

## Course 344: Agile Testing Methods for Developers and Testers (4 days)

### Course Description...

Agile software development means producing usable and tested code faster than with traditional methods. In addition to practices aimed at speeding up code production, such as pair programming, agile methods also define practices for improving the quality of the delivered code through testing, such as continuous integration, test first and test-driven development. This course explores the various agile practices for improving the quality of software development through testing and how those practices can be applied to any form of software development whether or not agile approaches are adopted. In this course we will explore Test Driven Development, continuous integration, pair programming, and other methods and techniques introduced by the agile software development approaches.

### Learning Objectives...

- Review generally accepted testing processes to determine the benefits and drawbacks of each
- Examine the agile software development approaches and how testing is done in an agile manner
- Practice refactoring and pair programming to improve code development
- Employ the principles of Test Driven Development to improve code quality
- Examine integration testing tools and the process of continuous integration
- Apply agile testing techniques to all development life cycles and all testing practices

### Who should attend...

Programmers, testers, software developers, QC managers, software project managers, and anyone else interested in the details of agile testing practices, tools, and techniques.

### Prerequisites...

An understanding of software development testing processes.

**See next page for a detailed course outline...**



## Course Outline...

### Unit 1 Testing in the SDLC

#### Unit Objectives

#### General Approach to Software Development

- The Software Development Process
  - *Exercise: Why don't we get quality software?*
  - Mandatory elements (2)
  - Transitions

#### The Classic SDLC timeline

- Elements and phases (2)
- Test stages
  - Unit
  - Integration
  - System
  - Acceptance
- Issues with Linear Life Cycles
  - Testing in isolation
  - Determining a true baseline
  - The problem with late testing
- General approaches to quality assurance
  - It compiles, therefore it must work
  - Problems with code expansion

#### Unit Summary

- Summary
- Keys to Success

### Unit 2: Test Methods

#### Unit Objectives

#### General Approaches to Testing

- *Exercise: testing methods*
- Prepare, Execute, Document (2)
- Test cases
- Elements of a test (2)
- Sources of test cases
- *Exercise: Defining a test case*

#### Classic Testing Methods

- Script-Driven testing (2)
- Destructive testing
- Cause-effect testing
- Requirements based testing (2)
- Root cause testing

#### Alternate Test Execution Approaches

- Exploratory testing (2)
- Scenario testing



## Testing in Agile Development

- Rapid Application Development
- Test First Approach
- Test-driven development
- Extreme Programming (XP)

## Unit Summary

- *Exercise: Which method?*
- Summary
- Keys to success
- Applying test methods to your testing process

## Unit 3: The Concept of Continual Testing

### Unit Objectives

### The Rise of Agile Methods

- Different reasons for agile software development
- Time – schedule orientation
- The Agile Manifesto
- Quality and Agile?

### Testing from the Beginning

- The Circle of Life
- Agile approach to requirements – the User Story (2)
- *Exercise: creating user stories*
- Can there be too much testing?
- Balanced approach: coding and testing

### Testing during Development

- Context Driven Testing (3)
- The testing frameworks
  - Test Fixtures
  - Unit testing
  - Integration testing
  - Acceptance testing
- Defining tests
  - Defining the unit test
  - Defining the acceptance test
  - *Exercise : defining a test from user story*
- Setting up and organizing test cases
  - The structure of test organization
  - Separate the test class from code class
  - Add test class to build class
  - Stubs, drivers, and mock objects
  - *Workshop : Defining stubs and drivers*

## Unit Summary

- Summary
- Applying continual testing and test organization to your testing process
- Keys to Success



## Unit 4: Test Driven Development Overview

Unit Objectives

### Test First Development

- Test first philosophy (2)
  - PDSA cycle
  - Design direction
  - Verification method
- A testing methodology?
- How it works (2)
- *Workshop: applying test first approach to requirements*

### Design, Code, or Test?

- A testing methodology ?
- Rationale for test-driven development
  - Timeboxing
  - Accommodating change
  - “Red, Green, Refactor”
- TDD as a design approach
  - Specification not validation
  - Alternative to modeling?
- Complementary approaches

### Assessment of TDD

- Benefits of TDD approaches
- Limitations of TDD
- TDD and automation
- TDD pragmatics: project size, team size, languages
- TDD and traditional testing practices

### Unit Summary

- Summary
- Keys to Success

## Unit 5: Refactoring

Unit Objectives

### What is refactoring?

- Do the simplest thing that can work (2)
- Principles of refactoring (2)
- *Exercise: refactoring*
- Bad smells in code (3)
  - Cargo cult programming
  - Voodoo chicken programming

### Simplification

- Simplifying conditionals (3)
- Simplifying methods (3)
- Generalization (3)

### Unit Summary

- Summary
- Applying refactoring to your development process
- Keys to Success



- *Workshop : refactoring*

## Unit 6: Applying TDD at the Unit Level

### Unit Objectives

#### Developer testing

- Developer, programmer, unit testing
- Where TDD fits
- Smallest increment of functionality
- Rules of TDD
  - Write the test - Red
  - Write the code - Green
  - Refactor - Refactor

#### Using the Unit testing frameworks

- Framework Class hierarchy
- Runner Class diagram
- JUNIT framework (3)
- Installing XUNIT (3)
- Red, Green Refactor in Junit
- Nunit Errors
- XUnit rest results
- Typical XUnit assertion statements (2)
- Other Xunit frameworks
  - CppUnit (2)
  - VbUnit (2)
  - PyUNit (2)
  - HTMLUnit (2)
  - Others (2)

#### Building tests

- Value of self-testing code
- Mock and fake objects (3)
  - Abstract methods
  - Mock Object tools
  - Method stub
- Organizing data (3)
- White box and black box testing with TDD
- Tips for building test classes
- *Exercise: Building Tests*

#### Pair Programming

- Basis for pair programming (2)
- Pair programming roles (2)
- The tester in pair programming (3)
- *Exercise: Pair programming*

#### Unit Summary

- Summary
- Applying TDD to your development process
- Keys to Success
- *Workshop: Applying TDD*



## Unit 7: Continuous Integration

### Unit Objectives

#### Background

- Classic integration
- Unit and integration testing with TDD
- Scheduled builds
- Automation (2)
- A Successful Build (2)

#### Test Organization

- Organizing test cases for integration and regression
- Accessing tests and code

#### ANT

- Background
- ANT configuration file (2)
- Similarities to make
- Installing ANT (2)

#### CruiseControl

- What CruiseControl is all about
- The build loop (2)
- The dashboard (3)
- Getting started (2)
- Reporting (3)

#### Other approaches

- Hudson

#### Unit Summary

- Summary
- Applying continuous integration to your development and testing process
- Keys to Success
- *Workshop: CruiseControl*

## Unit 8: Agile Acceptance Testing

### Unit Objectives

#### Agile acceptance testing

- Acceptance tests come first
- Testing business rules
  - Involving the customer
  - Test first paradigm
- The safety net effect

#### FIT and FITnesse

- Background
- Preparation
- What is FIT?
  - Framework for an integrated test (FIT) (3)
  - Fitnessse – FIT with a Wiki
- Context Driven Testing revisited



- Fixtures (2)
- Setting up fixtures
  - Column fixture (2)
  - Row fixture (2)
  - Action fixture (2)
- *Exercise: setting up fixtures*
- HTML tables (3)
- Parsing tables (2)
- FITnesse description (3)
- Using FITnesse
- Organization of test cases
- *Workshop: Setting up a FITnesse table*

## Unit Summary

- Summary
- Applying TDD and automated acceptance testing to your QC process
- Keys to Success
- *Workshop: An automated acceptance test*

## Unit 9: Some Additional Agile Practices

### Unit Objectives

### Agile Development and Testing practices

- Extracting and applying the practices
- Extreme Programming (XP)
  - XP development and testing practices
  - Planning game
  - Working in pairs
  - Testing and coding trade-off
  - *Workshop: XP Planning game*
- Scrum
  - Scrum development and testing practices
  - The daily Scrum
  - Burn down lists and Quality Control
  - Working in teams
  - Incorporating a quality control role on the team
  - *Workshop: Scrum burn down list*

### Incorporating the practices in your process

- Phased approach
- Increasing collaboration
- What to do first?

### Applying TDD to structured approaches

- Test First for design
- Testing structured code first
- *Workshop: Adding the Practices*

## Unit Summary

- Summary
- Keys to Success



## Unit 10: The Bottom Line

### Ideas to Use

- Best practices in testing
- Best practices in agile testing
- Highlights

### Where to go for more information (8)

Course summary (2)

*Please contact your ROI representative to discuss course customization!!!*