Software Testing Methods

Black box testing: Also called functional or specification-based testing, this method focuses on output. Testers aren't concerned with the internal mechanisms. They only check the software does what it's supposed to. Knowledge of coding isn't necessary, and testers work at user interface level.
White box testing: This method uses coding know-how as part of the test procedure. When a product fails, testers go as deep into the code as necessary to find the cause. The software developers do this themselves since they determine how the product should work. Structure-based and glass box testing are other names for this method.

Static testing: Testers examine the software's code and documentation but don't execute the program. Static tests start early in the product's development during the verification process.
Dynamic testing: The software is executed with various inputs, and testers compare outputs with expected behavior with this method.
GUI testing: This tests GUI characteristics -- text formatting, text boxes, buttons, lists, layout, colors, fonts, font sizes, and so on. GUI testing is time-consuming, and third-party companies often take on the task instead of developers.

Test levels

These are necessary to identify areas of weakness and overlap in each phase of the software development lifecycle.

Unit testing: Developers test the most basic parts of code like classes, interfaces, and functions/procedures. They know how their code should respond and can make adjustments depending on output.
Component testing: Other names are module or program testing. It's similar to unit testing but contains a higher level of integration. Modules of the software are tested for defects to verify their individual function.
Integration testing: This identifies errors when modules are integrated. Different integration tests are bottom up, top down, and functional incremental.

System testing: Components of a project are tested as a whole in different environments with this method. It falls under the black box method and is one of the final tests in the process. It determines if the system functions as it should to meet business and user needs.
Alpha testing: Internal staff test the software at the developer's site in a simulated or actual environment. After that, developers rectify bugs and other issues.
Beta testing: Known as field testing as well, the client tests the product on their own site in real conditions. The client may offer a group of end-users the opportunity to test the software via prerelease or beta versions. Feedback on possible improvements is then sent to the developer.

Acceptance testing: Also under the scope of black box testing, the client tests software to find out if the developer has created the program to desired specifications.

Test types

These software tests focus on specific objectives.

Installation testing: The software test engineer and the configuration manager conduct this test to ensure the end-user can install and run the program. It covers areas like installation files, installation locations, and administrative privileges.
Development testing: This implements a range of synchronized strategies to detect and prevent defects. It includes static code analysis, peer code reviews, traceability, and metrics analysis. The aim is to reduce risks and save costs.
Usability testing: User experience comes under the spotlight with this test. It measures how well the GUI is designed and its ease of use. The test checks accuracy and efficiency of functions and the emotional responses of test subjects.

Sanity testing: This indicates if the software is worth the time and cost to continue further tests. Too many flaws and more aggressive tests don't follow.
Smoke testing: Smoke testing reveals basic failures that are serious enough to prevent release. When this is carried out on a new build, it is called a build verification test.
Regression testing: When the system undergoes modification, regression testing monitors unexpected behavior. It points out adverse effects on modules or components.
Destructive testing: Testers input abnormal entries and discern the software's ability to manage unexpected input. This shows developers how robust the program is at error management.

Recovery testing: When hardware or other functions fail, this test shows how well the software can recover and continue operation.
Automated testing: This performs functions difficult to implement manually. It uses specific software to run the tests and to provide data on actual versus expected outcomes.
Compatibility Testing: Software must run in different computing environments, so this checks compatibility with different systems. For example, does the software work with various operating systems and web browsers?

Performance testing: This is an in-depth test that examines software performance in different scenarios. Information about responsiveness, stability, resource allocation, and speed is gathered. Moreover, sub-tests such as volume, capacity, and spike testing play a part in this process.
Security testing: This measures the software's ability to protect users' security. This means authorization functions, authentication, confidentiality, integrity, availability, and non-repudiation.
Accessibility testing: This is not the same as usability testing. This determines the extent to which users of differing abilities -- learning and physical disabilities included, can use the software.

Internationalization and localization testing: Results show how the software can adapt to different languages and regional demands. This includes adding components for specific locations and translating text.