Unit-1

About Software Development

About Me

Prof. Rajesh Pradhan

www.rb.gy/gvea5

@ProfRajeshP

@oyerajubhai

Job Roles in IT Field

- Entry-Level Positions:
- Junior Software Developer
- Junior Quality Assurance Analyst
- Junior Systems Administrator
- Junior Data Analyst
- Junior Network Technician

- Mid-Level Positions:
- Software Developer
- Quality Assurance Analyst
- Systems Administrator
- Data Analyst
- Network Engineer
- Business Analyst
- Database Administrator
- UI/UX Designer
- DevOps Engineer

- Senior-Level Positions:
- Senior Software Developer
- Senior Quality Assurance Analyst
- Senior Systems Administrator
- Senior Data Analyst
- Senior Network Engineer
- Senior Business Analyst
- Senior Database Administrator
- Senior UI/UX Designer
- Senior DevOps Engineer

- Management Roles:
- Project Manager
- Team Lead
- Delivery Manager
- IT Manager
- Product Manager

- Executive Leadership:
- Chief Technology Officer (CTO)
- Chief Information Officer (CIO)
- Chief Operating Officer (COO)
- Chief Financial Officer (CFO)

- Specialized Roles:
- Cybersecurity Specialist
- Machine Learning Engineer
- Cloud Solutions Architect
- Data Scientist
- Blockchain Developer
- ERP Consultant

- Support and Administrative Roles:
- IT Support Specialist
- Help Desk Technician
- Technical Writer
- System Analyst
- Administrative Assistant

Top 10 IT companies in India

Here's a quick look at India's top 10 IT powerhouses:

Name	Market Cap (Cr)	Employee Headcount (Approx)
Tata Consultancy Service	Rs1,244,681.21	6,00,000
Infosys	Rs583,291.29	3,36,294
HCL Technologies	Rs320,008.96	2,25,944
Wipro Limited	Rs216,450.93	2,40,000
LTIMindtree Ltd.	Rs152,318.83	82,000
Tech Mahindra Ltd.	Rs118,821.11	1,52,400
MphasiS Ltd.	Rs43,685.96	29,473
Persistent	Rs38,544.81	22,500
Oracle Fin Serv	Rs34,582.06	8,001
KPIT Tech	Rs31,016.63	10,000

21/08/2023
Source:Forbes

Top 10 IT Companies in World by MarketCap

Company	Sector	Market Cap (in USD)
#1 Apple	Technology	$2.744 trillion
#2 Microsoft	Technology	$$2.353 trillion
#3 Saudi Aramco	Oil & Gas	$2.224 trillion
#4 Alphabet (Google)	Technology	$1.624 trillion
#5 Amazon	E-commerce	$1.336 trillion
#6 Nvidia	Technology	$1.069 trillion
#7 Berkshire Hathaway	Diversified Investments	$770.43 billion
#8 Meta Platforms	Social Media	$725.89 billion
#9 Tesla	Automotive	$682.99 billion
#10 Eli Lilly	Pharmaceuticals	$518.71 billion

21/08/2023
Source:Forbes

Computer = Hardware + Software

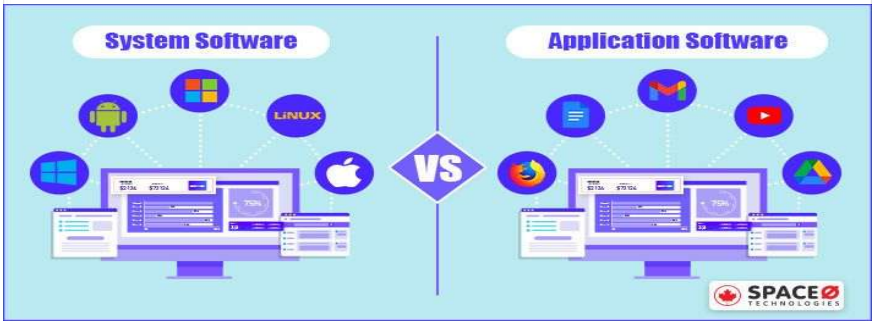
Hardware Vs Software

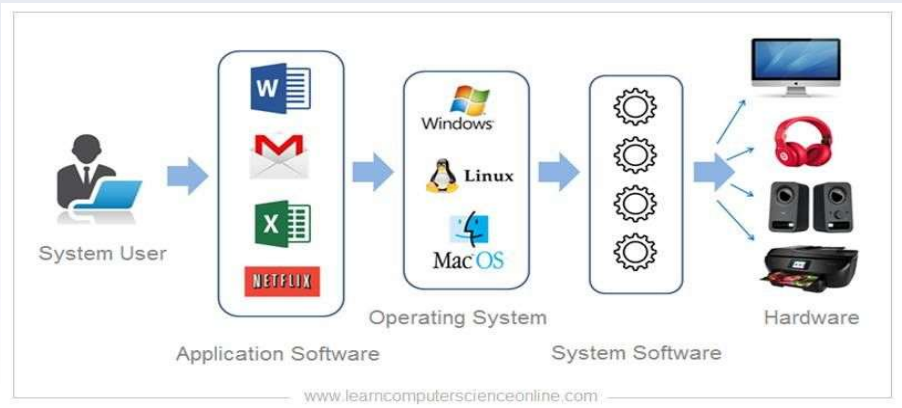
Category	Hardware	Software
Nature	Physical components and devices.	Instructions and programs.
Examples	Processors, memory, storage, I/O devices etc.	Operating systems, applications, scripts, code etc.
Functionality	Provides basic computing capabilities.	Leverages hardware capabilities to perform tasks.
Upgrades	Requires physical replacement or addition of new components.	Can be upgraded via installation of new versions/patches.
Cost	Generally has high initial cost.	Has low/no initial cost but requires hardware to run.
Failure	Components can fail physically over time.	Bugs/errors can cause failures but no physical damage.
Portability	Not portable, stationary within a system.	Portable across systems via installation.
Development	Requires electrical/mechanical engineering skills.	Requires programming/software engineering skills.
Ownership	Owned as tangible property.	Licensed/copyrighted intellectual property.
Examples of upgrades	Replacing CPU, adding more RAM.	Installing new OS version, updating apps.

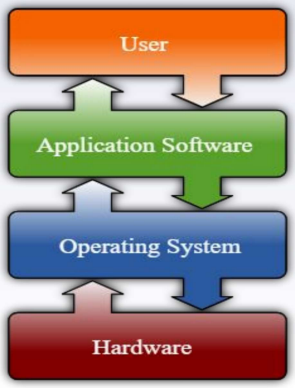
Software

- **(IEEE Definition of software)**
- **Software is a "Collection of computer programs, procedures, rules, associated documents and concerned data with the operation of data processing system."**

- "IEEE is Institute of Electrical and Electronics Engineers"







➔ **System software:** It is responsible for controlling and integrating the hardware components of a system. Hence, the software and users can work with them. It also controls the peripherals of computer like monitors, printers, storage devices etc.

Example: Operating system

➔ **Application software:** It is used to accomplish some specific task. It should be collection of small programs. path to It is a program or group of programs generally designed for end users.Example: Microsoft word, Excel, Railway reservation system etc.

1.1 Software Characteristics (Qualities of a Good Software)

(CCPU MMRF DRIVER)

- **Cost:** Software should be cost effective as per its usage.
- **Correctness:** Software should be correct as per its requirements.
- **Portability:** Software should have the capability to be adapted for different environments.
- **Understandability:** Software should be easy to understand, even to novice users. It should be efficient to use.
- **Maintainability:** Software should be easily maintainable and modifiable in future.
- **Modularity:** Software should have modular approach so it can be handled easily for testing.
- **Reliability:** It should have the capability to provide failure-free service.
- **Functionality:** Software should be functionally capable to meet user's requirements.

- **Documentation:** Software should be properly documented so that we can re-refer it in future.
- **Reusability:** It should be reusable, or its code or logic should be reusable in future.
- **Interoperability:** Software should be able to communicate with various devices using standard bus structure and protocol.
- **Verifiability:** Software should be verifiable with its properties and functionalities with its planning and analysis done in previous phase.

➔ For example, a banking system must be secure while a telephone switching system must be reliable.

1.1 Software Characteristics (Qualities of a Good Software)

(CCPU MMRF DRIVER)

1.1.1 Other/Special Software Characteristics

i. Software doesn't wear out:

a)Hardware Failure Curve

- ★ The relationship between time and failure called "bath-tub curve".
- ★ It indicates that hardware has relatively high failure rates early in its life, defects are corrected and the failure rate drops to a steady-state level for some period of time.
- ★ As time passes, however, the failure rate rises again as hardware components suffer from the affects of dust, vibration, abuse, temperature extremes, and many other environmental factors.
- ★ So simply, we can say hardware begins to wear out.

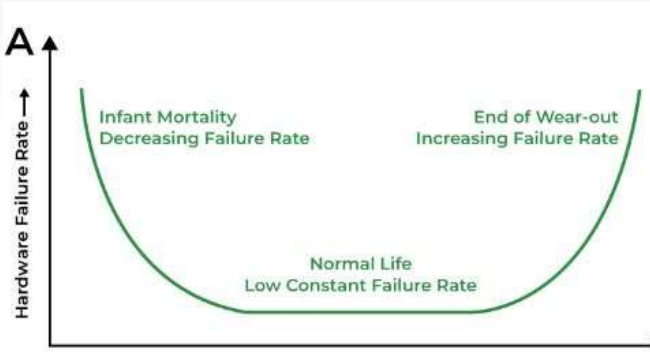


Figure 1.1 H/W failure curve

b) Software Failure Curve : Ideal Curve

- Software is not highly affected by environmental effects.
- The "idealized curve" shows software failure.
- In the early stage, due to lot many errors, software could have high failure.
- But it becomes reliable as time passes instead of wearing out.
- Once software is made it has a longer life span that we can see in idealized curve.

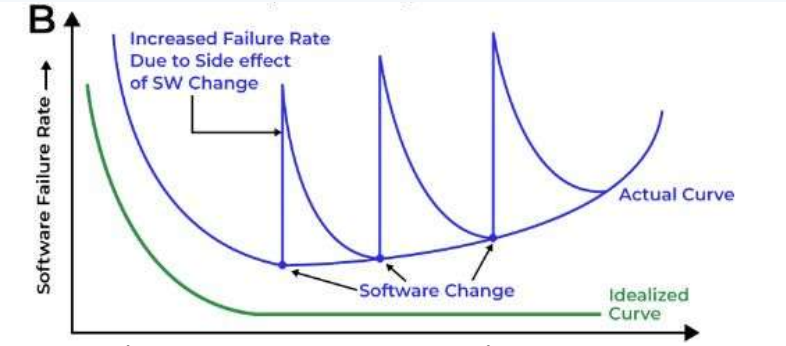


Figure 1.2 Software failure curve

c) Software Failure Curve : Actual Curve

★ In actual curve (above figure), we can see that software may have increased failure rate as it may become old as and when the new development environment changes, Spike in the curve is due to chance of maintenance and side effects.

★ Software may be retired due to new requirements, new expectations, new technologies etc.

★ Hence, software doesn't wear out, but it may get worse.

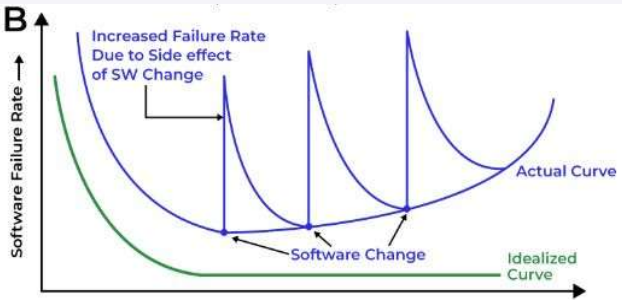


Figure 1.2 Software failure Curve

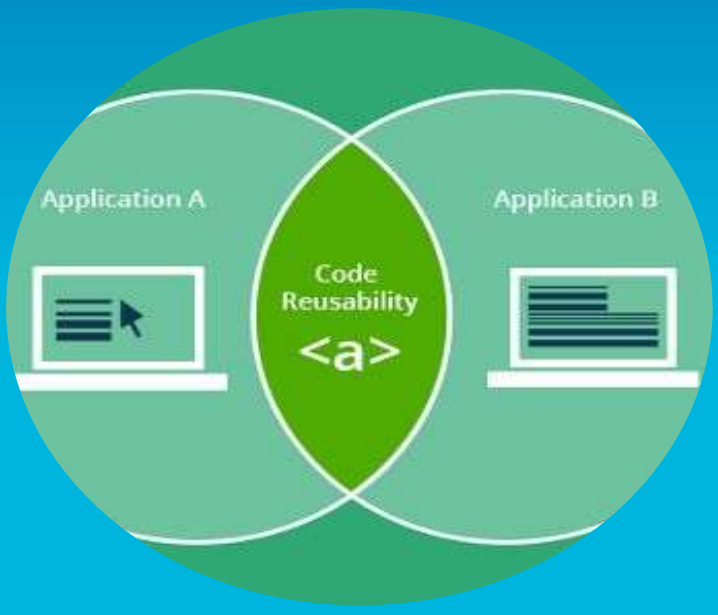
ii. Software is engineered, not manufactured like hardware:

- ★ Once a product is manufactured, it is not easy to modify or change it. While in case of software we can easily change or modify it for later use.
- ★ Even making multiple copies of software is a very easy task rather it is much more tuff in case of hardware.
- ★ In hardware, costing is due to assembly of raw material and other processing expenses while in software development, no assembly needed like hardware. Hence, software is not manufactured as it is developed or it is engineered.

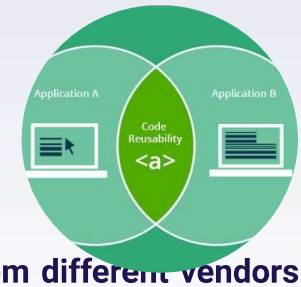
Software Engineer

Hardware Manufacturer

iii. Reusability of Components:



iii. Reusability of Components:



- If we assemble hardware, we will have every part and component from different vendors and then we may produce a finished product.
- In case of software, every project is a new project and we have to start from scratch and design every unit of ef blood software product. Huge amount of efforts required to develop a software system. So building a standard code or design is very useful for making many new projects. These codes are reusable.
- We can reuse software codes, modules or any logical components in any other related software projects. (For example, we prepared a payroll system for any organization. We can reuse some of the logical components while we are developing the related types of payroll systems for any other company or organizations.)
- Generally, GUI (graphical user interface) software is built using reusable components.

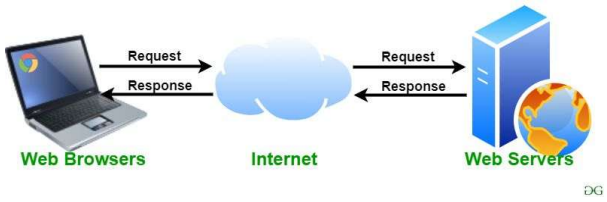
iv. Software is flexible for custom built:

- A software program can be developed to do anything. Any kind of change needed in software can be done easily.
- A software program or product can be built on user requirements basis or custom built. Even the developed software product can also be changed as per the user demands.
- We can say software is very much flexible for custom built rather than hardware.
- Hence, now a days industries are moving toward component based assembly, software continues to be custom built.

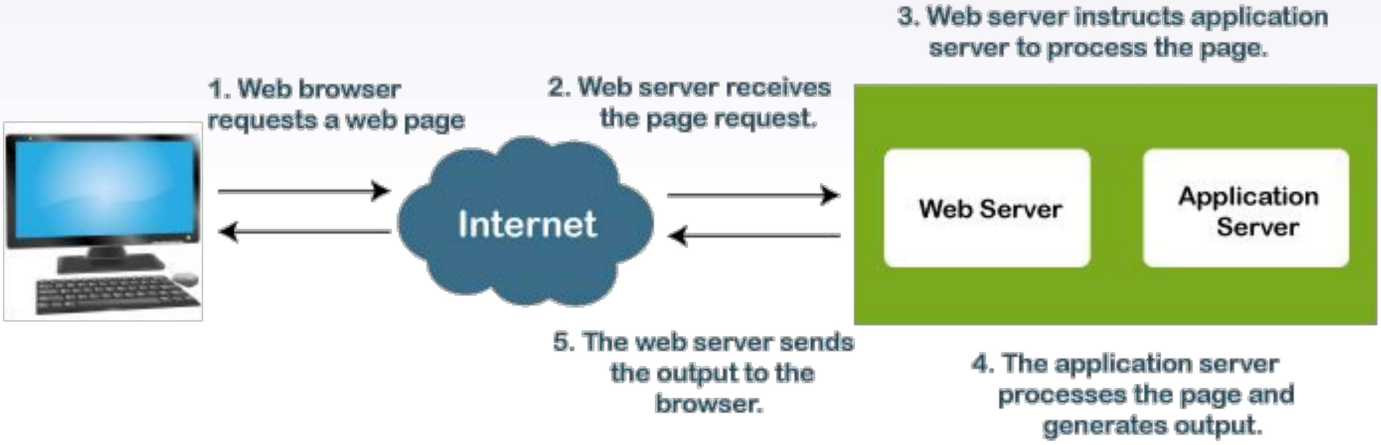
1.2 Characteristics of web-based applications

➢ Basic concept of web-based application

- A Web application (Web app) is an application program that is stored on a remote server and delivered over the Internet.
- User can access this applications with the use of software known as web-browser.
- Web application usually uses a combination of the server-side scripts such as PHP, ASP.
- for handling the information/ data storage and retrieval of the data. Some of them also use the client-side scripts such as JavaScript, HTML to represent the data/information in front of the users.



➢ How web application works ?



Application Server : processing the database, querying the databases & produces the result of the requested data.

Characteristics of web-based application

- **Availability:** The web applications are expected to be available round-the-clock from anywhere in the world.
- **Client driven:** the primary function of a web app is to use hypermedia to present text, graphics, audio, and video content to the end user.
- **Performance:** performance should be better with minimum delay in presenting the contents to the users.
- **Responsive:** it should be able to respond with any browser or device in proper way.
- **Informational:** Read-only content is provided with simple navigation and links.
- **Download:** A user can download information from the appropriate server.
- **Customizable:** The user customizes content to specific needs.
- **User input:** Forms-based input is the primary mechanism for communicating need.

- **Database access:** The user queries a large database and extracts information.
- **Data warehousing:** The user queries a collection of large databases and extracts information.
- **Transaction oriented:** The user makes a request (e.g., places an order) that is fulfilled by the web app.
- **Security:** it should be secure from the hackers or unauthorised users due to the important value of the information that goes through (i.e. passwords, details of banking etc.)
- **Usability:** It should be easy to use with friendly and quick navigation to move around different pages on site.

Characteristics of web-based application

ACER CPUS @DIT

1.1.3 Software engineering (A layered Approach)

- Definitions of Software Engineering
- Software Crisis is the difficulty of writing useful and efficient computer program in the required time.
- Software Engineering discipline began since 5 decade and provides solution to software crisis.
- Software Engineering (SE) is an engineering discipline that covers all the aspects of software from specification to maintenance.
- Software Engineering is an engineering discipline that delivers high quality software at agreed cost & in planned schedule.
- Software Engineering provides framework that guides the software engineers to develop the software.

- Software Engineering tells how software will work with machines.
- It covers technical and management issues.
- Three main aspects of Software Engineering are
- Provide quality product
- Expected cost
- Complete work on agreed schedule
- Software Engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

- Provide Quality product

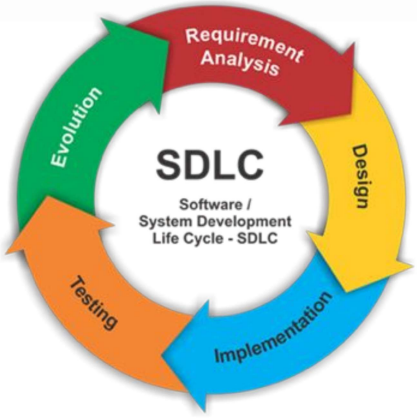
- Do expert Cost

- Work on agreed Schedule

Need of software engineering

1. To help developers to obtain high quality software product.

2. To develop the product in appropriate manner using life cycle models.



3. To acquire skills to develop large programs.

4. To acquire skills to be a better programmer.

5. To provide a software product in a timely manner.

6. To provide a quality software product.

7. To provide a software product at an agreed cost.

8. To develop ability to solve complex programming problems.

(IEEE Definition)

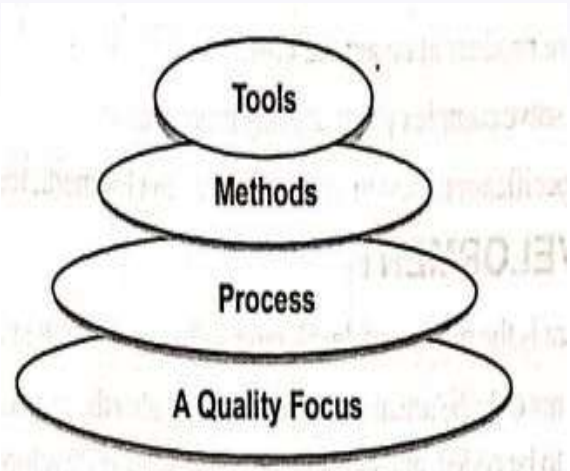
Software Engineering is the application of a systematic, disciplined and quantifiable approach to the development, operation and maintenance of software.

IEEE

1.1.3 Software engineering (A layered Approach)

Software engineering can be viewed as a layered technology.

It contains process, methods and tools that enable software product to be built in a timely manner.

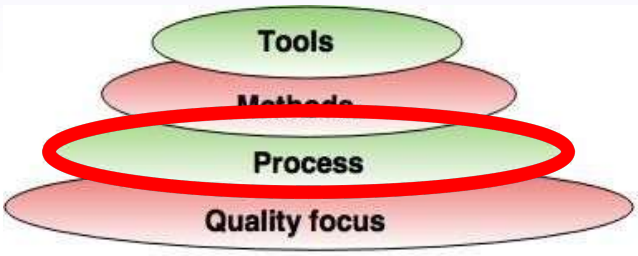


1. A Quality focus Layer

- Software engineering mainly focuses on quality product.
- It checks whether the output meets with its requirement specifications or not.
- Every organization should maintain its total quality management (TQM).
- This layer supports software engineering.

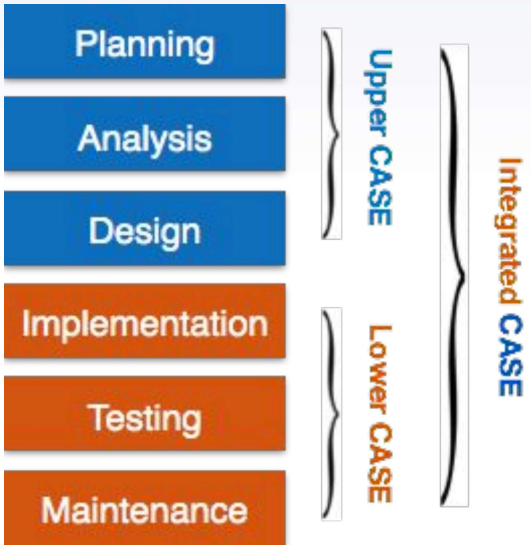
2. Process Layer

- Software process is a set of activities together if ordered and performed properly, the desired result would be produced.
- Main objective of this layer is to develop software in time.
- This layer is the heart of software engineering.
- It holds all the technology layers together like GLUE.
- It is also working as foundation layer.
- It defines the framework activities.



3. Method layer

- It provides technical knowledge for developing software. It describes 'how-to' build software product.
- It creates software engineering environment to software product using CASE tools. (CASE tools combines software, hardware and software engineering database).
- This layer includes requirements analysis, design, program construction, testing, and support

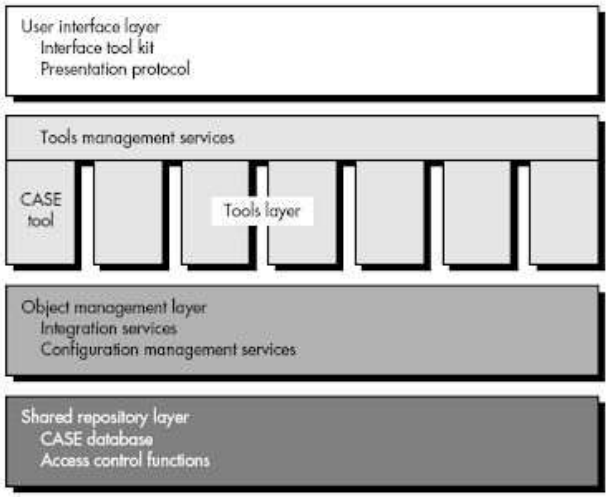


4. Tools layer

- **It provides support to below layers using automated or semi automated tools.**
- **Due to this layer, process is executed in proper manner.**

Key components of the Tools layer include:

- **Integrated Development Environments**
- **Version Control Systems (VCS): Tools like Git**
- **Testing Frameworks: Tools and libraries used for automated testing, such as JUnit for Java or pytest for Python.**
- **Project Management and Collaboration Software: Tools like Jira, Trello, and Slack that facilitate project planning, communication, and collaboration.**



Process Vs Method

Example 1 :
"Process" layer is like the recipe, giving you a high-level plan for making the cake, while the "Methods" layer is like the specific techniques and tools you use to carry out each step in the recipe.

Example 2:
"Process" layer is like the plan you make for your road trip, outlining the route and stops, while the "Methods" layer involves the actual actions and tools you use during the trip to navigate, ensure your vehicle is in good shape, and reach your destination safely.

Process Vs Method

Process Layer:

- In the Process layer, we're setting up the overall plan for how we're going to build the software.
- We decide on the steps, who does what, and when they do it.
- It's like making a schedule for a big project, so everyone knows what to do and when.

Methods Layer:

- In the Methods layer, we're focusing on how we actually build the software.
- We use specific techniques, tools, and rules for writing the code, designing the user interface, and testing everything.
- It's like following the instructions in a recipe to cook a delicious meal – we're using the right tools and methods to create the software effectively and correctly.

Aspect	Process Layer	Methods Layer
Focus	High-level planning and organization	Specific techniques and implementation
Purpose	Defines how the development process is managed	Specifies how individual tasks are performed
Concern	Workflow, project management, and coordination	Coding, design, testing, and technical details
Activities	Project planning, scheduling, risk management, and resource allocation	Writing code, designing, testing, debugging, and using tools
Level of Detail	High-level and strategic	Detailed and tactical
Role	Provides a framework for managing the entire project	Provides guidance for executing specific tasks
Example	Choosing Agile or Waterfall as the development methodology	Deciding to use a specific programming language, design pattern, or testing framework

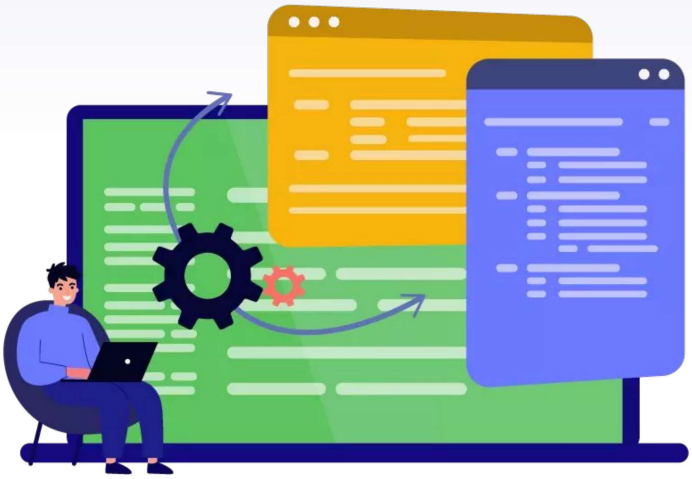
1.1.4 SOFTWARE MYTHS

Software myths propagated misinformation and confusion and that may cause serious problems.

Software is easy to change:

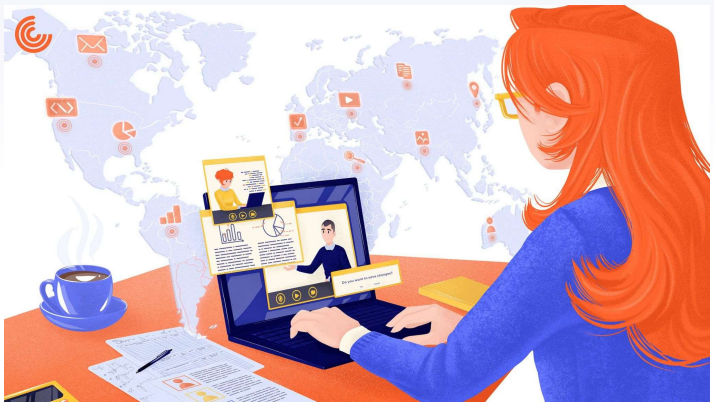
It is almost true that source code of the software is easy to change or edit.

But making changes in the code without making errors is quite difficult. And if we do not take care, then making change in code will be a tedious and expensive task.



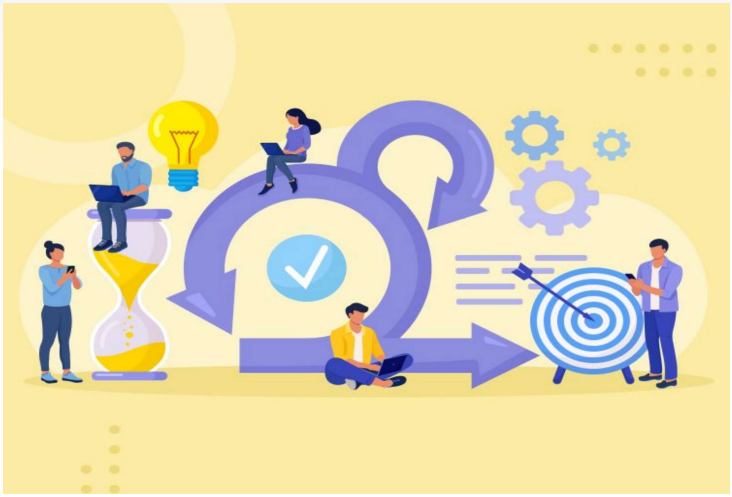
Outsourcing of software to a third party can relax the customers:

It is almost true. But if an organization does not understand to **manage and control** the software internally, then it can cause a **problem.**



Software can work right the first time:

It is not true all time that all software will work at the first time. As there may be many **run time errors** or anomalies created that may reduce the **reliability of the software.**



Increasing of software reliability will increase software safety:

It is much true as **software** becomes more reliable as we are increasing **security features**. That will increase **software safety**.

Reusing software increase safety:

It is not always true, because even though **reusability** is powerful tool, it required **analysis** at the time of feasing.

Best software is one which has more features:

It is not always true. Ofcourse as we are increasing more features in software, its usability increased. But as more the features a software has, it is not necessary that software will be best than others.

There are many other factors like reliability, security, flexibility, portability, etc. required for software to be best.

Testing of Software will remove all errors:

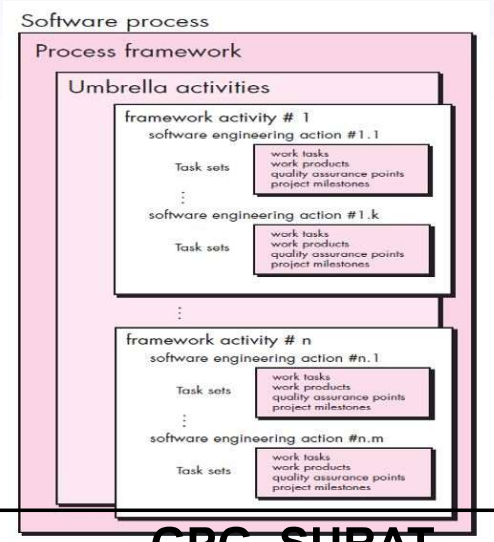
It is very true that if we test the software in all phases like unit test, system test, acceptance test etc., there should be almost no chance of occurring errors at run time.

Once the project is working, the job is done:

It is true. If have tested the software **physically and logically**. And after deployment it is working properly at sight, **job is done.**

1.1.5 Software process framework and umbrella activities:

A software framework provides a standard way to build deploy software product. And a software process is the set of activities and associated results that produce a software product.



★ **Any standard software process model would primarily consist of two types of activities:**

A set of framework activities, which are always applicable to all the projects and A set of umbrella activities which are non SDLC activities that are applicable throughout the process. It provides common process framework for all projects.

- **The Adaptable Process Model (APM) defines the following set of framework activities.**

Communication:

- The software development starts with the communication between customer and developer.

Planning:

- It consists of complete estimation, scheduling for project development and tracking.

Modeling:

- Modeling consists of complete requirement analysis and the design of the project like algorithm, flowchart etc.

- **The Adaptable Process Model (APM) defines the following set of framework activities.**

Construction:

- Construction consists of code generation and the testing part.

→ Coding part implements the design details using an appropriate programming language.
→Testing is to check whether the flow of coding is correct or not.
→Testing also check that the program provides desired output.

Deployment:

→ Deployment step consists of delivering the product to the customer and take feedback from them.
→ If the customer wants some corrections or demands for the additional capabilities, then the change is required for improvement.

Umbrella activities:

- The phases and related steps of the generic view of software engineering are complemented by a number of umbrella activities.
- Umbrella activities are performed throughout the process.
- These activities are independent of any framework activity.

Fig: Umbrella Activities in Software Engineering

1. Software project tracking and control:

- When project tracking and controlling done then software engineering tasks will enable to get the job done on time.

2. Formal technical review (FTR):

- This includes reviewing the techniques that has been used in the project.

3. Software quality assurance (SQA):

- This is very important to ensure the quality measurement of each part of software being developed.

4. Software Configuration Management (SCM):

- SCM is a set of activities designed to control changes made by identifying the work products that are likely to change, establishing relationships among them.

5. Document preparation and production:

- All the project planning and other activities should be hardly copied and the production gets started here.

6. Reusability management:

- This includes the backing up of each part of the software project they can be corrected or any kind of support can be given to them later to update or upgrade the software at user/time demand.

7. Measurement (estimation):

- This will include all the measurement or estimation of every aspects of the software project like: time estimation, cost estimation etc.

8. Risk management:

- As we know that 'tomorrow's problem is today's risk'. Risk management is very important activity for any type of software development.

- It identifies potential problems and deal with them when they are easier to handle before they become critical.
- Risk management allows early identification of risks and provide management decisions to the solutions, and improve quality of the product.

Thank You