LECTURE -2

Dronacharya College of Engineering

Topic Covered in this lecture:

1.Software Characteristics

2. Software Crisis

3. Software Myths

4. Software Application

.

SOFTWARE CHARACTERISTICS?

Software Characteristics are:-

(1). Software is developed or engineered, it is not manufactured:--

Unlike hardware, software is logical rather than physical. It has to be designed well before producing it.

In spite of availability of many automated software development tools, it is the skill of the individual, creativity of the developers and proper management by the project manager that counts for a good software product.

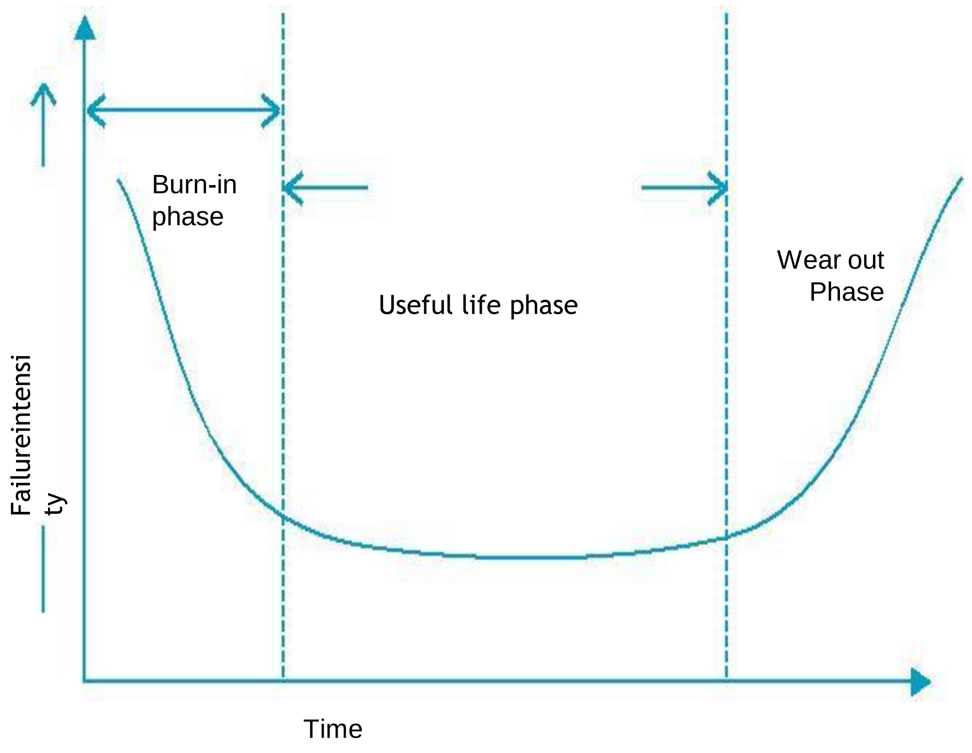
(2).Software does not "wear out":--

As time progresses, the hardware components start deteriorating-they are subjected to environmental maladies such as dust, vibration, temperature etc. and at some point of time they tend to breakdown.

The defected components can then be traced and replaced. But, software is not susceptible to the environmental changes. SDo, it does not wear out. The software works exactly the same way even after years it was first developed unless any charges are introduced to it.

(2).Software does not "wear out":--

There is a well-known "bath tub curve" in reliability studies for hardware products. The curve is given in Fig. The shape of the curve is like "bath tub"; and is known as bath tub curve.

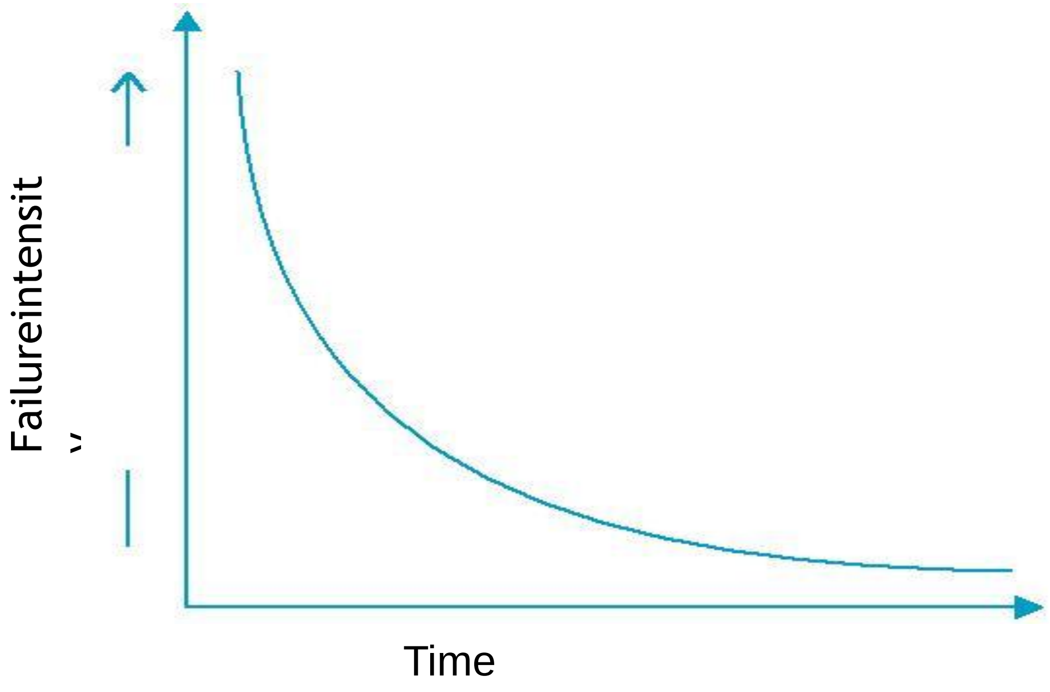


Hard ware curve

(2).Software does not "wear out":--

There are three phases for the life of a hardware product.:

- Initial phase is burn-in phase, where failure intensity is high. It is expected to test the product in the industry before delivery. Due to testing and fixing faults, failure intensity will come down initially and may stabilize after certain time.
- The second phase is the useful life phase where failure intensity is approximately constant and is called useful life of a product. After few years, again failure intensity will increase due to wearing out of components. This phase is called wear out phase.
- We do not have this phase for the software as it does not wear out. The curve for software is given in Fig.



(Software curve)

Important point is software becomes reliable overtime instead of wearing out. It becomes obsolete, if the environment for which it was developed, changes. Hence software may be retired due to environmental changes, new requirements, new expectations, etc.

(3).Most software is custom-built, rather than being assembled from existing components:--

Most of the engineered products are first designed before they are manufactured, Designing includes identifying various components for the product before they are actually assembled. Here several people can work independently on these components thus making the manufacturing system highly flexible.

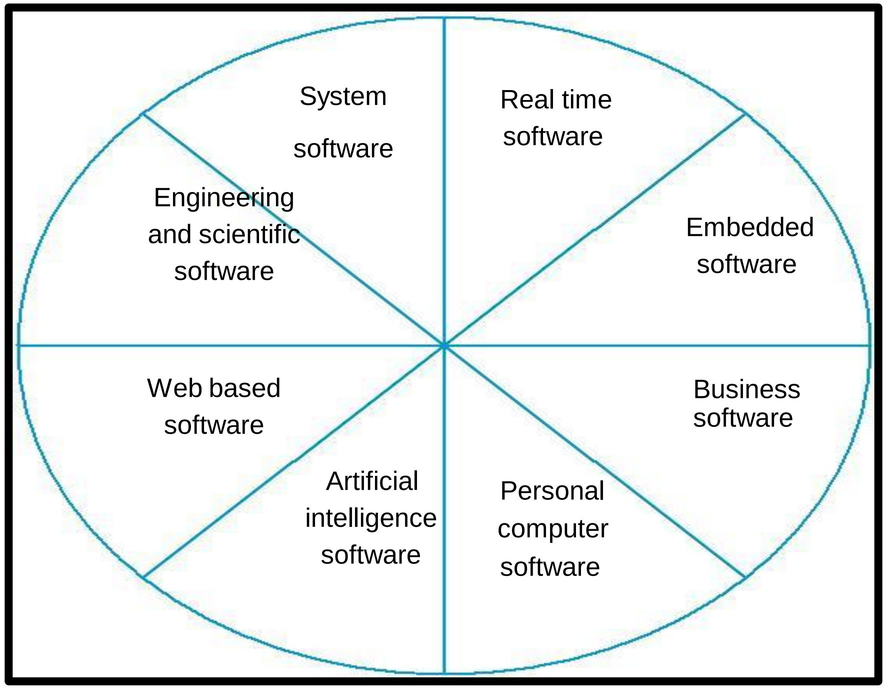
In software, breading a program into modules is difficult task , since each module is highly interlinked with other modules. Further, it requires lot of skill to integrate different modules into one. Now a days the term component is widely used in software industry where object oriented system is in use.

SOFTWARE APPLICATIONS

Software has become integral part of most of the fields of human life.

We name a field and we find the usage of software in that field.

Software applications are grouped In to eight areas for convenience as shown in Fig


System software
Real time software
Engineering and scientific software
Embedded software
Web based software
Business software
Artificial intelligence software
Personal computer software

Software applications

SOFTWARE APPLICATIONS

(*i*) **System software:** Infrastructure soft-ware come under this category like compilers, operating systems, editors, drivers, etc. Basically system software is a collection of programs to provide service to other programs.

(*ii*) **Real time software:** These software are used to monitor, control and analyze real world events as they occur. An example may be software required for weather forcasting. Such software will gather and process the status of temperature, humidity and other environmental parameters to forcast the weather.

SOFTWARE APPLICATIONS

(*iii*) **Embedded software:** This type of software is placed in "Read-Only-Memory (ROM)" of the product and control the various functions of the product. The product could be an aircraft, automobile, security system, signaling system, control unit of power plants, etc. The embedded software handles hardware components and is also termed as intelligent software.

(*iv*) **Business software:** This is the largest application area. The software designed to process business applications is called business software. Business software could be payroll, employee management, account management. It may also be a data warehousing tool which helps us to take decisions based on available data. Management information system, enterprise resource planning (ERP) and such other software are popular examples of business software.

SOFTWARE APPLICATIONS

(*v*) **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. This is a very upcoming area and many big organizations are concentrating their effort here due to large customer base.

(*vi*) **Artificial intelligence software:** Artificial Intelligence software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straight forward analysis [PRESOI]. Examples are expert systems, artificial neural network, signal processing software etc.

SOFTWARE APPLICATIONS

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

(*viii*) **Engineering and scientific software:** Scientific and engineering application software are grouped in this category. Huge computing is normally required to process data. Examples are CAD/CAM package, SPSS, MATLAB, Engineering Pro, Circuit analyzers etc.

SOFTWARE MYTHS

Software Myths – Management

- "We already have a book that is full of standards and procedures for building software. I provide my people with everything they need to know?"
  - Not used, not up to date, not complete, not focused on quality, time, and money
- "If we get behind, we can add more programmers and catch up"
  - Adding people to a late software project makes it later
  - Training time, increased communication lines
- "If I decide to outsource the software project to a third party, I can just relax and let that firm build it"
  - Software projects need to be controlled and managed

SOFTWARE MYTHS

Software Myths - Customer

- "A general statement of objectives is sufficient to begin writing programs – we can fill in the details later"
  - unclear statement of objectives spells disaster

- "Project requirements continually change, but change can be easily accommodated because software is flexible"
  - Impact of change depends on where and when it occurs in the software life cycle (requirements analysis, design, code, test)

SOFTWARE MYTHS

Software Myths - Practitioner

- "Once we write the program and get it to work, our job is done"
  - 60% to 80% of all effort expended on software occurs after it is delivered
- "Until I get the program running, I have no way of assessing its quality
  - Formal technical reviews of requirements analysis documents, design documents, and source code (more effective than actual testing)
- "The only deliverable work product for a successful project is the working program"
  - Software, documentation, test drivers, test results
- "Software engineering will make us create huge and unnecessary documentation and will invariably slow us down"
  - Creates quality, not documents; quality reduces rework and provides software on time and within the budget

SOFTWARE CRISIS

It was in late 1960's

- Many software projects failed.
- Many software projects late, over budget ,providing unreliable software that is expensive to maintain.
- Many software projects produced software which did not satisfy the requirements of the customer.
- Complexities of software projects increased as hardware capability increased.
- Large software system is more difficult and expensive to maintain.

All the attributes of what was called a 'Software Crisis" . So the term 'software Engineering' first introduced at a conference in late 960's to discuss the software crisis.

SOFTWARE CRISIS

The causes of the software crisis were linked to the overall complexity of hardware and the software development process. The crisis visible itself in several ways:

- Projects running over-budget.
- Projects running over-time.
- Software was very inefficient.
- Software was of low quality.
- Software often did not meet requirements.
- Projects were unmanageable and code difficult to maintain.
- Software was never delivered.

ASSIGNMENT

Question1-

“The software crisis is aggravated by the progress in hardware technology?” Explain with examples.

Question2 –

What is software crisis? Was Y2K a software crisis?