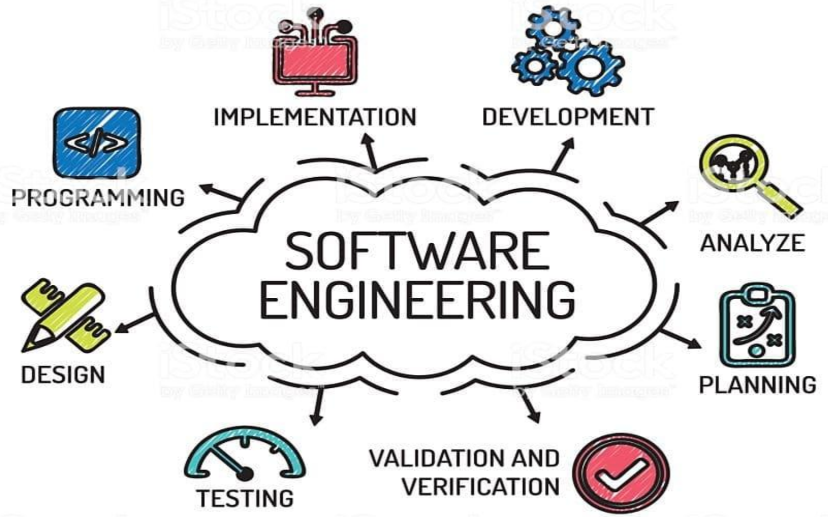
INTRODUCTION
Module 1- Chapt 1

Mrs. Madhu Nagaraj
Assistant Professor
Dept of CSE-Data Science
ATMECE


SOFTWARE ENGINEERING
IMPLEMENTATION
DEVELOPMENT
PROGRAMMING
ANALYZE
DESIGN
PLANNING
TESTING
VALIDATION AND VERIFICATION

ATME College of Engineering

Course Out Comes:

- CO1: Differentiate process models to judge which process model has to be adopted for the given scenarios.
- CO2: Derive both functional and nonfunctional requirements from the case study.
- CO3: Analyze the importance of various software testing methods and agile methodology.
- CO4: Illustrate the role of project planning and quality management in software development.
- CO5: Identify appropriate techniques to enhance software quality.

Module 1

- **Software and Software Engineering:** The nature of Software, The unique nature of WebApps, Software Engineering, The software Process, Software Engineering Practice, Software Myths.
- **Process Models:** A generic process model, Process assessment and improvement, Prescriptive process models: Waterfall model, Incremental process models, Evolutionary process models, Concurrent models, Specialized process models. Unified Process , Personal and Team process models

The Evolving role of software

Dual role of Software

1. **A Product**

- Transforms information - produces, manages, acquires, modifies, displays, or transmits information
- Delivers computing potential of hardware and networks

2. A Vehicle for delivering a product

- Controls other programs (operating system)
- Effects communications (networking software)
- Helps build other software (software tools & environments)

Where can you find software?

Software is Almost Everywhere.

facebook

Remember Me Forgot your password?

Email Password Login

Facebook helps you connect and share with the people in your life.

Sign Up

It's free and anyone can join

Full Name:

Your Email:

New Password:

I am: Select Sex:

Birthday: Month: Day: Year:

Why do I need to provide this?

Sign Up

To create a page for a celebrity, band or business, click here.

Software is defined as

1. Instructions - Programs that when executed provide desired function

2. Data structures -Enable the programs to adequately manipulate information

3. Documents -Describe the operation and use of the programs.

- Software is the collection of executable program codes, associated libraries and documentation.
- Software products may be developed for a particular customer or may be developed for a general market.

Software products may be

- **Generic** - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.
- **Custom** - developed for a single customer according to their specification.

Software Characteristics

- Software is developed or engineered; it is not manufactured in the classical sense.
- Software does not “wear out” but it does deteriorate.
- Software continues to be custom built, as industry is moving toward component based construction.

Changing Nature of Software

The nature of software has changed a lot over the years.

1.System software: It is a collection of programs to provide service to other programs. Infrastructure software come under this category like compilers, operating systems, editors, drivers, etc.

2. Real time software: These software are used to monitor, control and analyze real world events as they occur.

3. Embedded software: This type of software is placed inside the hardware of the system and is used control the various functions of the product.

Eg: Microwave oven controls

4. Business software : This is the largest application area. The software designed to process business applications is called business software.

5. Personal computer software: The software used in personal computers are covered in this category. Examples are word processors, computer graphics, multimedia and animating tools, database management, computer games etc

6. Artificial intelligence software: Artificial Intelligence software makes use of non numerical algorithms to solve complex problems.

Examples: Robotics, Gaming etc

7. Web based software: The software related to web applications come under this category. Examples are CGI, HTML, Java, Perl, DHTML etc.

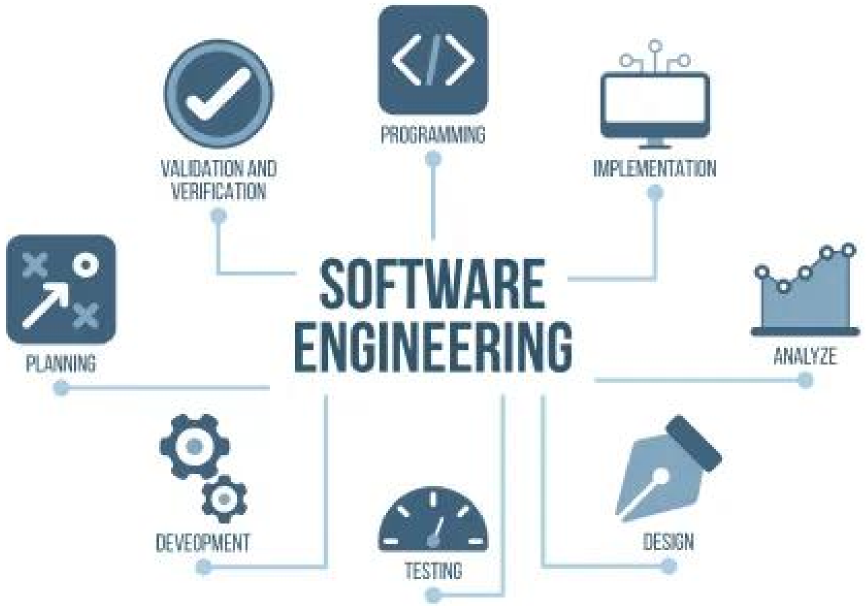
- What is Software Engineering?

➢ Software Engineering is an *engineering discipline* that is concerned with *all aspects of software production.*

➢ Software engineers should adopt a systematic and organized approach to their work.

➢ use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available

A T M E College of Engineering



Essential attributes of good software

- **Maintainability**
- Software should be written in such a way so that it can evolve to meet the changing needs of customers.
- This is a critical attribute because software change is an inevitable requirement of a changing business environment.

- **Dependability and security**
- 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. Malicious users should not be able to access or damage the system.

- **Efficiency**
- Software should not make wasteful use of system resources such as memory and processor cycles.
- Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.

- **Acceptability**
- Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Process Framework

- Framework is a Standard way to build and deploy applications.
- Software Process Framework is a foundation of complete software engineering process.
- Software process framework includes all set of umbrella activities. It also includes number of framework activities that are applicable to all software projects.

Generic Process Framework Activities

- **Communication:**
  - Heavy communication and collaboration with customers, stakeholders, team
  - Encompasses requirements gathering and related activities

- **Planning:**
  - Workflow that is to follow
  - Describe technical task, likely risk, resources will require, work products to be produced and a work schedule.

Modeling:
  - Help developer and customer to understand requirements (Analysis of requirements & Design of software)

Construction:
  - Code generation: either manual or automated or both
  - Testing – to uncover error in the code.

- **Deployment:**
  - Delivery to the customer for evaluation
  - Customer provide feedback

Umbrella Activities

1. **Software project tracking and control**
   - Assessing progress against the project plan.
   - Take adequate action to maintain schedule.

2. **Formal technical reviews**
   - Assessing software work products in an effort to uncover and remove errors before it goes into next action or activity.

3. **Software quality assurance**
   - Define and conducts the activities required to ensure software quality.

4. **Software configuration management**
   - Manages the effects of change.

5. **Document preparation and production**
   - Help to create work products such as models, documents, logs, form and list.

6. **Reusability management**
   - Define criteria for work product reuse
   - Mechanisms to achieve reusable components.

7. **Measurement & Metrics**
   - All measurement related activities like cost, time man power is done.
   - Assist the team in delivering software that meets customer’s needs.

8. **Risk management**
   - Assesses risks that may effect that outcome of project or quality of product (i.e. software)

General Principles of Software Engineering:

The word principle is “an important underlying law or assumption required in a system of thought.” David Hooker [Hoo96] has proposed seven principles that focus on software engineering practice as a whole.

1. The First Principle: The Reason It All Exists

A software system exists for one reason: to provide value to its users. All decisions should be made with this in mind.

2. The Second Principle: Keep It Simple, Stupid!

All design should be as simple as possible, but no simpler.

This facilitates having a more easily understood and easily maintained system.

3. The Third Principle: Maintain the Vision

A clear vision is essential to the success of a software project. Without one, a project almost unfailingly ends up being “of two [or more] minds” about itself.

4. The Fourth Principle: What You Produce, Others Will Consume

always specify, design, and implement knowing someone else will have to understand

what you are doing. The audience for any product of software development is potentially large.

5. The Fifth Principle: Be Open to the Future

Never design yourself into a corner. Always ask “what if,” and prepare for all possible answers by creating systems that solve the general problem, not just the specific one.

6. The Sixth Principle: Plan Ahead for Reuse

Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the systems into which they are incorporated.

7. The Seventh principle: Think!

Placing clear, complete thought before action almost always produces better results.

When you think about something, you are more likely to do it right. You also gain knowledge about how to do it right again.

Software Myths

Software myths are erroneous beliefs about software and the process that is used to build it.

We categorize myths from three different perspectives.

Management myths:

Myth: We already have a book that’s full of standards and procedures for building software. Won’t that provide my people with everything they need to know?

Reality: The book of standards may very well exist, but is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it adaptable? Is it streamlined to improve time-to-delivery while still maintaining a focus on

quality? In many cases, the answer to all of these questions is “no.”

Myth: If we get behind schedule, we can add more programmers and catch up (sometimes called the “Mongolian horde” concept).

Reality: Software development is not a mechanistic process like manufacturing. As new people

are added, people who are working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well coordinated manner.

Myth: If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it out-sources software projects.

Customer myths.

Myth: A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.

Reality: Although a comprehensive and stable statement of requirements is not always possible, an ambiguous "statement of objectives" is a recipe for disaster.

Myth: Software requirements continually change, but change can be easily accommodated because software is flexible.

Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly

Customer myths.

Myth: A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.

Reality: Although a comprehensive and stable statement of requirements is not always possible, an ambiguous "statement of objectives" is a recipe for disaster.

Myth: Software requirements continually change, but change can be easily accommodated because software is flexible.

Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly

Practitioner’s myths:

Myth: Once we write the program and get it to work, our job is done.

Reality: Someone once said that “the sooner you begin ‘writing code,’ the longer it’ll take you to get done.” Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.

Myth: The only deliverable work product for a successful project is the working program.

Reality: A working program is only one part of a software configuration that includes many elements.

Myth: Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.

Reality: Software engineering is not about creating documents. It is about creating a quality product. Better quality leads to reduced rework.