

## Testing for Programmers Course Outline

### General Description

From the popularity of the new agile methodologies that emphasize a “test first” approach to programming, to the long-established importance of solid programmer testing before handing the system to an independent test team, it’s clear that programmers need to understand the fundamentals of unit and integration testing. What are the key techniques, skills, and ideas programmers need? Based on professional experience and a survey of techniques, Rex Black will introduce programmers to successful unit and integration testing techniques.

### Learning Objectives

Through presentation, discussion, and hands-on exercises, attendees will learn to:

- Design and develop tests based on expected behavior (black box), using techniques including:
  - + Equivalence classes and boundary value analysis.
  - + Decision tables.
  - + Live data and customer workflow testing.
  - + State-transition diagrams.
  - + Domain testing.
  - + Orthogonal arrays.
- Measure and enhance test coverage based on implementation details (white box), using techniques like:
  - + Statement, branch, condition, and loop coverage.
  - + McCabe cyclomatic complexity and basis tests.
  - + Data and operation test checklists
  - + Object-oriented unit testing
- Perform effective and efficient reviews:
  - + Requirements
  - + Design
  - + Code.
- Understand how testing fits into the development lifecycle, addressing issues such as:
  - + Sequential and incremental lifecycle models.

- + Unit testing.
- + Integration testing.
- Design and develop appropriate integration tests, using techniques like:
  - + Backbone integration strategies.
  - + Object-oriented integration testing
- Select appropriate unit and integration test automation strategies, using tools like:
  - + API test harnesses (e.g., xUnit)
  - + CLI test scripts (e.g., Korn-shell)
  - + GUI “screen-scrapers” (e.g., WinRunner).

## Course Materials

This course includes the following materials:

<i>Name</i>	<i>Description</i>
Course Outline	A general description of the course along with learning objectives, course materials and an outline of the course topics, including approximate timings for each section.
Noteset	A set of over 150 PowerPoint slides covering the topics to be addressed.
Test Requirement Catalog	General-purpose clues (developer version) that are useful in a wide range of programs
Project Source Documents for Course Exercises	Specifications used in the realistic example project used in exercises for the course.
Scripts and programs - corrupter, ooticketcalc, ootree, pairs, and triangle	A set of scripts and programs which are used to demonstrate the topics and techniques which are described.
Bibliography and resources	A set of further readings, Web sites, tools and other resources to help implement the concepts.

The printed course materials are provided in a binder in a way which makes it convenient for course attendees to remove portions as needed for reference; e.g., during exercises. The scripts and programs are provided as electronic copies on CD-ROM.

## Session Plan

### *Day One\**

Introductions, objectives and overview

Fundamental black box tests

- Equivalence partitioning
- Boundary value analysis
- Customer data and workflows
- Decision tables

Exercise: Equivalence classes and boundary value analysis for oowordtree

Exercise: Decision tables for ooticketcalc

Advanced black box tests

- State transition diagrams
- Domain testing
- Orthogonal arrays and all-pairs tables
- Exploratory testing

Exercise: State-transition diagram tests for oowordtree

Exercise: Domain test for ooticketcalc

Exercise: All-pairs test for ooticketcalc

### *Day Two*

White box coverage and techniques

- Statement, branch, condition, and loop coverage
- McCabe complexity, basis paths, basis tests
- Testing typical operations
- Application program interface (API) testing
- Mutation and fault injection
- Object-oriented unit test concerns
- Other white-box tests

Exercise: Measuring coverage of oowordtree

Exercise: Condition coverage for the triangle function

Exercise: Calculating complexity of WordTree functions

Exercise: Checklist-based tests for oowordtree

Exercise: Testing oowordtree with corrupted data

Requirements, design, and code reviews

- Reviewing requirements and design documents
- Review techniques

---

\* Note: Exercises are selected based on time available as well as audience needs and interest. Not all exercises will be covered in the course.

- Code reviews
- Code review checklists

Exercise: Reviewing ooticketcalc or oowordtree

Programmer testing in the development process

- Sequential lifecycle models (e.g., V-model)
- Incremental lifecycle models (e.g., RUP)
- Unit testing and code reviews in the models
- What's a unit?
- Why do unit testing?
- How to introduce programmer testing
- Stub/driver/mock object considerations

Automated testing options

- Overview
- Theory and practice
- Manual versus automated
- GUI, API and CLI test automation
- Automated static testing

### **About the Units Tested in the Course**

These units are written in C or C++, using both procedural and object-oriented techniques. Each has known bugs hiding in it.

Clients may choose to customize the course by supplying their own code for testing. Customization charges will apply.

The course is run live. Attendees will need access to computers running Microsoft Visual C++ 6.0 or later.

### **Recommended Readings**

The class materials include an extensive bibliography of books related to software testing, project management, quality, and other topics of interest to the test professional.