IJCSE International Journal of Computer Sciences and Engineering Open Access

Review Paper Volume-2, Issue-5 E-ISSN: 2347-2693

A Study on Variations of Bottlenecks in Software Testing

S.Kannan[1]; T.Pushparaj[2]

[1] Department of Computer Applications, Madurai Kamaraj University, Madurai, Tamil Nadu, India
[2*]Department of Computer Applications, PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India
skannanmku@gmail.com; pusht82@gmail.com

www.ijcseonline.org

Received: 28/04/2014 Revised: 11/05/ 2014 Accepted: 22/05/May 2014 Published: 31/05/2014

Abstract-Software testing is a crucial activity in software development and it determines to completeness of a product. Being a crucial activity it needs to be carried out smoothly without any difficulties. But in reality it is difficult to carry out testing smoothly. There exists number of factors and issues that alter the smooth happening of testing. Those factors and issues are collectively called as bottlenecks[8]. This work aims at the process of identifying the bottlenecks that are very common to testing, bottlenecks that are application specific and testing type specific, analyse the reasons and provide recommendations that will make Software Testing[1] to happen without bottlenecks or with minimum bottlenecks. This work was carried out in two phases. Phase 1, was conducted to know and understand the existence of bottlenecks in software testing. Phase 2, deals with the analysis of bottlenecks and recommendations to avoid or minimize bottlenecks.

General Terms- Bottle Neck

Keywords- IST, UAT, SUT, SRS

I. INTRODUCTION

The paper work entitled **"Variations of Bottlenecks in Software Testing – An Analysis "**deals with the process of exploring, bringing out and analyzing the bottlenecks present is software testing. Followed by that suitable recommendations are provided based on the findings.

- Emphasizing the need for testing as an important activity in software development.
- Identification of Umbrella activities that has a direct impact on delivering a complete product.
- Identification of bottlenecks[42] that are very common to any type of testing and any kind of product (Software) developed.
- Identification of bottlenecks that are specific to the type of testing and specific to the kind of product developed.
- Analysing the reasons for the existence of these bottlenecks based upon the data collected.
- Providing suitable recommendations to avoid or to minimize the presence of bottlenecks[3].
- Identification of further scope of enhancing the work.
- Identify the various bottlenecks present is software testing.
- Identify the bottlenecks that have the origin in the early phases of software development life cycle.
- Analyse the reasons for the existence of bottlenecks.
- Classify the bottlenecks that are very common and specific to any kind of product (software) and any type of testing.
- Co-relate the reasons that act as the source for various bottlenecks.
- Identify the need for the proper selection, orientation and employment of Test Engineers[24].
- Emphasize the need for following engineering principles in software development.
- Making the software team to understand the need for providing a friendly testing environment.
- Making recommendations to avoid or to minimize bottlenecks.

II. LEVELS OF TESTING

Unit testing [UT]

Unit testing[5] is the process of testing a program module and running it in isolation from the rest of the software product by using prepared input and comparing the actual results with the results predicted by the specifications and design of the module.

Integrated System Testing [IST]

Integrated System Testing (IST) is a systematic technique for verifying the construction of the overall software structure. It is a set of activity that ensures, that the software correctly implements a specific function. IST[34] tests the conformance of the software to specifications.

User Acceptance Testing [UAT]

User Acceptance Testing (UAT) is performed by users or on behalf of the users to ensure that the application developed conforms to business requirements[43]. It is a set of activity that ensures, that software built is traceable to customer requirements.

Difference between IST and UAT

Particulars	IST	UAT
Base line document	Functional Specification	Business Requirement
Data	Simulated	Live Data
Environment	Controlled	Simulated Live
Orientation	Component	Business
Tester composition	Testing Firm	Testing Firm / users
Purpose	Verification	Validation

Corresponding Author: *T.Pushparaj*

III. TYPES OF TESTING

1.1 White-box Testing

Tests are based on coverage of internal code statements, branches, paths and conditions. Also known as glass-box testing[20].

1.2 Black-box Testing

Testing focuses on the functional aspects of the application. It enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

1.3 Regression Testing

Each time a new module is added as a part of integration testing, the software changes. Regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

1.4 Recovery Testing

This is a system test that forces the software to fail in variety of ways and verifies that recovery is properly performed. If recovery is automatic (performed by the system itself), re-initialization, checkpoint mechanisms[9], data recovery and restart are evaluated for correctness. If recovery requires human intervention, the mean-time-to-repair (MTTR) is evaluated to determine whether it is within acceptable limits.

1.5 Security Testing

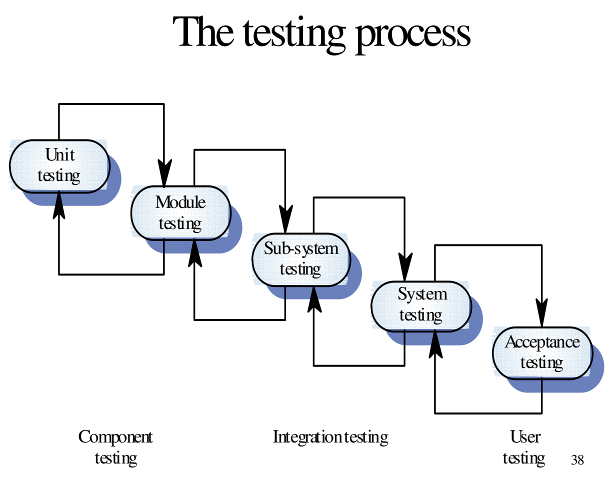
This testing attempt to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.

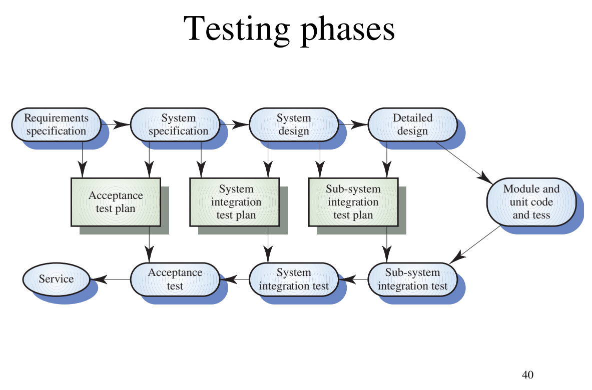
1.6 Compatibility Testing

The purpose of compatibility testing[10] is to evaluate how well software performs in a particular hardware, software, operating system, browser, or network environment.

1.7 Performance Testing

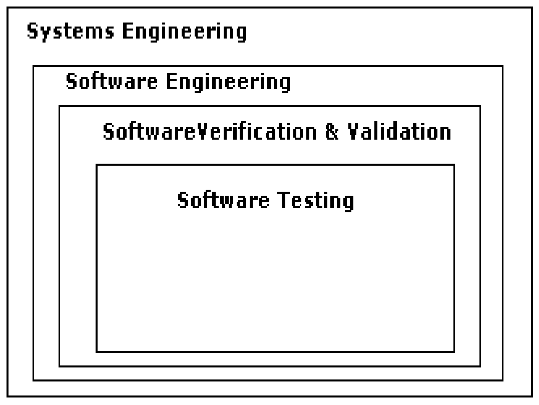
Performance testing[5] is designed to test the run-time performance of software within the context of an integrated system.




Testing phases
Requirements specification
System specification
System design
Detailed design
Module and unit code and tess
Acceptance test plan
System integration test plan
Sub-system integration test plan
Service
Acceptance test
System integration test
Sub-system integration test
40

IV. PUTTING SOFTWARE TESTING IN CONTEXT

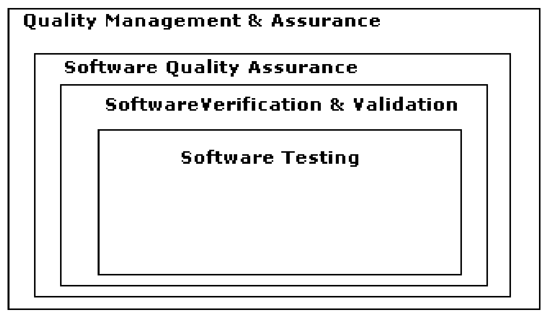
Software testing[6] is the process of exercising a software product to verify that it satisfies the specified requirements and to detect errors can be viewed as a part of system engineering. The relationship is shown in the following figure.



The Process Context of Software Testing

Quality context of Software Testing

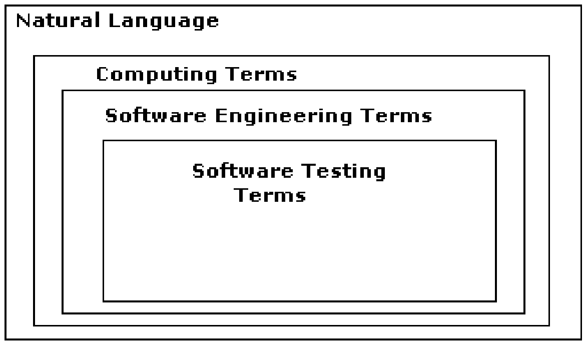
From a quality perspective, testing as part of verification and validation can be seen as an integral part of SQA[39]. If software is a part of a large system, then testing can also be considered as a part of overall quality assurance. Following is a diagrammatic representation of the same.


Quality Management & Assurance
Software Quality Assurance
SoftwareVerification & Validation
Software Testing

The Quality Context of Software Testing

Terminology context of Software Testing

This is a representation of software testing with a terminology perspective. It contains natural language at the highest level. Computing terms and software engineering terms are at the lower levels.



Terminology Context of Software Testing

V. CHALLENGES IN AUTOMATED TESTING

1. Test Tools Selection

Test tool selection[5] is a critical factor in the success of test automation. This requires the study of the scope of testing and test strategy and then selection of the right test tool, to meet the requirements of automating test-suite for a particular product and release. The factors to be taken into consideration while selecting a tool are reusability, reliability and cost.

A test tool should essentially support

- scripting Interface
- facility to give invalid input
- facility to Control IUT[17] (Implementation Under Test) & Peer
- result comparison & verdict declaration facility
- The level of automation targeted should be done along with the test tool selection

2. Customization of the Tool

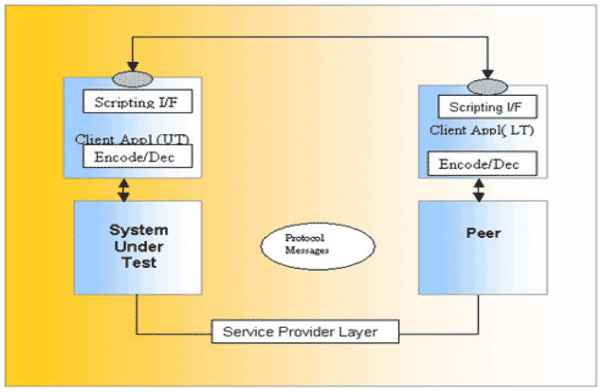
The Test Manager should always look for standard tools if available in the market, and go customize the existing tools[32]. New tools should be developed only as a last resort. Customization[9] is a most viable option with provisions for enhancement.

3. Development and Verification of Scripts

Scripting[5] effort includes development and testing of scripts. This depends upon the two factors. First, skill of the persons involved and secondly, the capability of the test tool to provide flexibility to develop scripts for all valid and invalid scenarios. However, it is recommended that scripting should always be done in a modular manner so that the same modules can be reused in different scripts. It is always advised that all scripts should be tested with a release prior to the actual test execution so that during the real testing no problems are found in scripts.

4. Implementation of a Test Management system

This is to facilitate complete automation in terms of managing the scripts at the System Under Test (SUT) and tester simultaneously and for consolidating the test results at one place[30].



VI. REAL – LIFE SITUATIONS

Following is a set of real-life situations taken into consideration for analysing and finding the bottlenecks pertaining to software testing. Bottleneck[28] is anything that hinders the process of software testing.

Bottlenecks common to any testing process.

a) One bottleneck is the change in specifications by the client. (Functionality testing bottleneck)
b) When the development time is more than expected, it becomes a general tendency to reduce the testing time. Testers are asked to test within a short duration of time. (General testing bottleneck)
c) Insufficient information in the SRS or test cases without proper coverage can be a bottleneck at times. (Functionality testing bottleneck)
d) When regression testing[17] is done, generally, the test cases and test plans may not be clear. When the person first develops it, it is understandable by him, but at a later point of time, the newer testers may not understand them.
e) Test plans[14] and test cases may not be updated. The tester will be testing it on a different version and the test fails because, the test case was for a previous version. Though the software has failed the test, the software is still working fine. This becomes an issue and it is sent back to the developer. He comes out with the result that it is correct and the test case is wrong.
f) One or two developers make more than acceptable mistakes. In that case, though the bug is found, the testers generally do not report all of them on the same day, because, doing so will reflect on the developer.
g) Difference in the development and execution environment. Testing may be done in a different version and development would have been in a different version. Thus the software fails. For example, a data migration tool was developed using JDK1.4[22], but the installations were done in JDK 1.2. The software did not even get installed in it.
h) Generally, testers are not familiar with the technology. Software or a module that is developed using a new technology always takes larger time to develop and to test.
i) The main problem between testers and developers is due to lack of communication. Sometimes, the developers see the testers as their adversaries. Lack of

professionalism among the testers may become a bottleneck for testing in this case.

VII. PERFORMANCE TESTING RELATED BOTTLENECK – DATA MIGRATION TOOL

When a software application is newly installed in a client location, a particular tool for migrating legacy database of the client, to the software's Database is used. This tool works correctly and the tool can handle a maximum of 10,000 records per minute. But, one of the recent clients, (perhaps the largest to date) (name is not mentioned because it is confidential) had a very huge database and when this tool was used, performance of the tool was very poor. It took five hours to migrate a single database. There are 500 such databases. So, the client will take 2500 hours to migrate everything. It will also take an additional hundred hours for setup. (325 days). This is a performance issue. This issue was found, when the tool was subjected to performance testing before giving it to the client. Now, this has become a high priority issue. This client uses ASA (Sybase) database in the legacy system[40] and the new system.

Now, the performance related issues that should be addressed are:
1. The database that took five hours for migration was ASA (Sybase) database and ASA[37] database generally resides in the client machine only. So, if ASA alone is going to be used, it is enough if the configuration of the machine in which the migration is run alone is upgraded.
2. Even if a dedicated server is set up, the speed of migration is dependent upon the speed of the network.
3. If the ASA database and the application run on the same machine it is faster than what it will be if the ASA DB and application run on different machines.
4. If a dedicated DB server has to be used, what DB should it run is an important concern, because there are three different Databases in use, Oracle, DB2 and Sybase?

VIII. DATABASE TESTING BOTTLENECK

Database is a major component of any information system. It is also called as the driving force of an information system. Database's performance plays an important role in the working of an information system. Database testing[16] is done in various ways. The major activity is Performance Tuning. Apart from that the verification and validations to be done is also important. Database tests[26] are supported via ODBC using the following functions: SQLOpen, SQLClose, SQLError, SQLRetrieve, SQLRetrieveToFile, SQLExecQuery, SQLGetSchema and SQLRequest.

Cursor type operations[12] can be carried out by incrementing arrays of returned datasets. All SQL queries are supplied as a string. Stored procedures can be executed.

In Database testing we need to check for
1. The field size validation[19]
2. Check constraints.
3. Indexes[25] are done or not (for performance related issues)
4. Stored procedures[35]
5. The field size defined in the application is matching with that in the database.

The database component is a critical piece of any data-enabled application. Today's intricate mix of client-server[31] and Web-enabled database[21] applications are extremely difficult to Test productively. Testing at the data access layer is the point at which the application communicates with the database. Tests at this level are vital to improve not only the overall Test strategy, but also the product's quality.

IX. CLIENT / SERVER APPLICATION RELATED BOTTLENECK

In a client server environment a client may get its requested processed by a server. At a particular moment a client may request and utilize a set of services from the server application. For this purpose, a linking file / thread will be maintained. This thread maintains the list of active applications / services requested and used by a client. The thread should maintain the programs that are currently in use, others should be removed from the active list by properly disconnecting them. Normally the developers / programmers miss this concept. This leads to a major performance bottleneck in-term of poor performance and more response time from the server, sometimes leading to server hangovers and crashes. This issue will be noticed only during User-acceptance testing[13] carried out with real data which is huge in amount and invokes all the services provided by the server.

X. REASONS OF BOTTLENECKS IN SOFTWARE TESTING

Bottlenecks occur in software testing for various reasons. Those bottlenecks are classified into the following four categories. Process Related[11], People Related, Planning Related and Technology related bottlenecks.

1. **Process Related:** It contains the bottlenecks that arise from the Software Development Life Cycle activities like Analysis, Design, Coding and other Umbrella Activities.
2. **People Related:** It contains the bottlenecks that arise because of personnel related issues. The success of any task depends upon the members to whom the responsibility of doing that is given. Good understanding is a major factor that breaks any obstacles in the path of success.
3. **Planning Related:** It contains the bottlenecks that have their origin from planning related mistakes. Any task carried out without a proper plan will fail miserably. Being a customer centric industry, software development is more dependent on plans.
4. **Technology Related:** It contains the bottlenecks[19] that arise because of technology related issues like non-availability of required environment, tools and so on.

Software Testing contains more number of Process and People related bottlenecks. Most of the bottlenecks are interdependent.

1. Process Related: Process related bottlenecks have their origin from the early phases of SDLC[7] like requirements gathering, SRS preparation[8] and other background or umbrella activities like project planning, scheduling, test planning and team structure formation.

1.1 Incomplete SRS: System Requirement Preparation[12] is the very basic activity that needs utmost care. If the fact finding methods are not adopted properly, then the resultant will be an incomplete and incorrect specification. Even effective validations[35] and reviews cannot identify these errors. Requirements that are wrongly specified will serve as the root cause for further problems. Testing a product that is developed by using an incomplete base will be definitely very tedious. It will result in lot of rework that will almost lead to re-initiation of the work from the scratch.

1.2 Lack of Communication: In most of the cases the communication gap between the client and the developer (Team) will be enormous. This gap needs proper attention[10]. If proper steps are not taken to remove or reduce the gap, the result will be a set of activities that are highly ambiguous in nature. These confusions will come right from the user specification to test cases.

1.3 Improper selection and usage of tools: The decision making activity[9] in the selection and usage of tools that are relevant to the different activities of software development like fact finding techniques, design and test tools will have a major impact.

1.4 Bad source code: The source code[8] that is developed by violating prescribed coding format and practices may result in a solution doing the intended function by consuming the resources unnecessarily, leading to an unmanageable code. Such a bad code will generate enormous bottlenecks at the time of testing.

1.5 Module not doing its intended function: A module[41] that is expected to do a particular function won't do so. This happens because of many reasons. But the test engineers will suffer a lot because of the mismatch between the specification and the implementation.

1.6 Improper integration of modules: If modules[32] that produce expected results are wrongly integrated, then the process of testing it to identify the mistake committed while integrating will be a major bottleneck.

1.7 Regression test Limits: Limits should be set for regression tests. Testing carried out even after reaching the prescribed limits will make the testing process to start as a fresh one. This will waste the resources in all aspects. Such a situation is an important bottleneck in software testing.

1.8 Lack of formal testing procedures: If the procedure adopted for testing software is not framed and followed in a formal way, chances of facing bottlenecks are more. This is because a test procedure that is not in a standard form will pave way for unwanted headaches.

1.9 Absence of system or functional specification: Software Requirements Specification is the base using which all the other phases of SDLC[31] can precede. Other activities of software development will get affected badly if such a base is not available. Testing will be a difficult task if performed without a proper system specification.

1.10 In-correct test cases: Test Case[37] is a triplet, that states the input, state and output of a system subject to testing. If these test cases are not designed properly, then the results may not be as expected. This leads to major setbacks in testing

1.11 Ambiguous test conditions: Test conditions that contain ambiguity[11] will misguide the test engineers. They won't be able to achieve their goals.

1.12 Frequent changes in requirements: Requirement specification[19] that gets altered due to frequent changes in customer requirements will affect the early activities of testing process very seriously. Frequent change request[8] is a major problem faced by test planners.

1.13 Implementation Compromises: Compromises that are made during the development stage in-order to meet the deadlines, to minimize resource usage, etc will create lot of bottlenecks in testing phase.

2. People related: Software development is teamwork. There exists a need for human beings in-spite of the availability of CASE Tools[17]. People related issues have its own impact on the success rate of any activity. Software testing has no exemption from this. Human Resources related issues also play a major role in determining the success of software testing.

2.1 Test engineers having no exposure to the work domain: People recruited, as test engineers should have prior hands-on exposure in the related work domain. Otherwise the process of training and making the people to get accommodated to the work environment will be a major bottleneck.

2.2 Misunderstood facts: "To err is human". Actual facts that are misunderstood by testers will direct them to perform activities in a wrong direction. This is applicable to the people involved in requirements gathering and specification development also. Realizing and correcting misunderstood facts is a major bottleneck.

2.3 Lack of objectives: The members of testing teamneeds a proper orientation[12] about the task assigned to them. If this is not done properly, then testers won't be aware of the expected results. This will make them to act as they like by moving away from the planned path. The process of identifying and taking corrective measures after a period of time is a cumbersome task.

2.4 Test engineers doing code coverage tests: Code coverage[14] is an activity that has to be performed by the person who developed it. Making test engineers to perform is not advisable. The reason is that an outsider cannot perform code coverage test without knowing the logic applied to arrive at the solution.

3. Planning related: The success of any activity with an objective is determined by the planning done in order to make things to happen as expected. Activities that are not planned properly won't yield desired results. Also the process of rectifying an outcome of a planned activity is very difficult to achieve.

3.1 Poor quality test plan: Test plan[23] is a document that serves as a guideline for carrying out testing smoothly. Quality of the test plan is a major factor that determines the smooth happening of software testing. Testing performed with a poor quality test plan will lead to lot of problems.

3.2 Usage of tools in a wrong manner: Software testing tools are meant for enhancing the productivity of a tester. Any tool that is not used properly will definitely damage the work done. The process of restoring the normal state after a wrong usage of a tool is a very difficult process.

3.3 Unorganized team structure: Team formation[38] is an important planning activity having a direct impact on the successful outcome of any activity. If proper planning is not done for identifying, selecting and forming team, the chances of getting into troubles are more. A team containing people with different perceptions about the assigned task and goal will have very low or no productivity at all. The code created by the members of such a team will be of bad quality. An unorganized testing team will also create problems leading to bottlenecks. Most of the bottlenecks created because of unorganized team structure are related to human factors, lack of understanding between members, lack of orientation about the assigned task, ego and other relates aspects.

3.4 Narrow Deadlines: Any activity possessing a goal, guidelines and a deadline[42] will lead to success. Proper understanding of the problem and resource requirements will enable the planner to identify a possible date and time for completing the task.

If deadlines are very short than the actual one, implementation compromises may occur, leading to poor quality code. Such compromises are the main source of bottlenecks for the quality assurance personnel. Test engineers will get pressure from the project head to finish testing in a short period of time because of the delay caused in the previous phases of SDLC.

3.5 Lack of training to team members:
Members of a testing team should have some prior knowledge about the work they are going to do. If team members are selected in the last minute without giving importance to their competency in the work to be allotted, their productivity will be very less. Sometimes having untrained people will create bottlenecks like misinterpretation of the problem[19], wrong usage of the tools and resources and lack of co-operation among the team members.

3.6 Target user not known: Testing team members should be made aware of the target users of the system under development. Providing complete information about the product, its target user etc will help the tester to have a better understanding about their task. In most of the cases, the information about the target user of the proposed system is either provided partially or not at all provided to the testing team members. This will make the test engineers to have their own assumptions about the target user and will proceed according to this. Such a proceeding will bring a mismatch between the actual system's features and the assumed features of the tester leading to unwanted and harmful bottlenecks[8].

4. Technology Related: Software testing is an activity that is more dependent on the technology used. Technology related issues like meeting the performance requirements, availability and stability of necessary hardware and environment are some of the factors that decide the success rate of testing[11].

Performance Bottlenecks: Performance testing[9] is conducted to measure the performance related issues of the software. Availability of the required environment, memory usage, etc., are the major factors. Performance tests conducted in an environment with insufficient resources will make the test engineers to face lot of bottlenecks.

Hardware Failures: A hardware failure during testing process is a major bottleneck[8]. This will lead to content loss, time loss, resource damage etc. This may happen because of executing the software in an environment having insufficient resources, improper handling of resources, damaged hardware elements etc.

Problems in simulation: Simulated software[25] like numeric controllers, flight simulators etc needs an environment that resembles the real-time situation. Creating and maintaining an exact replica of the real-time environment is a difficult task. Test engineers may have some constraints in working in such an environment.

XI. CONCLUSION

From the above mentioned findings, it is very clear that software testing[41] is an important activity in the software development process and it has got a strong relation with the other phases of software development. These phases may be both the core and umbrella activities. Making the members of the software development team to i) understand the importance of delivering high quality software and their responsibilities in the concerned phases of software development, ii) posses interest and involvement in the responsibilities assigned and iii) have a broader and in-depth view of he over-all objective of delivering a high quality product that meets the user requirements will definitely make the software development activity in general and software testing in particular a cake-walk[23].

This work explains the bottlenecks related to software testing, the reasons for the occurrence of such bottlenecks and the possibilities of avoiding such bottlenecks[8]. This report also analyses the causes for such bottlenecks and provides suggestions to avoid the bottlenecks.

The growth of any filed is purely based on the research made in that field. The filed of Computer Science will never go for any end process, it will always continue forever. This work can be extended to develop a set of guidelines for carrying out software testing in a smooth manner without any major bottlenecks. These guidelines will be based on the four major classifications of the bottlenecks identified and discussed here.

REFERENCES

[1]. Brain Marick, Classic Testing Mistakes, RSTAR97 conference.
[2]. Amit Paradkar, Automated Generation of Self-Checking Function Tests, ISSRE, July 2002.
[3]. Clay Williams, Thresa Kratschmer, A Technique for Generating Optimized System Test Suites, FSE 10th Symposium, November 2002.
[4]. B.Hailpern and P.Santhanam, Software Debugging, testing and verification IBM Systems Journal, Vol. 41, No. 1, February 2002.
[5]. Hongxia Jin and P.Santhanam, An approach to higher reliability using software components, ISSRE, November 2001.
[6]. Clay Williams and Amit Paradkar, Efficient Regression Testing of Multi-Panel Systems, ISSRE, November 1999.
[7]. Jeff Dunmall and Keith Clarke, Real-World Load Testing Tips to Avoid Bottlenecks when Your Web Apps Goes Live, MSDN , January 2003.
[8]. Jeff Straathorf, Load Testing Intranet Applications Finding Hidden Bottlenecks, www.pureatria.com.
[9]. [9] Nina Godbole, Try Not to Make these Classic Mistakes, Information Technology October 2000.
[10]. Mark Johnson,How to do good testing – an interview, STQE Magazine.
[11]. Doug Nickerson, Software Quality Assurance – not for Dummies, www.computerbits.com/archive/2000
[12]. BS 7925-1-1998, Software Testing Vocabulary
[13]. IEEE 610, Standard Computer Dictionary.
[14]. IEEE Std 1028-1997, Standard for software reviews.
[15]. www.stickyminds.com
[16]. www.rational.com
[17]. www.mercury-interactive.com
[18]. www.vlib.org
[19]. www.thinkquest.org
[20]. www.reality-test.com
[21]. www.riceconsulting.com
[22]. www.aptest.com
[23]. www.worksoft.com
[24]. www.computer.org
[25]. www.sqatester.com
[26]. www.stqemagazine.com
[27]. www.softwareqatest.com
[28]. www.rspa.com
[29]. www.qalinks.com
[30]. www.ibm.com
[31]. www.sdmagazine.com
[32]. www.cmu.edu
[33]. www.tamingthebeast.com
[34]. www.testing.com
[35]. www.unicomm.co.uk
[36]. www.testingreflections.com
[37]. www.perftestplus.com
[38]. http://www.sisqa.com/SISQA_downloads.htm
[39]. http://weblogs.asp.net/rosherove/articles
[40]. http://www.theserverside.com/articles
[41]. Roger S.Pressman, Software Engineering a Practitioner's Approach, TMH
[42]. W.E.Howden, A functional Approach to Program Testing and Analysis, IEEE Trans, Vol. SE- 12, No.10, October 1986
[43]. Mordechai Ben-menachem / Garry s.Marliss, Software Quality Producting Practical, Consistent Software, Thomson Learning

AUTHORS PROFILE

Prof. T.Pushparaj M.C.A., M.Phil., M.B.A., (Ph.D)., is working in PSNACET. He has rich experience in handling System Software, DBMS, Computer Organization, Software Engineering, Software Project Management. His hobbies are playing cricket, reading books and listening melodies. He is a member in ISTE, IAENG. He has rich teaching experience of 7 years from a reputed esteemed Institution. He is currently pursuing Ph.D in M.K.U in the area of Software Testing.