Module: 01

Introduction

Lecture: 1

Syllabus:

Lecture no	Motivation: Presented chapter is the foundation of the basic software process and therefore the motivation of this course is that many problems can plague a software project. Software planning involves estimating how much time, effort, money, and resources will be required to build a specific software system. Content	Duration (Hr)	Self-Study (Hrs)
1	Introduction to software engineering, Importance of Software, engineering Software Process	1	1
2	Various models for Software Development (Waterfall, Spiral, Agile (Scrum), V-Model, RAD, DevOps),	4	1
3	Capability Maturity Model (CMM).	1	2

Learning Objective:

This module explains that everyone involved in the software process—managers, software engineers, and customers— participate in development of software project

1. Students will be able to understand the Software engineering process paradigms.
2. Students will be able to understand the typical application of the process models: incremental and evolutionary.
3. Students will be able to understand the Agile methodology.
4. Students will be able to understand process and project metrics.

Key Definition:

Software Engineering: Software engineering encompasses a process, management techniques, technical methods, and the use of tools.
Software: Software is a set of items or objects that form a "configuration" that includes programs, documents, and data.

Software Engineering: Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

Agile software engineering: Represents a reasonable compromise between to conventional software engineering for certain classes of software and certain types of software projects

Software process & project metrics: Are quantitative measures that enable software engineers to gain insight into the efficacy of the software process & the projects that are conducted using the process as a framework.

Incremental Models: The incremental model combines elements of the linear sequential model (applied repetitively) with the iterative philosophy of prototyping.
Process and Project Metrics: Software process and project metrics are quantitative measures that enable software engineers to gain insight into the efficiency of the software process and the projects conducted using the process framework.
Software estimation: Software planning involves estimating how much time, effort, money, and resources will be required to build a specific software system.

Theoretical Background:

Overview

- The roadmap to building high quality software products is software process.
- Software processes are adapted to meet the needs of software engineers and managers as they undertake the development of a software product.
- A software process provides a framework for managing activities that can very easily get out of control.
- Different projects require different software processes.
- The software engineer's work products (programs, documentation, data) are produced as consequences of the activities defined by the software process.
- The best indicators of how well a software process has worked are the quality, timeliness, and long-term viability of the resulting software product.

Course Content

Software Engineering

- Software engineering encompasses a process, management techniques, technical methods, and the use of tools.

Generic Software Engineering Phases

- Definition phase - focuses on what (information engineering, software project planning, and requirements analysis).
- Development phase - focuses on how (software design, code generation, software testing).
- Support phase - focuses on change (corrective maintenance, adaptive maintenance, perfective maintenance, preventative maintenance).

1.1 Software Engineering and software paradigm

The term "software engineering" was coined in about 1969 to mean "the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines".
This view opposed the uniqueness and "magic" of programming in an effort to move the development of software from "magic" (which only a select few can do) to "art" (which the talented can do) to "science" (which supposedly anyone can do!). There have been numerous definitions given for software engineering (including that above and below).
Software Engineering is not a discipline; it is an aspiration, as yet unachieved. Many approaches have been proposed including reusable components, formal methods, structured methods and architectural studies. These approaches chiefly emphasize the engineering product; the solution rather than the problem it solves.
Software Development current situation:

- People developing systems were consistently wrong in their estimates of time, effort, and costs
- Reliability and maintainability were difficult to achieve
- Delivered systems frequently did not work
  - 1979 study of a small number of government projects showed that:
  - 2% worked
  - 3% could work after some corrections
  - 45% delivered but never successfully used
  - 20% used but extensively reworked or abandoned
  - 30% paid and undelivered
- Fixing bugs in delivered software produced more bugs
- Increase in size of software systems
  - NASA
  - StarWars Defense Initiative
  - Social Security Administration
  - financial transaction systems
- Changes in the ratio of hardware to software costs
  - early 60's - 80% hardware costs
  - middle 60's - 40-50% software costs
  - today - less than 20% hardware costs
- Increasingly important role of maintenance
  - Fixing errors, modification, adding options
  - Cost is often twice that of developing the software
- Advances in hardware (lower costs)
- Advances in software techniques (e.g., users interaction)
- Increased demands for software
  - Medicine, Manufacturing, Entertainment, Publishing

- Demand for larger and more complex software systems

- Airplanes (crashes), NASA (aborted space shuttle launches),
- "ghost" trains, runaway missiles,
- ATM machines (have you had your card "swallowed"?), life-support systems, car systems, etc.
- US National security and day-to-day operations are highly dependent on computerized systems.

Manufacturing software can be characterized by a series of steps ranging from concept exploration to final retirement; this series of steps is generally referred to as a software lifecycle.
Steps or phases in a software lifecycle fall generally into these categories:

- Requirements (Relative Cost 2%)
- Specification (analysis) (Relative Cost 5%)
- Design (Relative Cost 6%)
- Implementation (Relative Cost 5%)
- Testing (Relative Cost 7%)
- Integration (Relative Cost 8%)
- Maintenance (Relative Cost 67%)
- Retirement

Software engineering employs a variety of methods, tools, and paradigms.
Paradigms refer to particular approaches or philosophies for designing, building and maintaining software. Different paradigms each have their own advantages and disadvantages which make one more appropriate in a given situation than perhaps another (!).
A method (also referred to as a technique) is heavily depended on a selected paradigm and may be seen as a procedure for producing some result. Methods generally involve some formal notation and process(es).
Tools are automated systems implementing a particular method.
Thus, the following phases are heavily affected by selected software paradigms

- Design
- Implementation
- Integration
- Maintenance

The software development cycle involves the activities in the production of a software system. Generally the software development cycle can be divided into the following phases:

- Requirements analysis and specification
- Design
- Preliminary design
- Detailed design
- Implementation
- Component Implementation
- Component Integration

- System Documenting
- Testing
- Unit testing
- Integration testing
- System testing
- Installation and Acceptance Testing
- Maintenance
- Bug Reporting and Fixing
- Change requirements and software upgrading

Let's check the take away from this lecture

1. 1) Software is a product and can be manufactured using the same technologies used for other engineering artifacts.

A) True
B) **False**

2. Software deteriorates rather than wears out because…

A) Software suffers from exposure to hostile environments
B) Defects are more likely to arise after software has been used often
C) Multiple change requests introduce errors in component interactions
D) Software spare parts become harder to order

Exercise

Q.1 Discuss Importance of Software Engineering.

Questions/Problems for practice:

Q.2 What is software engineering?

Learning from this lecture: Learners will be able to understand basics of Software Engineering

Lecture: 2

Various models for Software Development (Waterfall, Spiral, Agile (Scrum), V-Model, RAD, DevOps)

Learning objective:

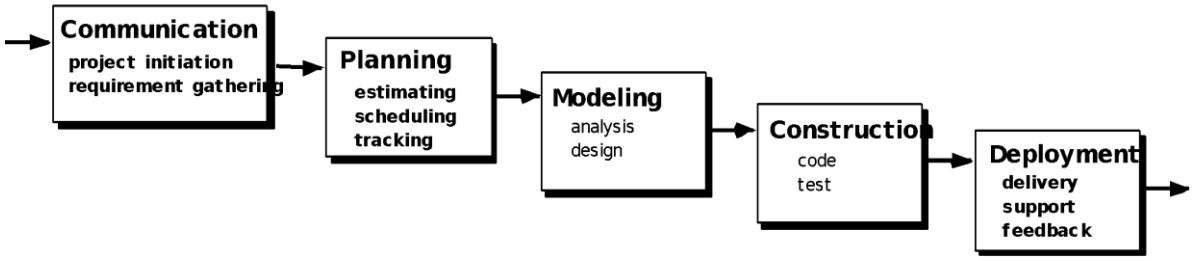
1. Students will be able to understand the typical application of the process models: incremental and evolutionary.
2. Students will be able to understand the Agile methodology.

1.2 Process Models –Incremental and Evolutionary models

Prescriptive Models

Many people (and not a few professors) believe that prescriptive models are "old school"—ponderous, bureaucratic document-producing machines.

1.2.1 The Waterfall Model



Many people dismiss the waterfall as obsolete and it certainly does have problems. But this model can still be used in some situations.
Among the problems that are sometimes encountered when the *waterfall* model is applied are:

- A Real project rarely follows the sequential flow that the model proposes. Change can cause confusion as the project proceeds.
- It is difficult for the customer to state all the requirements explicitly. The waterfall model requires such demand.
- The customer must have patience. A working copy of the program will not be available until late in the project time-span.

Advantages

1. Testing is inherent to every phase of the waterfall model
2. It is an enforced disciplined approach
3. It is documentation driven, that is, documentation is produced at every stage.

Disadvantages

1. Projects rarely flow sequentially as we cannot go back to introduce change.
2. Small changes or errors that arise in the completed software may cause a lot of problem.
3. it happens that the client is not very clear of what he exactly wants from the software. Any changes that he mentions in between may cause a lot of confusion.
4. The greatest disadvantage of the waterfall model is that until the final stage of the development cycle is complete, a working model of the software does not lie in the hands of the client

1.2.2 Incremental Process Models

The process models in this category tend to be among the most widely used (and effective) in the industry.

1.2.2.1The Incremental Model

The *incremental model* combines elements of the *waterfall* model applied in an iterative fashion. The model applies linear sequences in a staggered fashion as calendar time progresses.

Each linear sequence produces deliverable "increments" of the software. (Ex: a Word Processor delivers basic file mgmt., editing, in the first increment; more sophisticated editing, document production capabilities in the 2nd increment; spelling and grammar checking in the 3rd increment.

When an increment model is used, the 1st increment is often a *core product*. The core product is used by the customer.

As a result of use and / or evaluation, a plan is developed for the next increment.

The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality.

The process is repeated following the delivery of each increment, until the complete product is produced.

If the customer demands delivery by a date that is impossible to meet, suggest delivering one or more increments by that date and the rest of the Software later.

1.2.2.2 The RAD Model

Rapid Application Development (*RAD*) is an incremental software process model that emphasizes a short development cycle.

RAD is a "high-speed" adaptation of the waterfall model, in which rapid development is achieved by using a component based construction approach.

If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a fully functional system within a short period of time.

Advantages

1. To converge early toward a design acceptable to the customer and feasible for the developers

2. To limit a project's exposure to the forces of change

3. To save development time, possibly at the expense of economy or product quality

What are the drawbacks of the RAD model?

1. For large, but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
2. If developers and customers are not committed to the rapid-fire activities necessary to complete the system in a much abbreviated time frame, RAD project will fail.
3. If a system cannot properly be modularized, building the components necessary for RAD will be problematic.

1.2.3 Evolutionary Process Models

Software evolves over a period of time; business and product requirements often change as development proceeds, making a straight-line path to an end product unrealistic.

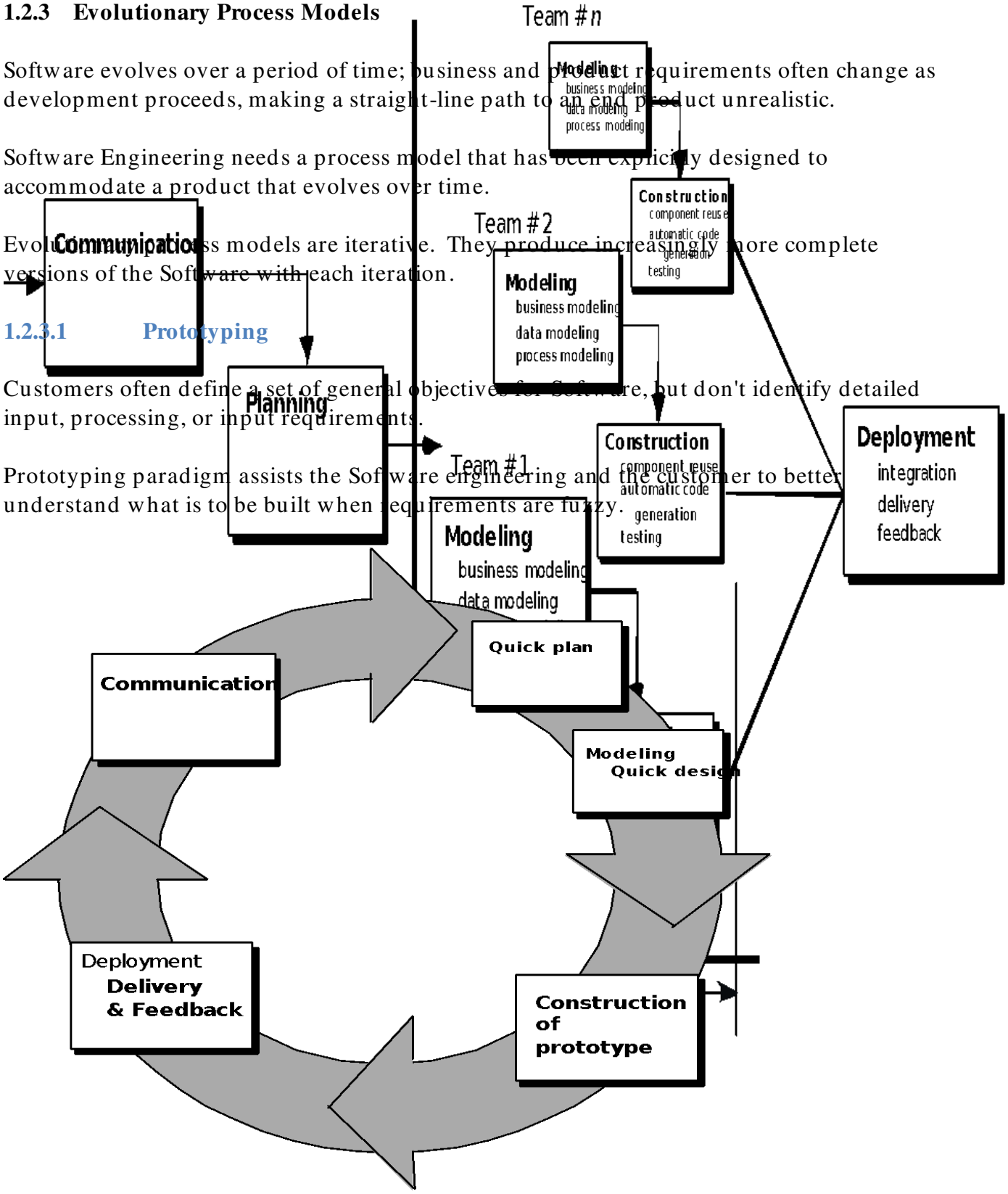
Software Engineering needs a process model that has been explicitly designed to accommodate a product that evolves over time.

Evolutionary process models are iterative. They produce increasingly more complete versions of the Software with each iteration.

1.2.3.1 Prototyping

Customers often define a set of general objectives for Software, but don't identify detailed input, processing, or input requirements.

Prototyping paradigm assists the Software engineering and the customer to better understand what is to be built when requirements are fuzzy.



The prototyping paradigm begins with *communication* where requirements and goals of Software are defined.
Prototyping iteration is *planned* quickly and modeling in the form of quick design occurs.
The *quick design* focuses on a representation of those aspects of the Software that will be visible to the customer "Human interface".
The quick design leads to the *Construction of the Prototype.*
The prototype is *deployed* and then *evaluated* by the customer.
Feedback is used to refine requirements for the Software.
Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while enabling the developer to better understand what needs to be done.
The prototype can serve as the "first system". Both customers and developers like the prototyping paradigm as users get a feel for the actual system, and developers get to build Software immediately. Yet, prototyping can be problematic:
1. The customer sees what appears to be a working version of the Software, unaware that the prototype is held together "with chewing gum. "Quality, long-term maintainability." When informed that the product is a prototype, the customer cries foul and demands that a few fixes be applied to make it a working product. Too often, Software development management relents.
2. The developer makes implementation compromises in order to get a prototype working quickly. An inappropriate O/ S or programming language used simply b/ c it's available and known. After a time, the developer may become comfortable with these choices and forget all the reasons why they were inappropriate.

The key is to define the rules of the game at the beginning. The customer and the developer must both agree that the prototype is built to serve as a mechanism for defining requirements.

1.2.3.2 The Spiral Model

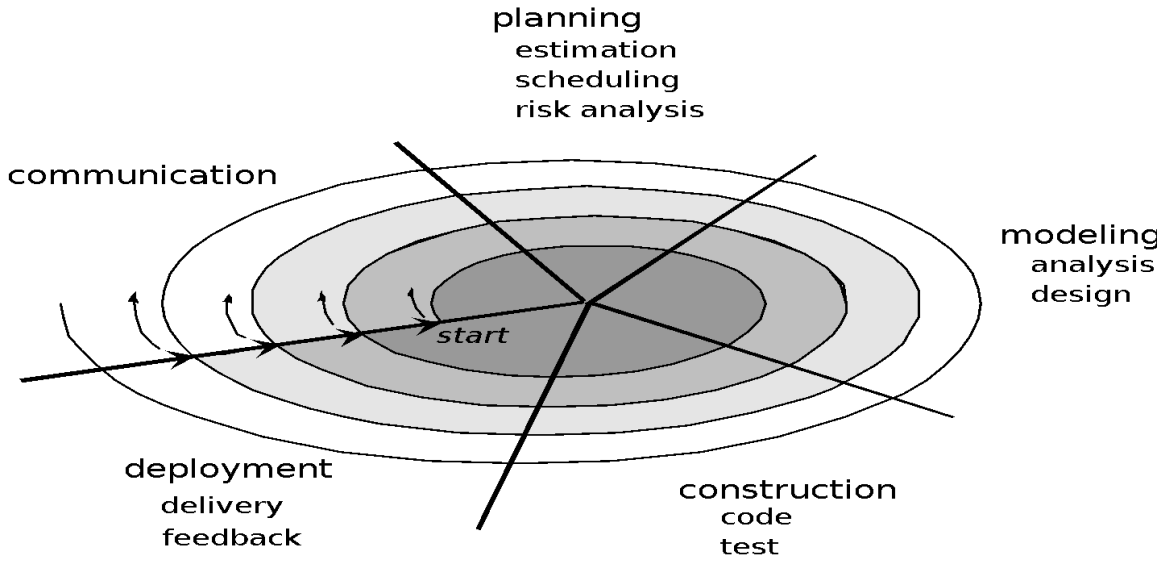
The spiral model is an evolutionary Software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.
It has two distinguishing features:
a. A cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk.
b. A set of *anchor point milestones* for ensuring stakeholder commitment to feasible and mutually satisfactory solutions.

Using the spiral model, Software is developed in a series of evolutionary releases.
During early stages, the release might be a paper model or prototype.
During later iterations, increasingly more complete versions of the engineered system are produced.
A spiral model is divided into a set of framework activities divided by the Software engineering team.
As this evolutionary process begins, the Software team performs activities that are implied by a circuit around the spiral in a clockwise direction, beginning at the center.
Risk is considered as each revolution is made.

Anchor-point milestones – a combination of work products and conditions that are attained along the path of the spiral- are noted for each evolutionary pass.

The first circuit around the spiral might result in the development of a product specification; subsequent passes around the spiral might be used to develop a prototype and then progressively more sophisticated versions of the Software.

Each pass through the planning region results in adjustments to the project plan. Cost and schedule are adjusted based on feedback derived from the customer after delivery.
Unlike other process models that end when Software is delivered, the spiral model can be adapted to apply throughout the life of the Software.



1.3 Typical Application for each model

– Waterfall model can be used in applications where the requirements are fixed and known in advance

– Iterative models can be used where the requirements are known and the product feedback has to be taken from the customer.

1.4 Agile methodology

• Agility

– The ability to both create and respond to change in order to profit in a turbulent business environment

• Companies need to determine the amount of agility they need to be competitive

• Chaordic

– Exhibiting properties of both *cha*os and *ord*er

• The blend of chaos and order inherent in the external environment and in people themselves, argues against the prevailing wisdom about predictability and planning

• Things get done because people adapt, not because they slavishly follow processes

– An agile view is a chaordic view

• "Balanced between chaos and order, perched on the precipice at the edge of chaos."

• Some people are not comfortable in this environment; others thrive on it

Current Problem in Program Management & Software development:

When to Apply Agile Methodologies?

- Problems characterized by change, speed, and turbulence are best solved by agility.
  – Accelerated time schedule combined with significant risk and uncertainty that generate constant change during the project.
- Is your project more like drilling for oil or like managing a production line?
  – Oil exploration projects need agile processes.
  – Production-line projects are often well-served by rigorous methodologies.

Why do we go for Agile process
1) the customer did not give all of the expected requirements and
2) engineers did not always understand the requirements.

Some Agile Methodologies
- Extreme Programming (XP)
- Scrum

Scrum-it is iterative in nature

- **Sprints- basic unit of development in scrum. It produces a working and tested software within given time limits.**

- **Sprints tend to last between on week to month**

1.5 Understanding Process and Project Metrics

A common process framework is established by defining a small number of framework activities that are applicable to all software projects, regardless of their size or complexity. A number of *task sets*—each a collection of software engineering work tasks, project milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement —overlay the process model. Umbrella activities are independent of any one framework activity and occur throughout the process.

1.5.1 Process Metric

Primary metrics are also called as Process metrics. This is the metric the Six Sigma practitioners care about and can influence. Primary metrics are almost the direct output characteristic of a process. It is a measure of a process and not a measure of a high-level business objective. Primary Process metrics are usually Process Defects, Process cycle time and Process consumption.
There are four basic types of metrics for product development:
1. **Process metrics** - short-term metrics that measure the effectiveness of the product development process and can be used to predict program and product performance
- Staffing (hours) vs. plan
- Turnover rate
- Errors per 1,000 lines of code (KSLOC)

2. **Program/project metrics** - medium-term metrics that measure effectiveness in executing the development program/project
- Schedule performance
- Program/project cost performance
- Balanced team scorecard

3. **Product metrics** - medium-term metrics that measure effectiveness in meeting product objectives - technical performance measures
- Weight
- Range
- Mean time between failure (MTBF)
- Unit production costs

4. **Enterprise metrics** - longer term metrics that measure the effectiveness of the enterprise in undertaking IR&D and developing new products
- Breakeven time
- Percent of revenue from products developed in last 4 years
- Proposal win %
- Development cycle time trend (normalized to program complexity)

Let's check the take away from this lecture

1. Software is a product and can be manufactured using the same technologies used for other engineering artifacts.

A) **True**
B) False

2. Software deteriorates rather than wears out because…

A) Software suffers from exposure to hostile environments
B) Defects are more likely to arise after software has been used often
C) Multiple change requests introduce errors in component interactions
D) Software spare parts become harder to order

3. Agility is nothing more than the ability of a project team to respond rapidly to change.

A) True
B) False

4. Which of the items listed below is not one of the software engineering layers?

A) Process
B) Manufacturing
C) Methods
D) Tools

5. Software engineering umbrella activities are only applied during the initial phases of software development projects.

A) True
B) False

Exercise

Q.1 Explain Waterfall Model.

Q.2 Explain Spiral Model.

Learning from this lecture: Learners will be able to understand and compare types of models for Software Development.

Lecture: 3

Various models for Software Development (Waterfall, Spiral, Agile (Scrum), V-Model, RAD, DevOps)

Learning objective:

1. Students will be able to understand process and project metrics.

Capability Maturity Model

To determine an organization's current state of process maturity, the SEI uses an assessment that results in a five point grading scheme. The grading scheme determines compliance with a *capability maturity model (CMM) that defines key activities required at different levels* of process maturity. The levels that are defined in the following manner

Level 1: Initial. The software process is characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

Level 2: Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 3: Defined. The software process for both management and engineering activities is documented, standardized, and integrated into an organization wide software process. All projects use a documented and approved version of the organization's process for developing and supporting software. This level includes all characteristics defined for level 2.

Level 4: Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled using detailed measures. This level includes all characteristics defined for level 3.

Level 5: Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from testing innovative ideas and technologies.This level includes all characteristics defined for level 4.

Let's check the take away from this lecture

1. Process models are described as agile because they

A) eliminate the need for cumbersome documentation

B) emphasize maneuverability and adaptability

C) do not waste development time on planning activities

D) make extensive use of prototype creation

2. Which of these terms are level names in the Capability Maturity Model?

A) Performed

B) Repeated
C) Reused
D) Optimized
E) both a and d

8. Software processes can be constructed out of pre-existing software patterns to best meet the needs of a software project.
A) True
B) False

9. Which of these are standards for assessing software processes?
A) SEI
B) SPICE
C) ISO 19002
D) ISO 9001
E) Both b and d

10. The best software process model is one that has been created by the people who will actually be doing the work.
A) True
B) False

Exercise

Q.1 Explain Capability Maturity Model (CMM)

Learning from this lecture: Learners will be able to understand Capability Maturity Model (CMM).

References:

1. Roger Pressman, Software Engineering: A Practitioners Approach, (6th Edition), McGraw Hill, 1997.

Objective Questions:

(Ans : 1-B,2-C,3-A,4-B,5-B,6-C,7-E,8-E,9-A,10-A)

Short Answer Questions:

1. Explain:
a) Waterfall model
Ans: Refer page.no. 6

2. State the difference between process & project metrics? Describe process metric in detail?
Ans: Refer page.no.18

3. Explain Agile development.
Ans: Refer page .no.17

Long Questions/Answers, Set of Questions for FA/CE/IA/ESE:

1. Compare Waterfall model and Spiral model of Software development. [May 2010] [10 marks]

Ans:

Waterfall

- Testing at end
- Changes cannot be incorporated in between
- Cascade effect
- Document driven
- Well understood products as requirements cannot change
- Linear sequential model or classical life cycle model

Spiral

- Testing is done at every step
- Iterations
- Not completely document driven
- the requirements are not defined in detail

2. Explain the Open Source software life cycle model. [May 2010] [10 marks]

Ans:

Open source is used by the open source initiative to determine whether or not a software license can be considered open source.

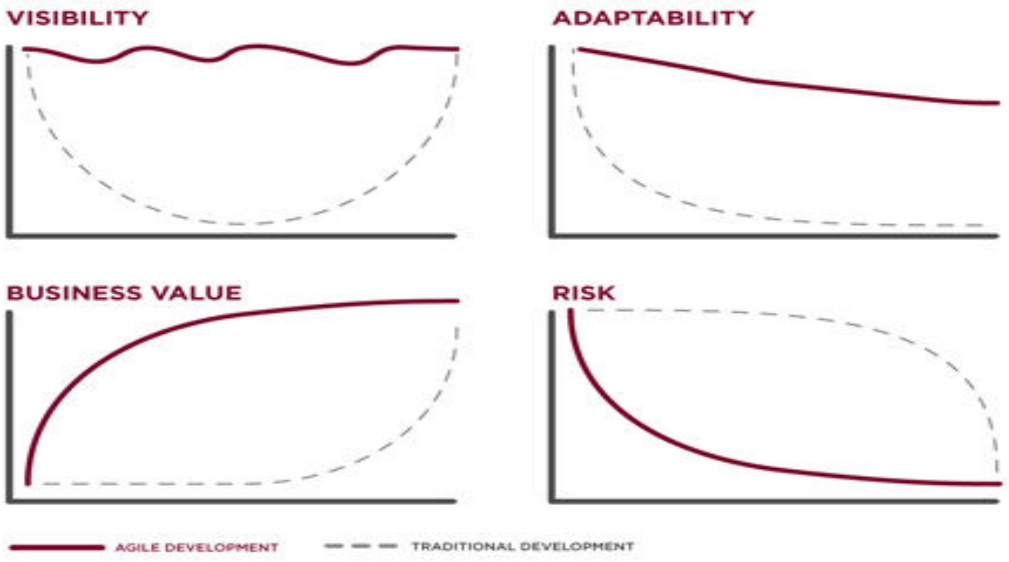
Distribution terms of open source

i. Free redistribution
ii. Source code
iii. Derived works
iv. Integrity of the author's source code
v. No discrimination against fields of endeavor
vi. Distribution of license
vii. License must not be specific to a product
viii. License must not restrict other software
ix. License must be technology-central

3. What are the advantages of agile methodology? [May 2010] [5 marks]

Ans:

Agile development delivers increased value, visibility, and adaptability much earlier in the life cycle, significantly reducing project risk.



4. What is an Agile Process? Explain any one Agile Process model with its advantages and disadvantages. [Nov. 2010] [10 marks]

Ans:

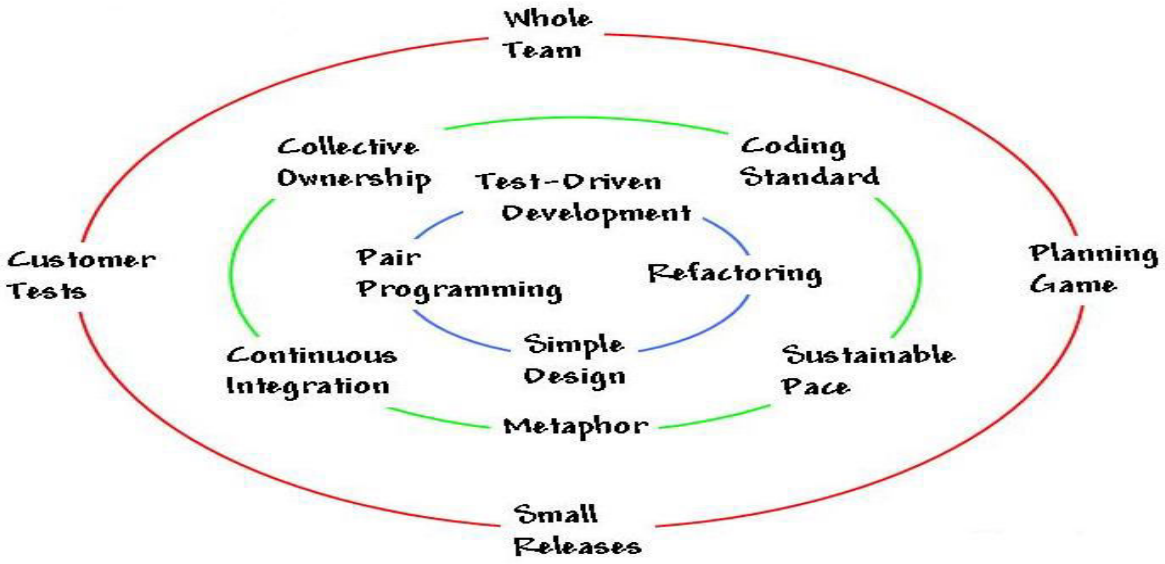
Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams

The different Agile Process models are

- Agile Unified Process (AUP)
- Extreme Programming (XP)
- Scrum

Extreme Programming:

Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.



XP Practices

- Core Practices: Whole Team

In Extreme Programming, every contributor to the project is an integral part of the "Whole Team". The team forms around a business representative called "the Customer", who sits with the team and works with them daily.

- Core Practices: Planning Game, Small Releases, Customer Tests

Extreme Programming teams use a simple form of planning and tracking to decide what should be done next and to predict when the project will be done. Focused on business value, the team produces the software in a series of small fully-integrated releases that pass all the tests the Customer has defined.

- Core Practices: Simple Design, Pair Programming, Test-Driven Development, Design Improvement

Extreme Programmers work together in pairs and as a group, with simple design and obsessively tested code, improving the design continually to keep it always just right for the current needs.

- Core Practices: Continuous Integration, Collective Code Ownership, Coding Standard

The Extreme Programming team keeps the system integrated and running all the time. The programmers write all production code in pairs, and all work together all the time. They code in a consistent style so that everyone can understand and improve all the code as needed.

- Core Practices: Metaphor, Sustainable Pace

The Extreme Programming team shares a common and simple picture of what the system looks like. Everyone works at a pace that can be sustained indefinitely.

5.) Write suitable applications of different software models [5M Jun 2015]
Ans: The development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.
The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.
There are various Software development models or methodologies. They are as follows:
Waterfall model
V model
Incremental model
RAD model

Agile model
Iterative model
Spiral model

Choosing right model for developing of the software product or application is very important. Based on the model the development and testing processes are carried out. Different companies based on the software application or product, they select the type of development model whichever suits to their application. But these days in market the 'Agile Methodology' is the most used model. 'Waterfall Model' is the very old model. In 'Waterfall Model' testing starts only after the development is completed. Because of which there are many defects and failures which are reported at the end. So,the cost of fixing these issues are high. Hence, these days people are preferring 'Agile Model'. In 'Agile Model' after every sprint there is a demo-able feature to the customer. Hence customer can see the features whether they are satisfying their need or not.

'V-model' is also used by many of the companies in their product. 'V-model' is nothing but 'Verification' and 'Validation' model. In 'V-model' the developer's life cycle and tester's life cycle are mapped to each other. In this model testing is done side by side of the development.

Likewise 'Incremental model', 'RAD model', 'Iterative model' and 'Spiral model' are also used based on the requirement of the customer and need of the product.

Self-evaluation

Name of Student		
Class		
Roll No.		
Subject		
Module No.		
S.No		Tick Your choice
1.	Do you understand the Introduction to software engineering?	○ Yes ○ No
2.	Do you understand the concept Software Process, Various models for Software Development?	○ Yes ○ No
3.	Do you understand Capability Maturity Model?	○ Yes ○ No