MCA CS1T03 : Unit 1

Chapter 1 : Introduction to Software Engineering

Ms. Sushmita Chakraborty

Assistant Professor, Deptt. of MCA

Patna Women's College

Bailey Road

What is Software?

- Some Professional defines software as a pattern that is **readable and executable by machine**.
- ***Readable*** means that there is an understandable material expression for the software that is readable by the computer.
- ***Executable*** means that the software is distinguished from data and noise.

What is Software?

*The product that software professionals **build** and then **support** over the long term.*

*Software encompasses: (1) **instructions** (computer programs) that when executed provide desired features, function, and performance; (2) **data structures** that enable the programs to adequately store and manipulate information and (3) **documentation** that describes the operation and use of the programs.*

Why Software is Important?

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment,)
- Software engineering is concerned with theories, methods and tools for professional software development.

Why Software is Important?

- Software does not wear out but deteriorates. Unlike hardware, Software does not get destroyed by usage or aging, but gets outdated.
- Any alteration in environment results in the failure of Software . Therefore alteration in the software is needed.
- Software is not manufactured but developed and customized.

Quality Attributes of Software

Software quality attributes can be measured on a **three- dimensional plot**.

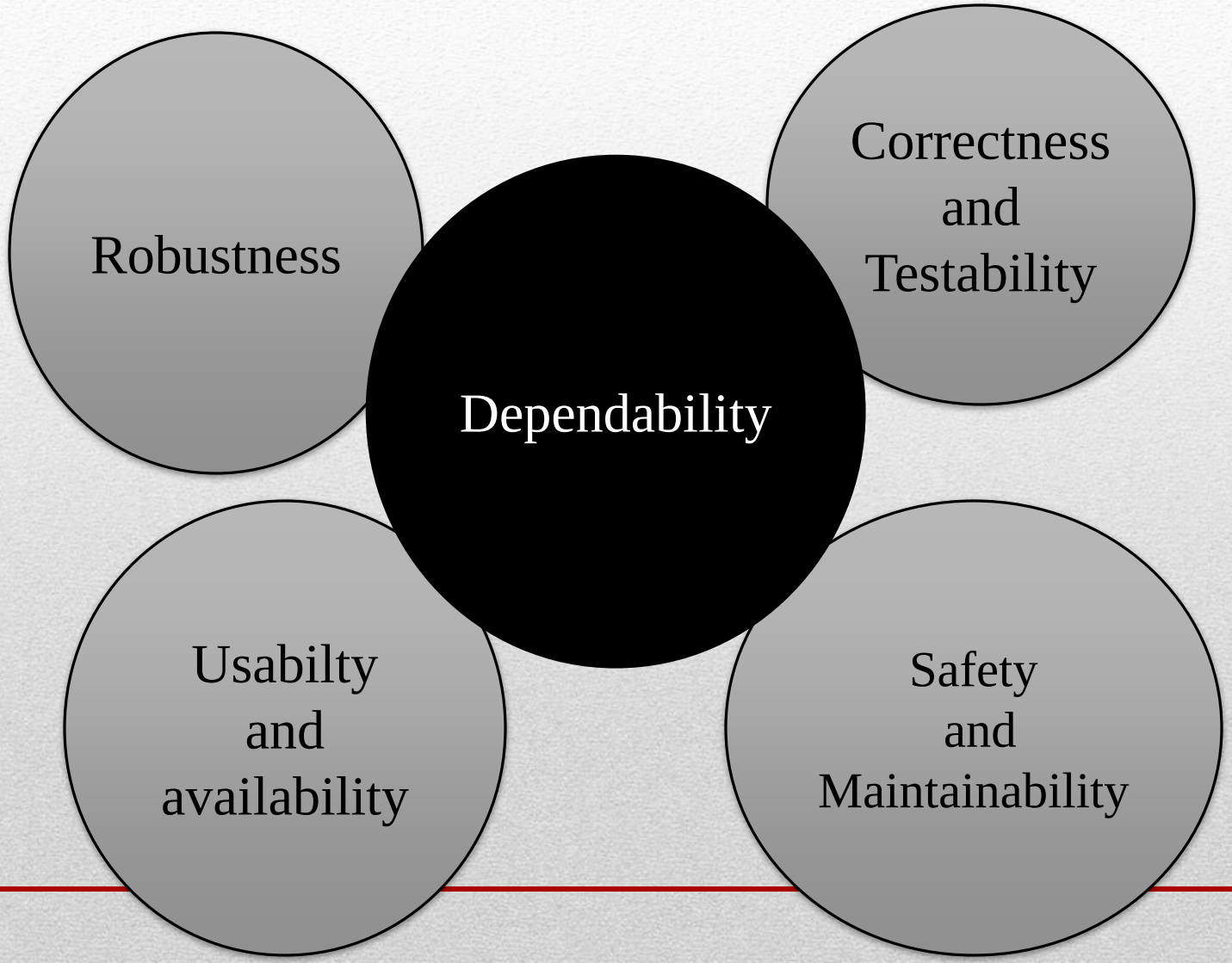
- **Performance :** Performance is a software quality attributes and refers to the timeliness aspect of how software system behave.
- **Dependability :** It states how much one can depend upon the software, to what extent one can trust the software and what it is intended to do when it is needed.

Quality Attributes of Software

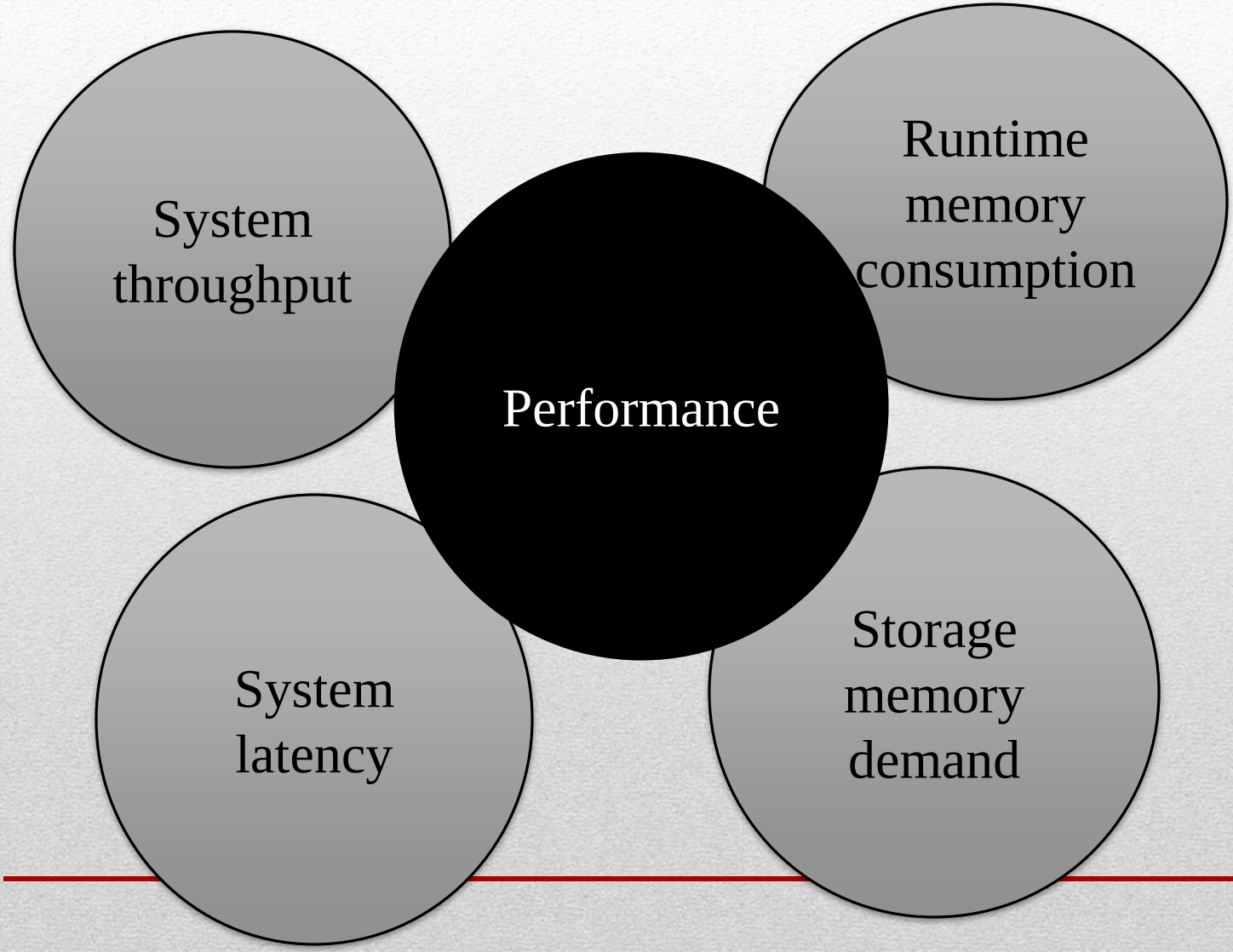
Software quality attributes can be measured on a **three- dimensional plot**.

- **Security :** The ability of the system to protect itself against deliberate or accidental intrusion.

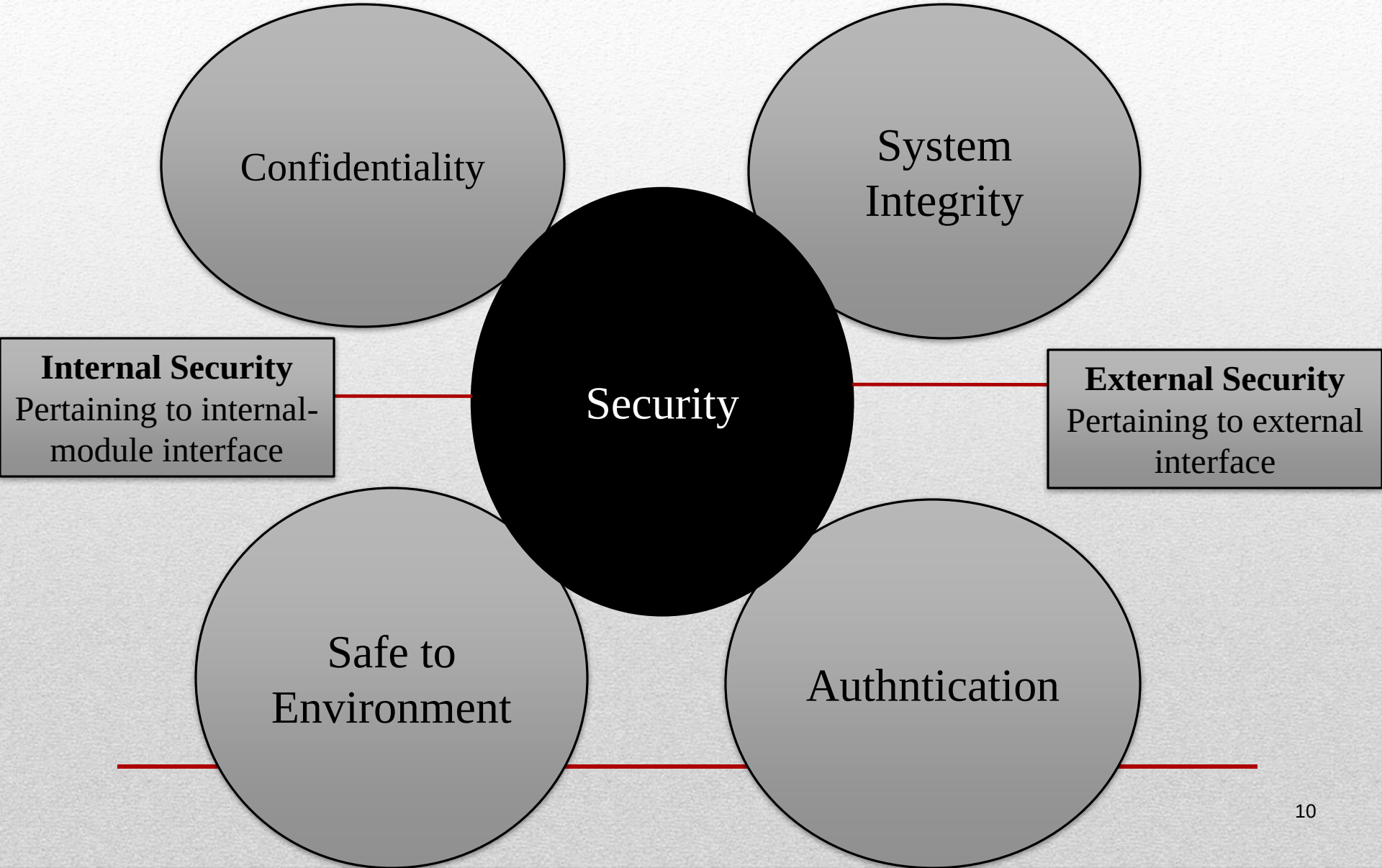
Quality Attributes of Software



Quality Attributes of Software



Quality Attributes of Software



Introduction to Software Engineering

- **The 1968 NATO Software Engineering Conference** is regarded as the origin of Software Engineering – till then regarded as the design of computer programs and software–intensive systems for a performance–efficient, high-quality, and economical framework.

- This conference popularized the term ***software engineering*** that encompasses knowledge, tools, and methods for defining software requirements and performing software design, construction, testing , and maintenance tasks.

Introduction to Software Engineering

- Definition of Engineering

-Application of science, tools and methods to find cost effective solution to problems

Definition of SOFTWARE ENGINEERING

- SE is defined as systematic, disciplined and quantifiable approach for the development, operation and maintenance of software

Software Engineering Definition

The seminal definition:

[Software engineering is] the establishment and use of ***sound engineering principles*** *in order to obtain* ***economically*** *software that is* ***reliable and works efficiently on real machines.***

The IEEE definition:

Software Engineering: (1) The application of a ***systematic, disciplined, quantifiable approach*** *to the* ***development, operation, and maintenance*** *of software; that is, the application of engineering to software. (2) The study of approaches as in (1).*

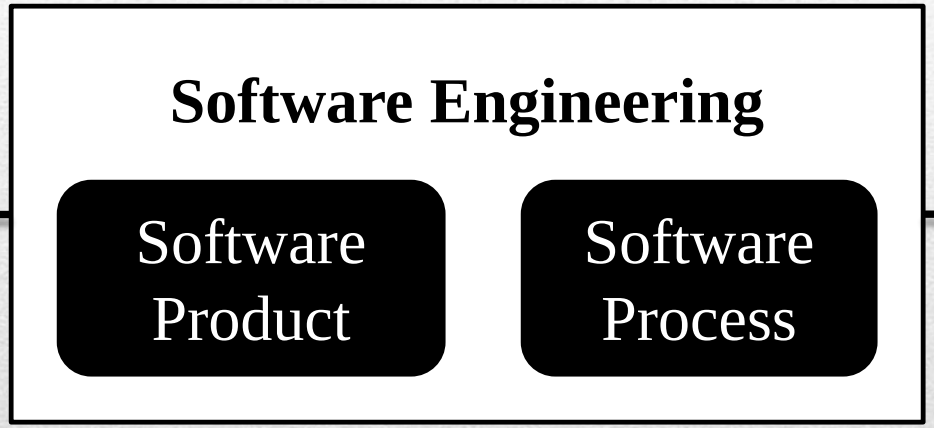
Activities in Software Engineering

Software Analysis

Software writing

Software Operation

Software maintenance



Software Process

Software process research and enhancement

Software process management and control

Software product management and control

Activities in Software Engineering

The following are the common teams who are involved in software development or in software engineering organization.

- User
- Stakeholders
- Sales & Marketing Personnel
- Mangers
- Domain Experts
- Analysts

Activities in Software Engineering

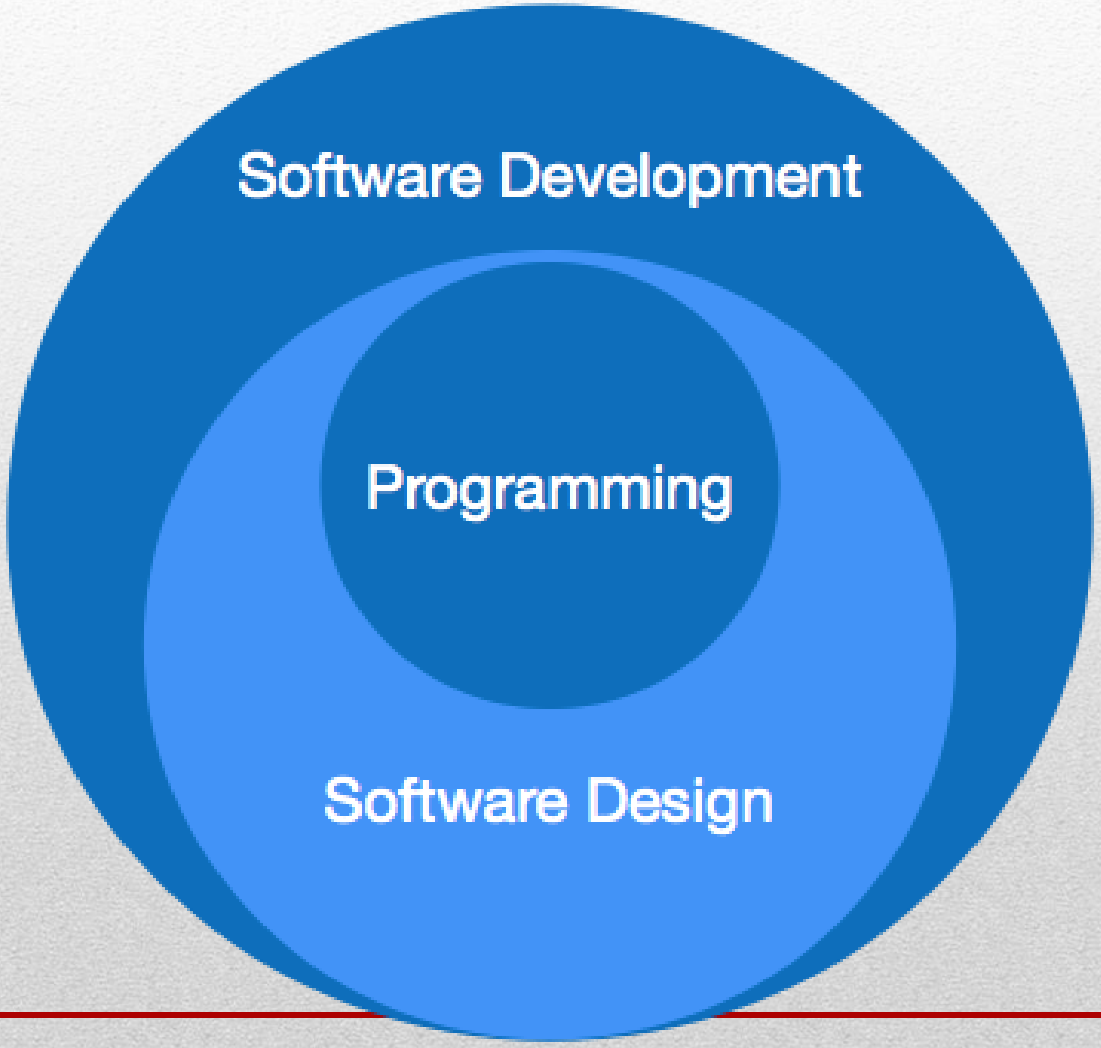
- Software Designer / Architects
- Technology Experts
- Programmers
- Testers and Debuggers
- User Support Staff

Software Paradigm

- Software paradigms refers to the methods and steps, which are taken while designing the software.

- There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another:

Software Paradigm



Software Paradigm

- **Programming paradigm** is a subset of Software design paradigm which is further a subset of Software development paradigm.
- **Software Development Paradigm** is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied.

Software Paradigm

It consists of –

- Requirement gathering
- Software design
- Programming

Software Design Paradigm is a part of Software Development and includes –

- Design
- Maintenance
- Programming

Characteristics of Software

A software product can be judged by what it offers and how well it can be used. Well-engineered and crafted software is expected to have the following characteristics:

- **Operational :** This tells us how well software works in operations. It can be measured on: Budget, Usability, Efficiency, Correctness, Functionality, Dependability, Security, Safety, Transitional, Maintenance

Characteristics of Software

- **Transitional :** This aspect is important when the software is moved from one platform to another : Portability, Interoperability, Reusability, Adaptability
- **Maintenance :** This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment: Modularity, Maintainability, Flexibility,Scalability

Characteristics of Software

So we conclude that Software Engineering is a **branch of computer science**, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products.

Software Applications

- **System software:** such as compilers, editors, file management utilities
- **Application software:** stand-alone programs for specific needs.
- **Engineering/scientific software:** Characterized by "number crunching"algorithms. such as automotive stress analysis, molecular biology, orbital dynamics etc
- **Embedded software** resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car)

Software Applications

- **Product-line software** focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management)
- **WebApps (Web applications) network centric software**. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.
- **AI software** uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition game playing

Need for Software Engineering

- The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

- More and more, individuals and society rely on advanced software systems. We need to be able to produce **reliable and trustworthy systems economically and quickly.**

Need for Software Engineering

- It is usually **cheaper, in the long run**, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the **costs of changing** the software after it has gone into use.

Need for Software Engineering

Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in Programming Paradigm This paradigm is related closely to programming aspect of software development. This includes –

- Coding
- Testing
- Integration which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

Legacy Software

- Legacy software are older programs that are developed decades ago.
- The quality of legacy software is poor because it has inextensible design,convoluted code,poor and nonexistent documentation,test cases and results that are not achieved.

Legacy Software

- As time passes legacy systems evolve due to following reasons:
  - The software must be adapted to meet the needs of new computing environment or technology.
  - The software must be enhanced to implement new business requirements.
  - The software must be extended to make it interoperable with more modern systems or database
  - The software must be rearchitected to make it viable within a network environment.

Software Myths

Erroneous beliefs about software and the process that is used to build it.

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth, ***but …***
- Invariably lead to bad decisions, ***therefore …***
- Insist on reality as you navigate your way through software engineering

Software Myths

Three types of Myths are:

- Management myth
- Customer myth
- Practitioner's myth

Software Myths Examples

- Myth 1: Once we write the program and get it to work, our job is done.
- Reality: the sooner you begin writing code, the longer it will take you to get done. 60% to 80% of all efforts are spent after software is delivered to the customer for the first time.
- Myth 2: Until I get the program running, I have no way of assessing its quality.
- Reality: technical review are a quality filter that can be used to find certain classes of software defects from the inception of a project.

Software Myths Examples

- Myth 3: software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.
- Reality: it is not about creating documents. It is about creating a quality product. Better quality leads to a reduced rework. Reduced work results in faster delivery times.
- Many people recognize the fallacy of the myths. Regrettably, habitual attitudes and methods foster poor management and technical practices, even when reality dictates a better approach.

34

Software Engineering – A Layered Technology

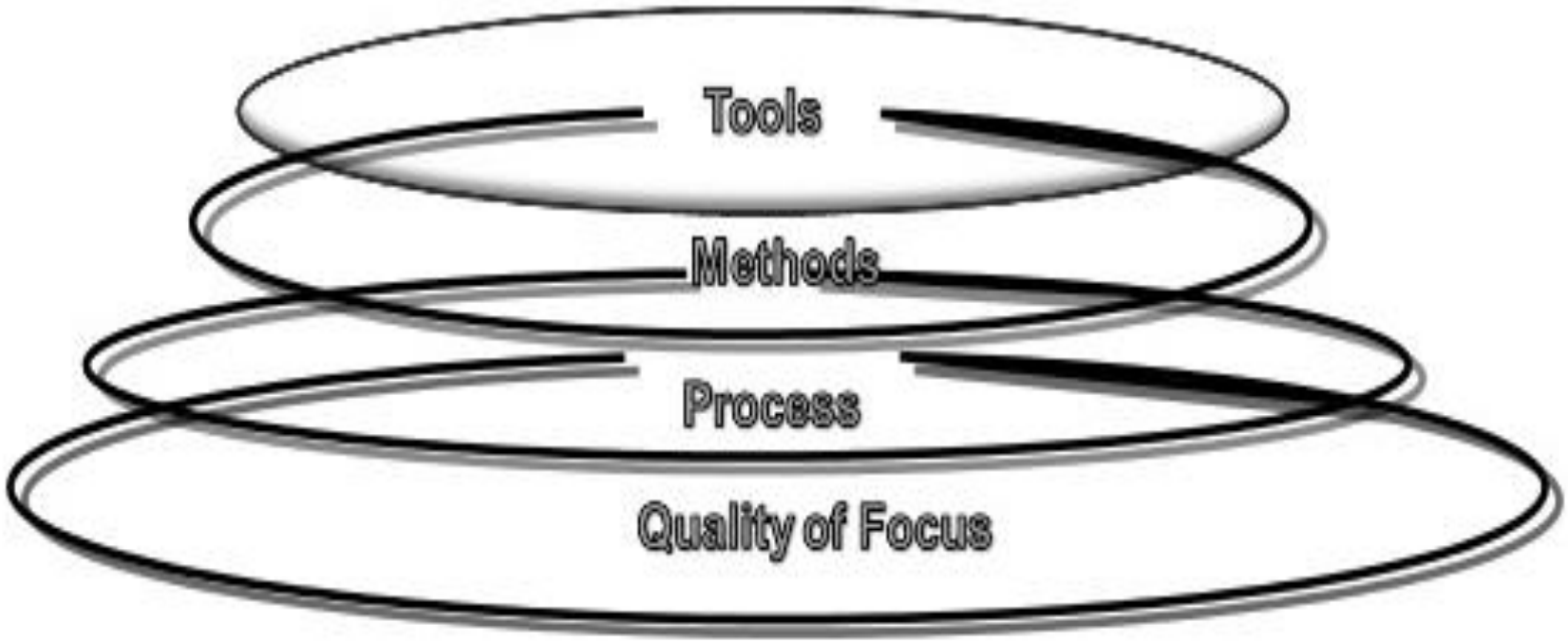


Fig: Software Engineering-A layered technology

Software Engineering – A Layered Technology

- Quality focus
  - Bedrock that supports software Engineering.
- Process
  - Foundation for software Engineering
- Methods
  - Provide technical How-to's for building software
- Tools
  - Provide semi-automatic and automatic support to methods

The Software Product

The objective of software engineering is to produce software products. Computer software is the product that software engineers design and built. Software products are software systems delivered to a customer with the documentation which describes how to install and use the system.

- Software products fall into two broad classes:
- **Generic products**: These are stand alone systems which are produced by a software development organizations/firms and sold on the open market to any customer who is able to buy them.
- **Customized products**: These are systems which are commissioned by a particular customer. The software is developed specially for that customer by some developer.

…. Software Products

- Until the 1980's, the vast majority of software systems which were sold that were customized and specially designed systems which run on large computers. They are expensive because all the development cost had to be met by a single client.

- After the development of PCs, this situation has completely changed. The PC market is totally dominated by software products produces by companies such as Microsoft. These account for the vast majority of software sales. These are usually relatively cheap because their development cost is spread across hundred or thousands of different customers.

Difference between generic and customized software

- The generic software product specifications are produced internally by the marketing department of the product company. They reflect what they think will sell. They are usually flexible and non-prescriptive.

- For customized systems are often the basis for the contract between customer and developer. They are usually defined in detail and changes have to be negotiated and carefully costed.

generic and customized software

- **Generic products:**
  - They are marketed and sold to any customer who wishes to buy them.

    Examples : Graphics Software- Photoshop, Office automation- Ms-Office; Accounting Software – Tally, Design software – Auto CAD , etc
- **Customized products :**
  - Software that is commissioned by a specific customer to meet their own needs.

    Examples : banking System, air traffic control software, railway ticket, reservation systems, Hospital management systems etc.

Software Product Attributes

The attributes of a software product are the characteristics displayed by the product, once it is installed and put in use. They are not the services provided by the product.

Rather, they are concerned with the products dynamic behavior and the use made of the product.

Examples of these attributes are therefore, **efficiency, reliability, maintainability, robustness, portability and so on.**

The relative importance of these characteristics obviously varies from system to system.

. . . product attributes

Product characteristics	Description
Maintainability	It should be possible to evolve software to meet the changing needs of the customer.
Dependability	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycle
Usability	Software should have an appropriate user interface and documentation

. . . product attributes

Optimizing the above attribute is difficult as some are exclusive.

For example, providing a better user interface may reduce system efficiency.

The relationship between cost and improvement in each of the attribute is not linear one.

Small improvement in any of these attributes may be devoted to optimizing particular attribute.

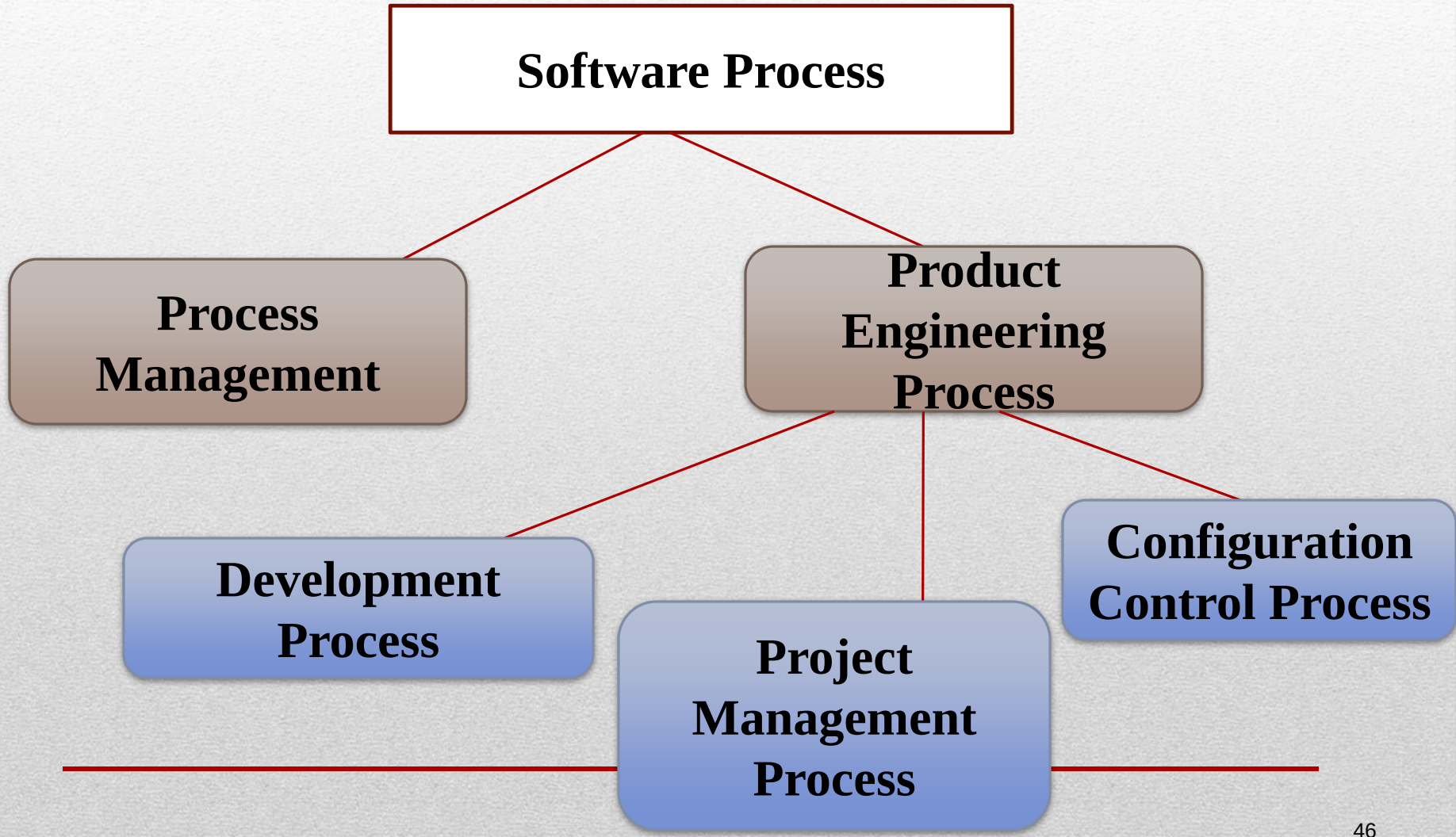
Unit 1 : Chapter : 2

Software Engineering Process and Models

Software Process

- A process is a collection of activities, actions and tasks that are performed when some work product is to be created. It is **not a rigid prescription** for how to build computer software. Rather, it is an adaptable approach that enables the people doing the work to pick and choose the **appropriate set of work actions** and tasks.
- Purpose of process is to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

Components of a Software Process



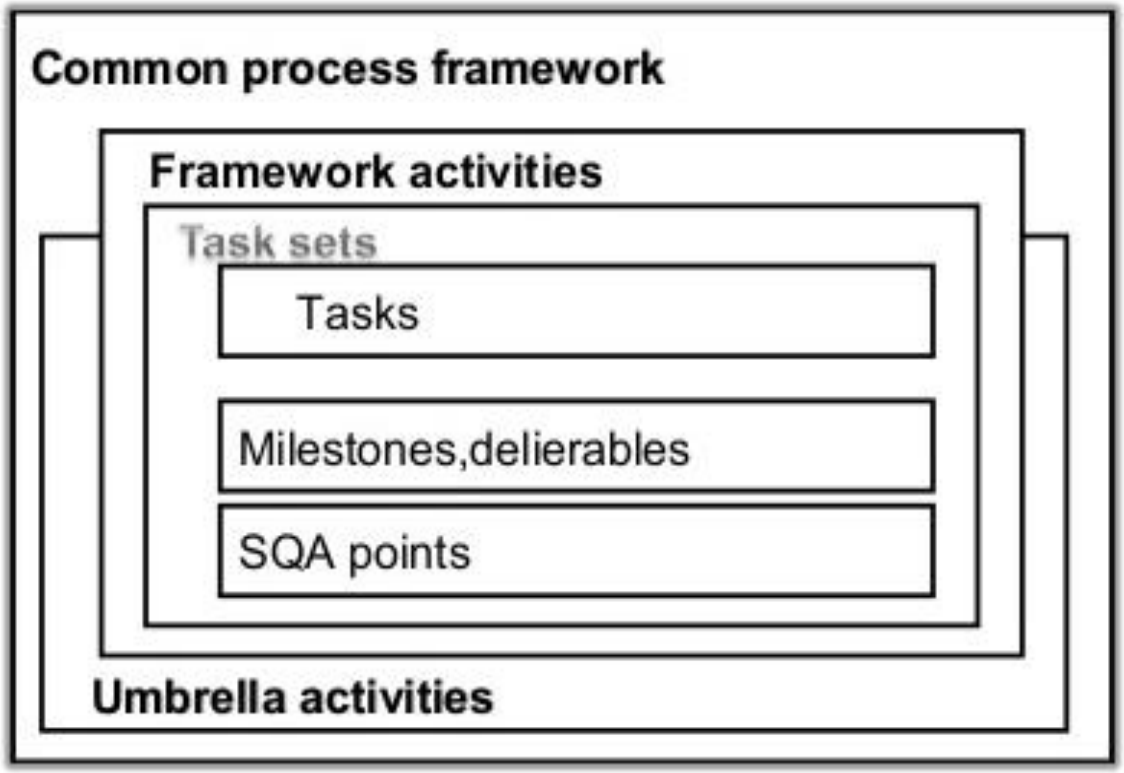
A Generic View of Process

- **Introduction to Software Engineering :** The evolving role of software, Changing Nature of Software, Software myths.
- **A Generic view of process :** Software engineering- A layered technology, a process framework, The Capability Maturity Model Integration (CMMI), Process patterns, process assessment, personal and team process models.

A Process Framework

- Establishes the foundation for a complete software process
- Identifies a number of framework activities applicable to all software projects
- Also include a set of umbrella activities that are applicable across the entire software process.

A Process Framework



Five Activities of a Generic Process framework

- Communication: communicate with customer to understand objectives and gather requirements
- Planning: creates a “map” defines the work by describing the tasks, risks and resources, work products and work schedule.
- Modeling: Create a “sketch”, what it looks like architecturally, how the constituent parts fit together and other characteristics.
- Construction: code generation and the testing.
- Deployment: Delivered to the customer who evaluates the products and provides feedback based on the evaluation.

Five Activities of a Generic Process framework

- These five framework activities can be used to all software development regardless of the application domain, size of the project, complexity of the efforts etc, though the details will be different in each case.

- For many software projects, these framework activities are applied **iteratively** as a project progresses. Each iteration produces a software increment that provides a subset of overall software features and functionality.

Umbrella Activities

Complement the five process framework activities and help team manage and control progress, quality, change, and risk.

- Software project tracking and control: assess progress against the plan and take actions to maintain the schedule.
- Risk management: assesses risks that may affect the outcome and quality.
- Software quality assurance: defines and conduct activities to ensure quality.
- Technical reviews: assesses work products to uncover and remove errors before going to the next activity.

Umbrella Activities

- Measurement: define and collects process, project, and product measures to ensure stakeholder's needs are met.
- Software configuration management: manage the effects of change throughout the software process.
- Reusability management: defines criteria for work product reuse and establishes mechanism to achieve reusable components.
- Work product preparation and production: create work products such as models, documents, logs, forms and lists.

Capability Maturity Model Integration

- Developed by SEI(Software Engineering institute)
- Assess the process model followed by an organization and rate the organization with different levels
- A set of software engineering capabilities should be present as organizations reach different levels of process capability and maturity.
- CMMI process meta model can be represented in different ways

1.A continuous model

2.A staged model

Capability Maturity Model Integration

Continuous model:

-Lets organization select specific improvement that best meet its business objectives and minimize risk

-Levels are called capability levels.

-Describes a process in 2 dimensions

-Each process area is assessed against specific goals and practices and is rated according to the following capability levels.

(CMMI)

Staged model

- This model is used if you have no clue of how to improve the process for quality software.
- It gives a suggestion of what things other organizations have found helpful to work first
- Levels are called maturity levels

Capability Maturity Model Integration (CMMI)

- Six levels of CMMI
  - Level 0:Incomplete
  - Level 1:Performed
  - Level 2:Managed
  - Level 3:Defined
  - Level 4:Quantitatively managed
  - Level 5:Optimized

(CMMI)

INCOMPLETE : Process is adhoc. Objective and goal of process areas are not known

PERFORMED : Goal, Objective, work tasks, work products and other activities of software process are came out.

MANAGED : Activities are monitored, reviewed, evaluated and controlled

DEFINED : Activities are standardized, integrated and documented

QUANTITATIVELY MANAGED : Metrics are indicators are available to measure the process and quality

OPTIMIZED :

- Continuous process improvement based on quantitative feed back from the user
- Use of innovative ideas and techniques, statistical quality control and other methods for process improvement

Adapting a Process Model

The process should be **agile and adaptable** to problems. Process adopted for one project might be significantly different than a process adopted from another project. (to the problem, the project, the team, organizational culture). Among the differences are:

- the overall flow of activities, actions, and tasks and the interdependencies among them
- the degree to which actions and tasks are defined within each framework activity
- the degree to which work products are identified and required
- the manner which quality assurance activities are applied
- the manner in which project tracking and control activities are applied

Adapting a Process Model

- the overall degree of detail and rigor with which the process is described
- the degree to which the customer and other stakeholders are involved with the project
- the level of autonomy given to the software team
- the degree to which team organization and roles are prescribed

Prescriptive and Agile Process Models

- The **prescriptive process** models stress detailed definition, identification, and application of process activates and tasks. Intent is to improve system quality, make projects more manageable, make delivery dates and costs more predictable, and guide teams of software engineers as they perform the work required to build a system.
- Unfortunately, there have been times when these objectives were not achieved. If prescriptive models are applied dogmatically and without adaptation, they can increase the level of bureaucracy.

Prescriptive and Agile Process Models

- **Agile process models** emphasize project "agility" and follow a set of principles that lead to a more informal approach to software process. It emphasizes maneuverability and adaptability. It is particularly useful when Web applications are engineered.


Thank You