Software Quality Assurance Principles and Practices

Course Summary

Description

This course is a two day overview of the theory and concepts of software quality assurance (SQA) and an exploration of the techniques and processes used to implement SQA programs within an organization.

Modern SQA has emerged from the fusion of several disciplines: the first is the body of quality assurance developed over the last century for business and manufacturing, the second is the more recent work on software development process maturity and related testing maturity models, and the third is the establishment of a number of formal quality standards such as ISO/IEC:25010.

The basic foundations of SQA are introduced with discussions on what quality is and how we derived actual working quality models that can be used and implemented in operational environments. In addition to the more formal standards, the current thinking on "good enough quality" and continuous quality are covered. These ideas are integrated with the current work on software process maturity, testing maturity models and how these concepts integrate with modern Agile, DevOps and other development models.

Each of the basic areas of software quality is explored. In the section on software quality management, students work through the high level activities of risk and requirements analysis, integrating quality management into the software project planning and developing the appropriate software testing plans and processes. Also covered is the establishment of metrics to evaluate how well testing and other quality activities are being performed in order to implement testing process quality improvements.

The software quality control section walks though how we translate the software quality requirements into specific testing goals, plans, and procedures and how testing activities are evaluated and modified to attain the quality objectives. The course does not cover any software testing techniques. A brief overview of statistical quality control, using six sigma as an example, is included in the discussion.

The section on software quality assurance focuses on the activities to evaluate the software development process including: integration of testing throughout the development life cycle, defect prevention, the development of best standards for continuous improvement, defect analysis, root cause analysis and defect analysis.

A variety of different techniques are presented that are used in a SQA environment: conducting reviews, inspections and walkthroughs, formal verification techniques, model validation, code metrics, the use of automated tools for various SQA functions. This provides an introduction into some of the challenges faced in modern mission critical software systems and how these are addresses using techniques focused on fault tolerance, robustness and what SQA means for real time systems.

The course concludes with a discussion the management issues and challenges faced when trying to implement a software quality program within an organization.

Topics

- What is software quality?
- Implementing software quality
- Software Quality Management
- Software Quality Control
- Testing Metrics
- Software Quality Assurance
- Beyond Testing I: Reviews and Inspections
- Beyond Testing II: Formal Verification
- Reliability Engineering and Fault Tolerant Systems
- Quality Roles in Organizations
- Summary

Software Quality Assurance Principles and Practices

Course Summary (cont’d)

Audience

The course is appropriate for software testers and QA staff, managers and team leads or anyone who needs to understand the basics of software quality assurance.

Prerequisites

The course has no prerequisites.

Duration

Two days

Software Quality Assurance Principles and Practices

Course Outline

I. What is software quality?

A. The problem of defining quality
B. Various approaches to defining quality
C. Quality attributes and stakeholder expectations
D. Quality dimensions: robustness, correctness, etc.
E. Quality frameworks and standards (ISO/IEC: 25010, etc.)
F. Effective software, efficient development, validation, and verification
G. The relationship between software testing and software quality

II. Implementing software quality

A. Beizer's tester's levels
B. Software development process maturity
C. The testing maturity model
D. Maturity level quality practices
E. Quality practices that drive software testing
F. Quality practices as part of project management
G. Quality practices integrated with software development
H. Quality practices and requirements interrelationships

III. Software Quality Management

A. "Good enough quality" (GEQ) model
B. Risk analysis, assessment, and mitigation
C. Deriving objectives from requirements
D. Assessments of dimensions of quality
E. Quality assessment strategies
F. Setting testing objectives and test planning
G. Developing quality metrics for a project
H. Challenges and best practices in software quality management

IV. Software Quality Control

A. Testing goals, method definition and establishing pass/fail criteria
B. Test planning and preparation
C. Test execution, data collection, and analysis
D. Analysis and follow-up
E. Test planning and a project management activity
F. Test automation and tools
G. Statistical quality control methods

V. Testing Metrics

A. Evaluation test effectiveness
B. Test process improvement
C. Correlation models
D. Direct and indirect measurement
E. Accuracy, reliability, validity and comprehensiveness
F. Testing cost-benefit analysis

VI. Software Quality Assurance

A. Life cycle testing and quality control
B. Proactive defect prevention
C. Root cause analysis
D. Analysis, documentation and reporting
E. Driving process improvement
F. Effectiveness measures and validity
G. Efficiency measures and verification
H. SQA and process maturity
I. Supporting best practices and standards

VII. Beyond Testing I: Reviews and Inspections

A. Reviews throughout the life cycle
B. Reviews, inspections and walkthroughs
C. Scenario analysis and persona development
D. Best practices for running reviews and inspections
E. Review planning, execution and follow-up

Software Quality Assurance Principles and Practices

Course Outline (cont'd)

VIII. Beyond Testing II: Formal Verification
- A. Axiomatic and logical techniques
- B. Model validation and testing
- C. Code metrics and automated tools

IX. Reliability Engineering and Fault Tolerant Systems
- A. How good is good enough?
- B. Modern distributed systems
- C. Mission critical software
- D. Fault containment and recovery
- E. Real time systems and quality

X. Quality Roles in Organizations
- A. Quality Management and organizational issues
- B. The business case for software quality
- C. People, tools, resources and policies
- D. Developing a quality profession in an organization
- E. Adopting industry standards and practices
- F. Special Challenges of DevOps and Agile

XI. Summary
- A. Review of the course
- B. How to move forward and apply what has been learned
- C. Other topics requested by students
- D. Assessment of knowledge and skills gained