UNIT-I
KCS-601
SOFTWARE ENGINEERING
PART-I

Software Engineering

Syllabus

Unit-I: Introduction

Introduction to Software Engineering, Software Components, Software Characteristics, Software Crisis, Software Engineering Processes, Similarity and Differences from Conventional Engineering Processes, Software Quality Attributes.

Software Development Life Cycle (SDLC) Models: Water Fall Model, Prototype Model, Spiral Model, Evolutionary Development Models, Iterative Enhancement Models.

Software Engineering

Software engineering (**SE**) is the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of Software, and the study of these approaches; that is, the application of engineering to Software.

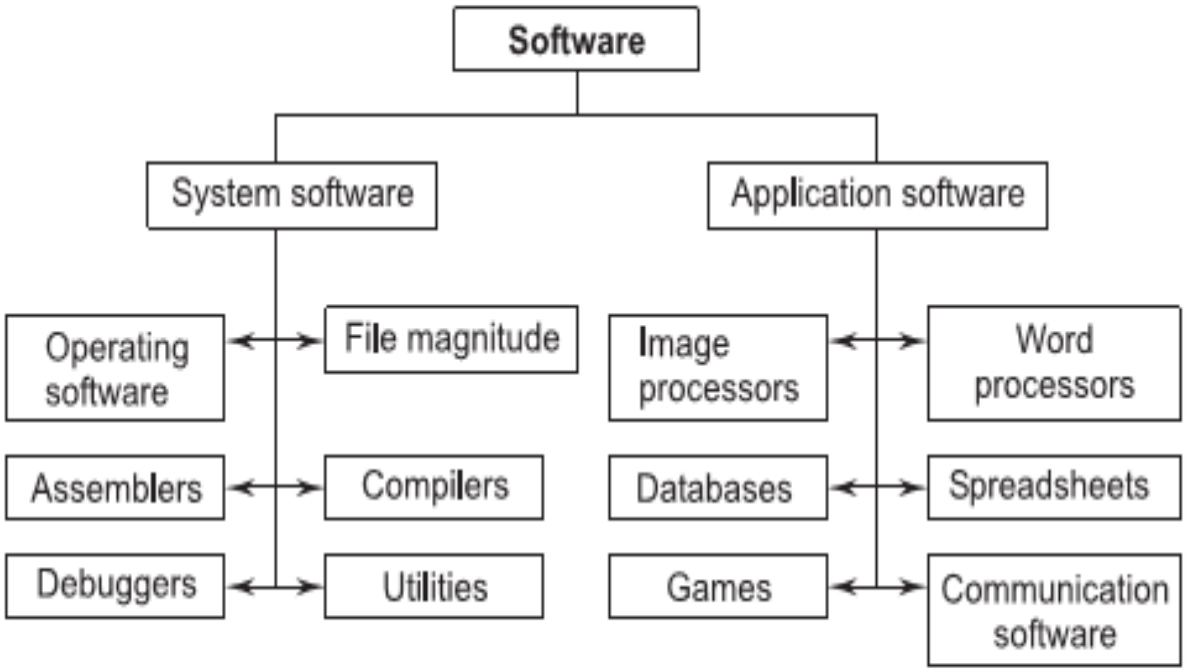
Classification Of Software

- Custom Software
  - For a specific customer
- Generic Software
  - Sold on open market
  - Often called
    - COTS (Commercial Off The Shelf)
    - Shrink-wrapped
- Embedded Software
  - Built into hardware
  - Hard to change

Types Of Software

- **System Software-**This Software includes the operating system and all utilities that enable the computer to function.
- **Application Software-**These consist of programs that do real work for users. For example, word processors, spreadsheets, and database management systems fall under the category of applications Software.

Types Of Software



Software Myth

It is Defined in these categories-

- Management Perspectives
- Customer Perspectives
- Practitioners (Developer) Perspectives

Software Myths

- Software is Easy to Change.
- Computer Software provides the greater reliability than the device they replaced.
- Testing Software Can remove all errors.
- Reusing Software increase safety.
- Software with more features is better Software.
- Once we write the program and get it to work, our job is done.
- The only deliverable work product for a successful project is the working program.

Software Myth

Pressman (1992) has compiled the following myths that prevail in the software industry:

A. Management Myths:

- We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know?
- My people do have state-of-the-art software development tools; after all, we buy them the newest computers.
- If we get behind schedule, we can add more men and catch up.

B. Customer's Myths:

- A general statement of objectives is sufficient to begin writing programs—we can fill in the details later.
- Project requirements continually change, but change can be easily accommodated because software is flexible.

C. Practitioner's Myths:

- Once we write the program and get it to work, our job is done.
- Until I get the program "running," I really have no way of assessing its quality.
- The only deliverable for a successful project is the working program.

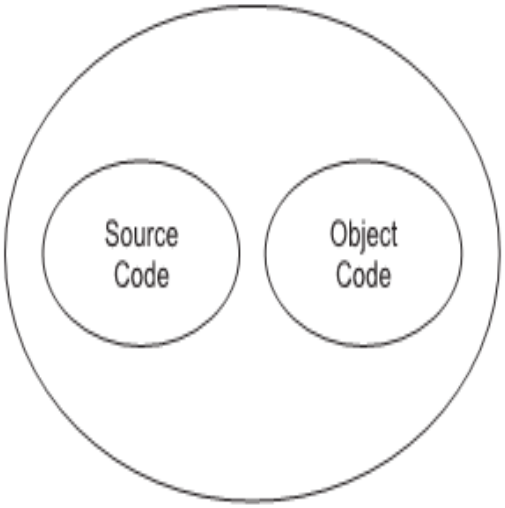
Software Myths

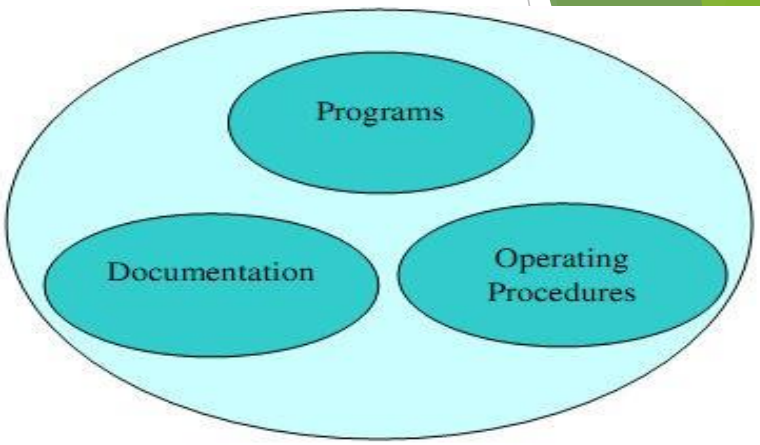
- “If my project is behind the schedule, I always can add more programmers to it and catch up …”(a.k.a. “**The Mongolian Horde concept**”)
- “We already have a book of standards and procedures for building software. It does provide my people with everything they need to know …”
- “If I decide to outsource the software project to a third party, I can just relax: Let them build it, and I will just pocket my profits …”

Software Components

- Off-the Self Components.
- Full Experience Components.
- Partial Experience Components.
- New Components.

Software Components



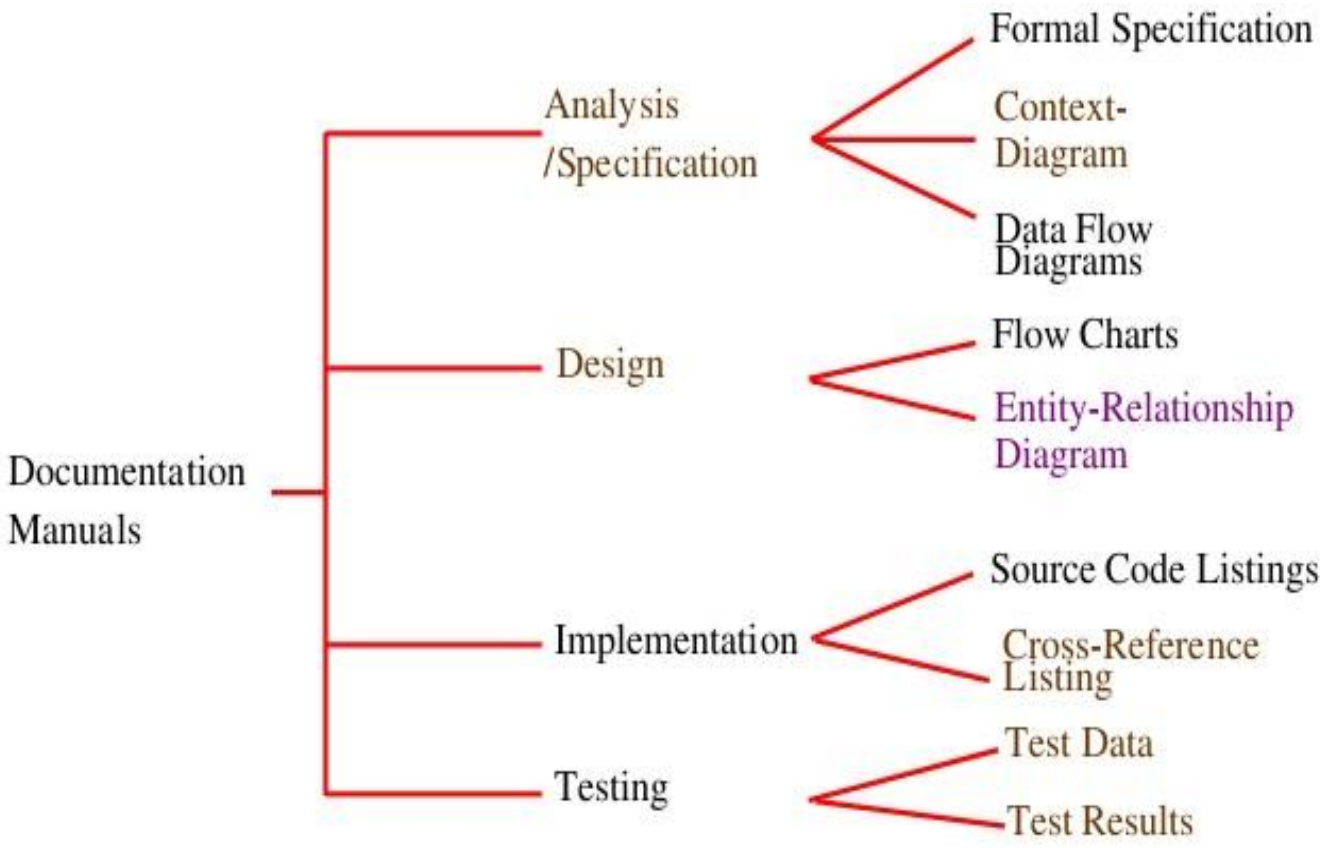


- **Program=**Source code + object code
- **Software=**Program + Documentation + Operating Procedures

Software Components

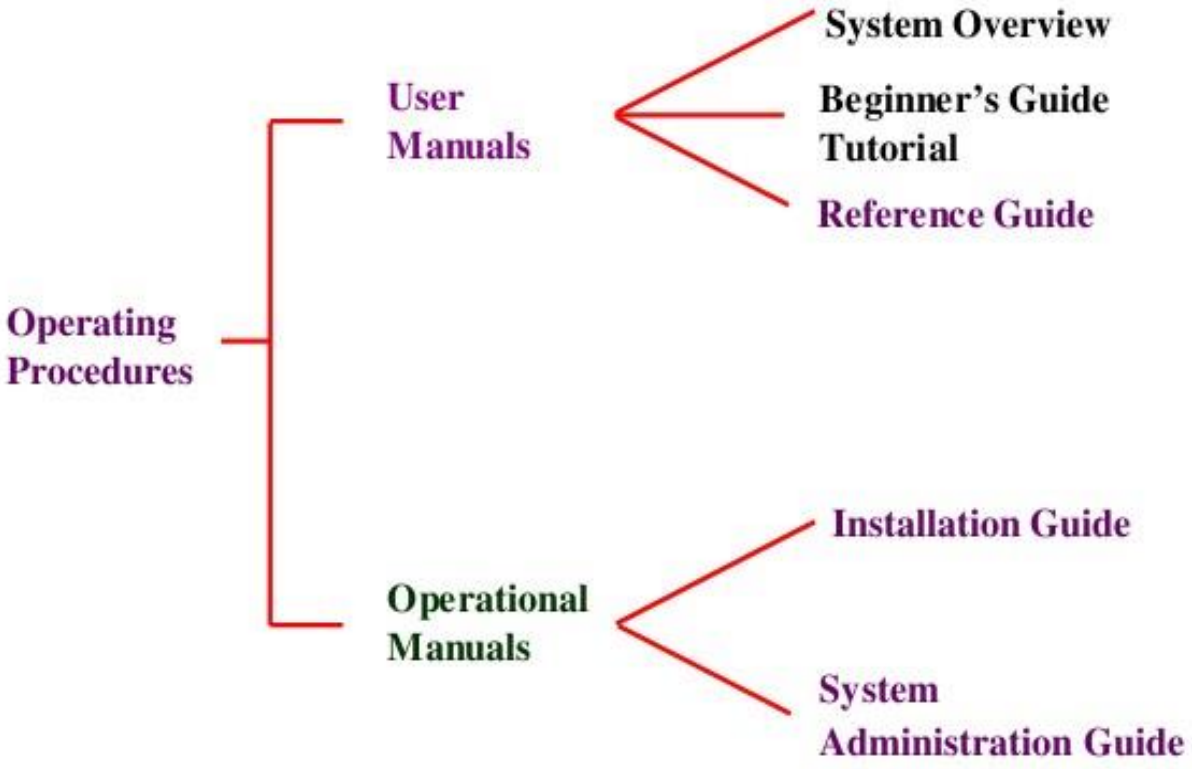
S. No.	Programs	Software Products
1.	Programs are developed by individuals for their personal use	A software product is usually developed by a group of engineers working as a team
2.	Usually small in size	Usually large in size
3.	Single user	Large number of users
4.	Single developer	Team of developers
5.	Lack proper documentation	Good documentation support
6.	Adhoc development	Systematic development
7.	Lack of user interface	Good user interface
8.	Have limited functionality	Exhibit more functionality

Documentation consists of different types of manuals are:



List of documentation manuals

Documentation consists of different types of manuals are:



List of operating procedure manuals.

Importance of Software

Computer Software has become a driving force-

- It is the engine that drives business decision making.
- It serves as thebasis formodern scientific investigation and engineering problem-solving.
- It is embedded in all kinds of systems, such as transportation, medical, telecommunications, military, industrial processes, entertainment, office products, etc.

Software Characteristic(important)

Important Characteristics are-

- Software does not wear out.
- Software is Flexible and Reliable.
- Software is Engineered, Not Manufactured.
- Reusability Of Components.

Software does not wear out-

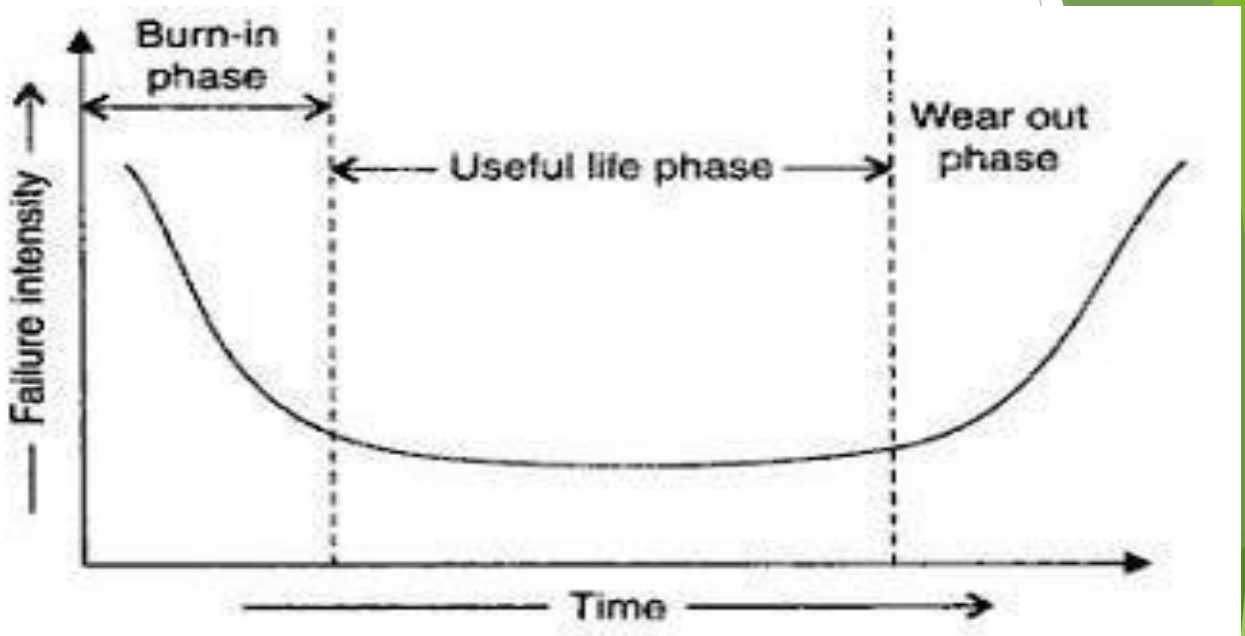


Fig. 1.5: Bath tub curve.

Software does not wear out-

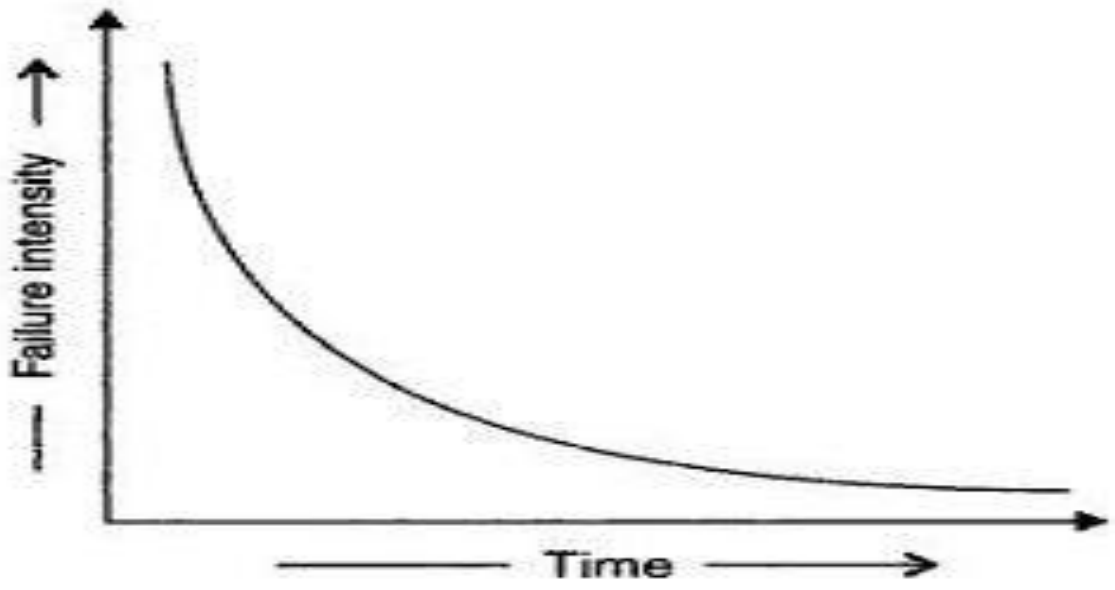
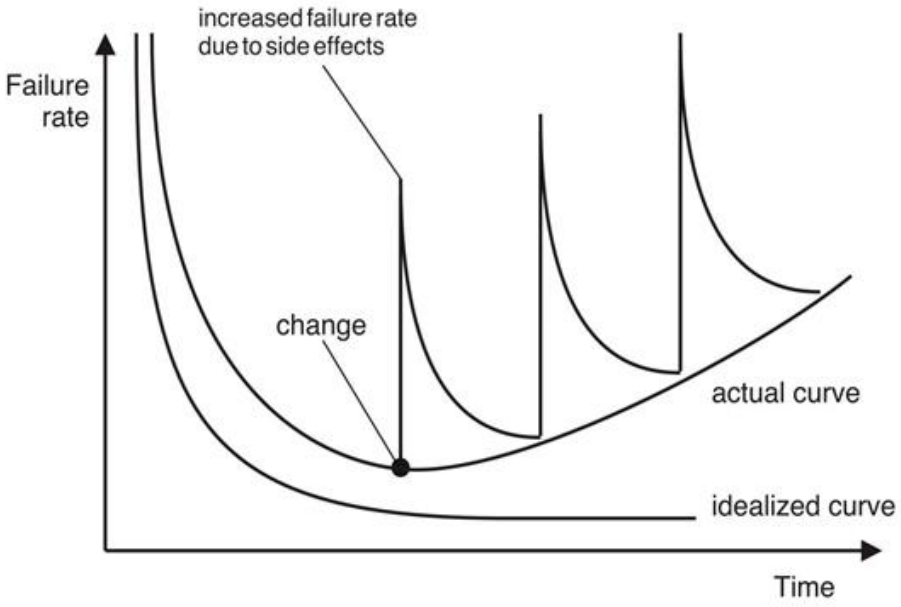
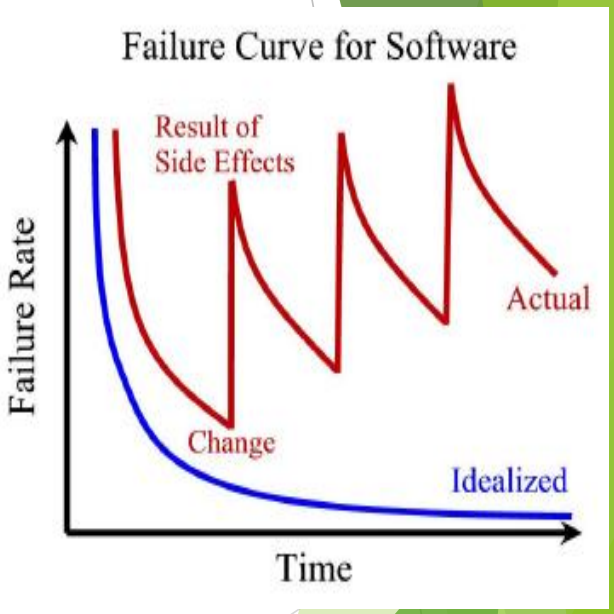


Fig. 1.6: Software curve.

Wear vs. Deterioration

Software deteriorates (Become worse) over time

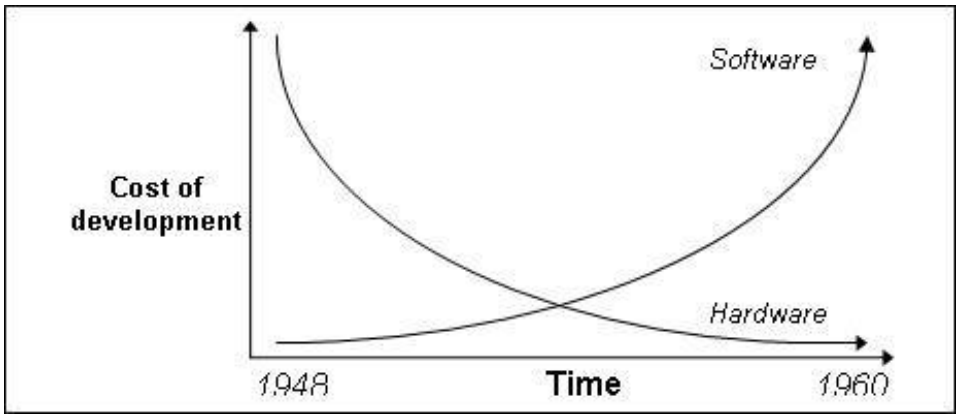




Software Crisis

The software crisis means the decisive time or turning point that software developers encounter during software development. During software development people face many problems such as development of software on time, development within the budget or it should be as per the given requirements. And if these conditions do not get fulfilled then the usefulness of the software deteriorates.

Software crisis is a term used in the early days of computing science for the difficulty of writing useful and efficient computer programs in the **required time**.



Problems faced by Customers due to S/W Crisis

- Schedule and cost estimates are often grossly inaccurate.
- The "productivity" of software people hasn't kept pace with the demand for their services.
- The quality of software is sometimes less than adequate.
- With no solid indication of productivity, we can't accurately evaluate the efficiency of new, tools, methods, or standards.
- Communication between the customer and software developer is often poor.
- Software maintenance tasks devour the majority of all software funds.

Software Crisis

S/W crisis from programmer point of view:-

i)Problem of compatibility.
ii)Problem of Portability.
iii)Proclaiming documentation.
iv)Problem of pirated s/w.
v)Problem in co-ordination of work of different people.
vi)Problem of proper maintenance.

S/W crisis from user point of view:-

i) s/w cost is very high.
ii)Price of h/w grows down.
iii)Lack of development specification.
iv)Problem of different s/w versions.
v)Problem of bugs or errors.

Causes and Solution of S/W Crisis

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

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

The **main cause** is that improvements in computing power had outpaced the ability of programmers to effectively utilize those capabilities.

Possible Solution:

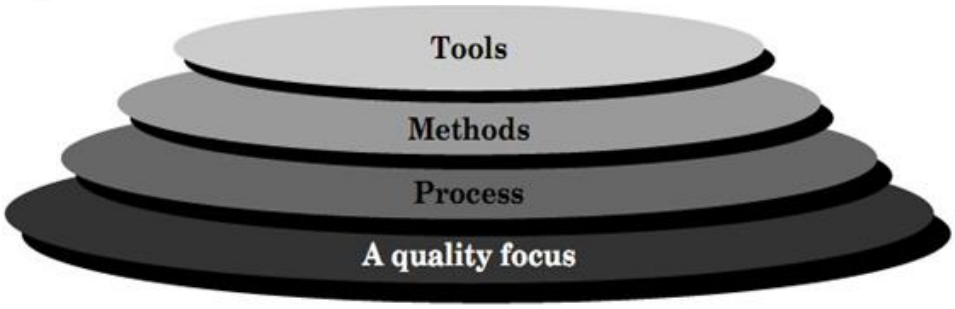
Various processes and methodologies have been developed over the last few decades to improve software quality management such as procedural programming and object-oriented programming. However software projects that are large, complicated, poorly specified, and involve unfamiliar aspects, are still vulnerable to large, unanticipated problems.

Software Engg: Layered Approach

Software engineering is a layered technology. Any software can be developed using these layered approaches. Various layers on which the technology is based are quality focus layer, process layer, methods layer, tools layer.

- A disciplined quality management is a backbone of software engineering technology.
- Process layer is a foundation of software engineering. Basically, process defines the framework for timely delivery of software.
- In method layer the actual method of implementation is carried out with the help of requirement analysis, designing, coding using desired programming constructs and testing.
- Software tools are used to bring automation in software development process.

Thus software engineering is a combination of process, methods, and tools for development of quality software.



Goals Of Software Engg

1. **Satisfy users requirements** - Many programmers simply don't do what the end user wants because they do not understand user requirements. Hence it becomes necessary to understand the demand of end user and accordingly software should be developed.
2. **High reliability** - Mistakes or bugs in a program can be expensive in terms of human lives, money, and customer relation. For instance Microsoft has faced many problems because earlier release of Windows has many problems. Thus software should be delivered only if high reliability is achieved.
3. **Low maintenance costs** - Maintenance of software is an activity that can be done only after delivering the software to the customer. Any small change in software should not cause restructuring of whole software. This indicates that the design of software has poor quality.
4. **Delivery on time** - It is very difficult to predict the exact time on which the software can be completed. But a systematic development of software can lead to meet the given deadline.
5. **Low production costs** - The software product should be cost effective.
6. **High performance** - The high performance software are expected to achieve optimization in speed and memory usage.
7. **Ease of reuse** - Use same software in different systems and software.

Challenges In Software Engg

The key challenges facing software engineering are :

- Coping with legacy systems

  Old, valuable systems must be maintained and updated. Hardware is evolved faster than software. If original developer have moved on managing, maintaining or integrating of software becomes a critical issue.

- Heterogeneity challenge

  Sometimes systems are distributed and include a mix of hardware and software. This implies that software systems must cleanly integrate with other different software systems, built by different organizations and teams using different hardware and software platforms.

- Delivery times challenge

  There is increasing pressure for faster delivery of software. As the complexity of systems that we develop increases, this challenge becomes harder.

As software is an integral part of computer based systems, it is essential to apply software engineering principles and practices while developing software. Hence the main objective of software engineering is to adopt systematic, disciplined approach while building high quality software.

Software Engg. Process

- Sequence of steps required to develop or maintain software.
- A Software process is the related set of activities and processes that are involved in developing and evolving a Software system.
- A set of activities whose goal is the development or evolution of Software.
- A Software process is a set of activities and associated results, which produce a Software product.
- A process defines who is doing what, when and how to reach a certain goal.

Fundamental Software Process Activities

Common to All Software process-

- **Software Specifications-**

  The functionality of the Software and constraints on its operation must be defined.

- **Software Development-**

  Software that meets the specifications must be produced.

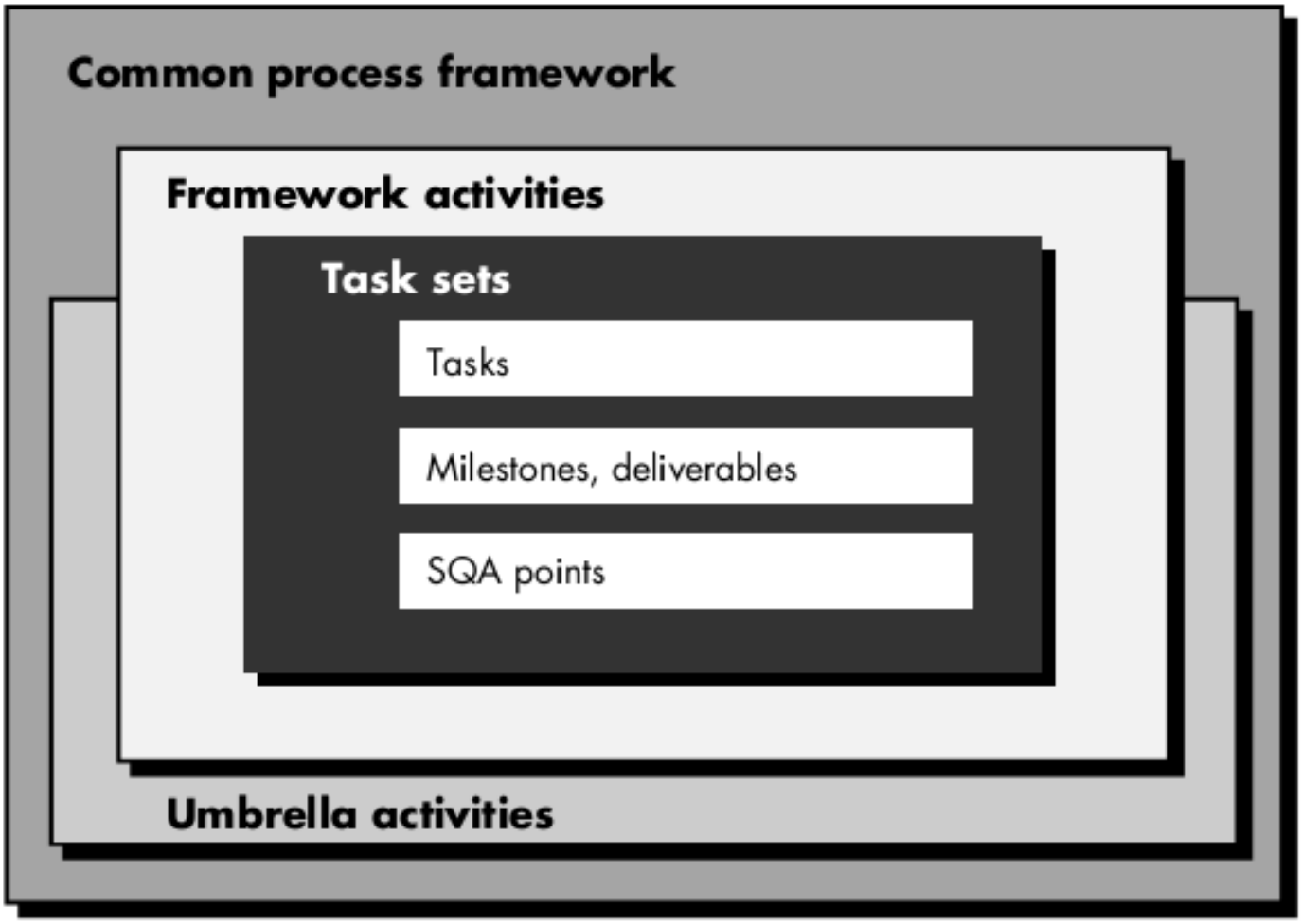
- **Software Validation-**

  The Software must be validated to ensure that it does what the customer wants.

- **Software Evaluation-**

  The Software must be develop to meet changing customer needs.

Software Process Framework



Software Engg.Process

Five Generic Common Process Framework activities are-

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction/Development
  - Code generation
  - Testing
- Deployment

Umbrella Activities (after deployment)

- Software project tracking and control.
- Formal technical reviews.
- Software quality assurance.
- Software configuration management.
- Work product preparation and production (activities to create models, documents, logs, forms, lists, etc.).
- Reusability management (defines criteria for work product reuse and establish mechanisms to achieve component reuse).
- Measurement.
- Risk management.

Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished
  - A list of the work products to be produced
  - A list of the quality assurance filters to be applied

Process Assessment

- **SPICE (ISO/IE15504) standard** defines a set of requirements for process assessment

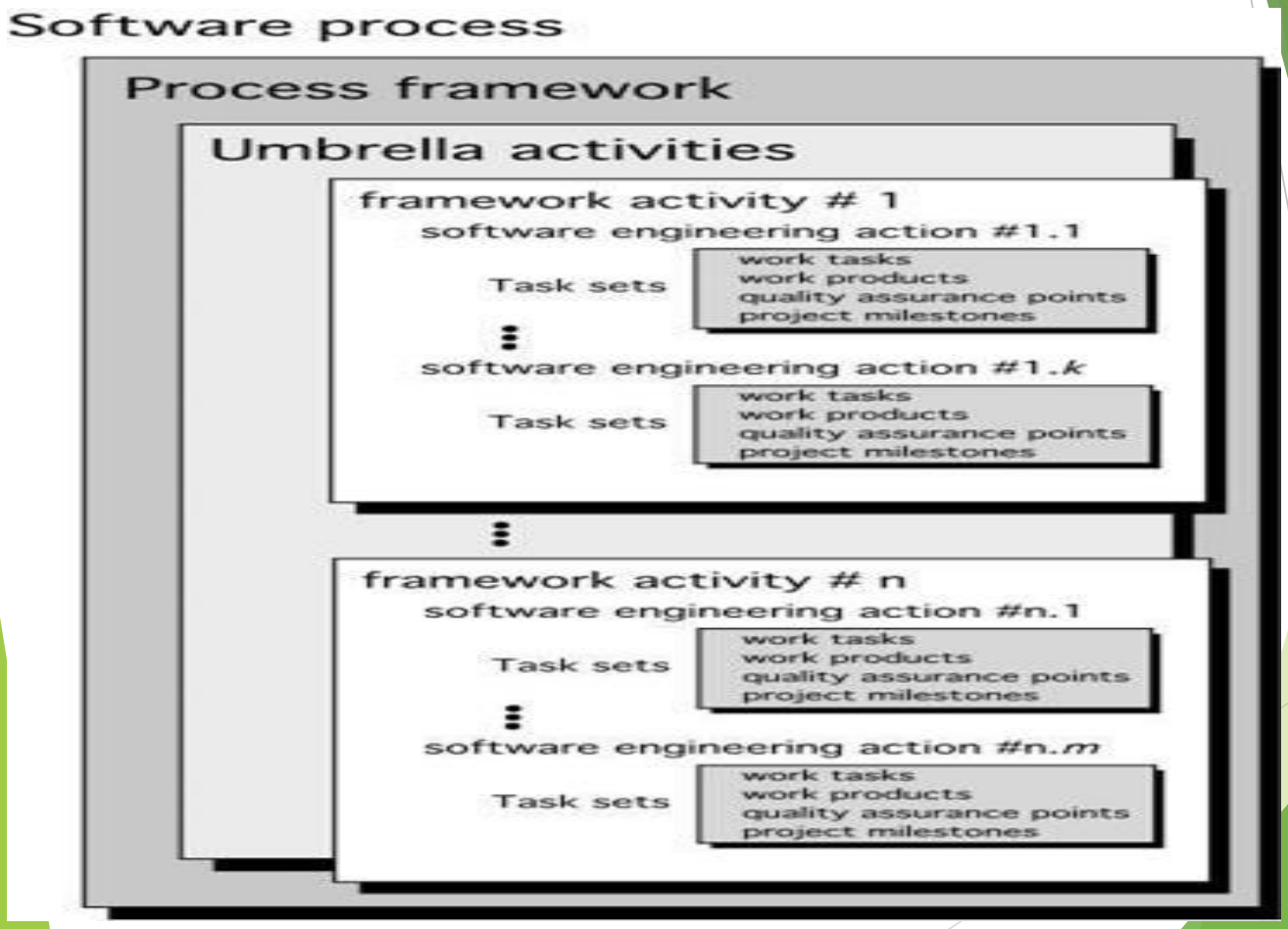
- **ISO 9001:2000** for Software defines requirements for a quality management system that will produce higher quality products and improve customer satisfaction

Software Process Characteristic

Software Process Characteristics are-

- Understandability
- Visibility
- Robustness
- Reliability
- Acceptability
- Maintainability
- Rapidity
- Supportability

A Generic Process Model

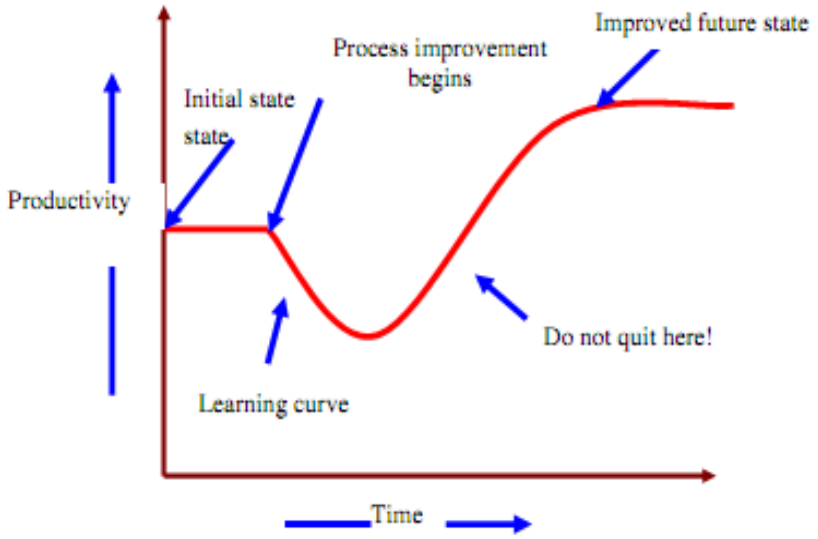


Software Process

The software process is the way in which we produce software.

Why is it difficult to improve software process ?

- Not enough time
- Lack of knowledge
- Wrong motivations
- Insufficient commitment



Software Process Models

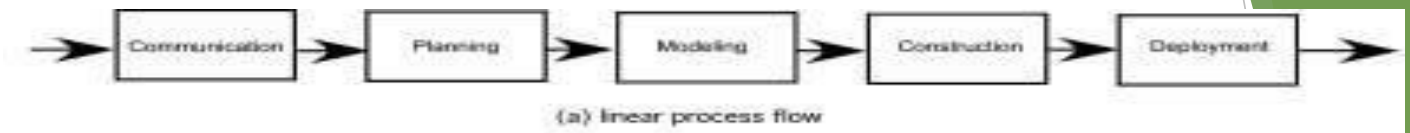
Based on s/w process, s/w products and the role of people involve in s/w engg.

1. Work Flow model
2. Data Flow model
3. Role/Action Model

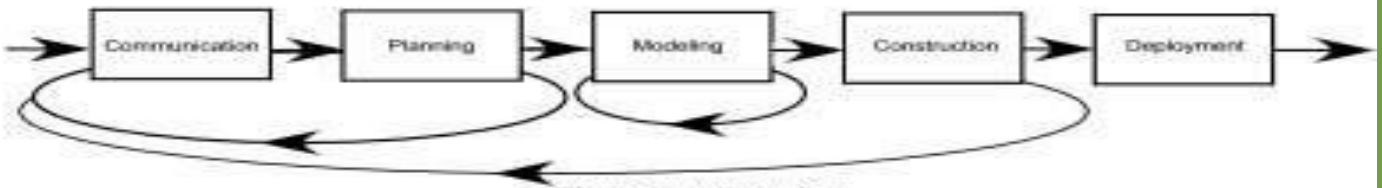
Most s/w process model are based on one of three general approaches of s/w development:

1. Waterfall approach
2. Iterative development approach
3. Computer Based S/W Engg. (CBSE) approach

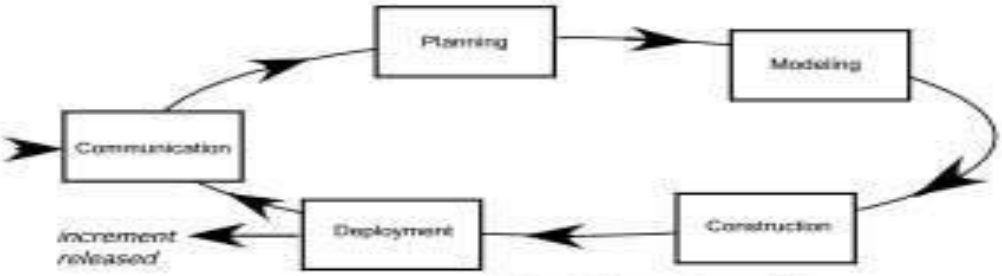
Process Flow



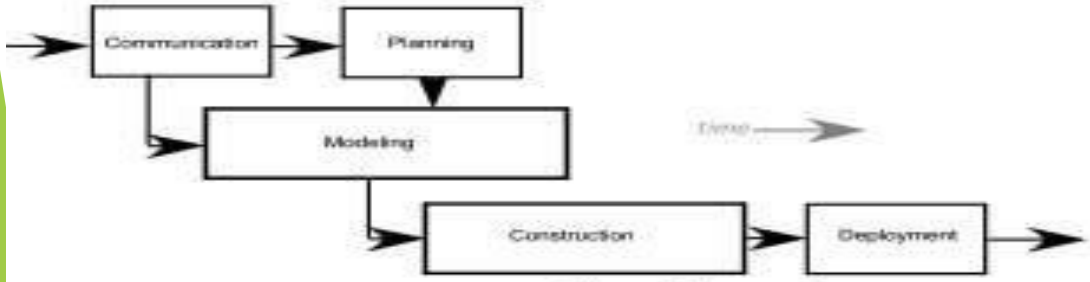
(a) linear process flow



(b) iterative process flow



(c) evolutionary process flow



(d) parallel process flow

Difference between Conventional and Software Engg

Issue	S/W Engg.	Conventional Engg.
Foundation	Based on CS, IS & Discrete maths.	Based on science,maths & emperical knowledge.
Cost	Project cost is higher than h/w.	Some projects material cost is high rather than manufacturing cost
Replication	Copy/paste, downloading, CD/DVD possible	Not possible, every time create new unit
Innovation	New and untested elements may be applied	Apply known and tested principles
Duration	Project lies for some years or decade	Projects endures for centuries e.g. bridges
Blame	Engineers blame themselves for problems	Engineers can blame to others
Age	SE is about 50 years old	CE is thousands of years old
Domain	Uses limited no. of concepts	Uses unlimited no. of concepts
Research	Research the unknown	Apply research result in known

Difference between Conventional and Software Engg.

Software Engineering: Abstract Design —→ Abstract Code

Conventional Engineering: Abstract Design —→ Concrete Products

Some Software failures

Ariane 5

It took the European Space Agency **10 years and $7 billion** to produce Ariane 5, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

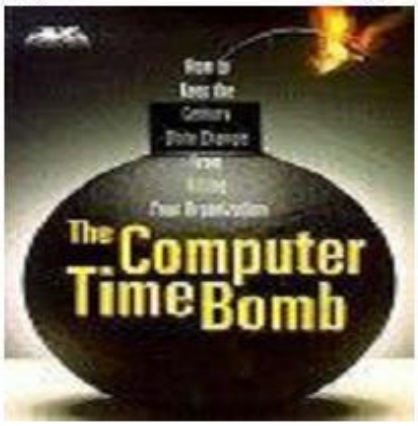
The rocket was destroyed after 39 seconds of its launch, at an altitude of two and a half miles along with its payload of four expensive and uninsured scientific satellites.

Some Software failures

Y2K problem:

It was simply the ignorance about the adequacy or otherwise of using only last two digits of the year.

The 4-digit date format, like 1964, was shortened to 2-digit format, like 64.



Some Software failures

The Patriot Missile

- First time used in Gulf war
- Used as a defense from Iraqi Scud missiles
- Failed several times including one that killed 28 US soldiers in Dhahran, Saudi Arabia

Reasons:

A small timing error in the system's clock accumulated to the point that after 14 hours, the tracking system was no longer accurate. In the Dhahran attack, the system had been operating for more than 100 hours.

Some Terminologies

- Deliverables and Milestones.
- Product and Process.
- Measures, Metrics, and Indicators.
- Feasibility.
- Requirements.
- Requirement elicitation.
- Software quality assurance.
- Cost analysis.
- Scheduling.

Deliverables and Milestones-

- Deliverables can be agreements or evaluations, normally deliverables are **objects**, such as **source code or user manuals.**

- Milestones are events that can be **used for telling the status of the project**. For example, the event of **completing the user manual** could be a milestone.

Product and Process-

- What is delivered to the customer is called the product. It may include source-code specification documents, manuals, documentation, etc. Basically, it is nothing but a set of deliverables only.
- Process is the way in which we produce Software. It is the collection of activities that leads to (a part of) a product. An efficient process is required to produce good quality products.
- if the process is weak, the end product will undoubtedly suffer.

Measures, Metrics, and Indicators-

- In Software engineering measures provide a **quantitative indication** of amount, dimension, capacity, or size of a given attribute of a product.
- Metrics are a **quantitative measure** of the degree to which a system, component, or process possesses a given attribute of a product.
- An indicator is a combination of metrics.

Difference between data and information

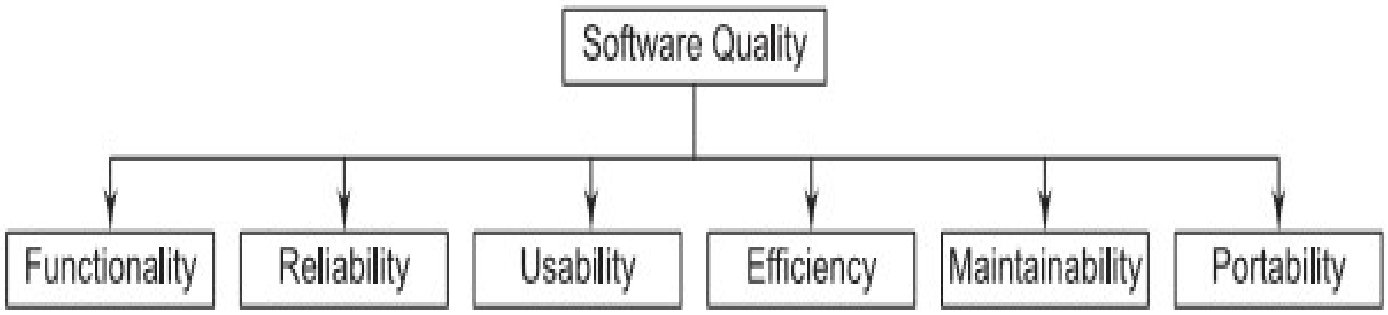
Data	Information
Data is collection of raw facts and figures.	Information is processed data.
Data is unarranged and unorganized..	Information is arranged and organized
Data is un-meaningful	Information is meaningful.
Data is input.	Information is output.
Data is used less frequently.	Information is used frequently.
Data represents facts before processing	Information represents results (after processing).
Data is not helpful in decision-making.	Information is helpful in decision-making.
Data is raw material for information.	Information is the final product of data.
Data are unprocessed records	Information contains processed records
Data is in large amounts	Information is in small amounts
If data is lost, it is very difficult (or even impossible in some situations) to re collect.	If information is lost, it is easily derived from stored data, again.
Data depends upon the sources	Information depends upon data

Feasibility, Requirements, Requirement elicitation, Software quality assurance, Cost analysis, Scheduling

- **Feasibility**—Determining if the proposed development is worth while.
- **Requirements-** Determining what functionality the Software should contain.
- **Requirement elicitation-** Obtaining the requirements from the user.
- **Software quality assurance-** Determining activities that will help ensure quality of the product.
- **Cost analysis**—Determining cost estimates.
- **Scheduling**—Building a schedule for the development.

Software Quality Attributes
[FRUEMP]

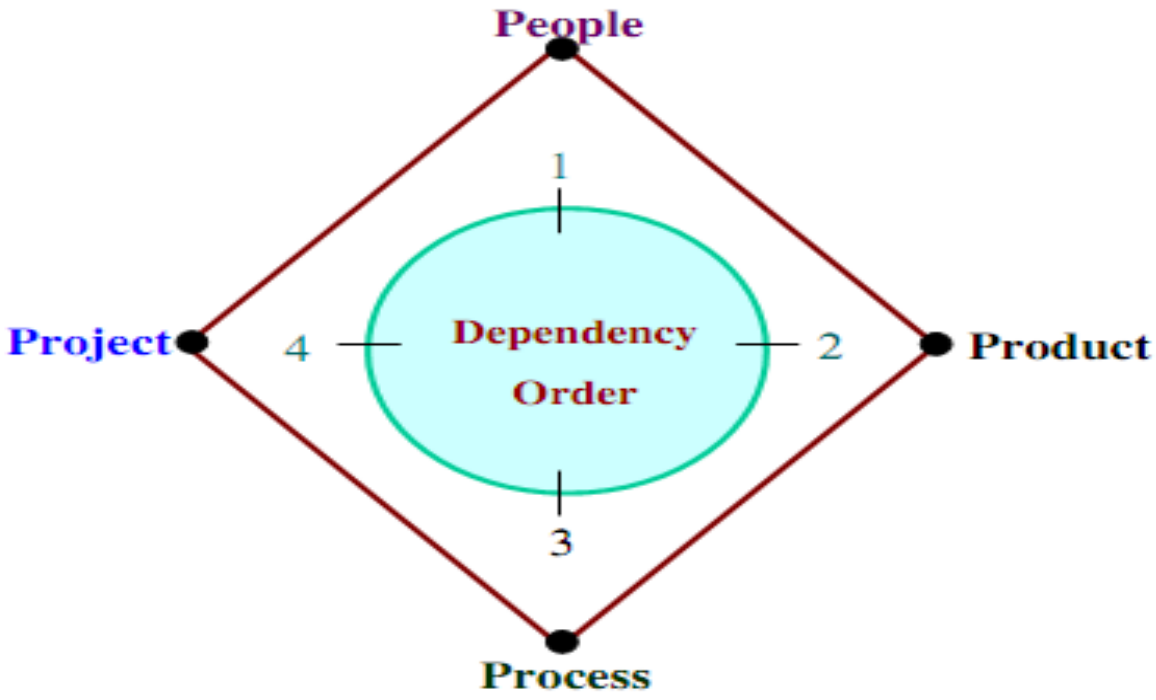
- Software quality is comprised of **six main** attributes (called characteristics)-
- for Software products, these attributes can be defined as follows.



Software Quality Attributes-

- **Functionality:** The capability to provide functions which meet stated and implied needs when the Software is used.
- **Reliability:** The capability to maintain a specified level of performance.
- **Usability:** The capability to be understood, learned, and used.
- **Efficiency:** The capability to provide appropriate performance relative to the amount of resources used.
- **Maintainability:** The capability to be modified for purposes of making corrections, improvements, or adaptation.
  - Corrective, Adaptive, Perfective.
- **Portability:** The capability to be adapted for different specified environments without applying actions or means other than those provided for this purpose in the product.

Role of Management in Software Development(4P's)



Role of Management in Software Development

- *People:* The architects, developers, testers, and their supporting management, plus users, customers, and other stakeholders are the prime movers in a software project. People are actual human beings, as opposed to the abstract construct of *workers,* which we will introduce later on.
- *Project:* The organizational element through which software development is managed. The outcome of a project is a released product.
- *Product:* Artifacts that are created during the life of the project, such as **models** (Appendix A), source code, executables, and documentation.
- *Process:* A software engineering process is a definition of the complete set of activities needed to transform users' requirements into a product. A process is a template for creating projects.
- *Tools:* Software that is used to automate the activities defined in the process.

Role of Management in Software Development