

EFFECTIVE DEVELOPER TESTING

Two-day workshop

Overview

The responsibility of uncovering all the defects does not lie only with the QA team. Potential defects can be clearly classified into those that can be caught early at implementation stage and those that may be detected post development. The key is to understand what types of defects can be caught at early stage vis-à-vis at later stage. This will help you evolve a formal approach to unit validation strategy and unit test design that will “guarantee” early quality.

Workshop Objective

This workshop will help you devise a strategy for testing an unit, discuss black-box techniques to design tests, help assess adequacy using white-box techniques, device test suites using a simple automation framework. Some common checklists/guidelines to aid cost-effective unit testing, OO/Structured-based units and how OO impacts unit testing will also be discussed.

Topics of discussion

Introduction to unit testing

- Objective of unit testing
- Definition of unit
- Types of faults that should be detected at unit testing

Black box-test design techniques

- Boundary value analysis
- Equivalence classes
- Special value
- Input/Output domain testing
- State based, Decision-table based testing

White box techniques

- Statement coverage
- Condition/Multiple condition coverage
- Path coverage
- File, Class, Function/Method coverage
- How much coverage do I am for?

Generating small yet effective test cases

- Single fault/Multi-fault
- Orthogonal arrays

Testing OO units (Classes)

- Information hiding and how this affects testability
- Validation of an individual method
- Testing polymorphic methods
- Testing class modal behavior i.e. inter-relationships of methods

Testing abstract classes

Testing generic classes

What to test in a derived class

Checklists/guidelines that aid in validation of unit

Units that are typically user interface

Units that are typically libraries

Units that parts of components/frameworks

Fault models in execution and techniques of detection

Resource leakage

Performance issues

Error recovery issues

Portability issues

Concurrency issues

State related issues

Test documentation

How much to document in a unit test plan

A simple style of document unit cases quickly and easily

Automating unit tests

Framework for unit test automation

Writing test drivers

Writing test oracles

Workshop benefits

Participants are exposed to disciplined and effective early “unit testing” resulting in lesser “headaches” later. At the end of the workshop, the participant will be able to:

- Design effective and reliable unit test cases
- Develop workable unit test plans
- Organize and manage his/her test efforts
- Evaluate test results
- Review the effectiveness of his/her testing

Who should attend?

Developers and Test engineers. **Prerequisites** - Good programming knowledge in at-least one language.

Delivery method

The workshop consists of classroom discussions and a problem solving/case-study session. It adopts our unique learning model. This enables participants listen to lecture on a topic, introspect using a structured questionnaire, explore the topic - hands-on session, discuss the learning after exploration and finally read the supplemental course notes.