Fundamentals of Unit Testing

Course Summary

Description

This is a hands on course intended to help programmers develop and improve their ability to write high quality unit tests and effectively use unit testing to produce better code during the development process.

Topics

- Unit Testing, Developers and Test Driven Development
- Software Testing Fundamentals.
- Types and Levels of Testing
- Unit Testing
- TDD: A unit test methodology for developers.
- Writing Good Functional Unit Tests.
- Writing Good Structural Unit Tests.
- Integrating Unit Testing into Development
- Refactoring.
- Advanced Testing for the Developer.
- Putting it All Together
- Optional Topics

Audience

This course is intended for programmers, QA staff and project managers or team leads.

Prerequisites

Students should have knowledge of at least one programming language and the ability to write programs in that language.

Duration

Three days

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Fundamentals of Unit Testing

Course Outline

I. Unit Testing, Developers and Test Driven Development
- A. Testing failures: Illustrations of why testing is critical.
- B. What software testing is: traditional and modern definitions.
- C. Common steps to all development processes.
- D. Where testing fits in the development process.
- E. Testing and development: orthogonal activities.
- F. The difference between testing and debugging.
- G. Why testers can't do effective unit testing.
- H. TDD: A unit testing methodology for developers.

II. Software Testing Fundamentals.
- A. Test terminology: tests, cases, suites, faults, bugs, pass and fail.
- B. The human problem: the need for testing protocols.
- C. Sources of bias and testing failures.
- D. Protocols to eliminate bias in test development.
- E. Protocols to eliminate bias in test execution.
- F. Protocols to eliminate bias in test evaluation.
- G. The problem of completeness versus limited resources.
- H. Coverage: the goal of testing.

III. Types and Levels of Testing
- A. Units, components, system, acceptance and regression testing.
- B. White, gray and black box testing.
- C. Functional versus operational testing.
- D. The specification: what we test against.
- E. Making requirements testable.

IV. Unit Testing
- A. Defining units to test.
- B. Interface and implementation.
- C. Programming by contract and using assertions.
- D. Testing units at the interface.
- E. Organizing unit tests: simplest first.
- F. Incremental coverage.
- G. Unit test coverage.
- H. Using unit tests to understand a system.

V. TDD: A unit test methodology for developers.
- A. A quick review of agile and extreme programming concepts.
- B. The TDD cycle: write test, write code, refactor.
- C. Why write tests first.
- D. Pair programming and test development,
- E. User stories, requirements become acceptance tests.
- F. Acceptance tests become unit tests.

VI. Writing Good Functional Unit Tests.
- A. What makes a good unit test suite.
- B. Black box coverage -- validating the spec.
- C. Equivalence classes and boundary values.
- D. Cause and effect graphs
- E. Decision tables.

VII. Writing Good Structural Unit Tests.
- A. The problem of path coverage.
- B. Levels of coverage.
- C. Branch coverage.
- D. Decision coverage.
- E. Multiple decision coverage.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Fundamentals of Unit Testing

Course Outline

VIII. Integrating Unit Testing into Development

A. Test early, test often and test automatically.
B. Automating testing: an overview.
C. Organizing and documenting the test suites.
D. Testing frameworks and common tools.
E. Fixture code, runners and suites.
F. Stubs and mock objects.
G. Developing a test schedule and process

IX. Refactoring

A. Refactoring versus code changes.
B. Refactoring drivers - code smells
C. Refactoring for simplicity and good design.
D. Types of refactorings and the refactoring catalogue.
E. “Red => Green => Refactor"

X. Advanced Testing for the Developer.

A. Stateful versus stateless testing.
B. Testing by scenario, method and state.
C. Exception testing.
D. Refactoring code for testing.
E. The Testing Maturity Model

XI. Putting it All Together

A. The influence of TDD on code design and development.
B. Establishing an integrated test/development environment.
C. Establishing and refining a unit testing process.
D. Integrating unit testing with QA.
E. Process improvement, how to use testing improve productivity.

XII. Optional Topics

A. Testing documentation.
B. Testing patterns.
C. More Code smells and refactoring.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically