A SURVEY ON SOFTWARE TESTING BASED ON COST REDUCTION METHODS

[1]J.SRIVIDHYA, [2]DR. K. ALAGARS AMY
[1] Research Scholar, Department of Computer Science, Karpagam University
[2] Professors, Department of Computer Science, Madurai Kamaraj University
[1]jsrividhya.ku2011@gmail.com

Abstract: Nowadays testing is an important role of any software development process. Testing process is having large cost. The reduction of cost is main problem during software testing process. In basic testing is executing a system with a specific end goal to recognize any crevices, lapses or missing prerequisites in spite of the real desire or necessities. In this paper is includes survey of the basic concepts of software testing, different levels of testing in software development process for better understand. Finally we try to give some suggestions for efficient way reduces the software testing cost.

Keywords: Software testing, STG, OTC, STM, Cost reduction.

1. INTRODUCTION

Testing is the procedure of assessing a system or its component(s) with the plan to find that whether it fulfills the specified prerequisites or not. This movement brings about the genuine, expected and distinction between their outcomes. In basic testing is executing a system so as to distinguish any holes, blunders or missing prerequisites in spite of the real desire or necessities. Software testing is an integral part of the software development life cycle that span over all the development phases [1]. One of the main challenges in software testing is deploying and maintaining a real–world test platform at the outset of a project. As a rule, emulating experts are included in testing of a system inside their individual limits : Software Developer, Software Tester, Project Lead/Manager and End User.An early begins to testing decreases the cost, time to revamp and failure free software that is conveyed to the customer. However in Software Development Life Cycle (SDLC) testing might be begum from the Requirements Gathering stage and keeps ticking work till the development of the software. However it likewise realize on upon the development methods is,utilized. For instance in waterfall modelformal testing is directed in the testing stage, yet in incremental model , testing is performed at the end of each augmentation cycle and at the end of entireProvision is tried. Testing is carried out in diverse structuresat each period of SDLC like throughout requirement gathering stage, the investigation and checks of requirements are likewise vied as testing. Reviewing the configuration in the design stage with goal to enhance the design is additionally acknowledged as testing. Testing performed by a designer on culmination of the code is additionally sorted as Unit sort of testing. Following are the viewpoints which ought to be recognized to stop the testing: Testing Deadlines, Completion of experiment execution, Completion of Functional and code scope to a certain point, Bug rate falls beneath a certain level and no high necessity bugs are distinguished and Management choice. The section II of this paper presents research background of cost reduction methods of software testing. Section III of this paper we are give some suggestions for improve testing system with cost reduction methods. Section IV is conclusion of this paper.

1.1 Testing types

Manual testing.

This sort incorporates the testing of the software physically i.e. without utilizing any automated tools or any script. In this sort the analyzer assumes control over the part of an end client and tests the software to distinguish any unexpected conduct or bug. There are distinctive stages for manual testing like unit testing, Integration is testing, System testing and User Acceptance testing.

Automation testing.

Automation testing which is otherwise called Test Automation is the point at which the analyzer composes scripts and utilizes alternate software to test the software. This procedure includes automation of a manual methodology. Automation Testing is utilized to re-run the test situations that were performed physically, rapidly and more than once.

Black Box testing.

The procedure of testing without having any information of the inside workings of the requisition is Black Box testing. The analyzer is absent to the system construction modeling and does not have entry to the source code. Typically, when performing a black box test, an analyzer will associate with the system's client interface by giving inputs and inspecting yields without knowing how and where the inputsare worked upon.

Table 1: Black Box *t*esting

Advantages	Disadvantages
Well suited and efficient for large code segments.	Restricted Coverage since just a selected number of test situations are really performed.
Code Access not required.	Ineffective testing, because of the way that the analyzer just has constrained knowledge about an application.
Unambiguously divides client's point of view from the developer's viewpoint through obviously characterized parts.	Blind Coverage, since the analyzer can't target particular code portions or mistake inclined ranges.
Expansive amounts of tolerably skilled analyzers can test the application with no learning of execution, programming dialect or working systems.	The test cases are difficult to design.

- **White Box Testing.**

White box testing is the definite examination of inside rationale and structure of the code. White box testing is likewise called glass testing or open box testing. so as to perform white box testing on an application, the analyzer needs to have information of the interior working of the code. the analyzer needs to observe inside the source code and discover which unit/lump of the code is behaving improperly.

Table 2: White Box testing

Advantages	Disadvantages
As the analyzer has knowledge of the source code, it gets simple to figure out which sort of information can help in testing the application viably.	Because of the fact that a skilled analyzer is required to perform white box testing, the costs are increased.
It helps in optimizing the code.	Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems as many paths will go untested.
Extra lines of code can be removed which can bring in hidden defects.	It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.
Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased.

- **Grey Box Testing**

Grey Box testing is a strategy to test the application with restricted knowledge of the inner workings of an application. In software testing, the term the more you know the better conveys a great deal of weight when testing an application. Mastering the area of a system dependably gives the analyzer an edge over somebody with constrained space knowledge. Unlike black box testing, where the analyzer just tests the application's client interface, in Grey box testing, the analyzer has entry to design records and the database. Having this knowledge, the analyzer can better get ready test information and test situations when making the test arrangement.

Table 3: Grey Box *testing*

Advantages	Disadvantages
Offers joined profits of black box and white box testing wherever conceivable.	Since the access to source code is not available, the capability to head over the code and test coverage is restricted.
Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.	The tests can be redundant if the software designer has already run a test case.
In light of the constrained data accessible, a grey box analyzer can plan astounding test situations particularly around correspondence conventions and data type handling.	Testing each conceivable input stream is doubtful on the grounds that it might take a preposterous measure of time; consequently, numerous program ways will go untested.
The test is carried out from the perspective of the client and not the designer.	

Table 4: Black Box vs Grey Box vs White Box

Black Box Testing	Grey Box Testing	White Box Testing
The Internal Workings of an application are not needed to be known.	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
Testing is based on external expectations - Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

1.2 Levels of testing

It incorporates the diverse procedures that might be utilized while directing Software Testing. Following are the fundamental levels of Software Testing:

- **Functional Testing.**
- **Non-Functional Testing.**

FUNCTIONAL TESTING

This is a kind of black box testing that is focused around the details of the software that is to be tried. The application is tried by giving input and after that the outcomes are inspected that need to comply with the practicality it was expected for. Useful Testing of the software is led on a complete, coordinated system to assess the system's agreeability with its specified requirements. there are five steps that are included when testing an application for practicality.

Table 5: Functionality

Steps	Description
I	The determination of the functionality that the intended application is meant to perform.
II	The creation of test data based on the specifications of the application.
III	The output based on the test data and the specifications of the application.
IV	The writing of Test Scenarios and the execution of test cases.
V	The comparison of actual and expected results based on the executed test cases.

Unit Testing.

This kind of testing is performed by the designers before the setup is given over to the testing group to formally execute the experiments. Unit testing is performed by the particular engineers on the unique units of source code relegated areas. The objective of unit testing is to disengage each one some piece of the program and show that distinct parts are right regarding requirements and purpose.

Limitations of Unit Testing.

Testing can't get every single bug in an application. It is difficult to assess each execution way in every software application. The same is the situation with unit testing.

Integration Testing.

The testing of joined parts of an application to figure out whether they work rightly together is Integration testing. There are two strategies for doing Integration Testing: Bottom-up Integration testing and Top-Down Integration tests.

Table 6: Integration Testing Method

Integration Testing Method
Bottom-up integration This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
Top-Down integration This testing, the highest-level modules are tested first and progressively lower-level modules are tested after that.

System Testing.

This is the next level in the testing and tests the system as a whole. System testing is so essential due to the accompanying reasons:

- System Testing is the initial step in the Software Development Life Cycle, where the application is tried in whole.
- The application is tried completely to check that it meets the technical and functional requirements.
- The application is tried in an environment which is near the creation environment where the application will be installed.
- System Testing empowers us to test, confirm and approve both the business requirements and in addition the Applications Architecture.

Regression Testing.

The goal of Regression testing is to guarantee that a change, for example, a bug fix did not bring about an alternate issue being uncovered in the application. Regression testing is so paramount due to the accompanying reasons:

- Minimize the holes in testing when an application with progressions made must be tested.
- Testing the new changes to check that the change made did not influence any possible area of the application.
- Alleviates Risks when regression testing is performed on the application.
- Test coverage is expanded without trading off timetables.
- Increase pace to market the item.

Acceptance Testing.
By performing acceptance tests on an application the testing group will reason how the application will perform in fabrication. There are additionally lawful and contractual prerequisites for acceptance of the system.

Alpha Testing.
Unit testing, integration testing and system testing when joined together are known as alpha testing. Throughout this stage, the accompanying will be tried in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

Beta Testing.
In beta testing a specimen of the target group tests the application. Beta testing is otherwise called pre-release testing. In this stage the spectators will be testing the accompanying

- Users will install, run the application and send their feedback to the project team.
- Typographical failures, befuddling application stream, and even crashes.
- Receiving the feedback, the project team can alter the issues before releasing the product to the genuine client.
- The more issues you settle that resolve true client issues, the higher the nature of your application will be.
- Having a higher-quality application when you release to the overall public, will build client fulfillment.

NON-FUNCTIONAL TESTING.
Non-functional testing of Software includes testing the Software from the prerequisites which are non-functional in nature related yet critical a well, for example, execution, security, client interface etc. some of the imperative and normally utilized non-functional testing sorts are said as follows:

Performance Testing.
It is basically used to distinguish any bottlenecks or execution issues as opposed to discovering the bugs in software. There are diverse reasons which help in bringing down the performance of the software.

- Network delay.
- Client side processing.
- Database transaction processing.
- Load balancing between servers.
- Data rendering.

Performance testing is considered as the vital and compulsory testing type in terms of following features:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

Qualitative or Quantitative testing

- **Load Testing.**

A methodology of testing the behavior of the Software by applying most extreme load in terms of accessing software and controlling large input information. It is possible at both ordinary and peak load conditions.

- **Stress Testing.**

This testing sort incorporates the testing of Software conduct under abnormal conditions. Taking away the assets, applying load beyond the actual load limit is Stress testing. This testing might be performed by testing diverse situations, for example,

- Shutdown or restart of Network ports randomly.
- Turning the database on or off.
- Running different processes that consume resources such as CPU, Memory, server etc.

- **Usability Testing.**

Usability testing is a kind of black box testing to identify any errors and enhancements in the software by means of observing the user operation.

- **Security Testing-**

Security testing involves the testing of Software in order to identify any flaws from security and vulnerability point of view. Following are the main aspects which Security testing should ensure

- Confidentiality.
- Integrity.
- Authentication.
- Availability.
- Authorization.
- Non-repudiation.
- Software is secure against known and unknown vulnerabilities.
- Software data is secure.
- Software is according to all security regulations.
- Input checking and validation.
- SQL insertion attacks.
- Injection flaws.
- Session management issues.
- Cross-site scripting attacks.
- Buffer overflows vulnerabilities
- Directory traversal attaks.

Portability Testing.

Portability testing includes the testing of Software with intend that it should be re-useable and can be moved from one Software to another as well. Following are the strategies that can be used for Portability testing.

- Transferred installed Software from one computer to another.
- Building executable (.exe) to run the Software on different platforms.

Following are some pre-conditions for Portability testing:

- Software should be designed and coded, keeping in mind Portability Requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

Fuzz Testing-

Fuzz testing is often called fuzzing, robustness [4] testing or negative testing. This technique feeds random input to application. The main characteristic of fuzz testing,

- The input is random
- The reliability criteria
- Fuzz testing can be automated to a high degree.

Compiler testing

The aim of compiler testing [5] is to verify that the compiler implementation conforms to its specifications, which is to generate an object code that faithfully corresponds to the language semantic and syntax as specified in the language documentation.

Visual testing

The aim of visual testing is to provide developers with the ability to examine what was happening at the point of software failure by presenting the data in such a way that the developer can easily find the information he or she requires, and the information is expressed clearly.

Smoke and sanity testing

Sanity testing determines whether it is reasonable to proceed with further testing.Smoke testing is used to determine whether there are serious problems with a piece of software, for example as a build verification test.

Testing Based On Precode Artifacts

Testing techniques [3] can be based on precode artifacts, such as design, requirements, and architecture specifications. Techniques that use these precode specifications for tasks such as test-case planning and development can help improve the overall testing process.

- **Testing artifacts-**The software testing process can produce several artifacts.
- **Test plan-**A test specification is called a test plan.
- **Traceability matrix-**A traceability matrix is a table that correlates requirements or design documents to test documents. It is used to change tests when related source documents are changed, to select test cases for execution when planning for

regression tests by considering requirement coverage.

- **Test case-**A test case normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output, expected result, and actual result.
- **Test script-**A test script is a procedure, or programming code that replicates user actions.
- **Test suite-**The most common term for a collection of test cases is a test suite. The test suite often also contains more detailed instructions or goals for each collection of test cases.
- **Test fixture or test data-**All the test values and changeable environmental components are collected in separate files and stored as test data.
- **Test harness-**The software, tools, samples of data input and output, and configurations are all refered to collectivityas a test hames.

2. RESEARCH BACKGROUND

Bounded-exhaustive testing [2] is an automated approach that checks the code under test for all inputs within given bounds. It consists of three activities. First, the user describes a set of test inputs and provides test oracles that check test outputs. Second, the tool generates all the inputs, executes them on the code under test, and checks the outputs using the oracles. Third, the user inspects failing tests to submit bug reports or debug the code.

2.1 Sparse Test Generation (STG)

It reduces the time to first failure. The time that the user has to wait after starting a tool for bounded - exhaustive testing until tool finds a failing test.

2.2 Structural Test Merging (STM)

It reduces the total time for test generation and execution. In bounded-exhaustive testing users typically describe a test set with a large number of small tests. While we advocate considering test sets with a smaller number of larger tests.

2.3 Oracle-based Test Clustering (OTC)

It reduces the human time for inspection of failing tests. Bounded-exhaustive testing can produce a large number of failing tests, and a tester/developer has to map these failures to distinct faults to submit bug reports or debug the code under test.

3. SUGGESTIONS

- Closely work with developers, do some parallel testing with them as the product/feature is getting developed.
- Identify and eliminate non-testing activities that occur in the name of process, documentation, management, metrics etc.
- Analyze and profile every application under the portfolio to determine “stable” and “well tested” areas of the application. These areas should receive the least or no testing effort.
- Analyze the test scripts suite and remove redundant, worn out ones. Aim to reduce scripted test repository as small as you can.
- Review and reduce “regression testing” on the basis of “well tested/stable areas” of the application.
- Switch from resource intensive and highly scripted testing approach to highly improvisational exploratory /rapid testing approaches.
- Plan testing in small but frequent cycles (Session based exploratory testing approach) – reduce planning and management overheads.
- Analyze and reduce the usage of costly tool licenses - especially those do not help in testing directly (test management tools).
- Cut down on lengthy test plans, testing reports, dashboards – switch to simple but frequent test reporting.
- Simplify defect management process – reduce defect life cycle – resort to informal/quick defect communication

Four ways to reduce software testing cost without sacrificing quality

- Manage by walking around and listening.
- Identify and remove barriers to high performance.
- Speed the test process.
- Eliminate excess work-in-progress inventory

4. Conclusion

In this section we surveyed the field of software testing by providing cost reductionmethods. The software development process is including the testing process for

quality assurance. In this paper, we have presented a survey on software testing based on cost reduction methods. We provide the basic concept of testing and different levels of testing for better understanding of software. We have presented the suggestions to improve the efficient way reduces the software testing cost.

REFERENCES

[1] KorayIncki, Ismail Ari, HasanSozer, "A Survey of Software Testing in the Cloud, " IEEE Sixth International Conference on Software Security and Reliability Companion, DOI 10.1109/SERE-C.2012.

[2] Vilas Jagannath, Yun Young Lee, Brett Daniel, and DarkoMarinov, "Reducing the costs of Bounded-Exhaustive Testing," .

[3] Mary Jean Harrold, "Testing: A Roadmap, " In Future of Software Engineering, 22nd International Conference on Software Engineering, June 2000.

[4] Jovanovic, Irena, "Software Testing Methods and Techniques, " 2008.

[5] A.S. Boujarwah, K. Saleh, " Compiler test case generation methods: a survey and assessment, " Information and Software Technology 39 pp 617-625,1997.