Importance of Software Testing in Software Development Life Cycle

Neeraj Kumar(15491), Ayush Singla(15462)

INFORMATION TECHNOLOGY

DRONACHARYA COLLEGE OF ENGINEERING

neeraj.136@gmail.com, shubhamsingla081@gmail.com

Abstract

Software development life cycle is a structure imposed on the development of a software product. Software development life cycle is closely linked to what has come to known as structured analysis and design. Testing is an important part of software development. Testing should be started as early as possible to make it a part of a process of deciding requirement .The major role of testing involves checking, that there should be no discrepancy in the software development process anywhere throughout. In my paper I have explained various phases and importance of testing in software development life-cycle. Keywords: Software Development Life Cycle, Software Development Process, Software Testing

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).

It involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments, and
- achieves the general result its stakeholders desire.

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 the 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.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

2. Importance of Testing in Software Development Process

The major role of software testing involves that there should be no discrepancy in the software development process. According to one survey software errors costs U.S economy 0.6 percent of the gross domestic product and about 80% of the software development costs of a project are spent on identifying and fixing errors. Each software development life cycle has passed through a set of common phases one or more times. So starting activities early means we can catch small problems before they become big problems later on. Starting testing activities early also provides the chance to review requirements for important quality attributes, to ask questions and to resolve issues. There are three different testing phases in SDLC are:

i. Test Analysis: tester tries to understand about the project.

ii. Test Design: tester design the test cases based on user requirement.

iii. Test Execution: tester execute the test cases and raise defects, if any.

Various types of testing involves throughout SDLC are:

Acceptance testing

Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a new build to the main testing process, i.e. before integration or regression.
2. Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as user acceptance testing (UAT). Acceptance testing may be performed as part of the hand-off process between any two phases of development.]

Alpha testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.[38]

Beta testing

Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team known as beta testers. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Beta versions can be made available to the open public to increase the feedback field to a maximal number of future users and to deliver value earlier, for an extended or even infinite period of time (perpetual beta).

Functional vs non-functional testing

Functional testing refers to activities that verify a specific action or function of the code. These are usually found in the code requirements documentation, although some development methodologies work from use cases or user stories. Functional tests tend to answer the question of "can the user do this" or "does this particular feature work."

Non-functional testing refers to aspects of the software that may not be related to a specific function or user action, such as scalability or other performance, behavior under certain constraints, or security. Testing will determine the breaking point, the point at which extremes of scalability or performance leads to unstable execution. Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users.

Destructive testing

Destructive testing attempts to cause the software or a sub-system to fail. It verifies that the software functions properly even when it receives invalid or unexpected inputs, thereby establishing the robustness of input validation and error-management routines. Software fault injection, in the form of fuzzing, is an example of failure testing. Various commercial non-functional testing tools are linked from the software fault injection page; there are also numerous open-source and free software tools available that perform destructive testing.

Software performance testing

Performance testing is generally executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Load testing is primarily concerned with testing that the system can continue to operate under a specific load, whether that be large quantities of data or a large number of users. This is generally referred to as software scalability. The related load testing activity of when performed as a non-functional activity is often referred to as *endurance testing*.*Volume testing* is a way to test software functions even when certain components (for example a file or database) increase radically in size. *Stress testing* is a way to test

reliability under unexpected or rare workloads. *Stability testing* (often referred to as load or endurance testing) checks to see if the software can continuously function well in or above an acceptable period.

There is little agreement on what the specific goals of performance testing are. The terms load testing, performance testing, scalability testing, and volume testing, are often used interchangeably.

Usability testing

Usability testing is to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application.

Accessibility testing

Accessibility testing may include compliance with standards such as:

- Americans with Disabilities Act of 1990
- Section 508 Amendment to the Rehabilitation Act of 1973
- Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

Security testing

The International Organization for Standardization (ISO) defines this as a "type of testing conducted to evaluate the degree to which a test item, and associated data and information, are protected to that unauthorised persons or systems cannot use, read or modify them, and authorized persons or systems are not denied access to them."[39]

3. Conclusion

Software development life-cycle is a structure imposed on the development of a software product. There are different activities involved in SDLC such as requirement, specification, architecture, software-construction, design, software testing, debugging, deployment and maintenance. Software testing plays a vital role in each and every phase of SDLC. An effective and efficient testing will provide timely visibility into the quality and readiness of the system and its input artifacts that is necessary to enable effective decision making throughout the SDLC. In my paper, I have explained various phases of SDLC and importance of testing in each and every phase.

References:

[1] Introduction to software testing available at http://www.onestopsoftwaretesting.com/introductionand-importance-of-software-testing-in-sdlc/

[2] Requirement analysis available at http://en.wikipedia.org/wiki/Requirements_analysis

[3] Functional specification available at http://en.wikipedia.org/wiki/Functional_specification

[4] Software architecture available at http://en.wikipedia.org/wiki/Software_architecture

[5] Software construction available at http://en.wikipedia.org/wiki/Software_construction

[6] Software design available at http://en.wikipedia.org/wiki/Software_design

[7] Software testing available at http://en.wikipedia.org/wiki/Software_testing