Review Session (L9)

Session 1

- Software – its characteristics
- Software Engineering Definitions
- Layered view of Software Engineering
- Framework & Umbrella activities
- Software Myths
- Process Patterns

Summary

- Software is a logical entity
- Software is engineered; not manufactured
- Software deteriorates in different ways than hardware
- Software Engineering could be looked upon as layers of elements
- Software Engineering includes framework activities and umbrella activities
- Patterns help gain from established solutions
- There are many myths associated with software engineering
- Software Engineering activities can be mapped to Polya's scientific approach to solve a problem
- David Hooker offers seven valuable software engineering principles

Session 2

- Process Model Taxonomy
- How process models differ?
- Prescriptive Process Models
- Linear Process Model
- Incremental Process Models
- Evolutionary Process Models
- Specialized Process Models
- Unified Process Model
- Personal and Team Software Process

Summary of Process Models

- Process models can be broadly categorized into Prescriptive process models (focusing on planned approach) and Agile process models (focusing on response to changing requirements)
- Plan-based models can be linear, incremental, or evolutionary
- There are specialized process models to address special situations.
- The Unified Model is developed by pioneers in Object-oriented systems development. It has taken best practices from other models and is complemented by the UML tools and work products.

Session 3

- Manifesto of Agile Development
- Agility Principles
- Agility Human Factors
- Agile Process Models
  - XP, ASD, DSDM, Scrum, Crystal, FDD
- Agile Concepts
  - Timeboxing, Spike solutions, Pair programming, Scrum Meetings, Refactoring, Agile Modeling
- Lean software development, Agile Unified Process, Agile processes with Offshore Development, Large Development & Maintenance

Summary of Agile Development

The agile process models have come out of frustration with plan-driven software models. The agile alliance challenged established order in the software industry then.

Agile philosophy stressed four key issues
Self-organizing teams that have control over work they perform
Communication and collaboration among stakeholders
Respectful and responsible attitude towards change
Emphasis on rapid delivery of software

While agile alliance have given many new ideas, they could not come up with a single Solution or approach to the industry problems.

Agile proponents caused change of thinking among adherents of prescriptive process models.

Extreme Programming introduced concepts of Pair programming, Test-driven programming, Spike Solutions. Scrum introduced idea of brief and purposeful meetings.

Agile process models have brought in microplanning-oriented timeboxing in place of macroplanning.

Session 4

- The Essence of Problem Solving
- Software Engineering Principles that guide Process, Practice, and Solution
- Communication Principles
- Planning Principles
- Modeling Principles
  - Analysis, Design
- Construction Principles
  - Coding, Testing
- Deployment Principles

Software Engineering Practice Summary

The Software Engineering practice includes

- concepts,
- principles,
- methods,
- tools

used by software engineers to develop software.

Software engineers must be concerned both with

- technical details of doing things and
- things needed to build high-quality computer software

Software process offers roadmap to build quality products

Software Engineering practice provides the detail needed to travel the road.

Session 5 – Quality Concepts

- Quality has several views from stakeholders
- Quality has several dimensions.
- The cost of quality includes
  - Prevention costs, Appraisal costs, Internal failure costs, External failure costs
- Quality dominates security, risk, liability considerations
- Quality is achieved through
  - Software engineering methods, Project management techniques, Quality control, Quality Assurance

Summary – Software Quality (L5)

- Software quality is the concern of every software process stakeholder.
- If a software team stresses quality in all software engineering activities, it reduces the amount of rework that must be done. This results in lower costs and improved time-to-market.
- To achieve high quality software, four elements must be present:
  - proven software engineering process and practice,
  - solid project management,
  - comprehensive quality control, and
  - the presence of a quality assurance infrastructure.
- Quality software meets its customer's needs, performs accurately and reliably, and provides value to all who produce or use it.

Session 5 – Project Management

- Software Project Management revolves around 4 P's :
  - People
  - Product
  - Process
  - Project
- Successful team leader qualities include Motivation, Organization, Innovation, and problem-solving skills
- Agile processes require self-organizing teams
- Quantitative management requires ability to decompose product and process
- W^5 HH principles offer necessary answers for a sound project plan.

Summary of Software Project Management (L5)

- Project management involves the planning, monitoring, and control of people, process, and events that occur during software development.
- Everyone manages, but the scope of each person's management activities varies according his or her role in the project.
- Software needs to be managed because it is a complex undertaking with a long duration time.
- Managers must focus on the fours P's to be successful (people, product, process, and project).
- A project plan is a document that defines the four P's in such a way as to ensure a cost effective, high quality software product.
- The way to be sure that a project plan worked correctly is by observing that a high quality product was delivered on time and under budget.

Session 6

- Requirement Engineering tasks include
  - Inception
  - Elicitation
  - Elaboration
  - Negotiation
  - Specification
  - Validation
  - Management
- Analysis Model includes
  - Scenarios, Classes, Data Flows, Behaviour
- Web Application Modeling includes
  - Content, Interaction, Functionality, Navigation, Configuration

Requirements Engineering Summary (L6)

- Requirements engineering helps software engineers better understand the problems they are trying to solve.
- Building an elegant computer solution that ignores the customer's needs helps no one.
- The requirements engineering process begins with inception, moves on to elicitation, negotiation, problem specification, and ends with review or validation of the specification.
- The intent of requirements engineering is to produce a written understanding of the customer's problem.
- Several different work products might be used to communicate this understanding (user scenarios, function and feature lists, analysis models, or specifications).
- Web application requirements focus on content, interactions, functionality, navigation, and configuration.

Session 7

- Analysis Modeling approaches can be
  - Structured or Object-oriented
- Analysis Model includes
  - Scenario, Class, Data Flow, Behavior
- Scenarios are represented by Use Cases
  - Alternative actions represented by Activity or Swimlane diagrams
- Class based modeling needs to identify
  - Objects, Operations, Relations, Collaborations
- Data Flow Model includes
  - Hierarchical data flow diagrams and process narratives
- Behavioral Model represents system using
  - State, Transition, Event, and Action

Analysis Model Summary (L7)

- The requirements model is the first technical representation of a system.
- Requirements modeling process uses a combination of text and diagrams to represent software requirements (data, function, and behavior) in an understandable way.
- Software engineers build requirements models using requirements elicited from customers. Building analysis models helps to make it easier to uncover requirement inconsistencies and omissions.
- Developer insights into software requirements grows in direct proportion to the number of different representations used in modeling
- Requirements modeling work products must be reviewed for correctness, completeness, consistency, and relevancy to stakeholder needs.

Session 8

- Design Model consists of
  - Data/Class Design
  - Architectural Design
  - Interface Design
  - Component Design
- The design model components are derived from analysis model
- Fundamental design concepts include
  - Abstraction—data, procedure, control
  - Architecture—the overall structure of the software
  - Patterns—”conveys the essence” of a proven design solution
  - Separation of concerns—any complex problem can be more easily handled if it is subdivided into pieces
  - Modularity—compartmentalization of data and function
  - Information Hiding—controlled interfaces
  - Functional independence—single-minded function and low coupling
  - Refinement—elaboration of detail for all abstractions
  - Aspects—a mechanism for understanding how global requirements affect design
  - Refactoring—a reorganization technique that simplifies the design

Design Model Summary(L8)

- A software design creates meaningful engineering representation (or model) of some software product that is to be built.
- Designers must strive to acquire a repertoire of alternative design information and learn to choose the elements that best match the analysis model.
- A design model can be traced to the customer's requirements and can be assessed for quality against predefined criteria. During the design process the software requirements model (data, function, behavior) is transformed into design models that describe the details of the data structures, system architecture, interfaces, and components necessary to implement the system.
- Each design product is reviewed for quality
  - —identify and correct errors, inconsistencies, or omissions,
  - —whether better alternatives exist, and
  - —whether the design model can be implemented within the project constraints

Thank You