Software Unit Test

Course duration: 1 day

Overview:

Unit Test is the testing applied to a small part of code independently of its environment. Focusing on atomic units allows to thoroughly checking that basic parts of a system behave as expected.

Developers use to apply light unit tests upon the code they just wrote for resolve possible language or libraries ambiguities. Problems arise from the fact that testing objectives are fundamentally different than system development objectives so that testing methods have nothing in common with development methods. For reaching a reasonable "test coverage" developers need to learn some relevant testing methods and techniques.

Developers and QA people may use debuggers for some laborious unit tests. Unit Test Tools help to generate stubs that activate the isolated piece of code in a wide range of extreme conditions. Tools help also analyzing, sometimes statistically, the results of intensive repetitive unit tests.

In this course, developers, testers and managers learn about Unit Test methodologies, methods, techniques and tools with practical examples.

Purpose:

The participants of this UT course will know to plan, manage, perform Unit Tests and analyze the results.

Who should attend:

Developers, Testers, Integrators, Development managers, Team leaders, Project managers and QA managers.

Topics:

1. Unit Test scope, purposes and means
    a. How early testing pays back
    b. Unit Testing challenges, UT is not Developer Test
    c. Unit test versus system and acceptance tests
    d. UT with embedded hardware
2. Unit Test dependencies, planning and roles
    a. Developer's UT and QA's UT in the SDLC
    b. UT prerequisites
    c. Responsibilities, training, tools and time
    d. UT reuse
3. Software scaffolding
    a. Unit in Procedural programming
    b. Unit in Object Oriented programming
    c. Stubs, exercisers, testbeds
    d. Mock Objects
4. Introduction to Test coverage, Test economics
    a. Definition of Test Coverage and its importance
    b. Maximizing coverage while minimizing the tests
    c. When test is enough?
    d. Sponsoring testing
5. Functional testing – Black box methods and techniques
    a. Purpose and principles
    b. Equivalence partitioning
    c. Boundary values
    d. Negative testing

6. White box methods and techniques
    a. Purpose and principles
    b. Path analysis, Control Flow
    c. Code coverage
    d. Dead code
7. Unit Test tools
    a. Symbolic debugger
    b. Unit testing frameworks
    c. Off the shelf tools
8. Test-driven programming, Extreme Programming
    a. Resolving uncertainty
    b. Test-driven development cycle
    c. Testathon
9. Complementary methods
    a. Defensive programming
    b. Static Analysis, Code review
    c. Pair programming
10. Probing further