Proactive Software Quality Assurance and Testing

Course Summary

Description

For many organizations, QA just means tail-end testing, catching errors right before they go out the door when they are too expensive and risky to fix. Effective systems organizations involve QA throughout the software process, catching and preventing errors early. It's not easy though—two-thirds of full-fledged QA groups fail. This workshop explains how QA can provide far greater value, analyzes why they so frequently have failed in IS, and presents practical approaches for successfully using QA effectively to produce high quality systems. Exercises reinforce learning.

Objectives

At the end of this course, students will be able to:

- Understand the reason for QA failures and factors critical to success of QA in IS development.
- Use the six functions that Proactive Software Quality Assurance should perform.
- Understand the proven quality management and review methods for promoting quality and preventing errors.
- Use a structured Proactive Testing model of which testing activities should be performed when and by whom with the life cycle to maximize testing efficiency and effectiveness.
- Test agile planning techniques that prevent showstoppers.
- Design tests that spot numerous ordinarily-overlooked defects in less time.
- Report defects so they get addressed.
- Measure system quality and SQA/Testing effectiveness.

Topics

- Why we have poor System Quality
- Software Process and Quality
- Quality Assurance Concepts
- Active Static Testing
- How Testing can cut effort and time
- Test planning value not busywork
- Detailed test planning
- Integration/System test planning
- Test Design: both Verb and Noun
- Measuring and Managing Testing

Audience

This course has been designed for systems and business managers, project leaders, systems analysts, business analysts, quality specialists, auditors, and others responsible for the quality of information systems.

Prerequisites

There are no prerequisites required for this course.

Duration

Three days

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

ProTech
protechtraining.com

"Charting the Course ...

... to Your Success!"

Proactive Software Quality Assurance and Testing

Course Outline

I. Why we have poor System Quality
- A. What is at stake--for me?
- B. Need for positive common quality definition
- C. Models of excellence, quality before TQM
- D. Quality factors and quality dimensions
- E. Engineered Deliverable Quality™

II. Software Process and Quality
- A. REAL vs. Presumed processes
- B. Silos that perpetuate problems
- C. Economics of quality problems in life cycle
- D. Defect injection, detection, ejection metrics
- E. Process capability, control, and variance
- F. Statistical process control in IS
- G. SEI Process Capability Maturity Model
- H. Exercise: TQM and Quality Circles

III. Quality Assurance Concepts
- A. Why SQA groups so often fail
- B. QA vs. Testing, common distinctions
- C. Proactive SQA™ for effectiveness
- D. Proactive Testing™ Life Cycle model
- E. 6 functions of effective software QA
- F. QA Plans, quality reviews of deliverables
- G. Engineering standards, conventions
- H. Quality controls at all key points
- I. Recordkeeping and auditing
- J. Metrics and analysis for improvement
- K. Promoting awareness and recognition

IV. Active Static Testing
- A. Often overlooked walkthrough limitations
- B. Formal technical reviews, procedures
- C. Why reviews so economically find defects
- D. Why review of requirements fails
- E. Exercise: Reviewing Requirements
- F. Unrecognized weaknesses of "Regular Way"
- G. Evaluating requirements form, testability
- H. REAL, business vs. system requirements
- I. Finding overlooked, incorrect requirements
- J. Reviewing design suitability and content
- K. Four powerful design review CAT-Scans

V. How Testing can cut effort and time
- A. Testing for correctness vs. testing for errors
- B. Reactive testing—out of time, but not tests
- C. V-model and objectives of each test level
- D. CAT-Scan Approach™ to find more errors
- E. Dynamic, passive and active static testing
- F. Developer vs. independent test group testing
- G. Strategy—create fewer errors, catch more
- H. Test activities that save the developer's time

VI. Test planning value not busywork
- A. Risk elements, relation to testing
- B. Proactive vs. reactive risk analysis
- C. IEEE Standard for Test Documentation
- D. Enabling manageability, reuse, selectivity
- E. Master Test Plan counterpart to project plan
- F. Approach, use of automated tools
- G. Entry/exit criteria, anticipating change
- H. Exercise: Anticipating showstoppers
- I. Risk-based way to define test units
- J. Letting testing drive development
- K. Preventing major cause of overruns

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Proactive Software Quality Assurance and Testing

Course Outline (cont'd)

VII. 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. Exercise: Defining use case test coverage
- G. Detailed Test Plan technical document

VIII. Integration/System test planning
- A. Graphical technique to simplify integrations
- B. Integration test plans prevent schedule slips
- C. Smoke tests; system and special testing
- D. Daily, top- and bottom-down builds strategy

IX. 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. Error guessing, condition combinations
- 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. Exercise: Writing test cases, script/matrix

X. Measuring and Managing Testing
- A. Defect reports that prompt suitable action
- B. Projecting when software is good enough
- C. Exercise: Measuring testing effectiveness
- D. Exercise: Post-Implementation Review

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically