



Course 300: Approaches to Software Engineering

Course Description...

Software engineering is a systematic, disciplined, quantifiable approach to developing, operating, and maintaining mission critical applications. This course explores the major components of software engineering – development, maintenance, and support activities, as defined in industry standards. The student with a professional interest in creating or maintaining software, or managing software projects, or monitoring outsourced development and maintenance efforts, will gain a “big picture” understanding of software engineering. The insights can point the way toward more effective approaches to producing high-quality maintainable software.

Learning Objectives...

- Understand the scope and goals of software engineering.
- Distinguish software engineering from conventional software development.
- Explore the characteristics of sound software engineering processes: requirements definition and analysis; design; software construction; and testing.
- Examine the nature and roles of support activities essential to successful software engineering projects: project management; configuration management; and quality assurance.
- Identify techniques that enable long-term, cost-effective maintenance of mission critical software: corrective, adaptive, and perfective maintenance.
- Understand the customer’s and contractor’s roles in an outsourced software engineering project.

Who should attend...

This course is suited for software practitioners, project managers, software engineering process designers, and quality assurance and configuration management professionals.

Prerequisites...

No specific prerequisites are assumed. A familiarity with information system concepts is recommended.



Course Outline...

Unit 1: Introduction to Software Engineering

Unit Objectives

Software Engineering Concepts

- Definitions of software and software engineering
- Purpose of software engineering
- Scope of software engineering
- Software engineering standards

Software Engineering Process

- Process structure
- Recursive nature of processes
- Process standards
- Phase names

Life Cycle Models

- Sequential models
- Iterative models
- Agile models
- How to pick a model for your project

Workshop: Picking a life cycle model for a project

Unit Summary and Best Practices

Unit 2: Requirements

Unit Objectives

Requirements Concepts

- Definitions
- Need for requirements
- Establishing the system context
- User, system, and software requirements
- Requirements standards

Eliciting Requirements

- Identifying stakeholders
- Methods of elicitation

Specifying Requirements

- Specification contents
- How to write functional requirements
- How to write performance requirements
- How to write system characteristics
- How to write design constraints

Workshop: Writing a structured use case description



Analyzing Requirements

- Purpose of the analysis phase
- Structured analysis: process modeling
- Structured analysis: data modeling
- Object-oriented analysis

Validating Requirements

- Inspections vs. reviews
- Prototyping
- Checklists
- Traceability

Workshop: Validating requirements

Managing Requirements

- Change management
- Using traceability matrices
- Requirements management tools

Unit Summary and Best Practices

Unit 3: Design

Unit Objectives

Design Concepts

- Definitions
- The design process

Design Drivers

- Identifying candidate drivers
- Selecting drivers for a given application

Design Views

- Structural views
- Behavioral views
- Design view standards

Architectural Design

- Definitions
- Design patterns
- Reference models
- Decomposition methods

Detailed Design

- Transforming functions into procedures
- Pseudocode vs. flowcharts



Design Verification

- Inspection vs. review
- Prototyping and simulation
- Design tools

Workshop: Choosing among design alternatives

Unit Summary and Best Practices

Unit 4: Construction

Unit Objectives

Construction Process

- Steps
- Code verification
- Language standards

Procedural Languages

- Where and when to use
- Examples

Non-Procedural Languages

- Where and when to use
- Examples

Visual Languages

- Where and when to use
- Examples

Coding Standards

- Need
- Examples

Tools

- Compilers
- Interpreters
- Code generators
- Debuggers

Workshop: Choosing a language

Unit Summary and Best Practices



Unit 5: Testing

Unit Objectives

Testing Process

- Definitions
- Goals
- Test planning
- Test scenario design
- Test execution and analysis
- Role of configuration management
- Testing standards

White Box Testing

- Definitions
- Unit testing
- Integration testing
- Regression testing

Black Box Testing

- System testing
- Acceptance testing
- Regression testing

Testing in the Maintenance Phase

- Maintenance test cases
- Regression test cases
- Sustaining the regression test suite
- Using the traceability matrices

Testing Tools

- Automation issues
- White box test tools
- Black box test tools
- Test management tools

Workshop: Creating a test strategy

Unit Summary and Best Practices

Unit 6: Maintenance

Maintenance Process

- Goals of the maintenance process
- Corrective maintenance
- Adaptive maintenance
- Perfective maintenance



Keys to Successful Maintenance

- Documentation
- Configuration management
- Regression test suite
- Traceability and other factors

Workshop: Monitoring maintainability

Unit Summary and Best Practices

Unit 7: Support

Unit Objectives

Support Concepts

- Definitions
- Need for support functions
- Support process standards

Software Engineering Management

- Planning
- Tracking

Software Configuration Management

- Identification
- Control
- Configuration audits
- Status accounting

Software Quality Assurance

- Contrast with testing
- Software process definition
- Process auditing
- Metrics

Unit Summary and Best Practices

Unit 8: The Bottom Line

Ideas to use

Where to go for more information

Please contact your ROI representative to discuss course tailoring!!!