

Software Testing: The Next Generation Virtual Course Outline

General Description

From traditional testers to software development engineers in test (SDETs) to programmers doing unit testing, everybody in the development team is doing testing now. You're probably spending lots of time and energy on testing, but are you getting the benefits you should? In this class, you learn the techniques that will allow you to create tests that will find important bugs when you develop new code, reduce regression risk when you change existing code, and build confidence when you deliver software. This class will allow you to defy conventional wisdom. Why move fast and break stuff when you can move even faster and not break stuff? Why release early and be ashamed of your first version when you can release even earlier and be proud of it? In three days, you'll learn the essential test techniques that will set you and your products apart from the crowd.

Course Outline

Day 1	
Morning	Introduction to the class Introduction to the exercise platform Introduction to black-box testing Equivalence partitioning and boundary value analysis Equivalence partitioning and boundary value analysis exercise
Afternoon	Decision table Decision table exercise State-based testing and model-based testing State-based testing and model-based testing exercise
Day 2	
Morning	Domain analysis Domain analysis exercise Pairwise testing Pairwise testing demonstration Pairwise testing exercise
Afternoon	Introduction to white-box testing Statement and branch coverage

	Statement and branch coverage demonstration Statement and branch coverage exercise Cyclomatic Complexity and path coverage Cyclomatic Complexity and path coverage demonstration Cyclomatic Complexity and path coverage exercise
Day 3	(optional additional day)
Morning	Introduction to test automation Test-driven development lecture Test-driven development demo Test-driven development exercise Behavior-driven development lecture Behavior-driven development demo
Afternoon	Behavior-driven development exercise Acceptance test-driven development lecture Acceptance test-driven demo Acceptance test-driven development exercise

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
 - + State-transition diagrams
 - + Domain testing
 - + Pairwise techniques
- Measure and enhance test coverage based on implementation details (white box), using techniques like:
 - + Statement and branch coverage.
 - + McCabe Cyclomatic Complexity and basis tests.
- If the optional third day is chosen, select appropriate unit and integration test automation strategies, using tools for:
 - + Test-driven development
 - + Behavior-driven development
 - + Acceptance test-driven development

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 materials.
Scripts and programs (e.g., corrupter, ooticketcalc, ootree, pairs, and triangle)	A set of scripts, Java, C, and C++ programs which are used to demonstrate the topics and techniques which are described and as a basis for the exercises. The programs are realistic in size and complexity, with understandable purposes.
Exercise solutions	A complete set of solutions for all of the exercises included in the course.
Open source tools	As part of the exercises, attendees will download and use various open source tools.
Platform description	A document describing how to configure a Linux system optimized to support the tools and exercises.

About the Programs Tested in the Course

These programs are written in C, Java, and/or C++, using both procedural and object-oriented techniques. Each has known bugs hiding in it. The programs work on Windows and Linux systems. Attendees should install the programs, compile them, and ensure the ability to use the compiled programs prior to the start of the class.

Clients may choose to customize the course by supplying their own code for testing.

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.