ProTech
protechtraining.com

"Charting the Course ...

... to Your Success!"

Proactive Testing Workshop

Course Summary

Description

Proactive Testing™ enables you to deliver better software in less time by doing more effective testing, while also providing the value that overcomes traditional user, manager, and developer resistance. Applying special strategies and techniques that spot many of the highest yet ordinarily-overlooked risks, Proactive Testing™ makes sure the most important unit/component, integration/assembly, system, and UAT testing is done in limited available time. Moreover, by managing within an overall Quality perspective that catches more defects earlier when they are easier to fix, and actually prevents many showstoppers and other errors, Proactive Testing™ also can cut developers' time, effort, and aggravation. After establishing core concepts, this interactive workshop shows Proactive ways to apply powerful proven structured test planning and design techniques that produce value, not busywork. To enhance learning, participants practice each key technique in a series of exercises with various aspects of a real case fact situation.

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

- Demonstrate a structured Proactive Testing model of testing that should be performed throughout the life cycle.
- Learn ways testing actually can cut, effort, and aggravation for users, developers, and managers.
- Write industry-accepted test plans, designs, and cases that make testing easier and more reliable.
- Learn multiple techniques/checklists to design more thorough tests and discover overlooked conditions.
- Manage test execution, including estimating/allocating resources and reporting defects and status.
- Apply risk analysis and reusable testware to perform more of the important testing in less time

Topics

- How testing can cut effort & time
- Test planning value not busywork
- Proactive master test planning (big risks)
- Detailed test planning (medium-sized risks)
- White box (structural) testing
- Integration/system/special test planning
- Test design: both verb and noun (small risks)
- Maintenance and regression testing
- Automated testing tools
- Measuring and managing testing

Audience

This course has been designed for testing professionals and others who manage and perform testing of software products, and also for analysts, designers, and system/project managers who need to know how Proactive Testing™ can cut software development time and effort.

Prerequisites

There are no prerequisites for this course.

Duration

Three days

ProTech
protechtraining.com

"Charting the Course ...

... to Your Success!"

Proactive Testing Workshop

Course Outline

I. How Testing Can Cut Effort & Time
- A. Testing for correctness vs. testing for errors
- B. Developer views of testing
- *C. Exercise: Your defined testing process*
- D. What is a process, why it matters
- E. REAL vs. presumed processes
- F. Why most IT process improvement efforts fail
- *G. Exercise: Your REAL testing process*
- H. Meaningful process measures, results, causes
- I. Defect injection, detection, ejection metrics
- J. Economics of quality problems in life cycle
- K. Keys to effective testing
- L. CAT-Scan Approach™ to find more errors
- M. Dynamic, passive and active static testing
- N. Developer vs. independent test group testing
- O. V-model and objectives of each test level
- P. Reactive testing—out of time, but not tests
- Q. Proactive Testing™ Life Cycle model
- R. Proactive user acceptance criteria
- S. Strategy—create fewer errors, catch more
- T. Test activities that save the developer's time
- U. Applying improvements

II. Test Planning Value Not Busywork
- A. Why test planning often is resisted
- B. Buzzword boilerplate platitudes paperwork
- C. Test plans as the set of test cases
- D. Six reasons to plan testing
- E. Risk elements, relation to testing
- F. Traditional reactive risk analysis, issues
- G. IEEE Standard for Test Documentation
- H. Overcoming controversial interpretations
- I. Testing structure's advantages
- J. Enabling manageability, reuse, selectivity
- K. Test plans, designs, cases, procedures

III. Proactive Master Test Planning (Big Risks)
- A. Exercise: Anticipating showstoppers
- B. Spotting overlooked large risks
- C. Involving key stakeholders, reviewing plans
- D. Formal and informal risk prioritization
- E. Dynamic identification of design defects
- F. Risk-based way to define test units
- G. Letting testing drive development
- H. Preventing major cause of overruns
- I. Stomach ache metric
- J. Testing highest risks more and earlier, builds
- K. Master Test Plan counterpart to project plan
- L. Strategy approach, use of automated tools
- M. Sequence of tests, sources of data
- N. Entry/exit criteria, anticipating change
- O. Test environment, supporting materials
- P. Estimating testing, avoiding traps
- Q. Roles, responsibilities, staffing, training
- R. Schedule, risks and contingencies, sign-offs
- S. Management document, agreements
- T. Maintaining the living document

IV. Detailed Test Planning (Medium-Sized Risks)
- A. IEEE Standard on Unit Testing
- B. Requirements-based functional testing
- C. Non-functional requirements challenges
- D. Black Box testing strategy
- E. 3-level top-down test planning and design
- F. Detailed Test Plans for large risks
- G. Exercise: Functionality matrix
- H. Test designs for medium-sized risks
- I. Use cases, revealing overlooked conditions
- J. Detailed Test Plan technical document

V. 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
- E. Flaws of conventional use-case testing
- *F.* Exercise: Additional use case conditions

VI. Integration/System/Special Test Planning
- A. Risks, issues integration testing addresses
- B. Graphical technique to simplify integrations
- C. Integration test plans prevent schedule slips
- D. Smoke tests, increasing their value
- E. Special tests
- F. Load, performance, stress testing
- G. Ongoing remote monitoring, reliability
- H. Security, configurations, compatibility
- I. Distribution and installation, localization
- J. Maintainability, support, documentation
- K. Usability, laboratories raising the bar

Proactive Testing Workshop

Course Outline (cont'd)

VII. Test Design: Both Verb And Noun (Small Risks)
- A. Why tests need to be designed
- B. Appropriate use of exploratory testing
- C. Exercise: Disciplined brainstorming
- D. Checklists, ad hoc exploratory pros and cons
- E. Data formats, data and process models
- F. Exercise: Applying checklists
- G. Business rules, decision tables and trees
- H. Exercise: Create a decision table
- I. Equivalence classes and boundary values
- J. Exercise: Identify logical equivalence classes
- K. Formal, informal Test Design Specifications
- L. Exercise: Defining reusable test designs
- M. Complex conditions, defect isolation
- N. Test Cases for small risks
- O. Test Case Specifications vs. test data values
- *P.* Exercise: Writing test cases, script/matrix

VIII. Maintenance And Regression Testing
- A. Maintenance vs. development, why so harder
- B. Improve attention and knowledge
- C. Regression testing, minefield effect
- *D.* Exercise: Testing maintenance changes

IX. Automated Testing Tools
- A. Key test automation issues
- B. Tools for a managed environment
- C. Coverage analysis, execution aids
- D. Test planning, design, administering
- E. Automated test execution tools, issues
- F. Scripting approaches, action words

X. Measuring And Managing Testing
- A. What is a test case survey
- B. Relevance for estimating test-based tasks
- C. Traceability concepts and issues
- D. Estimating non-test-based test project tasks
- E. Defect reports that prompt suitable action
- F. Determining defect age
- G. Status reporting people pay attention to
- H. Projecting when software is good enough
- I. Defect density, reductions
- J. Defect detection/removal percentages
- K. Exercise: Measuring testing effectiveness