

MT 18
2 Days

Test Strategy

This course provides an overview and rationale of the various kinds of testing performed during the lifetime of a project. Attention is paid to the ROI in terms of testing benefits and modern “incremental” testing is presented together with more traditional forms. Management software for testing is discussed and automatic test tools are presented and their advantages and disadvantages reviewed. Attendees will obtain a good overview of types of testing, when testing is applicable and how to measure its efficiency. More importantly, the question “how do you know when testing is finished” is answered.

Course Objectives

The course covers:

- Test Plans and the Master Test Plan
- Software lifecycles - RAD, XP, Staged, Waterfall and how testing is performed within them
- Testing Types
- Prioritised Testing
- Bug Classification
- Requirements gathering and management
- Acceptance test design
- Inspections and reviews
- Quality Processes
- Testing and the UML
- Reusable tests
- Creating a Testbed
- Continuous Process Improvement
- Decision Tables for test planning
- Problem avoidance

Audience

- Project Managers
- Lead analyst / programmers
- Staff needing a view on Test Planning
- Staff expecting to move test groups

Prerequisites

- Exposure to software development
- Familiarity with multi-person projects

Timetable

Register at 09:00 on day one for 09:30 start. 09:00 start on successive days. Finish at 17:00 each day.

Presentation Style

Lectures, demonstrations and group discussions.

Dates and Venues

Refer to *Course Schedules*.

MT 18
2 Days

Test Strategy

<p>Introduction to Testing</p> <ul style="list-style-type: none">What it isWhat it is notInfluence of LifecycleRADSpiralWaterfallV-ModelAgile ProcessesExtreme ProgrammingTDD <p>Test Strategies</p> <ul style="list-style-type: none">Error definitionsFault classificationsTraceability matrixRisk AssessmentTest ROIInspectionsReviewsProject Test StrategiesTest Design and Automation <p>Requirements Management</p> <ul style="list-style-type: none">Good Requirements and how to get themFunctional RequirementsNon-functional RequirementsDesigning Acceptance TestsTable-based specifications <p>UML and Testing</p> <ul style="list-style-type: none">UML IntroductionUML and Design ProcessUse Cases and RequirementsScenarios with Test DirectorTest Case DerivationLock-in to Acceptance TestsDesign Tools: Rational Rose	<p>Testing Types/Phases</p> <ul style="list-style-type: none">FunctionalSystemUser acceptancePerformanceUsabilityConfigurationSecurityStressRecoveryGUIRegression testing <p>Creating & Maintaining a Test Bed</p> <ul style="list-style-type: none">The Testing LifecycleRe-usable TestsEquipmentTest databasesConfiguration controlRegression testingData conversions <p>Code Testing</p> <ul style="list-style-type: none">Design of Code TestsDecision Tables for designBlack Box testingWhite Box testingEquivalence TestingDocumenting Code Tests <p>Test Metrics</p> <ul style="list-style-type: none">Test MetricsMetric GatheringUseful Metric AnalysisThe Metrics DatabaseROI and Ease of UseDrowning in Paper <p>The Master Test Plan</p> <ul style="list-style-type: none">Contents and useANSI/IEEE standardsIdentification & prioritizationReporting Results	<p>Test Completion</p> <ul style="list-style-type: none">When is enough?Quality vs. testingAnalysisLessons LearnedFeedbackContinuous Improvement
--	--	--