Course ID
SWENG2
Course Duration
3 days

Course Title
Software Engineering: An Advanced Tutorial

Related Courses

- Project Management Workshop (PROJMGT2D, 2 days)
- Project and Team Management Workshop (PROJMGT4D, 4 days)

Aimed At

Individuals involved in software development who have a general understanding of the software development lifecycle and wish to deepen their knowledge of software engineering techniques and tools.

Group Size

5-25

Prerequisites

- Principles of Software Engineering (SWENG1, 2 days)

Course in a Nutshell

This, the second of our two courses on software engineering, focuses on the software engineering toolkit.

Building on the overview of the software lifecycle provided by the first course, it undertakes an in-depth study of the methodology and practice of software engineering. Upon completion of this course, you will have acquired a good understanding of the tools and techniques used throughout the software lifecycle from the conception to the installation and maintenance.

Customize It!

- *Are you a project manager, analyst, programmer, testing/debugging, or integration specialist and* need a better understanding of how your work fits into the software lifecycle? We can emphasize the phases relevant to your particular needs while also trying to give you a broad understanding of the full lifecycle.
- *Are you involved in certain areas or phases of software development?* The course can be tailored to focus on those areas or phases.
- *Are you interested in particular techniques of software development?* If so, we can emphasize those parts of this course that focus on the areas and tools pertinent to your project or product.
- *Are you involved in the marketing, sales, or post-sales support of software* and need a better understanding of how your job fits into the overall software lifecycle? We can customize the course to focus on those issues.

Learn How To

Describe and apply the key concepts and techniques for:

- Software inception: Feasibility and estimation.
- Requirements elicitation and definition.
- Analysis and different design approaches and patterns.
- Management of configuration, change, and risk.
- Joint Application Development (JAD)
- Prototyping and user interfaces.
- Object oriented analysis, design, and programming.
- Coding and testing.
- Deployment and maintenance.
- Project management.
- Verification

Course Outline

- Introduction: Why Study Software Engineering?
- Life Cycle Models
    - Software Project Life Cycle Model
    - Software Process Life Cycle
    - Organization Process Assets
    - Software Project Life Cycle Process
    - Model Descriptions
    - Strengths and Weaknesses
    - Case Study
- Phase Artifacts/Outputs
- Software Life Cycle vs Project Management Life Cycle
- Objectives, Principles, Attributes
    - Objectives: Reusability, Maintainability, Concurrent Documentation, Testability, Correctness, Reliability, Portability
    - Principles: Concurrent Documentation, Hierarchical Decomposition, Functional Decomposition,Information Hiding, Stepwise Refinement, Structured Programming, Life-Cycle Verification
    - Attributes: Reduced Coupling, Enhanced Cohesion, Reduced Complexity, Well-Defined Interfaces, Readability, Ease of Change, Traceability, Visibility of Behavior, Early Error Detection
- Documentation Principles
    - Concurrency, Maintainability, Correctness, Portability, Reusability, Reliability, Testability, Traceability, Adaptability, Accessibility
- Types of Interfaces
- Project Initiation
    - Description of the Business Needs/Problem to Be Solved, Objective(s),

Scope
    - Feasibility Studies and Estimating
    - People Management (Socio-Technical Systems)
        - Technical and social skills and skill sets
        - Team size and communication
        - Developing, maintaining, and updating a resourced project schedule and resource plan.
    - Configuration Management
    - Change Management
    - Risk Management

- Process Improvement: Software Engineering Institute's Capability Maturity Model (SEI CMM)
- Object Oriented Analysis and Design
- Requirements Definition
    - Stakeholders, Buy-in
    - Joint Application Development (JAD)
    - Requirements Management
    - User Interfaces, Prototyping
    - Alternatives Analysis
    - Validation and Verification
- Design
    - Architectural Design
    - Application Architectures
    - Decomposition: Functional and Modular
    - Object Oriented Analysis and Design
    - Procurement
    - Design Patterns
- Development and Testing
    - Object Oriented Programming
    - Validation and Verification
    - Types of Testing
        - Unit, integration, white-box, black-box, path, boundary value analysis, equivalence classes, regression
- Implementation and Support
    - Deployment
    - Types of Maintenance: Corrective, Perfective, Preventive, Adaptive
- Wrap-up:
    - Future of Software Engineering
    - Course Recap, Q/A, and Evaluations

How You Will Learn

- A seasoned software engineer/instructor will present this course in interactive lecture format.
- Along with lecture, we use exercises, puzzles, case studies, and interesting group activities to enrich the instruction and drive home the essential points.
- If you already know something about software engineering, we build on that.
- If your background is less technical, we use examples and analogies to simplify the complex subject matter.
- You will receive a printed Participant Handbook, which will help you remember and retain what you learned in class and apply it on your job. This will also include useful web links to equip you with additional resources for future reference and study.

Revised *February 28, 2007*