www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume - 3 Issue -9 September, 2014 Page No. 8188-8197

A Review On Software Testing In SDlC And Testing Tools

T.Amruthavalli*, S.MahaLAkshmi*, K.HariKrishnan*

Assit Prof, SKR Engineering college. amrutha1201@gmail.com

Assit Prof, SKR Engineering college. maha3088@yahoo.com

UG Scholar,SKR Engineering College.hari1993skr @gmail.com

Abstract- The paper reviews the software testing and the different tools used for testing the software.The testing is based on the methods of its attributes,security and its usability.Testing is an important part in software engineering where the coding gets deployed based on the testing.The testing can be done with different parameters of different types .I do not mean to give here a complete survey of software testing.Rather I intend to show how unwieldy mix of theoretical and technical challenges faced by the testers between the state of art and practice.

Keyword-Software testing,Testing and debugging,Testing methods,Testing levels.

1.Introduction:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs (errors or other defects)[1].It involves the execution of a software component or system to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test. As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects).Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

2.Software Testing Levels:

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process[2].

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Black box testing – Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

White box testing – This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

Unit testing – Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses.

Incremental integration testing – Bottom up approach for testing i.e continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately. done by programmers or by testers.

Integration testing – Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

Functional testing – This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.

System testing – Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

End-to-end testing – Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

Sanity testing - Testing to determine if a new software version is performing well enough to accept it for a major testing effort. If application is crashing for initial use then system is not stable enough for further testing and build or application is assigned to fix.

Regression testing – Testing the application as a whole for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.

Acceptance testing -Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer do this testing to determine whether to accept application.

Load testing – Its a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

Stress testing – System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

Performance testing – Term often used interchangeably with 'stress' and 'load' testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

Usability testing – User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

Install/uninstall testing - Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment.

Recovery testing – Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

Security testing – Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

Compatibility testing – Testing how well software performs in a particular hardware/software/operating system/network environment and different combination s of above.

Comparison testing – Comparison of product strengths and weaknesses with previous versions or other similar products.

Alpha testing – In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

Beta testing – Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

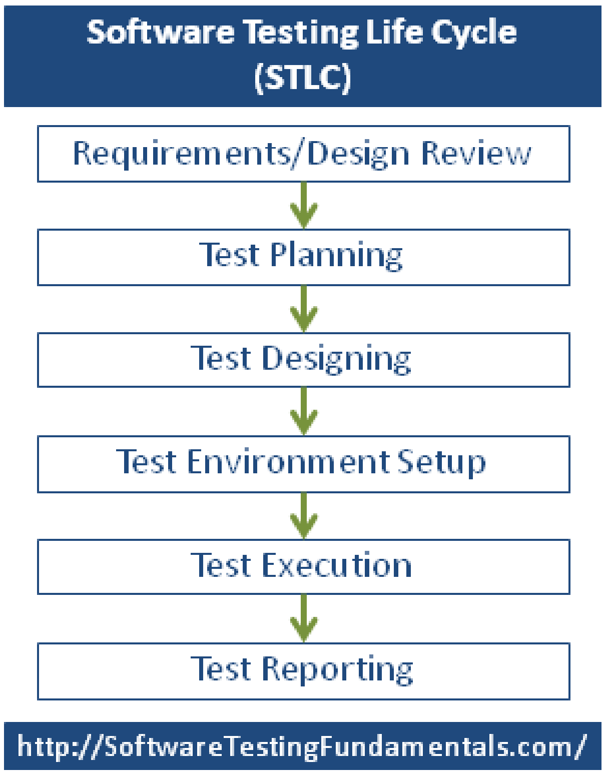

Software Testing Life Cycle (STLC)
Requirements/Design Review
Test Planning
Test Designing
Test Environment Setup
Test Execution
Test Reporting
http://SoftwareTestingFundamentals.com/

Fig:1 software testing fundamentals

3.Phases of Testing :

Phase	**Activity**	**Deliverables**	**Necessity**
Requirements/Design Review	You review the software requirements/design (Well, if they exist.)	▪ Review Defect Reports	Curiosity
Test Planning	Once you have gathered a general idea of what needs to be tested, you 'plan' for the tests.	▪ Test Plan ▪ Test Estimation ▪ Test Schedule	Farsightedness
Test Designing	You design/detail your tests on the basis of detailed requirements/design of the software (sometimes, on the basis of your imagination).	▪ Test Cases/ Test Scripts/Test Data ▪ Requirements Traceability Matrix	Creativity
Test Environment Setup	You setup the test environment (server/client/network, etc) with the goal of replicating the end-users' environment.	▪ Test Environment	Rich company
Test Execution	You execute your Test Cases/Scripts in the Test Environment to see whether they pass.	▪ Test Results (Incremental) ▪ Defect Reports	Patience
Test Reporting	You prepare various reports for various stakeholders.	▪ Test Results (Final) ▪ Test/Defect Metrics ▪ Test Closure	Diplomacy

		Report ▪ Who Worked Till Late & on Weekends Report	

4.Testing Roles:

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in the context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing[3]. In most cases, following professionals are involved in testing of a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Different companies have difference designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, and QA Analyst etc.

It is not possible to test the software at any time during its cycle. The next two sections state when testing should be started and when to end it during the SDLC.

4.1 Testing time:

An early start to testing reduces the cost, time to rework and error free software that is delivered to the client. However in Software Development Life Cycle (SDLC) testing can be started from the Requirements Gathering phase and lasts till the deployment of the software. However it also depends on the development model that is being used. For example in Water fall model formal testing is conducted in the Testing phase, but in incremental model, testing is performed at the end of every increment/iteration and at the end the whole application is tested.

Testing is done in different forms at every phase of SDLC like during Requirement gathering phase, the analysis and verifications of requirements are also considered testing. Reviewing the design in the design phase with intent to improve the design is also considered as testing. Testing performed by a developer on completion of the code is also categorized as Unit type of testing.

Unlike when to start testing it is difficult to determine when to stop testing, as testing is a never ending process and no one can say that any software is 100% tested. Following are the aspects which should be considered to stop the testing:

- testing Deadlines.
- Completion of test case execution.
- Completion of Functional and code coverage to a certain point.
- Bug rate falls below a certain level and no high priority bugs are identified.
- Management decision.

5.Testing, Quality Assurance and Quality Control:

Most people are confused with the concepts and difference between Quality Assurance, Quality Control and Testing. Although they are interrelated and at some level they can be considered as the same activities, but there is indeed a difference between them. Mentioned below are the definitions and differences between them:

S.N.	Quality Assurance	Quality Control	Testing
1	Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software.

2	Focuses on processes and procedures rather then conducting actual testing on the system.	Focuses on actual testing by executing Software with intend to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3	Process oriented activities.	Product oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a corrective process.
5	It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

6.Testing and Debugging :

TESTING:

It involves the identification of bug/error/defect in the software without correcting it. Normally professionals with a Quality Assurance background are involved in the identification of bugs[4]. Testing is performed in the testing phase.

DEBUGGING:

It involves identifying, isolating and fixing the problems/bug. Developers who code the software conduct debugging upon encountering an error in the code. Debugging is the part of White box or Unit Testing. Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs.

6.1Testing Types :

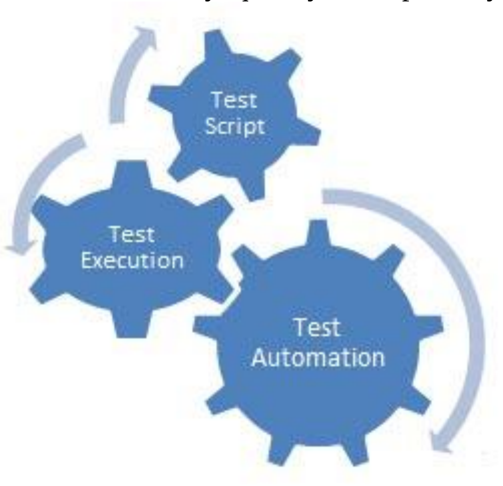
Manual testing

This type includes the testing of the Software manually i.e. without using any automated tool or any script. In this type the tester takes over the role of an end user and test the Software to identify any un-expected behavior or bug[5]. There are different stages for manual testing like unit testing, Integration testing, System testing and User Acceptance testing.

Testers use test plan, test cases or test scenarios to test the Software to ensure the completeness of testing. Manual testing also includes exploratory testing as testers explore the software to identify errors in it.

Automation testing

Automation testing which is also known as *Test Automation*, is when the tester writes scripts and uses another software to test the software. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly and repeatedly.



Apart from regression testing, Automation testing is also used to test the application from load, performance and stress point of view. It increases the test coverage; improve accuracy, saves time and money in comparison to manual testing.

7.Testing Methods:

Black Box Testing

The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

White Box Testing

. In order to perform white box testing on an application, the tester needs to possess White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also knowledge of the internal working of the code[6].

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Grey Box Testing

Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the term *the more you know the better* carries a lot of weight when testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to better prepare test data and test scenarios when making the test plan.

Functional Testing

This is a type of black box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional Testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements[7].

There are five steps that are involved when testing an application for 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.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

Unit Testing

This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is separate from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Integration Testing

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top Down Integration testing.

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

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in customers' computers, systems and network.

System Testing

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

System testing is so important because of the following reasons:

- System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment which is very close to the production environment where the application will be deployed.
- System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

Regression Testing

Whenever a change in a software application is made it is quite possible that other areas within the application have been affected by this change. To verify that a fixed bug hasn't resulted in another functionality or business rule violation is Regression testing. The intent of Regression testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application.

Regression testing is so important because of the following reasons:

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the change made did not affect any other area of the application.
- Mitigates Risks when regression testing is performed on the application.
- Test coverage is increased without compromising timelines.
- Increase speed to market the product.

Acceptance Testing

This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client.s requirements. The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or Interface gaps, but also to point out any bugs in the application that will result in system crashers or major errors in the application.

By performing acceptance tests on an application the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

ALPHA TESTING

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested 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

This test is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.

On the other hand Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end user. UI testing can be considered as a sub part of Usability testing.

Security Testing

Security testing involves the testing of Software in order to identify any flaws ad gaps 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 attacks.

Portability Testing

Portability testing includes the testing of Software with intend that it should be re-useable and can be moved from another Software as well[8]. 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.

Portability testing can be considered as one of the sub parts of System testing, as this testing type includes the overall testing of Software with respect to its usage over different environments. Computer Hardware, Operating Systems and Browsers are the major focus of Portability testing. 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.

8.Software Testing Documentation:

Testing documentation involves the documentation of artifacts which should be developed before or during the testing of Software.

Documentation for Software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing etc. This section includes the description of some commonly used documented artifacts related to Software testing such as:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

Test Plan

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A test plan will include the following.

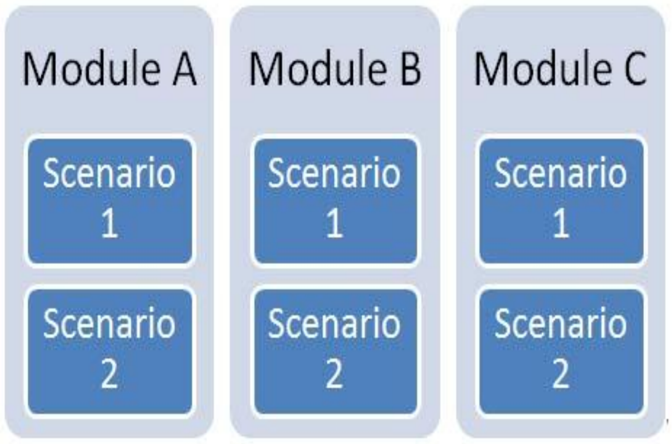
- Introduction to the Test Plan document
- Assumptions when testing the application
- List of test cases included in Testing the application
- List of features to be tested
- What sort of Approach to use when testing the software
- List of Deliverables that need to be tested

- The resources allocated for testing the application
- Any Risks involved during the testing process
- A Schedule of tasks and milestones as testing is started

Test Scenario

A one line statement that tells what area in the application will be tested. Test Scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application[9].

The term test scenario and test cases are used interchangeably however the main difference being that test scenarios has several steps however test cases have a single step. When viewed from this perspective test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.



Test Case

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects. There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc.

Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing[10]. However, following are the main components which are always available and included in every test case:

- Test case ID.
- Product Module.
- Product version.
- Revision history.
- Purpose
- Assumptions
- Pre-Conditions.
- Steps.
- Expected Outcome.
- Actual Outcome.
- Post Conditions.

Many Test cases can be derived from a single test scenario. In addition to this, some time it happened that multiple test cases are written for single Software which is collectively known as test suites.

9.conclusion

Thus the paper reviews the testing of different levels and different methods of testing were been discussed.Each and every testing has some uniqueness in their own way.Thus testing which makes the module with goodness and satisfy the requirements of the customers.

Reference:

1. Exploratory Testing, Cem Kaner, Florida Institute of Technology, *Quality Assurance Institute Worldwide Annual Software Testing Conference*, Orlando, FL, November 2006.

2. Software Testing by Jiantao Pan, Carnegie Mellon University

3.Leitner, A., Ciupa, I., Oriol, M., Meyer, B., Fiva, A., "Contract Driven Development = Test Drive Development – Writing Test Cases", Proceedings of ESEC/FSE'07: European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering 2007, (Dubrovnik, Croatia), September 2007

4. Kaner, Cem; Falk, Jack and Nguyen, Hung Quoc (1999). *Testing Computer Software, 2nd Ed*. New York, et al: John Wiley and Sons, Inc. pp. 480 pages. ISBN 0-471-35846-0.

5. Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in*

Software Management. Wiley-IEEE Computer Society Press. pp. 41–43. ISBN 0-470-04212-5.

6. Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press. p. 426. ISBN 0-470-04212-5.

7. Section 1.1.2, Certified Tester Foundation Level Syllabus, International Software Testing Qualifications Board

8. Principle 2, Section 1.3, Certified Tester Foundation Level Syllabus, International Software Testing Qualifications Board

9. "Proceedings from the 5th International Conference on Software Testing and Validation (ICST) Software Competence Center Hagenberg. "Test Design: Lessons Learned and Practical Implications.".

10. Software errors cost U.S. economy $59.5 billion annually, NIST report