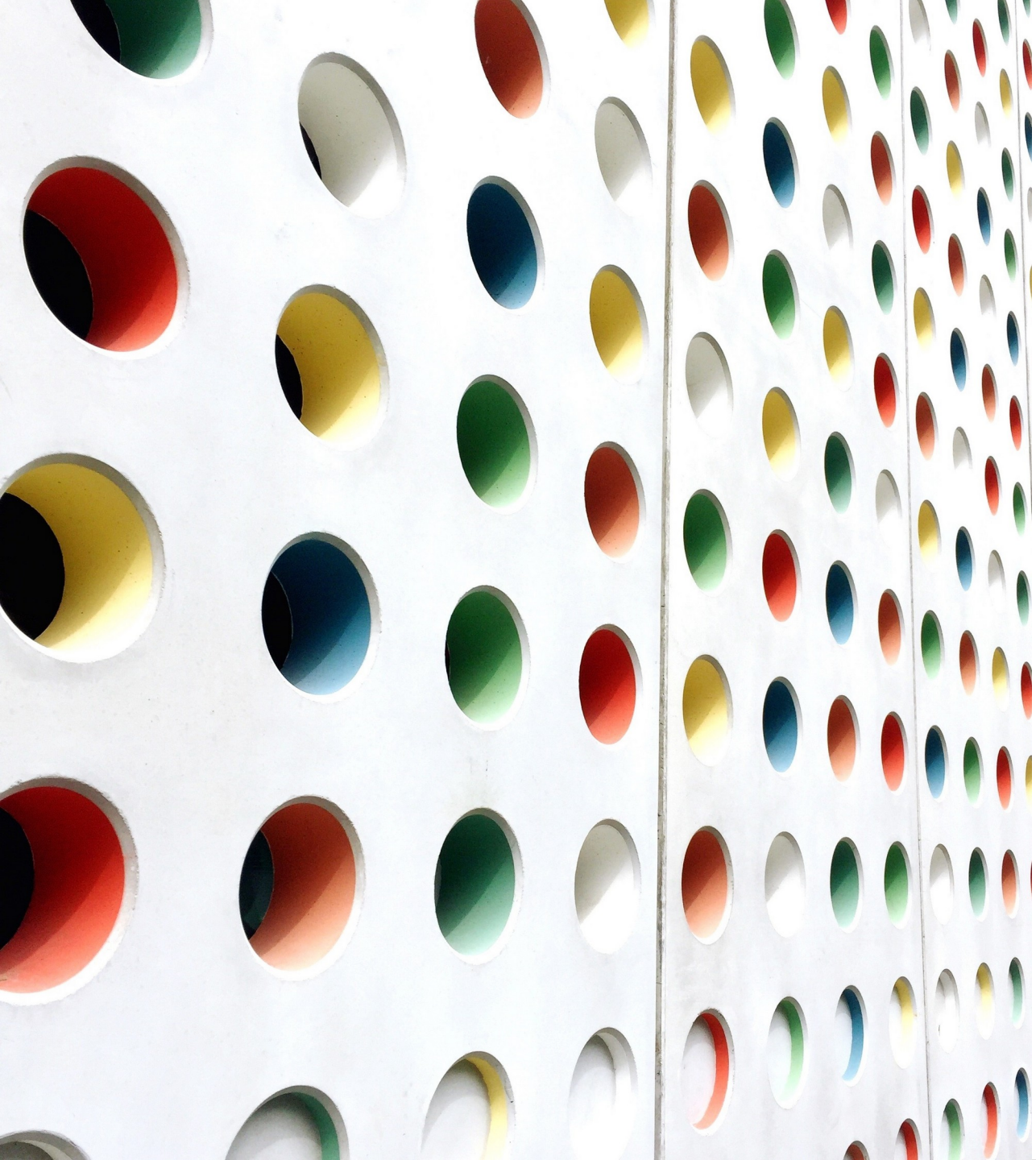


# Overview of Software Engineering

# Software -

- The product that software professionals build and then support over the long term .
- A professional software is developed by a group of software developers working together in a team. It is therefore necessary for them to use some systematic development methodology.
- Today, software takes on a dual role. It is a product and, at the same time, the vehicle for delivering a product

# Characteristics

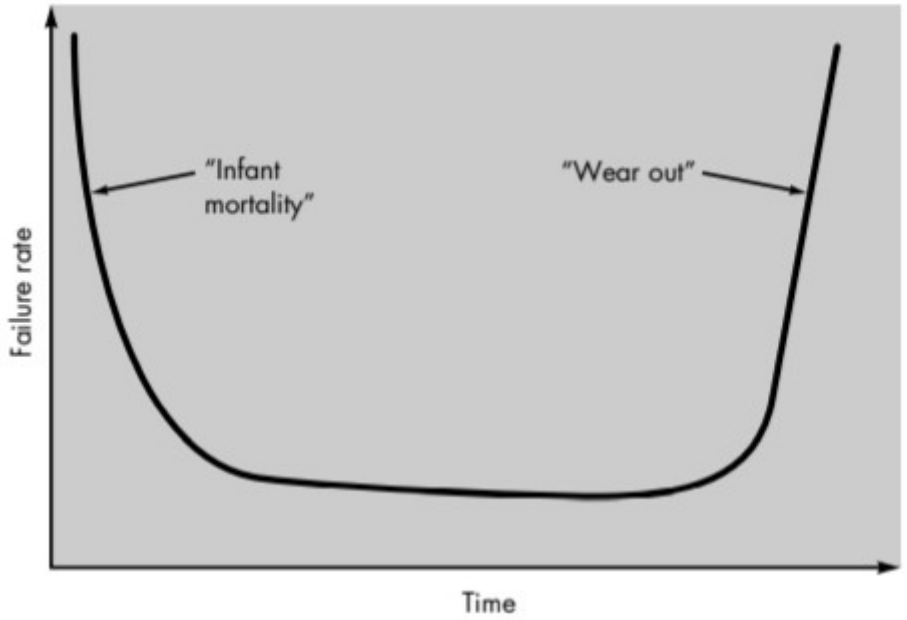
- Software is developed or engineered, it is not manufactured .

Both software and hardware  activities are dependent on people, but the relationship between people applied and work accomplished is entirely different. Both activities require the construction of a "product" but the approaches are different

- Software doesn't "wear out.“

The relationship, often called the "bathtub curve," indicates that hardware exhibits relatively high failure rates. Defects  are corrected and the failure rate drops to a steady-state level (ideally, quite low) for some period of time.

Graph of failure rate is shown .




- Software continues to be custom built.

# Applications

- System Software

It is a collection of programs written to service other programs.

e.g., compilers, editors.

- Real-time software.

Software that monitors real-world events as they occur is called real time.

- Business software.

It is the largest single software application area. Eg : Discrete systems (payroll, accounts receivable/payable, inventory) , MIS

- Engineering and scientific software.

Engineering and scientific software have been characterized by "number crunching" algorithms . However, modern applications within the engineering/scientific area are moving away from conventional numerical algorithms.

- Embedded software

It resides in read-only memory and is used to control products and systems for the consumer and industrial markets. Embedded software can perform very limited and esoteric functions .
e.g., keypad control for a microwave oven, dashboard displays, braking system .

- Personal computer software.

Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications, external network, and database access.

- Web-based software.

The Web pages retrieved by a browser are software that incorporates executable instructions (e.g., CGI, HTML, Perl, or Java), and data (e.g., hypertext and a variety of visual and audio formats).

- Artificial intelligence software.

Artificial intelligence (AI) software is to solve complex problems that do not have straightforward analysis.
Eg: Expert systems, pattern recognition (image and voice), ANN, Game playing.

# Software Engineering

- Definition :

The application of systematic, disciplined ,quantifiable approach to the development operation and maintenance of software ; i.e , application of engineering to software.

Three software paradigms:

- Programming paradigm
- Software design paradigm
- Software development paradigm

# Need of Software Engineering

- Software engineering tools guides scheduling and controlling.
- Different models are required for designing and analysis without which developing software is impossible.
- Resources needs to be managed.
- Software projects are complicated . They need scientific and engineering approach to develop.
- Project teams have to continuously deal with team and new technology challenges.

# Relationship between System Engineering and Software Engineering

## SYSTEM ENGINEERING

- It is the set of activities which takes place before software engineering process starts
- It mainly focuses on the " System". System is the set of components where software product resides.
- It is necessary to understand the role of people, procedures, database, hardware, software and other components
- Includes analysis, modelling, validating and management etc.

## SOFTWARE ENGINEERING

- It is derived from System Engineering.
- It mainly focuses on software product and development process.
- It is overall study of a system where software is going to be placed and used.
- Includes software development, and related issues.

# Software Engineering : A Layered Approach

- Software engineering is a  four layered technology.

- Tools :

It provides automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering, is established.

- Methods:

“How “ to implement .It consists of collection of tasks:

i. Communication – Between customer and developer

ii. Requirement Analysis – To state the requirements.

iii. Analysis and design modelling

Program construction

Testing and support

- Process

i. What activities are to be carried out?

ii. What activities will be taken?

iii. What tasks are to be carried out in given action?

- Quality:

Quality is nothing but “degree of goodness”. It includes characteristics as:

i. Correctness

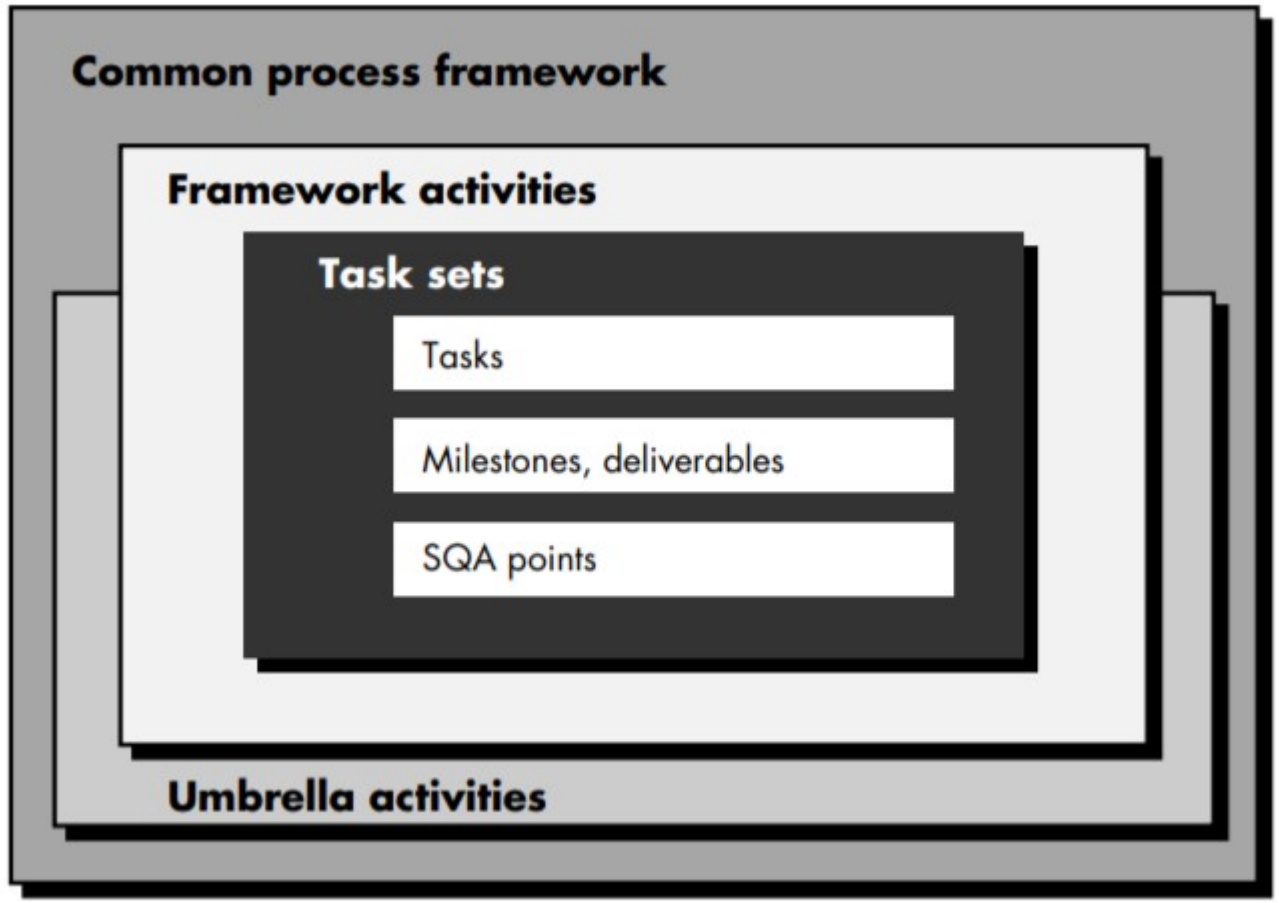
ii. Maintainability

iii. Integrity

iv. Usability

# Software Process

- 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.
- 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.


Common process framework
Framework activities
Task sets
Tasks
Milestones, deliverables
SQA points
Umbrella activities

- Common Process Framework :

The process are chosen by software engineers and their team heads to accomplish the task of software product development.

Different processes can be used to develop different software products , depending on what type of software .

Work Product includes different models, documents , data, reports , forms, and many things.

- Software Product

i. It is a collection of programs that runs on computer designed and built by software engineers .

ii. It also contains documents in form of hard copy and soft copy. Data in the form of text, numbers, pictures , audio , video , etc.

- Software work Product

i. Work products are collection of programs, documents , and resultant data produced by software processes.( from the view point of developers / engineers)

ii. Work products can be defined as ensured information that makes their world better. ( From view point of users.)

- Framework activities :

i. Communication

ii. Planning

iii. Modelling

iv. Construction

v. Deployment

- Tasks sets :

For ex : requirements of the project , activities are :

i. Create and maintain list of stakeholders

ii. Arrange a meeting of all stakeholders

iii. Allow stakeholders to make list of features and functions required .

iv. Discuss all the requirements.

v. Classify requirements on priority basis.

vi. Find areas of uncertainity/ risks.

- Umbrella activities are :

i. Software project tracking and control.

ii. Risk management.

iii. Software quality assurance

iv. Technical reviews

v. Measurement

vi. Reusability Management

vii. Work product preparation and production

viii. Software configuration management

# SOFTWARE COMPONENTS

Functionality: It refers to the degree of performance of the **software** against its intended purpose

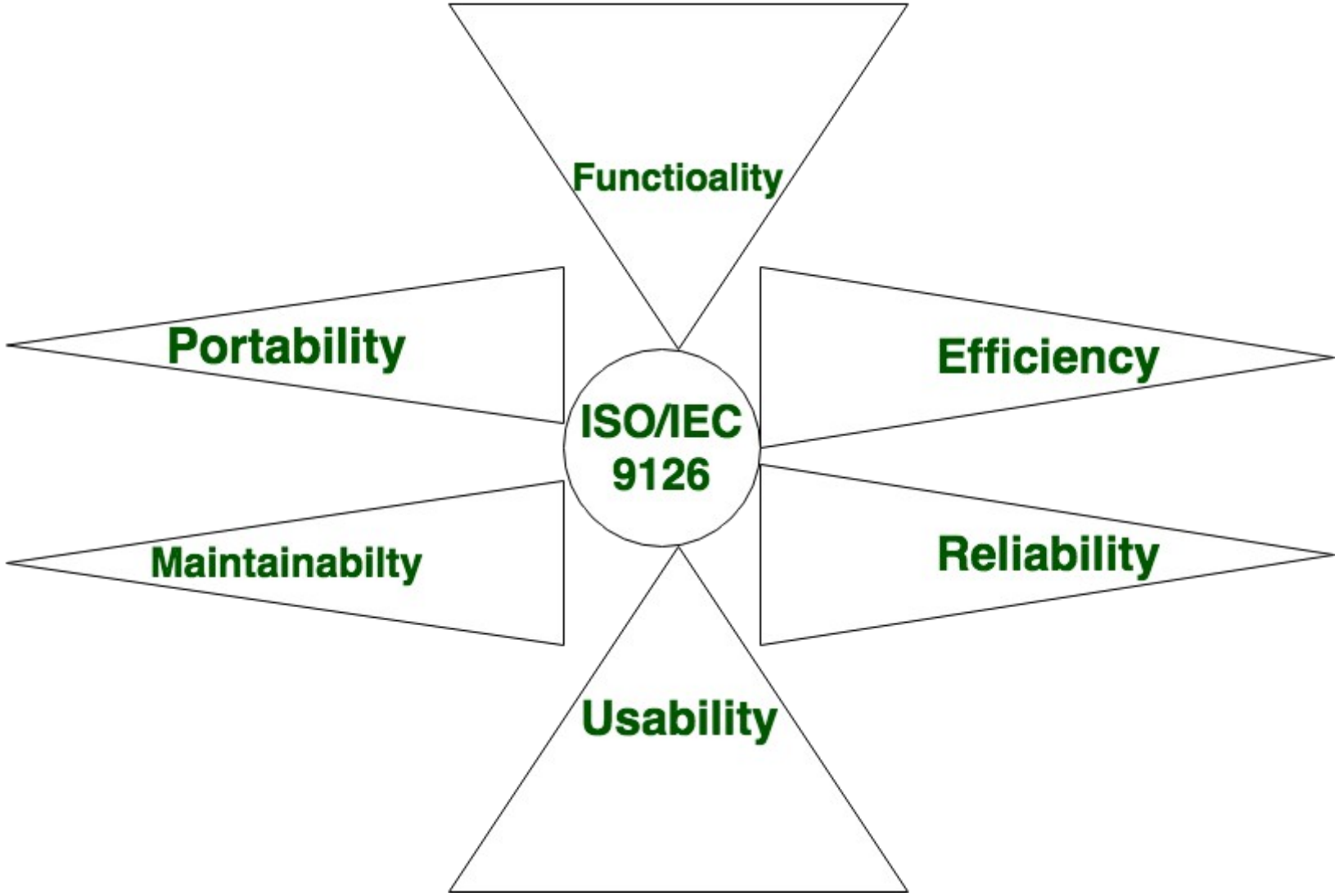
Reliability:

Efficiency:

Usability:

Maintainability:

Portability:


Functioality
Portability
Efficiency
ISO/IEC
9126
Maintainabilty
Reliability
Usability

## Functionality:

is the sum or any aspect of what a product, such as a software application or computing device, can do for a user.

A product's functionality is used by marketers to identify product features and enables a user to have a set of capabilities.

Functionality may or may not be easy to use.

It refers to the degree of performance of the **software** against its intended purpose

## Reliability:

The probability of failure-free **software** operation for a specified period of time in a specified environment.

It can be an important factor affecting system **reliability**.

Different from hardware **reliability** since it doesn't age, wear out, rust, deform or crack!

Software usually stays in the same condition as when it was created and unless there are changes caused by hardware

**Efficiency:**

**Efficiency** measures the amount of each **engineer's** “productive” code, or code that provides business value.

. An **engineer** creating a whole new solution or implementing

sweeping code changes will likely deal with lots of trial and error with a low **efficiency** rate.

**Usability:**

**Usability** refers to the quality of a user's experience when interacting with products or systems,

including websites, **software**, devices, or applications.

**Usability** is about effectiveness, efficiency and the overall satisfaction of the user

**Maintainability:**

More formally, the IEEE Standard Glossary of **Software Engineering** Terminology defines **maintainability** as

The ease with which a **software** system or component can be modified to correct faults,

improve performance or other attributes, or adapt to a changed environment.

**Portability:**

Portability, in relation to software, is a measure of how easily an application can be transferred from one computer environment to another.

A computer software application is considered portable to a new environment if the effort required to adapt it to the new environment is within reasonable limits.

**Portability** in high-level computer programming is the usability of the same **software** in different environments.

**Module:**

A **module** is defined as the unique and addressable components of the **software** which can be solved and modified independently without disturbing ( or affecting in very small amount ) other **modules** of the **software**.

Thus every **software design** should follow modularity.

A module is a software component or part of a program that contains one or more routines.

One or more independently developed modules make up a program.

An enterprise-level software application may contain several different modules, and each module serves unique and separate business operations.

Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application.