Inflectra: Software Built For You

Our one goal is to help you succeed. We care deeply about giving you the best quality service and support you've ever had.

As many users, projects, tests, releases, items, API calls as you want. All pricing is based on concurrent users.

Flexible options to make your life easier. Use on desktop or mobile; your servers or our cloud, sensible add-ons, fairly priced.

Software Testing Methodologies

Why Choose SpiraTeam as Your Software Testing Tool?

SpiraTest® manages your project's requirements, test cases, bugs and issues in one integrated environment, with full traceability throughout the software testing lifecycle.

- SpiraTest is a complete, out-of-the-box quality management platform, with requirements management, test case management and bug tracking fully integrated from day one
- SpiraTest makes the managing and tracking of your testing easy, it allows you to quickly configure different test plans for the different hardware/software combinations and make sure that all of the requirements and test cases have been fully tested on them
- SpiraTest is an open, extensible platform, with a rich library of add-ons and extensions that let SpiraTest integrate with all of the most popular unit, functional and load testing tools on the market
- SpiraTest comes with a powerful set of customizable dashboards that consolidate all of the most importing testing and QA metrics onto a single page, so that you can make real-time decisions.

10 Top Things to Look for in a Software Testing Tool

1. manage your project's requirements, test cases, bugs, and issues in one integrated environment
2. can manage, schedule, and execute manual and automated tests
3. open architecture that lets you integrate with different testing tools
4. flexibility to use third-party requirements-management and defect-tracking tools when needed
5. supports templates, tests, and parameterized data-driven testing
6. works across a range of platforms and devices, including mobile
7. fully customizable workflows and configurable field values
8. rich dashboards of key information that drilldown to the source data
9. customizable reporting available in a variety of different formats
10. end-to-end traceability from requirements to tests to defects.

Visit us at www.inflectra.com for a free trial
Or get in touch with us: sales@inflectra.com, 1-866-572-5878 or +1 202-558-6885 (international)

inflectra®

What Are Software Testing Methodologies?

Software testing methodologies are the different approaches and ways of ensuring that a software application in particular is fully tested. Software testing methodologies encompass everything from unit testing individual modules, integration testing an entire system to specialized forms of testing, such as security and performance.

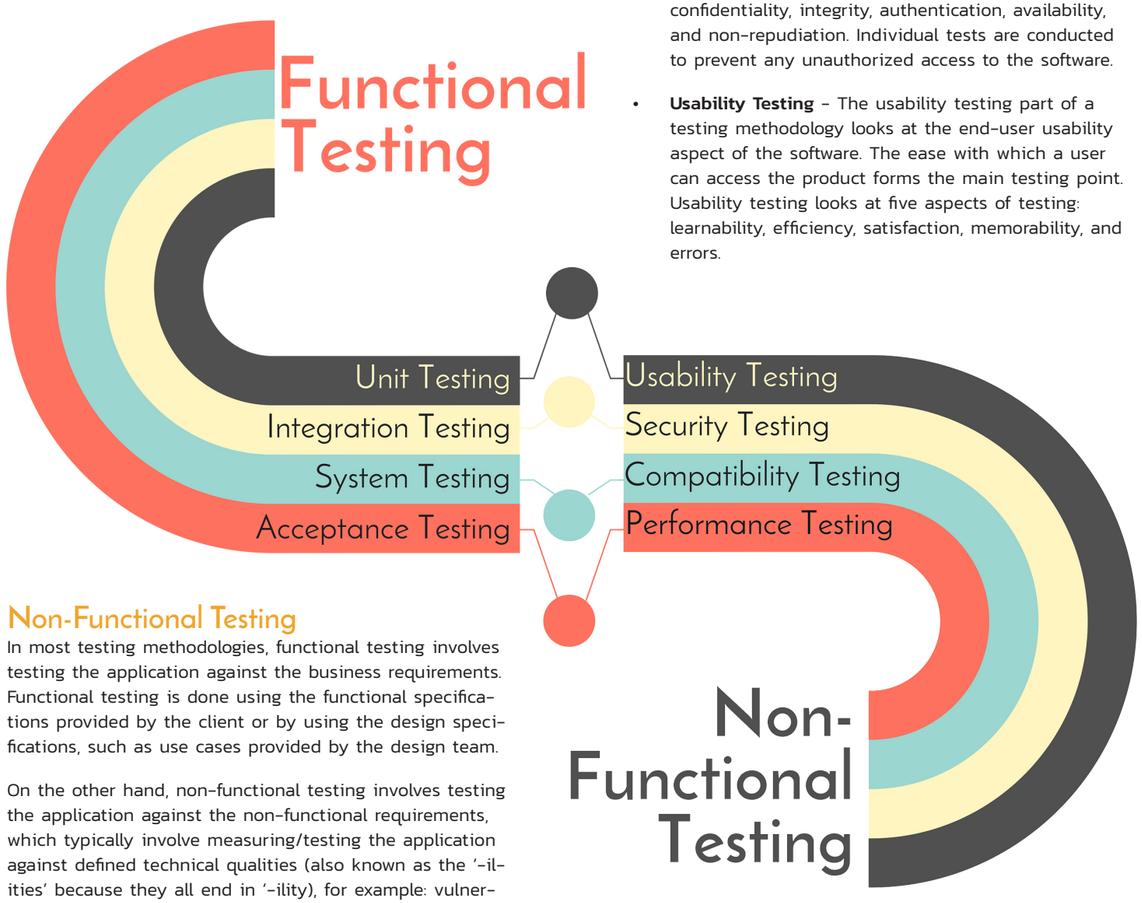
Importance of Testing Methodologies

As software applications get ever more complex and intertwined and with the large number of different platforms and devices that need to get tested, it is more important than ever to have a robust testing methodology. The more robust a testing system, the easier is to ensure that software products/systems being developed have been fully tested, meet their specified requirements and can successfully operate in all the anticipated environments with the required usability and security.

Functional Testing

The functional testing part of a testing methodology is typically broken down into four components – unit testing, integration testing, system testing and acceptance testing – usually executed in this order. Each of them is described below:

- **Unit Testing** – this refers to testing of individual software modules or components that make up an application or system. These tests are usually written by the developers of the module and in a test-driven-development methodology (such as Agile, Scrum or XP) they are actually written before the module is created as part of the specification. Each module function is tested by a specific unit test fixture written in the same programming language as the module.
- **Integration Testing** – these test the different modules/components that have been successfully unit tested when integrated together to perform specific tasks and activities (also known as scenario testing). This testing is usually done with a combination of automated functional tests and manual testing depending on how easy it is to create automated tests for specific integrated components.
- **System Testing** – this involves testing the entire system for errors and bugs. This test is carried out by interfacing the hardware and software components of the entire system (that have been previously unit tested and integration tested), and then testing it as a whole. This testing is listed under the black-box testing method, where the software is checked for user-expected working conditions as well as potential exception and edge conditions.
- **Acceptance Testing** – The acceptance testing is the final phase of functional software testing and involves making sure that all the product/project requirements have been met and that the end-users and customers have tested the system to make sure it operates as expected and meets all their defined requirements.



Non-Functional Testing

In most testing methodologies, functional testing involves testing the application against the business requirements. Functional testing is done using the functional specifications provided by the client or by using the design specifications, such as use cases provided by the design team.

On the other hand, non-functional testing involves testing the application against the non-functional requirements, which typically involve measuring/testing the application against defined technical qualities (also known as the '-ilities' because they all end in '-ility), for example: vulnerability, scalability, usability. Some examples of non-functional testing are described below:

- **Performance, Load, Stress Testing** – There are several different types of performance testing in most testing methodologies, for example: performance testing is measuring how a system behaves under an increasing load (both numbers of users and data volumes), load testing is verifying that the system can operate at the required response times when subjected to its expected load, and stress testing is finding the failure point(s) in the system when the tested load exceeds that which it can support.
- **Security, Vulnerability Testing** – Previously, security was something that was tested after-the-fact. With the rise in cyber-crime and the awareness of the risks associated with software vulnerabilities, application security is something that should be designed and developed at the same time as the business functionality. Security testing tests the software for confidentiality, integrity, authentication, availability, and non-repudiation. Individual tests are conducted to prevent any unauthorized access to the software.
- **Usability Testing** – The usability testing part of a testing methodology looks at the end-user usability aspect of the software. The ease with which a user can access the product forms the main testing point. Usability testing looks at five aspects of testing: learnability, efficiency, satisfaction, memorability, and errors.
- **Compatibility Testing** – The compatibility testing refers to testing that the product or application is compatible with all the specified operating systems, hardware platforms, web browsers, mobile devices, and other designed third-party programs (e.g. browser plugins). Compatibility tests check that the product works as expected across all the different hardware/software combinations and that all functionality is consistently supported.

Testing Practices

Test Automation

Testing is one of the greatest consumers of manpower in software development projects. Anything that can reduce this manual effort is going to seriously reduce the cost of overall development. Test automation saves time and effort because, unlike some project activities, it is repeated, sometimes frequently, so you would do well to consider test management tools with in-built support for automation. If you plan to use an XUnit testing framework such as TestNG, NUnit and JUnit, make sure your test management tool has an integration to the one(s) you have chosen.

When reviewing test management tools with some built-in test automation, look for the ability to manage test scripts as well as schedule and launch tests both locally and on remote hosts to help with remote testing. If the test tool can pass parameters to your test cases, whether they are manual or automated tests, then score it bonus marks!

Take away: If you plan on using test automation, integrate it with test management.

Exploratory Testing

The problem with planned testing is that it needs to be predictive. Planned functionality can, of course, be anticipated and tests created to verify it, but testing for the unexpected or testing to ensure there is no additional, unwanted functionality, is much harder to plan for. Sometimes the best technique is to simply allow the test engineer to see where his or her instinct leads. Exploratory testing does not advocate arbitrary or ad hoc behavior on the part of the tester, but promotes less structured, experience driven testing which changes and grows as it is performed. It is also a technique to be used in addition to scripted testing, not in place of it. Consequently, the capabilities required from a test management tool for the support of exploratory testing are rather different from those of planned and scripted testing.

Look for tools that provide a means to track exactly what the tester does so that the conditions can be reproduced should a defect be discovered or in case it seems that the test should become part of the scripted test set. Capturing key strokes, screen shots and user comments provides a good way to integrate exploratory testing with scripted testing in a controlled way, especially when it comes to adding new tests to future regression testing. If the tool is also able to provide comparative metrics of scripted versus exploratory testing, it can help you determine the effectiveness of each and where best to put additional test effort.

Take away: Explore the options that support exploratory testing and integrate that with formal testing.