Agile Risk-Based Proactive Testing

Course Summary

Description

Agile development on the one hand emphasizes integrating especially automated testing with development. On the other hand, though, Agile teams often lack key testing processes and important test planning, design, execution, and management skills. Without them, testing can be neither effective nor efficient. Agile's narrowly-focused short iterations aggravate but also mask such weaknesses. Projects benefit when cross-functional teams include skilled testing specialists and also when all members suitably understand effective testing processes and techniques. This interactive course introduces the powerful Proactive Testing methodology that enables you to do more effective testing in less time than traditional reactive testing, while also helping overcome traditional user, manager, and developer resistance. Applying special techniques that spot many of the highest yet ordinarily-overlooked risks, Proactive Testing truly "puts Agile testing on steroids." After a thorough grounding in Proactive Testing, concepts are extended to Agile's test-driven approaches. Participants practice each key concept/technique with a real case fact situation.

Objectives

At the end of this course, students will learn:

- A structured Proactive Testing model of testing that benefits Agile and traditional life cycles.
- Ways testing actually can cut time, effort, and aggravation for users, developers, and managers.
- Low-overhead industry-accepted test plans, designs, and cases that aid effectiveness and reliability.
- Multiple techniques/checklists to design more thorough tests and discover overlooked conditions.
- Analyzing risk and reusing testware to perform more of the important testing in less time.
- Practical ways to apply the concepts/techniques to short-iteration Agile test-driven development.

Topics

- How testing can cut effort & time
- Test planning value not busywork
- Detailed test planning
- White box (structural) testing
- Test design: both verb and noun
- Managing test execution
- Agile, user story fundamentals
- Requirements are requirements—
- Or maybe not
- Writing more suitable user stories
- Test-driven development
- User story acceptance tests

Audience

This course has been designed for developers, testing professionals, subject matter specialists, managers, and others who need more effective testing in Agile or other projects.

Duration

Three Days

Agile Risk-Based Proactive Testing

Course Outline

I. ***How Testing Can Cut Effort & Time***
- A. Why Test? Critical Concepts
- B. Testing For Correctness Vs. Testing For Errors
- C. Defect Injection, Detection, Ejection Metrics
- D. Reactive Testing—Out Of Time, But Not Tests
- E. Proactive Testing_ Life Cycle Model
- F. Agile, Test-Driven Unit And Acceptance Tests
- G. Uat, Cots Vs. Development/Technical Tests
- H. Cat-Scan Approach_ To Find More Errors
- I. V-Model And Objectives Of Each Test Level
- J. Dynamic Tests, Passive/Active Static Review
- K. Developer Vs. Independent Tester Testing
- L. Strategy—Create Fewer Errors, Catch More
- M. Test Activities That Save The Developer's Time

II. ***Test Planning Value Not Busywork***
- A. Proactive Vs. Reactive Risk Analysis
- B. Identify Overlooked Large Risks, Test Earlier
- C. IEEE Standard For Test Documentation
- D. Master Test Plan, Scoping
- E. Strategy Approach, Use Of Automated Tools
- F. Entry/Exit Criteria, Anticipating Change, Trace
- G. Exercise: Preventing Showstoppers
- H. Risk-Based Way To Define Best Test Units
- I. Letting Testing Drive Development
- J. Stomach Ache Metric; Agile?

III. ***Detailed Test Planning***
- A. IEEE Standard On Unit Testing
- B. Functional (Black Box) Testing Strategy
- C. 3-Level Top-Down Test Planning And Design
- D. Exercise: Functionality Matrix
- E. Use Cases, Revealing Overlooked Conditions
- F. Detailed Test Plan Technical Document

IV. ***White Box (Structural) Testing***
- A. Structural (White Box) Degrees Of Coverage
- B. Flowgraphing Logic Paths
- C. Applying Structural Paths To Business Logic
- D. Exercise: Defining Use Case Test Coverage

V. ***Test Design: Both Verb And Noun***
- A. Exercise: Disciplined Brainstorming
- B. Checklists Find More Overlooked Conditions
- C. Data Formats, Data And Process Models
- D. Business Rules, Decision Tables And Trees
- E. Equivalence Classes And Boundary Values
- F. Exploratory Tests Supplement Not Substitute
- G. Defect Isolation, Reproducibility,
- H. Formal, Informal Test Design Specifications
- I. Exercise: Defining Reusable Test Designs
- J. Test Case Specifications Vs. Test Data Values
- K. Writing Test Cases, Script/Matrix

VI. ***Managing Test Execution***
- A. Test Environment, Automation, Regression
- B. Defect Isolation, Analysis Reproducibility
- C. Defect Reports That Prompt Suitable Action
- D. Projecting When Software Is Good Enough
- E. Exercise: Measuring Test Effectiveness

Agile Risk-Based Proactive Testing

Course Outline

VII. ***Agile, User Story Fundamentals***
- A. Agile Manifesto's Relevant Points
- B. Characterization Of Traditional Approaches
- C. Agile's Sprints And Backlogs Alternative
- D. Prioritizing And Allocating To Backlogs/Sprint
- E. Agile Vs. Traditional Project Team Roles
- F. User Story "As A <Role>…" (Card)
- G. Invest User Story Guideline
- H. Exercise: Write User Stories
- I. User Story Acceptance Criteria (Confirmation)
- J. Given, When, Then Format
- K. Exercise: Define Their Acceptance Criteria
- L. Exercise: Review Your User Stories/Criteria
- M. Right User Stories For Right Acceptance Criteria
- N. Sizing, Splitting And Grooming Stories
- O. Exercise: Split/Groom Your User Story

VIII. ***Requirements Are Requirements—***

IX. ***Or Maybe Not***
- A. User Stories Are Backlog Items, Features
- B. Chicken And Egg Relation To Use Cases
- C. Business Vs. Product/System Requirements
- D. "Levels Model" Of Requirements
- E. Other Mistaken Presumptions
- F. Requirements Overview
- G. Where User Stories Should Fit, May Fit Instead
- H. Product Owner And Business Analyst Roles
- I. Responsibility For Defining User Stories
- J. Who Defines Features, Tasks To Create
- K. (Conversation) Conundrum
- L. Exercise: Write Business User Story

X. ***Writing More Suitable User Stories***
- A. Focus On Value
- B. Users, Customers, And Stakeholders
- C. Exercise: Identify Overlooked Stakeholders
- D. Problem Pyramid™ Tool To Find Value
- E. Exercise: Using The Problem Pyramid™
- F. Exercise: Write Business Value User Story
- G. Exercise: Identify Features For User Story
- H. Exercise: Design Code For Feature

XI. ***Test-Driven Development***
- A. Developer Writes Test-First Unit Tests
- B. Automating Tests, Regression Testing
- C. Applying Test Design Techniques
- D. Exercise: Define Test-First Unit Tests
- E. Exercise: Evaluate Unit Tests' Adequacy

XII. ***User Story Acceptance Tests***
- A. User Story Acceptance Criteria, Tests
- B. Differences From Test-First Unit Tests
- C. Automating, Cucumber Limits And Issues
- D. Tests Help Define Requirements
- E. Risks Of Relying On Tests As Requirements
- F. Suitability Of Using For Quality Factors
- G. Confirming Vs. Defining Requirements
- H. Where Are Tests For Missed Criteria
- I. Exercise: Write User Story Acceptance Criteria
- J. Exercise: Review Acceptance Criteria
- K. Turning Criteria Into Tests, Issues
- L. How Many Tests Are Really Needed, Risk
- M. Applying Test Design Techniques
- N. Exercise: Write User Story Acceptance Tests
- O. Exercise: Review Acceptance Tests
- P. Agile Executable Code Demos Vs. Uat
- Q. Testing That User Story Focus Misses
- R. Managing Test-Driven Process