Smoke Testing – A Necessary Evil!

Smoke Testing – A Necessary Evil!

Introduction

Increase in competition and leaps in technology have forced companies to adopt innovative approaches to have an application which is 100% error free (well almost!). Having more features and functionality may not necessarily translate into success for a product or application—what ultimately counts is how stable the end user finds these features. This can be achieved by applying different types of testing methods and one such type of testing is "Smoke Testing".

Smoke testing is a good practice to ensuring that a build meets minimum quality requirements. It instills confidence, develops a good understanding between development and testing team, provides a structured approach and shortens the testing cycle while improving the overall quality of the build(s) or release. The quality and functioning of the developed software is improved as functional, architectural and component level design defects are found earlier.

A good test strategy includes smoke testing as one of the key activities and is often defined as acceptance criteria before a build is deployed for further detailed testing by a testing team.

What Is Smoke Testing?

A **smoke test** is a series of test cases that are run prior to commencing with full-scale testing of an application. Smoke testing is non-exhaustive testing, ascertaining that the most crucial functions of a program work, but not bothering with finer details.

The idea is to test the high-level features of the application to ensure that the essential features work. If they do not work, there is no need to continue testing details of the application, so the testing team will refuse to do any additional testing until the smoke test suite passes. This puts pressure on the development team to ensure that the testing team does not waste valuable testing time with code that is not ready to be tested.

Once smoke tests are implemented as part of your software development life cycle, you will see the overall quality of the product improve and the sensitivity to producing high quality software increased.

Smoke Tests - How to Implement

To implement smoke tests, the testing team will develop a set of test cases that are run any time a new release is provided from the development team. These set of test cases are used to test the major functional areas of the system. They are not nit-picky test cases that test low-level detail of the application; they are higher level test cases that test overall functionality.

It will be more productive and efficient if the smoke test suite is automated or it can be a combination of manual and automated testing.

To ensure quality awareness, the smoke test cases are publicized to the development team ahead of time, so that they are aware of the quality expectations.

Components of a Smoke Test Suite

It is important to focus on the fact that smoke test suite is a "shallow and wide" approach to the testing. The test suite should comprise all areas of the application without getting too deep, looking for answers to basic questions like, "Can I launch the test item at all?", "Does it open to a window?", and "Do the buttons on the window perform actions as intended"? These written tests can either be performed manually or using an automated tool.

One way to ensure that smoke test suite comprises of "shallow and wide" test cases is by focusing on positive tests only. A positive test is used primarily, if not solely, to validate that a given system,

Smoke Testing – A Necessary Evil!

function, operation, etc. works as designed when a user enters the right data, in the right place, at the right time, clicks the right buttons, etc.

The smoke test suite should ideally include at least one test case for every feature or function of all the modules. Smoke test suite need not be part of the regression test suite; this reduces or avoids the redundancy or repeatability of the tests that do not add value during regression testing. Typical issues uncovered by a good smoke test suite are:

- Initial environment setup or configuration issues
- Overall application stability issues
- Features those are not yet available or not functioning

Responsible Parties for Smoke Testing

Smoke testing is either done by developers before the build is released or by testers before accepting a build for further testing. In either case, it is important to define the roles and responsibilities at the beginning of the project. If indeed the development has a detailed unit test suite, then generally the smoke testing responsibility is shifted to testing team.

Summary

Typical Characteristics of Smoke Testing

- It exercises the entire system from end-to-end. Stress should be given to detect those errors which can stop the system from functioning further
- It is not exhaustive but should be capable of exposing major problems
- It ensures that the major functionality is working and the build is stable enough for further testing thoroughly.

Advantages of Smoke Testing

- **It minimizes integration risk.** One of the greatest risks that a team project faces is that, when the different team members combine or "integrate" the code they have been working on separately, the resulting composite code does not work well. Depending on how late in the project the incompatibility is discovered, debugging might take longer than it would have if integration had occurred earlier, program interfaces might have to be changed, or major parts of the system might have to be redesigned and re-implemented. In extreme cases, integration errors have caused projects to be cancelled. The daily build and smoke test process keeps integration errors small and manageable, and it prevents runaway integration problems.
- **It reduces the risk of low quality.** Related to the risk of unsuccessful or problematic integration is the risk of low quality. By minimally smoke-testing all the code daily, quality problems are prevented from taking control of the project. You bring the system to a known, good state, and then you keep it there. You simply don't allow it to deteriorate to the point where time-consuming quality problems can occur.
- **Uncover major problems.** A good designed smoke test can increase the probability of finding a major problem when software is built early in the cycle. It also uncovers defects not related to coding but that arise because of wrong or incomplete setup or configuration; such as database, server, firewall, application parameters etc. Thus you catch defects earlier in the cycle.
- **Save time and cost.** If a major problem is detected at an earlier stage, it can save huge time and cost than if the same error was discovered late in the cycle.