Improving Quality with Proven Software Assurance Techniques

This is a 3-day workshop promoting a cohesive approach to testing: a "how-to" approach with exercises, examples, and templates that can be applied immediately to testing. It addresses the roles and responsibilities of each participant in the software development process. It outlines responsibilities, expectations, and mechanisms to measure performance and progress. The workshop emphasizes a practical approach to testing in order to create better products and addresses the ever -changing needs and resources of an organization. Students will learn how to move testing and QA techniques from "gut feelings & instinct" towards an engineering discipline. The workshop discussion is directed towards practical solutions to quality assurance problems. Techniques to ensure that an information system protects data and maintains functionality are discussed. The question of best ways of involving a QA team in the requirements definition process is investigated. We specifically address the problems of: a lack of resources, insufficient user community involvement, no budget for test automation, poor performance tracking metrics, overlaps in responsibilities, and other common pitfalls in a testing process.

Course Objectives:

- Examine the differences between unit testing and system testing.
- Review testing in an Agile methodology such as eXtreme Programming and ‘Programming by Contract.
- Translate requirements into tests, and demonstrate the value of early testing vs. late testing in a project.
- Ensure that testers are testing scenarios that are of concern to the end users.
- Develop strategies to implement better approaches to quality assurance.
- Ensure that an information system protects data and maintains system functionality.
- Use diagraming techniques to identify testable conditions from specifications.
- Identify the appropriate metrics to measure progress and performance in your organization.
- Write test plans that assure the desired amount of test coverage.
- Assess readiness to acquire test tools and automate the testing process.
- Establish criteria to start testing and determine when it is completed.

Audience: Quality assurance specialists, quality control analysts, system testers, programmers, end users, business analysts, project managers, team leaders, support analysts, engineers and acceptance testers.

Prerequisites: None

Number of Days: 3 days

1 Introduction: Defect Detection or Defeat Prevention:

Objectives / Observations
Impediments to Quality
Role of the Tester
Responsibilities
Early Testing vs. Late Testing
Quality Assurance Assessment
Quality Issues and Elements
Quality Improvement Suggestions
Quality Tools and Steps
Opportunities to Improve the Testing Process
System Development Life Cycles

Waterfall SDLC
Spiral SDLC
Agile
Phase Objectives
Measuring Performance
Reliability Metrics
Testing Success Factors
Product Development and Testing Phases

2 Major Software Development and Testing Issues
Preparing Specifications
Writing User Manuals
Writing Skills
QA / QC Responsibilities
Reviewing Project Specifications
Scripts and Cases
Unit vs. System (or Acceptance) Testing
Scripted vs. Exploratory Testing
Building Confidence in Testing
Positive and Negative Testing
Blind Testing
Unit-level Test Scripts
System-level Test Scripts
Managing Change
Levels of Testing
Responsibilities by Test Type

3 Test Methodologies & Checklists
Setting Test Objectives and Identifying Tests
Test Planning
Methodologies
Boundary Value Analysis
Path Analysis
Decision Tables
State Machines
State Transition
Factor Analysis
OATS
Pairs and Magic Squares
Embedded Systems
Checklists

4 Risk Analysis
Categorical Analysis
Factor Breakdown
Business Rule Analysis
Operational Matrix

5 Test Planning
Unit Testing (Early Testing)
Integration Testing and System Testing
System / Acceptance Testing
Creating the Regression Test
Traceability Matrix
Usability Testing
Palm Compliance Testing Checklist
Stopping Rules for Testing
Estimating with Function Points
How Do I Estimate the Testing Effort?
Data Dictionaries
Approaches to Testing
Regression Testing
Alternatives to Testing
Test Notebook

6 Test Plan Reviews

7 Promotion Rules for Applications

8 Test Modifications
Maintenance Issues
Maintenance Testing
Estimating the Modifications
Cost Benefit Identification

9 Upstream/Downstream Testing

10 Integration Testing Errors
Checklist of Integration Issues
Error Prevention Checklist

11 Degraded Mode Testing Techniques

12 Defect Prevention
Checklists
Functional Specification Defects
Designing Defects
Coding Defects
Testing Defects
Coding / Testing Rules

13 Test Management
Test Logs
Sample Defect Tracking Report
Test Log Scenarios
Retesting and Follow-up Procedures
Root Cause Analysis

14 Problem Solving Techniques
Error Isolation
Variable Tracers
Flowcharts

Deductive Questioning
Structured Walkthroughs
JAD – Joint Application Design

15 Object-Oriented Testing
Definitions
00 vs Traditional Testing
Managing Complexity
Abstraction
Encapsulation
Inheritance
Object-Oriented Systems Testability Issue
Object-Oriented Testing Approach
Using Test Clients
Other Testing Issues

16 Second Chapter Title
Automated Testing Considerations
Test Tools

17 Web-based client/Server Testing
Web-based Testing-Where to Begin?
What Will Be New?
Determining What to Test
Where to Test: Client-side or Server-side?
Web Testing Responsibilities
Web Testing Checklist
Agile Methodology and Testing Agile Manifesto
What Changes with Agile?
Agile Principles
eXtreme Programming
Productivity Measure: Velocity XP Basic
Rules and Definitions Testing in XP Shops
Basic XP Practices

18 Systems Architecture
Test GUIs/APIs
Local (One Node) Testing
Multiple clients/Multiple Servers

19 Cross Functional Analysis

20 Database Integrity Testing
Declarative Checking
Procedural Checking
Checklists

21 Appendix
Increasing Productivity
Testing Environment
System Testing Without a Specification
Job Responsibilities of System Testers
Web Sites

22 Glossary