Index

★★★

Unit 1

About Software Development

Unit Outcomes (UOs) :

After completion of this unit, students will be able to :

1.1 Software and software development

1.1 Software (Definition and Characteristics)

Software

We all know that Computer is made up of hardware and software. The things in our computer that are visible and physical called hardware and things which are visible but not physical (it is logical) calledsoftware.

In other terms:*Anything that can be stored electronically is called 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."

It also includes representations of pictorial, video, and audio information.

Software is divided into two broad categories :

- **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. It is a program or group of programs generally designed for end users.

Example : Microsoft word, Excel, Railway reservation system etc.

Computer hardware is built from peripherals, device or components, while computer software is logical rather than physical.

Software has following distinct characteristics.

1.1.1 Software Characteristics

Different software characteristics decide whether the software is good or bad. These attributes reflect the quality of a software product. and these are depended on the application of the software also.

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

Following are the characteristics of good software. (Qualities of good software).

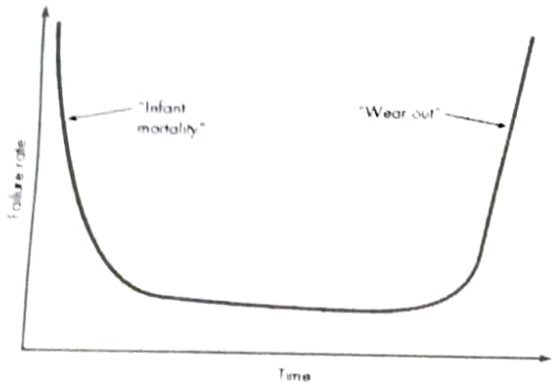
- *Understandability* : software should be easy to understand, even to novice users. It should be efficient to use.
- *Cost* : software should be cost effective as per its usage.
- *Maintainability* : software should be easily maintainable and modifiable in future.
- *Modularity* : software should have modular approach so it can be handled easily fo testing.

- *Functionality* : software should be functionally capable to meet user's requirements.
- *Reliability* : it should have the capability to provide failure-free service.
- *Portability* : software should have the capability to be adapted for different environmen
- *Correctness* : software should be correct as per its requirements.
- *Documentation* : software should be properly documented so that we can re-refer it 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 usin standard bus structure and protocol.
- *Verifiability* : software should be verifiable with its properties and functionalities with it planning and analysis done in previous phase.

Along with these characteristics, software has some special characteristics which ar explained below :

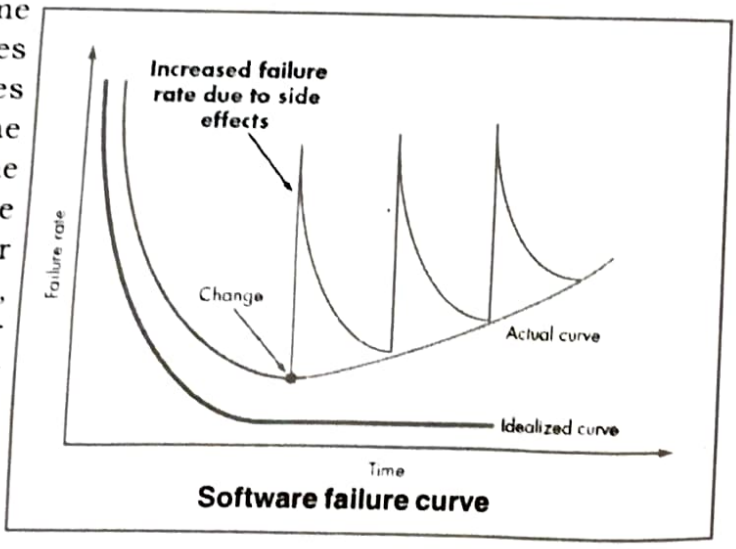
➥ **Software doesn't wear out**

This can be well understood with the help of figure given below.



h/w failure curve

In above figure, 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 effects of dust, vibration, abuse, temperature extremes, and many other environmental factors. So simply, we can say hardware begins to wear out.



Software failure curve

Above figure shows the software failure rate.

- 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.
- 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 etc.

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

➥ **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.

➥ **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 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, if we prepared a payroll system for any organization. We can reuse some of the logical componentswhile 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.

➥ **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 day's industries are moving toward component based assembly, software continues to be custom built.

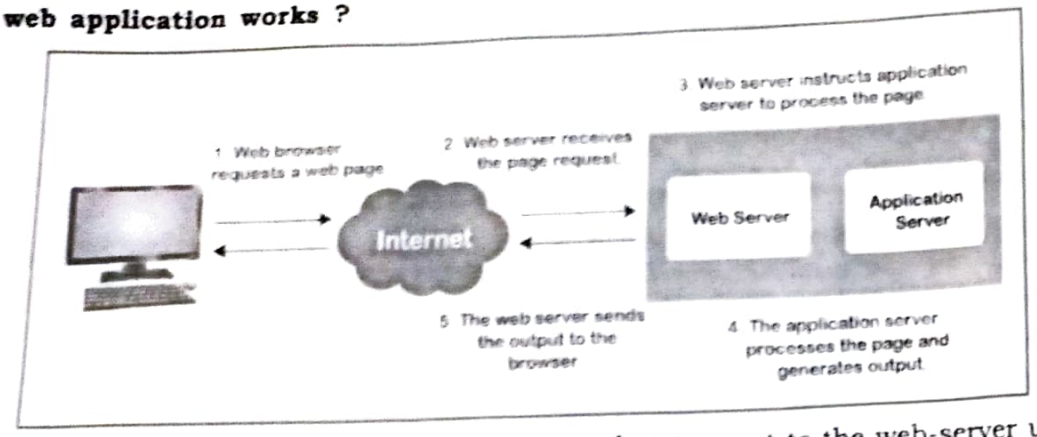
1.1.2 Characteristics of web-based applications

➥ **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 ?



In above figure, we can see that first a user sends a request to the web-server using web browsers.

Then the request is forwarded to the appropriate web application server by the web-server.

Then web application server processing the database and generate the result. And then the result is sent to the web server along with the requested data.

Then web server pass on the result to the end user on the screen.

aracteristics of web-based application

me basic characteristics are explained below:

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.

nformational : Read-only content is provided with simple navigation and links.

Download : A user can download information from the appropriate server.

ustomizable : The user customizes content to specific needs.

ser input : Forms-based input is the primary mechanism for communicating need.

atabase access : The user queries a large database and extracts information.

ta warehousing : The user queries a collection of large databases and extracts ormation.

nsaction oriented : The user makes a request (e.g., places an order) that is fulfilled by web app.

rity : it should be secure from the hackers or unauthorised users due to the rtant value of the information that goes through (i.e. passwords, details of banking

- *Usability* : It should be easy to use with friendly and quick navigation to move around different pages on site.

1.3 Software Engineering – A layered approach

Before understanding the layered approach of Software Engineering, lets first see what exactly Software Engineering (SE) is.

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 planed schedule.***
- Software Engineering provide 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 or to obtain economically software that is reliable and work efficiently on real machines.

Need of software engineering

The reasons why software engineering is needed to develop software products are gi below :

- To help developers to obtain high quality software product.
- To develop the product in appropriate manner using life cycle models.
- To acquire skills to develop large programs.
- To acquire skills to be a better programmer.
- To provide a software product in a timely manner.
- To provide a quality software product.
- To provide a software product at an agreed cost.
- To develop ability to solve complex programming problems.

Also learn techniques of specification, design, user interface development, testing, management, etc.

(IEEE Definition) Software Engineering is the application of a systematic, disci and quantifiable approach to the development, operation and maintenance of so

- **Myth: 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.
- **Myth: Outsourcing of s/w 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.
- **Myth: 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.
- **Myth: 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.
- **Myth: Reusing software increase safety :** It is not always true, because even though reusability is powerful tool, it required analysis at the time of reusing.
- **Myth: Best software is one which has more features :** It is not always true. Of course 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.
- **Myth: 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.
- **Myth: once the project is working, 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.

❖ **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.

SE Layers

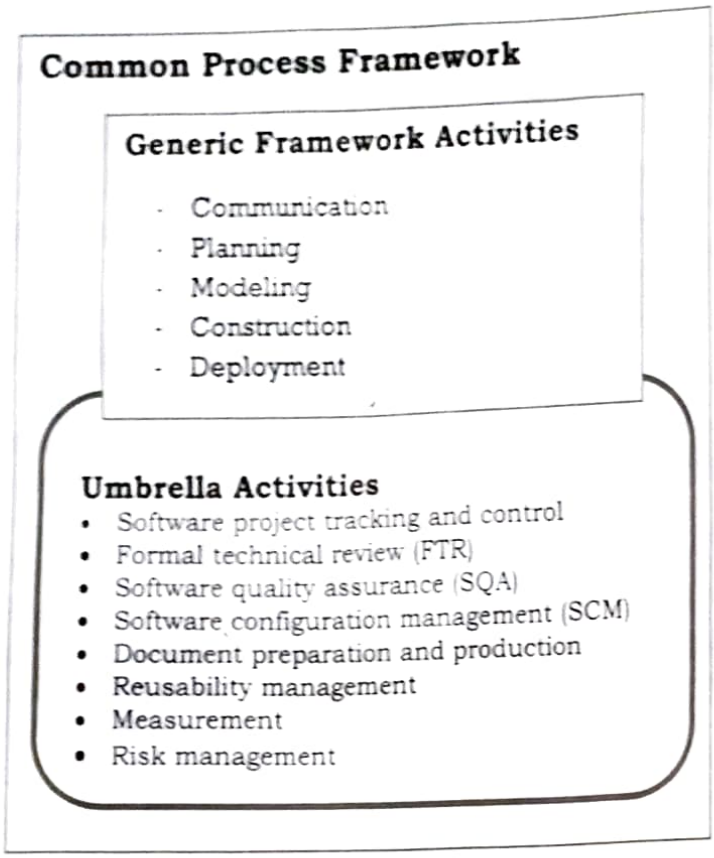
As we can see in the above figure, this layered approach contains mainly four layers.

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.

1.1.4 Software Myths

Many causes of a software application can be traced to a mythology throughthe development of software. Software myths propagated misinformation and confusionand that may cause serious problems.

There are many myths of software and software engineering in software development community. Here is the discussion of these myths and realities.



Generic framework and Umbrella activities

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.

- **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 in the quality of the software.

❖ **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.

Typical activities in this category include :

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. **Softwareconfigurationmanagement (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. **Documentpreparationandproduction**

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

6. **Reusabilitymanagement**

- 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.

ASSIGNMENT

MCQs

1. Microsoft word is a kind of ..... software.
   (a) system (b) SRS **(c) application** (d) programming
2. Full form of TQM is .....
   (a) Total Quality Measurement (b) Time Quality Measurement
   (c) Time Quality Management (d) **Total Quality Management**
3. ..... layer of SE layered approach is working as foundation layer.
   (a) process (b) quality focus (c) method (d) tools
4. ..... layer of SE layered approach defines framework activities.
   (a) tools (b) quality focus **(c) process** (d) method
5. Which of the following umbrella activities designed to control changes.
   (a) SCM (b) FTR (c) SQA (d) RM

True / False

1. Railway reservation system is a kind of system software. **Answer : False**
2. Software doesn't wear out. **Answer : True**
3. Risk management is one of the layers of SE layered approach. **Answer : False**
4. Tools layer of SE layered approach works as a foundation layer. **Answer : False**

Short Questions

1. Give the full form of :
   - TQM • FTR • SCM • SQA
2. Define software.
3. Define software engineering.
4. What is system software ?
5. What is software crisis ?
6. Define web based application (web app).
7. List different layers of software engineering layered approach.
8. List out umbrella activities.
9. List different software process framework activities.

Descriptive Questions

1. Explain software characteristics. **OR** List qualities of good software.
2. Write a short note on web based applications.
3. Explain customer software myth.
4. Justify: software doesn't wear out.
5. Explain why there is a need of software engineering.
6. Explain software engineering layered approach.
7. Explain software process framework activities.

* * *