Define software.

Software is -

1. Instructions (computer programs) that when executed provide desired features, function and performance.

2. Data structures that enable the programs to adequately manipulate information and

3. Documents that describe the operation and use of the programs.

Describe the characteristic of software.

1. Software is developed or engineered; it is not manufactured in the classical sense.

Software is a design of strategies, instruction which finally perform the designed, instructed tasks. and a design can only be developed, not manufactured. On the other hand, design is just a material of hardware.

In the case of software, a good design will introduce good software, if done through the design. But in the case of hardware, the design is only the half way. In the manufacturing phase it may introduce quality problem and this phase doesn't exist in the case of software.

2. Software doesn't "wear out".

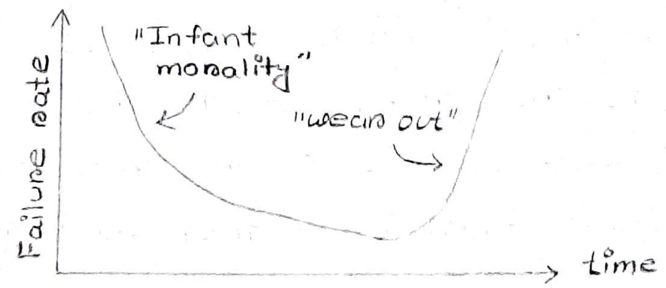


Fig 1: Failure curve of hardware

Considering fig-1. This is often called the "buthtub curve". It indicates that, at the beginning of the life of hardware it shows high failure rate as it contains many defects. By time, the manufacturers or the designers repair these defects and it becomes idealized or gets into the steady states and continues. But after that, as time passes, the failure rate rises again and this may be caused by excessive temperature, dust, vibration, improper use and so on and at one time it becomes totally unusable. This state is the "wear out" state.

On the other hand, software doesn't wear out. Like hardware software also shows high failure rate at its infant state. Then it gets modifications and the defects get corrections and thus it comes to the idealize state.

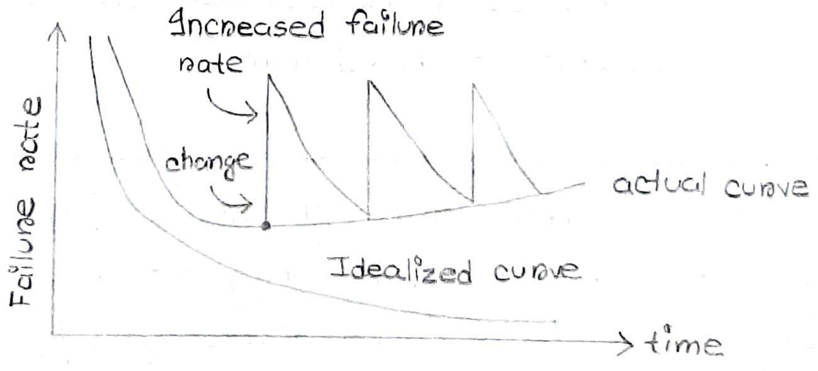


Fig 2: Failure curve for software

But though a software not having any defects it may get the need of modification as the users demand; from the software may change. And when it occurs, the unfulfilled demands will be considered as defects and thus the failure rate will increase After one modification another may get the necessity. In that way, slowly, the minimum failure rate level begin to rise which will cause the software deteriorated due to chang, but it does not "wear out".

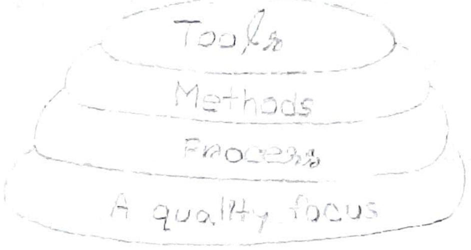
In fig-2, the idealized curve for software has been shown. Also the change to failure rate due to users' demand and the rise of the minimum failure rate has been shown on the actual curve.

3. Reusability

It is the ease with which software can be reused in developing other software. By reusing existing software, developers can create more complex software in a shorter amount of time.

A simple example of software reuse could be the development of an efficient sorting routine that be incorporated in many future applications.

Software engineering is a layered technology - explain.



Software engineering is divided into four layers

A quality focus:

- Any engineering approach must rest on an quality.
- The "Bed Rock" that supports software engineering is quality focus.

Process:

- Foundation for software engineering is the process layer.
- Software engineering process is the glue that holds all the technology layers together and enables the timely development of computer software.
- It forms the base for management control of software project.

Methods:

- Software engineering methods provide the "Technical Questions" for building software.
- Methods contain a broad array of tasks that include communication requirement analysis, design modeling, program construction testing and support.

Tools :

- Software engineering tools provide automated or semi-automated support for the "Process" and the "Methods".
- Tools are integrated so that information created by one tool can be used by another.

Describe the three phases of the generic view of software engineering.

The work associated with software engineering can be categorized into three generic phases, regardless of application area, project size or complexity.

Definition Phase:

The definition phase focuses on "what". That is, during definition, the software engineer attempts to identify what information is to be processed, what function and performance are desired, what system behaviour can be expected, what interfaces are to be established, what design contraints exist and what validation criteria are required to define a successful system. During this, three major tasks will occur in some form: system or information engineering, software project planning and requirement analysis.

Development Phase:

The development phase focuses on "how". That is, during development a software engineer attempts to define how data are to be structured, how function is to be implemented within a software architecture, how interfaces are to be characterized, how the design will be translated into a programming language and how testing will be performed. During this, three specific technical tasks should always occur; software design, code generation and software testing.

Support phase:

The support phase focuses on "change" associated with error correction, adaptions required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. Four types of change are encountered during the support phase:

Correction: Even with the best quality assurance activities, it is likely that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

Adaption: Over time, the original environment, that is, CPU, operating system, business rules etc for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accomodate changes to its external environment.

Enhancement: As software is used, the customer/users will recognize additional functions that will provide benefit. Perfectible maintenance extends the software beyond its original functional requirments.

Prevention: Computer software deteriorates due to change, and because of this, preventive maintenance, often called software re-engineering, must be conducted to enable the software to serve the needs of its end users.

Describe the umbrella activities.

The phases and related steps of the generic view of software engineering are complemented by a number of umbrella activities. Typical activities in this category include:

Software project tracking and control:

When plan, tasks, models all have been done then a network of software engineering tasks that will enable to get the job done on time will have to be created.

Formal Technical Reviews:

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

Software Quality Assurance:

This is very important to ensure the quality measurement of each part to ensure them

Software Configuration Management:

Software configuration management (SCM) is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products.

Document preparation and production:

All the project planning and other activities should be handly copied and the production get started here.

Re-usability Management:

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

Measurement & Metrics:

This will include all the measurement of every aspects of the software project.

Risk Management:

Risk management is a series of steps that help a software team to understand and manage uncertainty. It is a really good idea to identify it, assess its probability of occurrence estimate its impact and establish a contigency plan that - 'should the problem actually occur'.

What is KPA?, Describe its characteristics / why it is used?

The KPAs (key Process Area) describe those software engineering functions that must be present to satisfy good practice at a particular level. Each KPA is described by identifying the following characteristics:

Goals - the overall objectives that the KPA must achieve

Commitments - requirements (imposed on the organization) that must be met to achieve the goals or provide proof of intent to comply with the goals.

Abilities - those things that must be in place to enable the organization to meet the commitments.

Activitise - the specific tasks required to achieve the KPA function.

Methods for monitoring implementation - the manner in which the activities are monitored as they are put into place.

Methods for verifying implementation - the manner in which proper practice for the KPA can be verified.