

Pragmatic Software Testing Course Outline

General Description

Testing even a simple system is a potentially infinite task. With tight budgets and schedules, testers need a practical set of tools that will allow them to effectively and efficiently test software and find bugs.

This course will give attendees those tools. We'll look at practical techniques like risk analysis and test planning. We'll talk about static testing of requirements and code. We'll get hands-on experience with black-box techniques including equivalence classes, decision tables, state-transition diagrams, and orthogonal arrays. We also get hands-on experience with white-box techniques like statement and branch coverage, basis paths, and set-use pairs. We'll cover documenting test cases, writing bug reports, and tracking test status. We'll also look at the skills, traits, and attitudes that make testers successful.

This three-day course may be extended with a one-day hands-on exercise module (using a computer) and/or a one-hour exam.

Learning Objectives

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

- Understand the goals, strategies, and tactics of effective and efficient testing.
- Analyze, prioritize, and document risks to the quality of the system.
- Design, develop, and document static, white-box, and black-box tests, using techniques including:
 - + Requirements and code reviews.
 - + Equivalence classes and boundary value analysis.
 - + Decision tables.
 - + Live data and customer workflow testing.
 - + State-transition diagrams.
 - + Domain testing.
 - + Orthogonal arrays.
 - + Statement, branch, condition, and path coverage.
 - + McCabe unit basis tests.
 - + Data flow coverage.
 - + Integration test strategies (including backbone integration).
 - + McCabe integration basis tests.

- Execute test cases and track their results.
- Research observed anomalies and write actionable, high quality bug reports.
- Be effective and efficient professionals within the test operation and the company.

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 600 PowerPoint slides covering the topics to be addressed.
Project Source Documents for Course Exercises	Specifications used in the realistic example project used in exercises for the course.
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.

Session Plan

*Day One**

Introductions, objectives and overview

What is effective and efficient software testing?

- Effectiveness
- Efficiency
- Viewpoints on testing
- Quality

Exercise: The triangle tests

Understanding risks to system quality

- An overview of various quality risks
- The challenge of focusing on the right risks

Aligning testing with quality risks

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

- Testing real-world use
- Testing real-world configurations
- Sources of information on risk
- Using risk to guide the areas and extent of testing

Exercise: Quality risk analysis

Aligning testing with the project

- Why organizations test
- Testing and software development lifecycles
- Testing and other kinds of projects

Understanding test strategies, tactics, designs

- A survey of test strategies
- Regression test strategies
- A survey of test tactics and techniques
- Good test system design

Effective and efficient static testing

- Static testing requirements
- Types, goals, and benefits of reviews
- Static testing documentation
- Static testing code

Exercise: Reviewing requirements

Day Two

Effective and efficient white-box (“structural”) testing

- Code coverage
- McCabe complexity, basis paths, basis tests
- Data flow coverage (“set use pairs”)
- Application program interface (API) testing
- Mutation and fault injection
- Object-oriented unit test concerns
- Other white-box tests
- Drivers and stubs
- Integration techniques
- Backbone integration
- Integration complexity, basis paths, basis tests
- Object-oriented integration test concerns
- Other considerations

Exercise: Code-coverage and data coverage input classifier tests

Exercise: White-box triangle tests

Exercise: Internet kiosk integration tests

Basic effective and efficient black box (“behavioral”) testing

- Equivalence partitioning
- Boundary value analysis
- Use cases
- Nouns and verbs
- Customer data and workflows

- Decision tables

Exercise: Equivalence classes and boundary value analysis

Exercise: Internet kiosk decision table tests

Exercise: Decision table, boundary value analysis, and scenarios

Exercise: Building a decision table and tests

Advanced effective and efficient black box (“behavioral”) testing

- State transition diagrams
- Risk-driven tests
- Exploratory tests
- Domain testing
- Orthogonal arrays and all-pairs tables
- Syntax testing

Exercise: Internet kiosk state-transition diagram tests

Exercise: ATM state-transition diagram tests

Exercise: Internet kiosk compatibility test

Exercise: Internet kiosk domain test

Exercise: Print server tests using state models

Effective and efficient automated testing

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

Case study: A mixed commercial-custom automated test project

Exercise: Picking an oracle

Day Three

Documenting, calibrating, and assessing tests

- What to document
- Test case templates
- Detail and precision of documentation
- Traceability
- Predicting test effectiveness
- Continuous improvement

Exercise: Traceability matrix for kiosk tests

Exercise: Print server tests in a template

Effective and efficient test plans

- Developing test plans
- A test plan template
- Entry and exit criteria

- Test execution
- Planning risks for test plans
- Selling the plan

Exercise: Test plan for the Internet kiosk

Effective and efficient test tracking

- A basic test tracking spreadsheet
- Test tracking process
- Extending the test tracking system
- Adding coverage traceability

Case study: A test tracking example

Basic effective and efficient bug reporting

- Bug reporting basics
- Ten steps to better bug reports

Advanced effective and efficient bug reporting

- Classification
- Bug lifecycles
- Bug triage

Case study: A sample bug report

Exercise: Bug reporting

Being an effective and efficient tester

- Adopting good attitudes and outlooks
- Planning for skills growth
- Training and education

Glossary, bibliography, and resources

Recommended Readings

Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional, by Rex Black. In addition, 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.