Effective testing: An investigative approach for improving test efficiency

Abdul Rauf EM

Abstract— Software testing efficiency is very important factor for all test organizations. By conducting a case study in data base environment. this paper is talking about how we can improve the test execution process by considering the factors like automation using standard framework, introducing risk based testing, parallel execution , modularization , avoiding code redundancy and through proper test management. The recommendation given at the end of the document can easily implement for any domains of testing for getting an improvement in test execution.

Index Terms— Indexing tool, test efficiency, test automation, process improvement, risk based testing, test tracking tool, test point method, parallel execution.

1 INTRODUCTION

This article focuses on how we can improve the test efficiency in an existing test environment. Test efficiency measures the cost-effectiveness of a test organisation. Article also provide a justification for identifying the importance of this topic, the theoretical basis for this project, research methods to be employed in conducting this case study, and expected future benefits

2 BASIS OF THIS ARTICLE

Even though software development industry spends more than half of its budget on software testing and maintenance related activities; software testing has received little attention in our curricula. This suggests that most software testers are then either self taught or they acquire needed skills on the job perhaps through informal and formal mechanisms used commonly in the industry. Lack of proper attention in acquiring testing skills is resulting in less utilization of test resources and thus results in less test efficiency of organisation. Review of extant literature on software testing lifecycle (STLC) identifies various Software testing activities and ways in which these activities can be carried out in conjunction with the software development process. This literature also identifies various skills that software testers need to possess in order to perform activities effectively in a given phase of STLC. Similar to development lifecycle (SDLC), STLC also suggests the phases of analysis, design, implementation, execution, and evaluation in software testing lifecycle. The V - model, which is the most popular testing model, provides a basis for the identification of various testing activities. Based on the V- model, Vijay (2001) and Waligora and Coon (1996) suggest the need to conduct testing in parallel with many of the SDLC phases so that testing efforts in later stages can be minimized

With this case study, we are targeting on how we can improve the test efficiency of database indexing tool and thus to prepare generic guide lines for improving the test efficiency of an organisation. Database (DB) indexing tool, which helps users and application programmers a fast, versatile, and quick method of searching full-text documents stored in DB and file systems using SQL queries. The initial test environment of this tool was not fully automated, that results in lot of manual intervention for executing system test cases and thus results in lot of manpower utilization. This tool has multiple releases and service packs, each service packs is consuming around 100 man days of system testing due to the execution of regression scenarios in test cycle. This case study was targeted to come out with new regression environment that can use existing test setup and test tools for reducing 50 % of system test effort. Case study was conducted in database environment, but the solution will be generic and can use in other test environments after customization

3 INITIAL APPROACH

The indexing tool don't have automated regression environment and this had negative impact on the effectiveness of the testing of the product. We used to run regression test cases manually in system test cycle. Due to the time limitation and resource shortage only selected regression scenarios considered in system test cycle that was resulting a risk of less coverage for regression scenarios. Below are some of the challenges faced by the test team

- Manual intervention for running the test suites
- Manual download and installation of DB and indexing tool drivers.
- Parallel execution of different scenarios on different machines from central point
- Scheduled execution of multiple scenarios
- Tracking the test status for multiple service packs

Abdul Rauf EM is working in IBM India Software Labs, Bangalore office. He received his bachelor's degree in Computer Engineering from the University of Cochin, Kerala -India (2000), master's degree in Telecommunication and Software Engineering from Bits – Pilani (2004). He has total 11+ years of software industry experience with proven expertise in different skills and with involvement in various life cycles of project development

- Tracking the test progress for each service packs
- Tracking the test status of the individual test team members

4 SOLUTION

To address the problem of automated regression environment, team evaluated many tools and processes and finally short-listed the build forge tool. IBM Rational Build Forge is an adaptive process execution framework that automates, orchestrates, manages, and tracks all the processes between each handoff within the assembly line of software development, creating an automated software factory. Rational Build Forge integrates into your current environment and supports major development languages, scripts, tools, and platforms; allowing you to continue to use your existing investments while adding valuable capabilities around process automation, acceleration, notification, and scheduling.

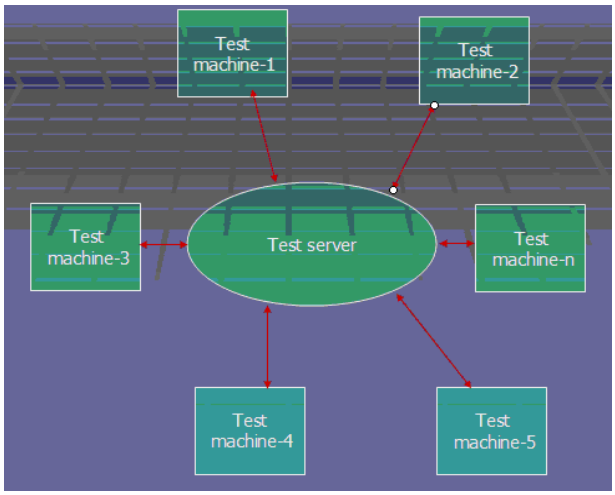

Fig. 1. Existing test setup

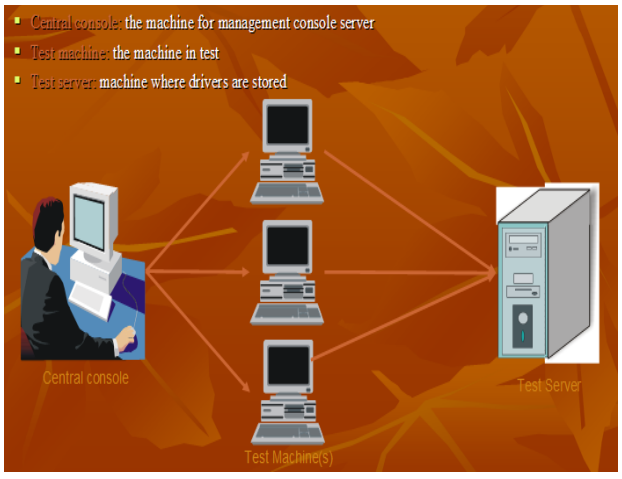

Fig. 2 Proposed solution

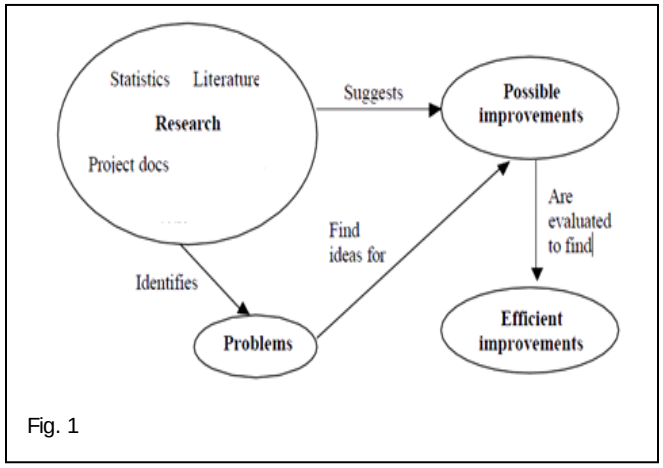

Fig. 1

The above figures shows the current test setup (Figure -1), new setup implemented for regression environment (Figure-2) and the approach used for studying and implementing the setup (Figure-3). New regression environment implementation done after a deep analysis of the current test frame work. Some of the improvement areas identified in the initial setup are mentioned below

- Parallel execution of test scenarios
- Scheduling of different jobs
- Log verification from a central console
- Monitoring of long running scenarios from central console
- Report generation
- Code redundancy
- Modularization

Most of the above reported issues are addressed by the new regression environment. With the new regression setup we are able to move many system test cases to automated regression environment and thus reduced the system test cycle time. Now the team is able to save around 30-50% of manual effort and more test coverage for each service pack release

5 INSTALL AUTOMATION

Manual download and installation of DB and index tool drivers are one of the painful issue we faced during system and install verification testing. We addressed this issue with development of new tool named Test Install automation Tool (TIAT). With the introduction of the new tool we automated the following items

- FTP download of selected levels of drivers to centralized location (Test server)
- Copy the specified driver to the test machine

- Unzip the compressed drivers and extract the same
- Run silent install using response file generated
- DB installation on multiple machines
- Index tool installation on multiple machines
- Un installation
- Generates reports that summarizes results of the job
- Verification
- Scheduling

5.1 TIAT Concepts

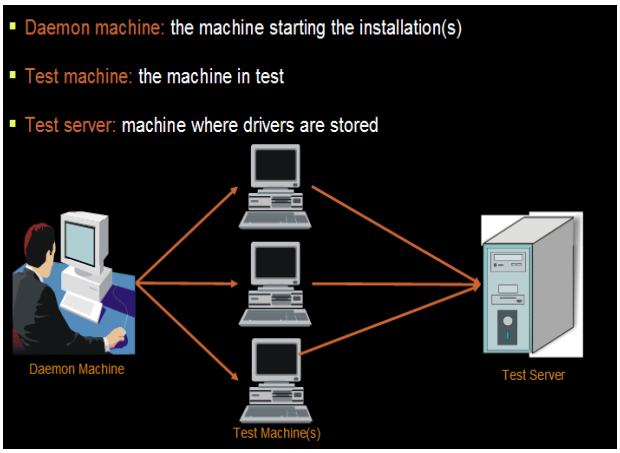

Fig. 4

Figure- 4 shows the concepts of the TIAT tool, and Figure -5 shows the typical work flow of the TIAT tool. With this tool we are able to solve almost all the issues related with driver download and installation.

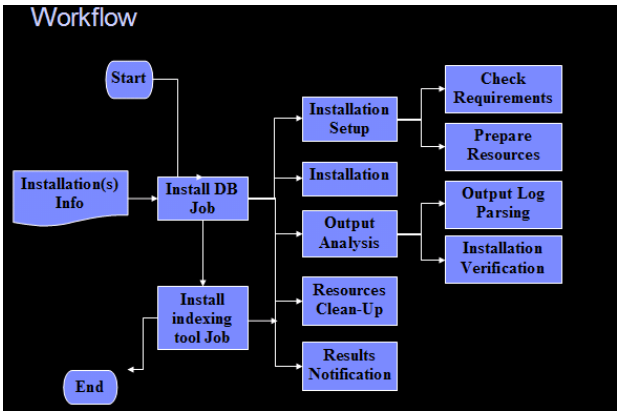

Fig. 5

6 BEST PRACTICES FOLLOWED IN PROCESS IMPROVEMENT

6.1 Introduced Risk Based Testing

Risk is the possibility of a negative or undesirable outcome, so a risk could negatively affect customer, user, or stakeholder satisfaction. Through testing, we can reduce the overall level of quality risk. Analytical risk-based testing uses an analysis of quality risks to prioritize tests and allocate testing effort. By introducing this approach we are able to achieve the following things

- Find out the important bugs earlier in test execution that reduces the risk of schedule delay
- Finding important bugs than unimportant bugs, reducing the time spent chasing trivialities.
- Provide the option of reducing the test execution period in the event of a schedule crunch without accepting unduly high risks

6.2 Parallel Execution

In our process improvement, we considered many parallelization techniques some of them are mentioned below

- Identified serially executing independent steps and modified them as separate program executing in parallel. This will reduce the wait time of the independent processes thereby allowing execution of independent steps in parallel
- Multi threading concepts introduced
- Executing multiple jobs on same machine using the instance concepts of Unix
- Modularizing complex steps: Separate the independent steps and avoid writing huge lines of code for performing similar tasks. Creating macros for performing repeated tasks simplifies the process and makes the

code easy to read. Moreover, it also reduces code redundancy and in case of a change to process, allows users to update code in only one location instead of many

- Reducing code redundancy: Removing unwanted code; use of functions and macros for performing repetitive tasks makes the code much simpler and easier to read. Also any processes, differing in only a certain parameter or a set of parameters but having all the other processing logic similar, need to be converted to a macro. This would make the coded simpler and more readable
- Syntactic optimization of the codes.

6.3 Use of Test Tracking Tool (TTT)

TTT, an IBM test tracking tool helped us to track the test status of each tester separately. TTT has multiple option there we can customize the tracking view based on our requirements. We used to follow a test point method; there we assigned test points to the scenarios based on the importance and duration of execution. This helped us to easily predict the duration of the test cycle

6.4 Test Management Using Test Point Method

Using this method, test manager will get a clear idea, how the testing is progressing. Based on the test points completion manager can take early decisions like, whether the testing will meet the project dead line, whether the team is overloaded, is there any extra resource needed, etc.. And he can make adjustments in manpower utilization based on the test point's completion. Initially we were facing problems for reporting daily progress of the testing, due to the incompletion of long running scenarios, tracking progress on each platform (product used to test on 20 + platforms) and also getting the correct report from each tester. This causes lot of confusion in test management for rotating resources to other works, also for re allocation of scenario to different testers. After implementing test point method and graphical report using test points, we are able to solve the all the test management issues

Below example shows, how a test manager is planning the testing using test point method

10 test points (TP) = 1 man day

After analyzing the selected scenarios of the test phase, manager got total test points of 560 TP

This means he need total 56 man days of execution. Based on test start date and end date manager can easily decide the number of testers need to allocate for this test cycle. For example the manager want to finish the execution in 28 days, he can allocate 2 people for this test phase

Time allotted for test completion = 28
Total test points in test cycle = 560
Number of test points need to cover in one day = 560/28 = 20 TP
Total tester needed = 20/10 = 2 person

Based on this calculation manager can easily monitor the test progress and if there any shortage in execution he can easily adjust the resources with below calculations

Planned number of test points completion on Nth day of execution = N * 20

Actual number of test points executed on Nth day = M
Difference in expectation = N*20 –M

Eg: - After 10 th day as per the plan we need to complete 10* 20 TP s. But actual number executed is only 140.
Difference in plan is 200-140 = 60 TP s

That means as per the plan, testing lagging 3 days.
Here test manager can change his plan by adjusting the days/resources etc.

This early planning will help the manager to avoid missing dead lines of test execution. Also with this approach he can prepare pictorial representation of test progress. See the below sample chart of execution. Using the below graph (Figure-6), manager will get a clear progress of the execution. Also he will get idea of total defects found in test phase . With this approach manager can handle any number of releases with out any management issues

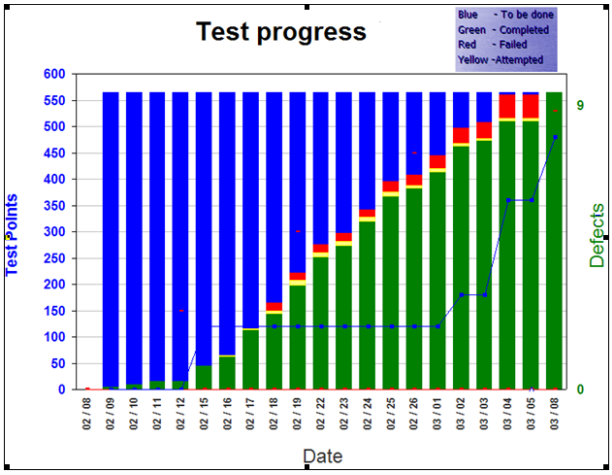

Fig. 6

7 MEASURABLE RESULTS

Based on the action taken to improve the test execution and management of indexing tool, team was able to address the challenges specified and actually benefited in terms of test effectiveness and productivity improvement. Overall, it improved the test management. Below are some of the significant benefits and improvements achieved by the team.

- Implemented new regression frame work and thus able to increase the testing coverage
- Reduced manual intervention for the system test and thus saved 30-50% system test execution time
- Introduced scheduled execution of the scenarios on different platforms
- Parallel execution of the scenarios helps better utilization of the test machines
- Log verification for all platforms can be done from a centralized screen
- Monitoring long running scenarios can be done from a central console
- Report generation
- Introduced new tool TIAT and able to address the issues with manual download and installation of drivers with high efficiency improvement

CONCLUSION

Based on the work done by the team in the test execution and management area, below are some of the recommendations which the author would like to highlight.

- Know your efficiency to know what to improve
- Institute risk based testing for catching defect in early test cycle
- Setup regression frame work as early as possible and move repeatedly executing system test scenarios to regression
- Introduce light weight test automation
- Use IBM Rational Build Forge , that you can easily integrates into your current environment and supports major development languages, scripts, tools, and platforms; allowing you to continue to use your existing investments while adding valuable capabilities around process automation, acceleration, notification, and scheduling.
- Automate your install verification test (IVT) so that you can run the IVT for each and every build with out manual intervention
- Introduce proper test tracking system using easily understandable graphical approach

ACKNOWLEDGMENT

I would like to offer my deepest gratitude to the following people for their help through out the preparation of this paper and case study
Mohammed Shaffi for giving me an opportunity for conducting the case study
Nikunja B Das for helping me in automation and his 'ready to help' attitude
V Balaji and Kayalvizhi Ganesan for her straightforward and constructive feedback on the article

.

REFERENCES

1. Cem Kaner, Jack Falk , Hung Quoc Nguyen, *'Testing Computer Software'* 2nd Edition , 2001 ,ISBN:81-7722-015-2
2. Boris Beizer , '*Software Testing Techniques*' , 1st Reprint Edition,2002, ISBN: 81-7722-260-0
3. Booz Allen Hamilton, Gary McGraw , '*Software Security Testing*' , IEEE SECURITY & PRIVACY , 2004 ,PP 1540-7993
4. 'Test Plan Template (IEEE 829-1998 Format),2001, Software Quality Engineering -Version7.0
5. TOSHIAKI KUROKAWA , MASATO SHINAGAWA, *'Technical Trends and Challenges of Software Testing'* , SCIENCE & TECHNOLOGY TRENDS, 2008 -QUARTERLY REVIEW No. 29
6. IBM Rational build forge V 7.13 – Information Center doc
7. Viraj Kumbhakarna, '*A Practical Approach to Process Improvement Using Parallel Processing*', PharmaSUG2011 - Paper PO03
8. Lars-Ola Damm, '*Evaluating and Improving Test Efficiency*' , Master Thesis ,Software Engineering , June 2002 ,Thesis no: MSE-2002-15