2.0 Course Name : Software Quality Assurance-Implementing Consistent Quality

Course Duration : 300 hours

Course Outline :

Introduction to Quality Assurance

- Contrasting roles: Quality Assurance, Testing, Verification and Validation
- Comparing software development life cycles
- Documenting processes
- Defining the goals of Quality Assurance

Quality Assurance Components

- Creating processes
- Choosing the best practices and implementing process improvement initiatives
- Comparing Agile and traditional QA roles and methods
- IEEE
- CMMI
- ISO 9001
- Selecting and documenting standards
- Participating in reviews and audits
- Maintaining records

Planning for Quality Assurance

- Evaluating verification and validation techniques
- Analyzing life cycle products
- Implementing a QA policy and plan
- Exploring testing levels
- Defining the inspection process
- Planning and conducting an inspection
- Communicating inspection results

Conducting Audits

- Comparing process, product and projects
- Implementing quality system and configuration audits
- Documenting audit findings in a report
- Complying with industry standards and models: ISO 9001 and CMMI
- Comparing the work products against industry best practices
- Planning and preparing for the audit
- Reporting the results
- Monitoring noncompliance
- Reviewing documentation
- Documenting and confirming audit findings
- Presenting audit findings

Applying Configuration Management (CM)

- Identifying the workflow and work products
- Assessing and managing components with release management
- Communicating product status using reports

- Controlling QA and/or test plans
- Monitoring quality reports, audit findings and peer review documentation

Continuous Process Improvement

- Defining and implementing process improvement
- Planning process improvement initiatives
- Selecting and analyzing metrics
- Communicating organizational progress
- Determining the possible causes of problems utilizing a Fishbone diagram
- Narrowing down the correct course of action by creating a Pareto Analysis diagram
- Examining flow charts during the root cause analysis process
- Implementing corrective actions
- Focusing on prevention techniques

Testing in an Agile Environment

- Evaluating the key testing principles
- Differentiating between Agile and traditional practices
- Introducing the theory and purpose of Agile Testing
- Mapping Agile principles and values to testing
- Inspecting Agile testing quadrants
- Benefiting from Test Driven Development (TDD)
- Automating testing for better Agility

Confirming Customer Satisfaction

- Testing the charter and key features
- Focusing on customer value and user personas
- Writing useful test cases from user stories
- Developing Story Acceptance Criteria
- Designing the anatomy of an Agile Test
- Creating a Test Idea Catalog
- Refining a Definition of Done and Ready
- Anticipating validation criteria through Behavior Driven Development (BDD)
- Specifying by examples and scenarios
- Enabling Usability and Exploratory Testing
- Performing Story-Mapping for better coverage
- Managing the UAT processes

Implementing Developer and Technology Testing

- Defining the unit candidates for testing
- Achieving green-light success
- Identifying good tests
- Dealing with large systems
- Finding non-functional testing patterns
- Simulating interfaces and conducting performance testing through automation
- Evaluating legacy systems

Test Management

- Creating a risk-based approach to release planning
- Coordinating day-to-day through a lightweight test plan

- Declaring rules on zero-bug tolerance and defect tracking
- Using metrics to measure success
- Implementing continuous integration and deployment
- Setting-up JIT test environments
- Organizing post-release testing
- Working toward a whole team approach to quality
- Designing software with high testability attributes
- Sharing testing responsibilities between developers and testers
- Collaborating on complex and distributed teams projects