Software Engineering Tutorial Class

Chapter 1

Basic objectives

On completion of the course successfully, students will be able to:

- ❑ Understand the basic principles of Software Engineering
- ❑ Differentiate between structured system analysis and object oriented approach
- ❑ Identify the object oriented techniques
- ❑ Identify and apply various software process model
- ❑ Understand process assessment model and the metrics used in software process

Two Orthogonal view of software

1) Traditional Technique – focuses on data and functions.

- ❑ Traditional system development technique, time-tested and easy to understand. Uses a series of phases, called SDLC, to plan, analyze, design, implement, and support an IS. Predictive approach, organized into phases, with deliverables and milestones to measure progress

2) Object oriented methodologies – focuses on objects that combine data and Functionality.

- ❑ Object oriented systems development develop software by building objects that can be easily replaced, modified and reused.
- ❑ An object has attribute (data) and methods (functions).

SDLC

Basic Steps of SDLC

1) Requirement Engineering
2) System Analysis
3) System Design
4) Implementation
5) Testing and
6) Maintenance that leads to the development of an application

Object Oriented Analysis and Design Concept

- ❑ OO analysis and design become popular b/c
- ❑ Represent data and process together, object
- ❑ Objects may represent actual people, things, concept transactions and events…etc.
- ❑ Views the system in terms of objects that combine data and processes that act on the data together
- ❑ Decompose the system according to key abstractions in the problem domain (domain objects)
- ❑ Model the structural and behavioral aspects of objects as they interacts each other

Object Oriented Analysis and Design Concept

- ❑ OO analysis and design become popular b/c
- ❑ Represent data and process together, object
- ❑ Objects may represent actual people, things, concept transactions and events…etc.
- ❑ Views the system in terms of objects that combine data and processes that act on the data together
- ❑ Decompose the system according to key abstractions in the problem domain (domain objects)
- ❑ Model the structural and behavioral aspects of objects as they interacts each other

Principles of Object-Oriented Systems

- The conceptual framework of object–oriented systems is based upon the object model. There are two categories of elements in an object-oriented system:
- **Major Elements :** By major, it is meant that if a model does not have any one of these elements, it ceases to be object oriented. The four major elements are:
  - Abstraction
  - Encapsulation
  - Modularity
  - Hierarchy
- Minor Elements: By minor, it is meant that these elements are useful, but not indispensable part of the object model. The three minor elements are:
  - Typing
  - Concurrency
  - Persistence

Software Process

❑ The software process defines the way in which software development is organized, managed, measured, supported and improved

- A set of activities, methods, practices, and transformations
  - that people use to develop and maintain software and the associated products
  - (e.g., project plans, design documents, code, test cases, and user manuals)

Software Process Models

- ❑ A process model is an abstract representation of a process. It presents a description of a process from some particular perspective
- ❑ No matter which process model is followed, the basic activities are included in all life cycle models

Need for Software Process Model

- ❑ The software development team must have a clear understanding about when and what to do
- ❑ Without using of a particular process model
  - – the development of a software product would not be in a systematic and disciplined manner.
  - – it becomes difficult for software project managers to monitor the progress of the project.

Software Process Models

- A process model for Software Engineering is chosen based on the
- Nature of project and application
- Methods and tools to be used
- Controls and deliverables that are required
- Project requirements
- Project size
- Project complexity
- Cost of delay
- Customer involvement
- Familiarity with technology
- Project resources.

Software Process Models

1) Classical Waterfall Model
2) V-shaped model
3) Prototyping Model
4) RAD Model
5) Incremental Model
6) When to use iterative model:
7) Spiral Model
8) Agile Model

When to Use The Waterfall Model?

- All the requirements are known, clear, and fixed.
- There are no ambiguous requirements.
- The project is short and simple.
- The development environment is stable.
- Resources are adequately trained and available.
- The necessary tools and techniques used are stable, and not dynamic

When to use the V-model?

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.
- High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

When to use Prototype model

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users
- It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system.
- They are excellent for designing good human computer interface systems.

When to use RAD model

- ❑ RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- ❑ It should be used if there's high availability of designers for modelling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- ❑ RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

When to use the Incremental model

- ❑ This model can be used when the requirements of the complete system are clearly defined and understood.
- ❑ Major requirements must be defined; however, some details can evolve with time.
- ❑ There is a need to get a product to the market early.
- ❑ A new technology is being used
- ❑ Resources with needed skill set are not available
- ❑ There are some high risk features and goals.

When to use iterative model:

- ❑ Requirements of the complete system are clearly defined and understood.
- ❑ When the project is big.
- ❑ Major requirements must be defined; however, some details can evolve with time.

When to use Spiral model

- ❑ When costs and risk evaluation is important
- ❑ For medium to high-risk projects
- ❑ Long-term project commitment unwise because of potential changes to economic priorities
- ❑ Users are unsure of their needs
- ❑ Requirements are complex
- ❑ New product line
- ❑ Significant changes are expected (research and exploration)

When to use Agile model

- ❑ When new changes are needed to be implemented.
- ❑ New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- ❑ To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
- ❑ Unlike the waterfall model in agile model very limited planning is required to get started with the project.
- ❑ Agile assumes that the end users' needs are ever changing in a dynamic business and IT world.
- ❑ Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- ❑ Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way.

SEI-CMM and Levels

1. The lowest level is the initial level.

- Success of organizations at this maturity level depends on the skills and individual efforts of developers rather than on properly defined and managed processes.

2.Second level processes of CMM are repeatable.

- The organizations establish project management policies and procedures to carry out a project.
- A quality assurance function controls that the policies and the procedures are being practiced.
- This discipline ensures repeatability of earlier success on similar projects.

3. Third level processes of CMM are said to be the defined level

- It defines project management and software engineering processes, and it is tailored to each project. An organization adopting and tailoring, for instance ISO 12207, would be at this level.

CMM Level

4. The fourth level of CMM is the managed level:

- the process and product quality is measured, predictable and quantifiable.
- By using these measures, the managers can identify the causes of exceptional events and can correct the situation.

5.The fifth level of CMM is the optimizing level

- The process is continuously improved on the basis of quantitative feedback from earlier instantiations of the process.
- Process improvement is obtained by introducing new methods and new technologies, and
- It is planned like the ordinary management activities.

CMM Level

4. The fourth level of CMM is the managed level:

- the process and product quality is measured, predictable and quantifiable.
- By using these measures, the managers can identify the causes of exceptional events and can correct the situation.

5.The fifth level of CMM is the optimizing level

- The process is continuously improved on the basis of quantitative feedback from earlier instantiations of the process.
- Process improvement is obtained by introducing new methods and new technologies, and
- It is planned like the ordinary management activities.

Process Metrics

- ❑ **A metric as " a quantitative measure of the degree to which a system, component, or process possesses a given attribute**
- ❑ Private process metrics (e.g. defect rates by individual or module) are known only to the individual or team concerned.
- ❑ Public process metrics enable organizations to make strategic changes to improve the software process.
- ❑ Metrics should not be used to evaluate the performance of individuals.
- ❑ Statistical software process improvement helps and organization to discover where they are strong and where are week

Project Metrics

- ❑ Software project metrics are used by the software team to adapt project workflow and technical activities.
- ❑ Project metrics are used to avoid development schedule delays, to mitigate potential risks, and to assess product quality on an on-going basis.
- ❑ Every project should measure its inputs (resources), outputs (deliverables), and results (effectiveness of deliverables).

Software Measurement

- ❑ Direct measures of software engineering process include cost and effort. Direct measures of the product include lines of code (LOC), execution speed, memory size, defects per reporting time period.
- ❑ Indirect measures examine the quality of the software product itself (e.g. functionality, complexity, efficiency, reliability, maintainability).

Continued…

Size-Oriented Metrics

- ❑ Derived by normalizing (dividing) any direct measure (e.g. defects or human effort) associated with the product or project by LOC.
- ❑ Size oriented metrics are widely used but their validity and applicability is widely debated.

Function-Oriented Metrics

- ❑ Function points are computed from direct measures of the information domain of a business software application and assessment of its complexity.
- ❑ Once computed function points are used like LOC to normalize measures for software productivity, quality, and other attributes.
- ❑ Feature points and 3D function points provide a means of extending the function point concept to allow its use with real-time and other engineering applications.
- ❑ The relationship of LOC and function points depends on the language used to implement the software.

Continued…

Size-Oriented Metrics

- ❑ Derived by normalizing (dividing) any direct measure (e.g. defects or human effort) associated with the product or project by LOC.
- ❑ Size oriented metrics are widely used but their validity and applicability is widely debated.

Function-Oriented Metrics

- ❑ Function points are computed from direct measures of the information domain of a business software application and assessment of its complexity.
- ❑ Once computed function points are used like LOC to normalize measures for software productivity, quality, and other attributes.
- ❑ Feature points and 3D function points provide a means of extending the function point concept to allow its use with real-time and other engineering applications.
- ❑ The relationship of LOC and function points depends on the language used to implement the software.

Thank you!
Questions?