ProTech *"Charting the Course ...*

protechtraining.com

... to Your Success!"

Systems Analysis and Design

Course Summary

Description

This course is a comprehensive overview of systems analysis and design concepts and techniques and their application to IT projects. The course compares and contrasts the major systems development life cycles (SDLCs) commonly used in software systems development as well as the dominant development paradigms – the structured versus object oriented approaches.

The presentation of the material focuses on the engineering pragmatics of getting a working system built, deployed and operational without attempting to promote one specific paradigm or technology over another. The benefits and costs of the Agile approach versus the waterfall or an iterative SDLC, and the choice of the OO approach or the structured approach are presented. The goal is to teach students to make realistic cost-benefit analysis of these technologies in a real world environment.

How the standard engineering development cycle is implemented in each of these SDLCs is traced from requirements to solution specification to design and planning to construction and project management. The issues of operational maintenance, change management and system retirement are also examined in each SDLC.

The OO and structured paradigms are both ways of analyzing, thinking about and designing systems, and each has its own set of modelling methods, documentation styles, requirements representation and design approaches. The underlying ideas of each of these are examined to provide students with both an ability to choose the approach that is most appropriate for new projects, and to understand how to work with modernization of legacy systems. However the bulk of the material presented is based on the object oriented paradigm and Agile processes.

All of the theory presented is reinforced with examples, real world case studies, hands on exercises and projects.

Topics

- The three major SDLCs in systems development: Waterfall, Iterative and Agile.
- Adaptive versus predictive projects
- Object oriented concepts and design approaches
- Structured analysis and design overview
- Using UML to model OO designs
- Agile process implementations: SCRUM, XP, Kaban, Crystal, DSDM and others
- Project Management for Agile versus traditional projects
- Implementing the standard engineering process in different SDLCs
- Risk analysis models for different SDLC choices
- Project Management for Agile versus traditional projects
- Requirements techniques and documentation
- Domain and business process modelling and business process improvement
- Data requirements, analysis and design
- Design architectures
- Quality assurance practices and techniques for systems design
- Current best practices and standards
- Automation tools for systems development

Systems Analysis and Design

Course Summary (cont'd)

Audience

The course is appropriate for project managers, developers, architects, business and systems analysts, data modelers and anyone else who is involved in the planning, design and building of software systems.

Prerequisites

The course has no prerequisites other than a general understanding of information technology.

Duration

Five days

ProTech protechtraining.com *"Charting the Course ... ... to Your Success!"*

Systems Analysis and Design

Course Outline

I. Introduction to Modern Systems
- A. Mission critical software
- B. Industrial strength software and complexity
- C. Costs of software failures
- D. Root causes of project failures
- E. What we have learned: best practices and anti-patterns

II. The Engineering Process
- A. The basic engineering process defined
- B. The difference between programming and software engineering: the iron triangle
- C. Requirements: problem definition and scope
- D. Analysis: specification of potential solutions
- E. Design: planning the solution with available resources
- F. Construction: software project management
- G. Deployment: field support and change management
- H. Retirement: transitioning to replacement systems

III. Process and SDLC Types
- A. Adaptive versus Predictive SDLCs
- B. Predictive SDLCs: The Waterfall
- C. Adaptive SDLCs: Agile and Spiral
- D. Mixed SDLCs: Iterative (RUP like)
- E. Strengths and weaknesses of each SDLC
- F. Criteria for choosing an SDLC
- G. Best practices and anti-patterns

IV. Process maturity demystified
- A. What process maturity is
- B. The SEI CMM levels explained
- C. What CMM really describes in the real world
- D. The path to process maturity

V. Development Paradigms
- A. Ways of analyzing and designing systems
- B. The structured approach – basic concepts and methods
- C. The object oriented approach – basic concepts and methods
- D. Strengths and weakness of each paradigm
- E. Criteria for choosing a specific paradigm
- F. Using each paradigm with each process type
- G. Cost-benefit analysis for choosing a paradigm and process type

VI. The Structured Paradigm
- A. The kinds of problems it solves
- B. How it was used to build our legacy systems
- C. Basic modelling activities and documentation
- D. Data flow diagrams and flow charts
- E. Entity-relationship models

VII. The Object Oriented Paradigm
- A. The principle of iconicity
- B. Interface and implementation
- C. The principle of recursive design
- D. The object model
- E. The six basic axioms of OO design

VIII. Models, Views and Architectures
- A. Modelling as quantitative activity
- B. Fundamental properties of models
- C. Modelling best practices
- D. Developing orthogonal model sets
- E. Architectures as organization of models
- F. Current architectures in common use

IX. Using UML for OO models
- A. The basic diagrams and their use
- B. How the diagrams fit into the SDLC
- C. Diagrams as views into the architectural design

Systems Analysis and Design

Course Outline (cont'd)

D. Modelling "just enough" with UML
E. UML best practices and anti-patterns

X. Software Systems Project Preliminaries
A. Deciding on an SDLC
B. Deciding on a design approach
C. Establishing project scope
D. Definition of the problem to be solved
E. Establishing a quality baseline
F. Infrastructure choices: tool sets, configuration management and portfolio analysis

XI. Structured Project Design
A. Defining and modelling the data
B. Identifying the business or domain processes
C. Decomposition of the problem
D. Modelling the business logic with flow charts
E. Modelling the data flow with DFDs
F. Modelling "data in motion" with relational models

XII. Agile Project Methodologies
A. SCRUM and its variants
B. Extreme Programming
C. Crystal
D. Dynamic Systems Development Method (DSDM)
E. Other Agile approaches
F. The Agile Manifesto and Software Craftsmanship

XIII. OO Requirements I
A. Functional versus Nonfunctional requirements
B. Requirements engineering models for nonfunctional requirements
C. Functional requirements as user stories and use cases
D. Discovering and documenting user requirements
E. Problems and issues in requirements elicitation
F. Requirements elicitation methods

XIV. OO Requirements II
A. Domain Driven Design – creating a domain model
B. Modelling the domain objects and their structure
C. Modelling the domain processes
D. Iterative model development and knowledge crunching
E. Collaborative process reengineering and redesign during the domain modelling process

XV. OO Requirements III
A. Creating acceptance tests for functional requirements
B. Requirements robustness analysis
C. Stakeholder analysis and reviews
D. Collaborative methods (e.g. Acceptance Test Driven Development)
E. Developing data requirements from the domain model
F. Unified statement of requirements 15. OO Analysis I
G. Developing an Agile specification
H. Specification reviews and IEEE standards (complete, correct, etc.)
I. Acceptance tests and UML diagrams as specification documentation
J. Early Usability mocks

XVI. OO Analysis II
A. Developing an functional architecture
B. Describing the functional architecture with UML class, communication and other diagrams
C. Robustness analysis for the architecture
D. Modularization and subsystem definition

XVII. OO Design I
A. Choosing a software architecture (i.e. Programming environment, frameworks, micro-services or web services etc.)

Systems Analysis and Design

Course Outline (cont'd)

B. Developing a physical architecture
C. Documenting high level design with UML (deployment diagram etc.).
D. Defining interfaces for subsystems

XVIII. OO Design II
A. Developing design classes
B. Modelling design class interfaces (SOLID principles)
C. Modelling the design classes and their interactions with class and sequence diagrams
D. Concurrency and designing for nonfunctional requirements

XIX. OO Design III
A. Understanding Design Patterns
B. Refactoring to Design Patterns

XX. OO Construction I
A. Project Management under different Agile methodologies
B. Test Driven Development
C. Clean code and software reviews
D. Continuous integration
E. Code base maintenance best practices

XXI. OO Construction II Automated Tools
A. Build environments
B. Continuous integration and testing
C. Configuration management tools
D. Code analyzers

XXII. Data Design
A. The logical data model – defined data structure
B. Relational models – data in use
C. Dimensional models – data under analysis
D. Data management and security
E. Data archiving and retention

XXIII. Production Support
A. Release management
B. Change management best practices
C. Regression testing
D. Beta and acceptance testing

XXIV. 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