* = 100) and *S2* (*na* = 20, *nm* = 220) on the production possibilities frontier defined by *R1*. Note: While all points on *R1* between the *S1* and *S2* satisfy the objectives of this scenario, the point representing the optimal solution depends on the definition of the contribution to risk mitigation of automated and manual testing, *Ra(na)* and *Rm(nm)*.
- **Scenario C** – The testing objectives in this scenario are, on the one hand, to test at least two releases completely and, on the other hand, to test the most critical 50 percent of the system for all releases. Graph for scenario C has been shown in Figure 10.

These objectives correspond to the restrictions *R3.2* and *R2.2* in our example model. As shown in Figure 10. A solution that satisfies both restrictions cannot be found.

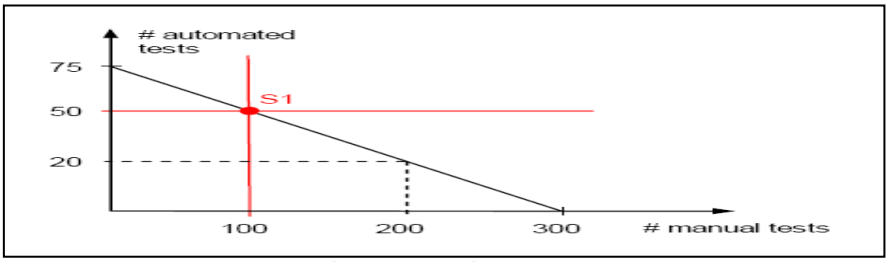

Figure 8: Graph of scenario A for automated vs. manual

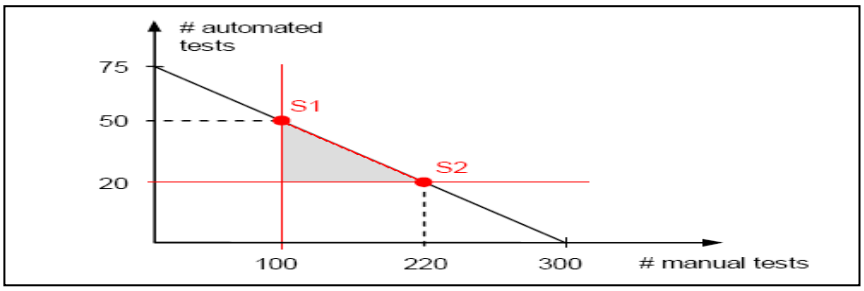

Figure 9: Graph of scenario B for automated vs. manual

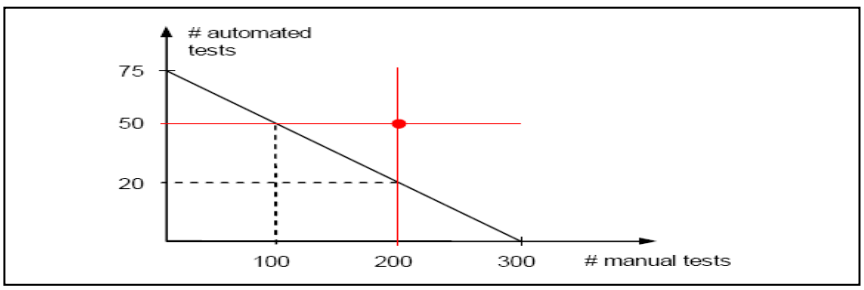

Figure 10: Graph of scenario C for automated vs. manual

11 Conclusions

The Conclusion of this research and review paper is analyze to the manual testing drawback in software testing rather more benefits of automated software testing tools. The enlightened of this modern approaches leads to the new Methodologies of software test automation. The destination of software testing is considered to succeed when an error is detached. Effective Conclusions are given below. Software testing is an art. Most of the testing methods and practices are not very different from 20 years ago. In the current era there are many tools and techniques available to use. Good testing also requires a tester's creativity, experience and intuition, together with proper techniques. Testing is more than just debugging. Testing is not only used to locate defects and correct them. It is also used in validation, verification process, and reliability measurement. Although manual testing is not expensive but it is no more effective rather automated testing because automation is a good way to cut down cost and time. Testing efficiency and effectiveness is the criteria for coverage-based testing techniques.

Acknowledgments

Dr. V. N. Maurya; mentor author of the present paper has served as founder Director of Vision Institute of Technology, Aligarh (U. P. Technical University, Lucknow (India) and as Principal/Director at Shekhawati Engineering College (Rajasthan Technical University, Kota) after having vast experience in the field of Technical and Management Institutions in the cadre of Professor & Dean Academics such as at Institute of Engineering & Technology, Sitapur, UP, India; Haryana College of Technology & Management (Kuruchhetra

University, Kuruchhetra) and Singhania University, Rajasthan. During his tenure as the Director, Vision Institute of Technology, Aligarh (Uttar Pradesh Technical University, Lucknow) and as the Principal, Shekhawati Engineering College (Rajasthan Technical University, Kota); massive expansion of infrastructure, research facilities, laboratories upgradation/augmentation and other relevant facilities and services for B.Tech./M.Tech./MBA academic programmes in different branches had taken place to accommodate and facilitate the campus students. He is the Chief Editor of Editorial Board of American Journal of Modeling and Optimization; Science and Education Publishing, New York, USA and Statistics, Optimization and Information Computing; International Academic Press, Hong Kong and Advisory Editor of World Research Journal of Mathematics; Bioinfo Publications, Pune, India and Member of Editorial and Reviewer Board of over 50 Indian and Foreign International leading journals published in USA, Italy, Hong Kong, Austria, Germany, U.K., Algeria, Nigeria and other European and African countries. He has been associated with leading Indian Universities-U. P. Technical University, Lucknow during 2005-06 and Chhatrapati Shahu Ji Maharaj University, Kanpur for three terms during 2000-2004 for significant contribution of his supervision as Head Examiner of Central Evaluation for Theory Examinations of UG (B.Tech./B.Pharm.) and PG (MA/M.Sc.) programmes.

Dr. Maurya was born on 15th July 1974 in District Basti of Uttar Pradesh in India and he possesses an outstanding and meritorious academic record. He earned his M.Sc. (1996) with First Division and Ph.D. Degree (2000) in Mathematics & Statistics with specialization in Operations Research from Dr. Ram Manohar Lohia Avadh University, Faizabad, UP, India and thereafter he accomplished another two years Master's Professional Degree-MBA with First Division (B^+ Grade) with specialization in Computer Science from NU, California, USA in 2003. He did Ph.D. on topic titled "A study of use of stochastic processes in some queueing models" under supervision of Prof. (Dr.) S.N. Singh, Ph.D. (BHU). He started his teaching career as Lecturer in 1996 to teach post-graduate courses MBA, MCA and M.Sc. and later he was appointed as Professor & Head, Department of Applied Sciences and Engineering at Singhania University, Rajasthan in the year 2004. Since then, Prof. V. N. Maurya has rendered his services as Professor & Head/Dean as well as keen Researcher for Post-Doctoral research and he has devoted his entire scientific and professional career in teaching at various premier technical institutions of the country such as at Haryana College of Technology & Management, Kaithal (Kuruchhetra University, Kuruchhetra); Institute of Engineering & Technology, Sitapur and United College of Engineering & Research, Allahabad. During the last 17 years of his professional career, Prof. V. N. Maurya has authored three textbooks and published more than 55 scientific and academic research papers including 25 research papers as Principal Author based on his Post-Doctoral work and D.Sc. Thesis in Indian and Foreign leading International Journals in the field of Mathematical and Management Sciences, Industrial Engineering & Technology. He is an approved Supervisor of UGC recognized various Indian Universities for Research Programs leading to M. Phil. & Ph.D. such as Shridhar University, Pilani (Rajasthan), Singhania University,

Rajasthan and CMJ University, Sillong, Meghalaya and JJT University Jhunjhunu, Rajasthan and U.P. Technical University Lucknow etc. and since last 7 years, he is actively engaged as Research Supervisor of M. Phil. & Ph.D. Scholars in wide fields of Operations Research, Optimization Techniques, Statistical Inference, Applied Mathematics, Operations Management and Computer Science. He has guided as Principal Supervisor and Co-Supervisor to several Research Scholars of M. Phil. and Ph.D. Most of his published research papers in India, USA, Algeria, Nigeria, Malaysia and other European and African countries are recognized as innovative contributions in the field of Applied Mathematics and Statistics including Operations Research. By virtue of his innovative research contribution to many mathematical, statistical, computer science and industrial engineering related areas basic as well as application oriented, Prof. V. N. Maurya is known as one of the Indian leading experts in Applied Mathematics, Statistics and Operational Research.

Apart from this, Prof. Maurya is also on active role of Fellow/Senior/Life Member of various reputed National and International professional bodies of India and abroad including Operations Research Society of India, Kolkata; Indian Society for Technical Education, New Delhi; Indian Association for Productivity, Quality & Reliability, Kolkata; Indian Society for Congress Association, Kolkata; International Indian Statistical Association, Kolkata; All India Management Association, New Delhi; Rajasthan Ganita Parishad, Ajmer and International Association of Computer Science & Information Technology, Singapore and many more.

Dr. Rajender Kumar Bathla, corresponding author of the present paper is working as a Senoir Assistant Professor in Computer Science & Engineering Department at Haryana Institute of Engineering & Technology, Kaithal (Haryana). He accomplished his master's degree M.Tech. in Computer Science from M.M. University Mullana, Haryana in 2009 and earned Ph.D. Degree in Computer Science & Engineering, Faculty of Technology from CMJ University, Shillong (Meghalaya) under supervision of Prof. (Dr.) V. N. Maurya, Ex. Founder Director, Vision Institute of Technology Aligarh (U.P. Technical University, India) in 2012. His area of specialization is Software Testing. He has published several research papers based on his M.Tech. and Ph.D. Thesis in Indian and Foreign Journals and Conferences. Apart from this, Dr. Rajender Kumar Bathla is also serving as Editor and Reviewer of several Foreign leading International Journals such as International Journal of Electronics Communication and Electrical Engineering, Algeria; World Academy of Science, Engineering and Technology, Italy

Diwinder Kaur Arora; co-author of the present paper accomplished two years Master's Degree of MBA with specialization in Human Resources Management from

Pondicherry Central University, Pondicherry and she graduated with B.Sc. (Medical/ZBC Group) Degree in 1987 from Kanpur University, Kanpur, India and did two years Diploma also from Government Polytechnic College, Amethi, U.P. throughout in First Division. She has vast experience of more than 22 years of general administration and management as Police Officer in Central Reserve Police Force, Ministry of Home Affairs, Govt. of India. She was selected as Assistant Sub-Inspector (Non-Gazetted Officer) in 1991 and after successful completion of her services she was promoted as Sub-Inspector in 2004 and since 2012 she is working in the grade of Inspector of Police at Group Centre, Central Reserve Police Force, Lucknow, U.P. Apart from this, she has published more than 15 research papers in Indian and Foreign International journals of repute in the field of Management, Computer Science & Information Technology and Physical Sciences such as in World of Sciences Journal, Engineers Press Publishing Group, Vienna, Austria; International Journal of Engineering Research and Technology, Engineering Science & Research Support Academy (ESRSA), Vadodara, India; International Journal of Electronics Communication and Electrical Engineering, Algeria; International Journal of Information Technology & Operations Management, Academic and Scientific Publisher, New York, USA; International Open Journal of Operations Research, Academic and Scientific Publisher, New York, USA.

Er. Avadhesh Kumar Maurya; co-author of the paper is having an outstanding academic record and accomplished his M.Tech. Degree with specialization in Digital Communication Engineering from Uttarakhand Technical University, Dehradun, UK and he was graduated with B.Tech. Degree in Electronics and Communication Engineering from Rajasthan Technical University, Kota (Rajasthan). He is recipient of four First Divisions in his Student career with flying colors. Since last one year, Er. A. K. Maurya is serving as Assistant Professor in Department of Electronics and Communication Engineering at Lucknow Institute of Technology, U.P. Technical University, Lucknow. Prior to assuming the post of Assistant Professor at Lucknow Institute of Technology, U.P., he served as a Network Engineer for two years in National Informatics Centre, Department of Information Technology, Govt. of India with collaboration of HCL Co. He has worked on some projects such as Movable Target Shooter using Ultrasonic Radar and Hartley Oscillator. Apart from this, he has got industrial training in Door Darshan Kendra, Lucknow, U.P. in the field of TV Program Generation and Broadcasting of different channels for partial fulfilment of his Degree and published also over 18 research papers in various Indian and Foreign International journals of repute in the field of Electronics & Communication Engineering, Computer Science & Information Technology and Physical Sciences such as in International Journal of Electronics Communication and Electrical Engineering, Algeria; World of Sciences

Journal, Engineers Press Publishing Group, Vienna, Austria; International Journal of Information Technology & Operations Management, Academic and Scientific Publisher, New York, USA; International Journal of Mathematical Modeling and Applied Computing, Academic and Scientific Publisher, New York, USA; International Journal of Engineering Research and Technology, Engineering Science & Research Support Academy (ESRSA), Vadodara, India; International Journal of Software Engineering & Computing, Serials Publications, New Delhi, India and many more.

References

[1] Boehm, B.: Value-Based Software Engineering: Overview and Agenda. In: Biffl S. et al.: Value-Based Software Engineering. Springer (2005)
[2] Cerv Antes Alex: Exploring the Use of a Test Automation Framework, IEEEAC paper #1477, version 2, up dated January 9 (2009)
[3] Dustin E. et. al.: Automated Software Testing, Addison- Wesley (1999)
[4] Fewster M., Graham D.: Software Test Automation: Effective Use of Text Execution Tool, Addison- Wesley (1999)
[5] Godefroid N., Patrice K. and Sen Koushik, In: DART- Directed Automated Random Testing, Proceedings of Conference on Programming Language Design and Implementation, ACM SIGPLAN (2005)
[6] Grechanik M., Xie Q. and Fu Chen: Maintaining and Evolving GUI- Directed Test Scripts, IC SE'09, IEEE, Vancouver, Canada, 978-1-4244-3452-7, May 16-24, (2009)
[7] Ieshin A., Gerenko M. and Dmitriev V.: Test Automation- Flexible Way, IEEE, 978-1-4244-5665-9 (2009)
[8] Marinov D. and Khurshid S.: Test Era: A Novel Framework for Automated Testing of Java Programs, Proceedings of 16th IEEE International Conference on Automated Software Engineering (ASE), pp. 22-34 (2001)
[9] Maurya V.N., Bathla R.K., Maurya A.K. and Arora D. Kaur: A Dynamic Innovative Scenario of Automated Regression Testing Using Software Testing Tool, International Journal of Information Technology & Operations Management, Academic & Scientific Publisher, New York, USA, Vol. 1, No.1, pp. 1-10 (2013)
[10] Maurya V.N., Bathla R.K.: Analytical Study on Manual vs. Automated Module Testing Using Critique on Overly Simplistic Cost Model, International Journal of Electronics Communication & Electrical Engineering, Algeria, Vol. 2, Issue 1, pp. 203-216 (2012)
[11] Mustafa Khaled M., Al-Qutaish Rafa E. and Mohammad I. Muhairat: Classification of Software Testing Tools Based on the Software Testing Methods, 2nd International Conference on Computer and Electrical Engineering, 978-0-7695-3925-6 (2009)

[12] Nagowah Leckraj and Purmanand Roopnah: A Simple Automated System Testing Tool", IEEE, 978-1-4244- 5540-9/10 (2010)
[13] Pressman R.S.: Software Engineering A Practitioner's Approach, McGraw-Hill International Edition, ISBN: 007-124083-7
[14] Rajender Kumar, Maurya V.N. and Maurya A.K.: A Cost-Benefit Model for Evaluating Regression Testing Technique, International Journal of Software Engineering & Computing, Vol. 4, No. 2, pp. 84-89 (2012)
[15] Ramler R., Biffl S. and Grünbacher P.: Value-Based Management of Software Testing. In: Biffl S. et al. Value-Based Software Engineering. Springer (2005)
[16] Schwaber C. and Gilpin M.: Evaluating Automated Functional Testing Tools, Forrester Research (2005)
[17] Tonella P.: Evolutionary Testing of Classes, International Symposium on Software Testing and Analysis, Boston, Massachusetts, USA (2004)