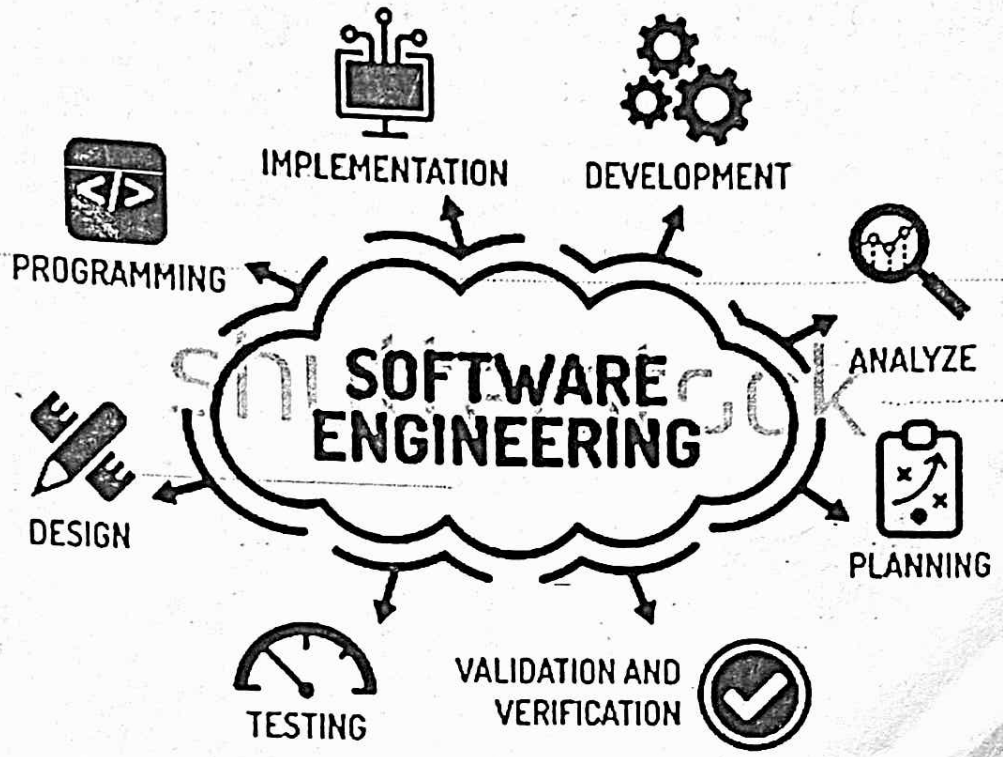
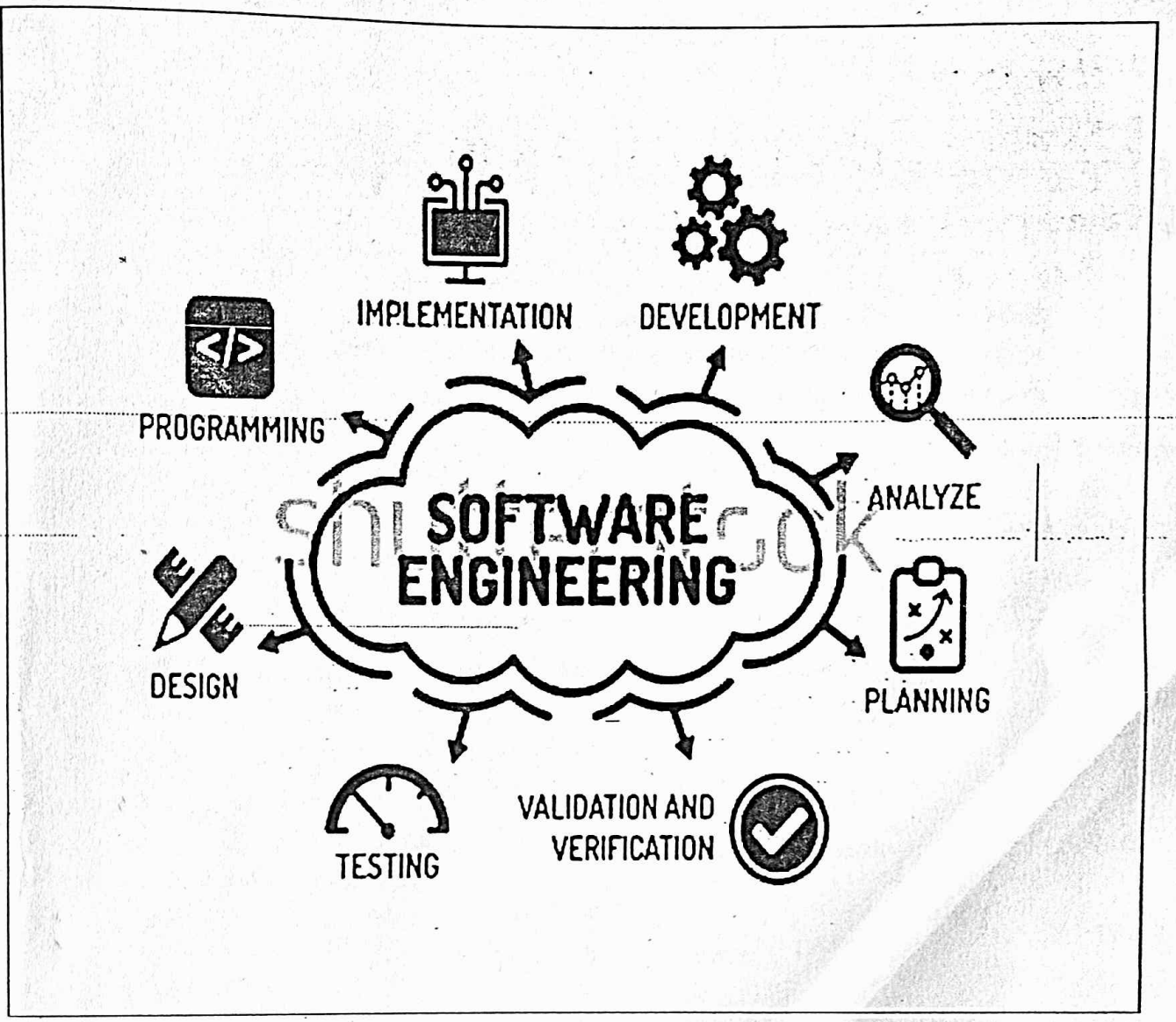
SRI HARSHINI DEGREE COLLEGE



SRI HARSHINI DEGREE COLLEGE



III B.Sc V SEMESTER

Paper VI: Software Engineering

UNIT I: INTRODUCTION:

Software Engineering Process paradigms - Project management - Process and project Metrics – software estimation - Empirical estimation models - Planning - Risk analysis - Software project scheduling.

UNIT II: REQUIREMENTS ANALYSIS:

Requirement Engineering Processes – Feasibility Study – Problem of Requirements – Software Requirement Analysis – Analysis Concepts and Principles – Analysis Process – Analysis Model

UNIT III: SOFTWARE DESIGN:

Software design - Abstraction - Modularity - Software Architecture - Effective modular design - Cohesion and Coupling - Architectural design and Procedural design - Data flow oriented design.

UNIT IV: USER INTERFACE DESIGN AND REAL TIME SYSTEMS:

User interface design - Human factors - Human computer interaction - Human - Computer Interface design - Interface design - Interface standards.

UNIT V: SOFTWARE QUALITY AND TESTING:

Software Quality Assurance - Quality metrics - Software Reliability - Software Testing - Path testing – Control Structures testing - Black Box testing - Integration, Validation and system testing - Reverse Engineering and Re-engineering. CASE tools – projects management, tools - analysis and design tools – programming tools - integration and testing tool - Case studies.

INDEX

UNIT – I

INTRODUCTION

Software Engineering Process paradigm

Q) What is Software Engineering? Explain.

Software :

software is a collection of program code(set of instructions) that enable the user to interact with a computer, its hardware, perform tasks. In other words
Software is :
and provides functionality and performance.

Engineering is developing products, using well-defined, scientific principles and methods.

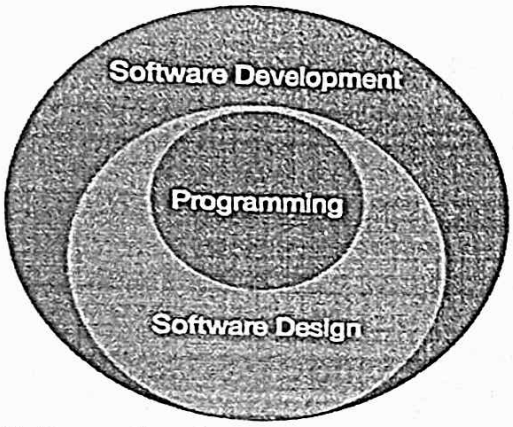
Software engineering is an engineering associated with development of software product using well-defined scientific methods ,processes and tools.

Software Evolution The process of developing a software product using software engineering principles and methods is referred to as **software evolution.** This includes the initial development of software and its maintenance and updates, till it satisfies the expected requirements.

Evolution starts from the requirement gathering process. After which developers create a prototype the software and show it to the users to get their feedback at the early stage of software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.
Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly.

Software Paradigms

Software paradigms refer to the methods and steps, which are taken while designing the software. These can be combined into various categories, though each of them is contained in one another:



Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

Software Development Paradigm

In this Paradigm all the engineering concepts are applied in development of software. It consists of –

- Requirement gathering
- Software design
- Programming

Software Design Paradigm

This paradigm is a part of Software Development and includes –.

- Design
- Maintenance
- Programming

Programming Paradigm

This paradigm is related closely to programming aspect of software development. This includes –

- Coding
- Testing
- Integration

****

Q) What are Software Myths? Explain.

A 'myth' is a statement which is not real or a belief with no logical reason. Software myths are beliefs with no logical reasons about software development. These are classified as

1. Management Myths :

Managers with software responsibilities are under pressure, due to this they may believe software myths. That include...

Myth : Already, we have a book with full of standards and procedures to build software, won't that provide the software developers with everything they need to know ?.

Reality :The book of standards is very good, but is it used ? does it reflect modern software development practice? Is it complete ? In many cases, the answer to all of these questions is 'NO'.

Myth : My people do have state-of-the-art software development tools. After all ,we have newest computers.

Reality : The high quality softnware development by using latest PCs or workstation and CASE tools are very important, but the majority of software developers still do not use them.

Myth : If it is not possible to finish the software in time, we can add more developers and finish the project.

Reality: Software development is not a mechanic process like manufacturing. People can be added, but only in a planned and well coordinated manner.

2. Customer Myths :

In many cases, the customer believes myths because software responsible managers and developers do little effort to correct misinformation.

Myth : To write programs, a general statement of objective is sufficient. Later we can fill in the details.

Reality: Poor definition is the major cause of software failure. A formal and detailed description of everything is essential.

Myth : Because the software is flexible , any changes in the project requirements can be easily accommodated.

Reality: It is true that the software requirements continually change with the needs of the customer, but the impact of changes will more on time, risk , and cost

3. Practitioner's Myths: Some myths are believed by software practitioner , that include

Myth: Our job can be finished by writing the program and working it.

Reality : Software indicates that between 50% to 70% of all effort expended on a program after it is delivered to the customer for the first time.

Myth: After running the program only, we can asses its quality.

Reality: Most effective software quality assurance mechanism can be applied from the inception of the project.

Myth: The only deliverable for a successful project is the running program.

Reality: A working program is only one part of the software project, you must configure programs, data and documents.

what is risk?

A A risk is an undisired event or circumstance that occurs while project is underway. A risk is a ~~Production~~ Potential Problem it might happen or might not happen

Q) What is the need of Software Engineering? Explain.

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

- **Large software** - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
- **Scalability**- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
- **Cost**- As huge manufacturing of hardware has lower down the price of computer and electronic hardware, the cost of software remains high if proper process is not adapted.
- **Dynamic Nature**- If the nature of software is always changing; new enhancements need to be done in the existing one. This is where software engineering plays a good role.
- **Quality Management**- Better process of software development provides better and quality software product.

Q. What are the characteristics of a software?

The software has the following characteristics that differs from the characteristics of hardware characteristics.

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

This means developing and manufacturing are fundamentally different in the context, but some similarities exists between software development and hardware manufacture , that include

- Both activities depend on the 'people' but their nature of work is different.
- Both activities for construction of a 'product' but the strategies are different.

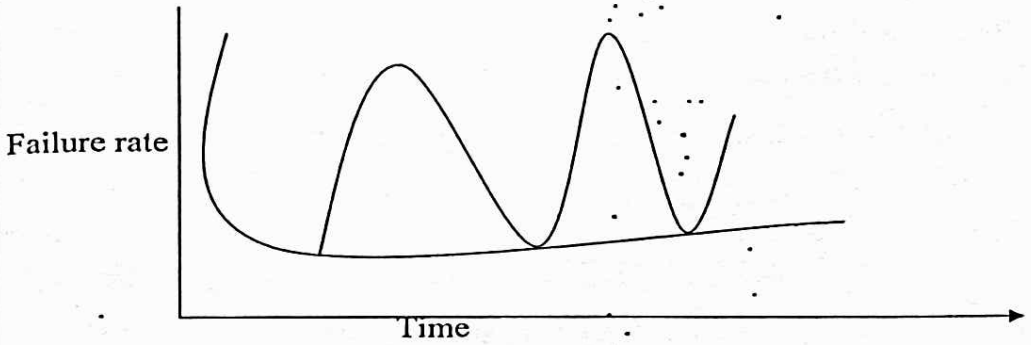
2. Software does not wear out

This means software does not degrade with time as hardware does. By drawing the graph of failure rates of hardware as a function of times , we get the 'bath tub' curve as

By the above curve, the high failure rates early in its life. As the hardware components suffer from affects of dust,vibrant, temperature extremes, power extremes, misuse and other environmental affects the hardware begins to wearout.

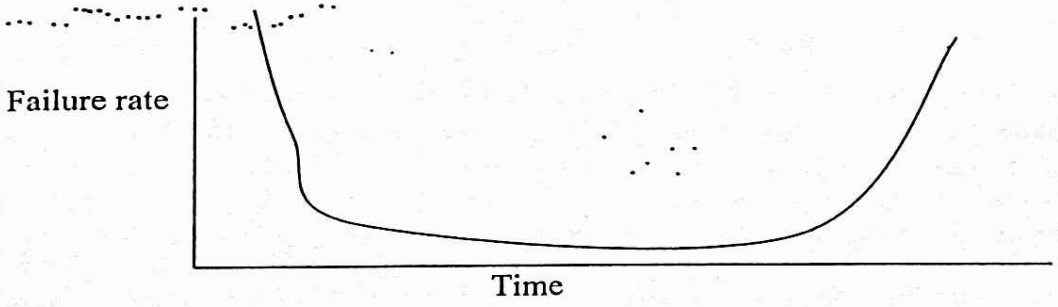
Software failure curve

The following shows the graph of software failure.



During the life software will undergoes changes, due to this changes in the software the failure rate is high at beginning, but it will be streamlined. The frequent changes causes more complex maintaining software than hardware.

Hardware Failure Curve:-

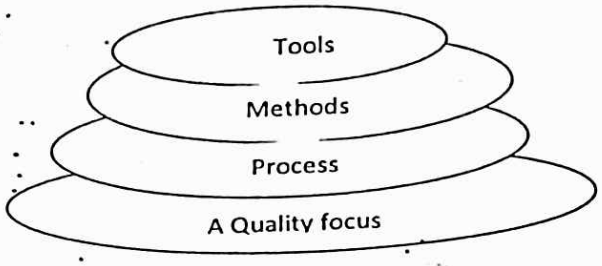


3. Software is custom built, rather than being assembled from existing component

The manufacturer of hardware uses catalogs of digital components, procedures in hardware manufacturing. But software developers are not have the digital components , procedures etc.

Q) What are different layers of software engineering ?

Software engineering is a layered technology. A software engineering approach must rest on an organizational commitment to quality. Three important layers of software engineering are Process, Methods and Tools.



Process :

Process defines a framework for a key process areas that are necessary for effective delivery of software engineering technology. A small no, of framework activities are applicable to all software process regardless of their size. A process task sets-tasks, milestones, deliverables and quality assurance.

Method :

The methods of software provides the technical " How to's" for building software. Methods include a broad array of tasks that include requirements, analysis, design, coding, testing and maintenance.

Tools :

The Tools of software provides automated or semi automated support for the process and methods. There is a system for the support of software development called CASE(Computer Aided Software Engineering). The CASE combines software, hardware and databases about analysis, design, coding and testing.

Q) What are 'Organizational paradigm' for Software Engineering Team ?

Or

Write about software teams ?

There are four types organizational paradigms for a software engineering team that include

The 'Closed paradigm' structures a team along a traditional hierarchy of authority.They produce software that is similar to past effort they less innovative.

The 'Random paradigm' structures a team that depends on individual innovative of the team members. It is needed when innovation is reaquired. These teams may struggle when ordered performance is required.

The 'Open paradigm' structures a team associated with the closed paradigm and random paradigm. These are well suited to the solution of complex problems.

The 'Synchronous paradigm' structures a team to work on pieces of the problem with little active communication.

Project Management

Q) Explain the management spectrum ?

Or

*** Explain the 4P's of Project Management?**

The software project management focuses on the four P's: P*eople, Product, Process and Project.*

- **The People:**

The cultivation of motivated, highly skilled software people is so important.The SEI(Software Engineering Institute) has developed a People Management-Capability Maturity Model(PM-CMM) to improve software development capability. It defines Key Practice Areas(KPA) for software people recruiting, selection, performance management, training, career development ,design,team/culture development.

The IEEE published that three major companies were asked to most important contribution to a successful software project , they answered in the following way...

VP1: The most important ingredient that was successful in the project was having smart people.

VP2: I had to pick most important in our environment, I had say its not tools that we use,it's the People.

VP3: The only rule ,I have in management is to ensure I have good People.

The Players(The Stock Holders):

The software process and project is populated by players who can be classified as

1.Senior Project Managers:

They defines the business issues that have significant influence on the project.

2.Project(team) Managers:

They must plan, motivate, organize, and control the practitioners who do software work.

3.Practitioners:

They delivers the technical skills that are necessary to engineer a product or anapplication.

4.Customers

They specify the requirements for the software to engineered/

5 End Users:

They interact with the software once it is released for production.

Role of Team Leader :

Project Management is a people intensive activity, recruiting or selecting someone to lead a software Project needs some approach. There is an MOI model of leadership.

Motivation :

The ability to encourage (by push or pull) technical people to prove to their best ability.

Organization:

The ability to organize and translate an existing model to final product.

Ideas or Innovation :

The ability to encourage people to create a creative application or product.

The other key Traits of Project Manager :

Problem Solving :

An effective software project manager can diagnose the technical and organizational issues and provide solution to that problems.

Achievements :

They must demonstrate through their own activities that produce a successful project.

Team Building

They must be able to read people and construct a good team.

Management Identity :

They must have confidence to assure and allow good technical people.

The Process : \

A Process is a collection of activities, actions, and tasks that are performed when some work product is to be created. An 'activity' is to achieve a broad objective(eg. Communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity. A 'task' focus on a small ,but well-defined objectives that produce a better outcome. A process framework establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects. A process framework has the following activities.

Communication :\

It is important to communicate with customers and other stakeholders to understand the objective.

Planning;

It defines the software engineering work by describing the technical tasks to be conducted, work products to be produced and a work schedule.

Modeling :

A software engineer has to create models to understand of software requirements and design that will include those requirements.

Deployment:

The software product has to deliver to customers who evaluate and provide feedback on the evaluation.

The Product

A 'product' is a developed software. Before a project can be planned, product objective and scope should be established, alternative solutions should be considered, technical and management constraints should be identified. A software developer and other stakeholders must meet to define product objective and scope. The 'objective' identify the overall goals for the product(from the stakeholders point view), and 'scope' identifies the primary data, functions and behavior that characterizes the product. Once the product objective and scope are understood, alternative solutions are considered which enables managers and practitioners to select the best approach for the constraints, delivery deadlines, budget restrictions and other factors.

The Project:

The project includes all and everything of the total development process and to avoid project failure the manager has to take some steps, has to be concerned about some common warnings etc.

Ten Signs that the project is in Jeopardy:

- ✓ Software people don't understand their customer's needs
- ✓ The product scope is poorly defined
- ✓ Changes are managed poorly
- ✓ The chosen technology changes
- ✓ Business needs change (or are poorly defined)

- ✓ Deadlines are unrealistic
- ✓ Users are resistant
- ✓ Sponsorship is lost (or was never properly obtained)
- ✓ The project team lacks people with appropriate skills
- ✓ Managers (and practitioners) avoid best practices and lessons learned

****

Q) Write about the W^{5}HH principle ?.

Boehm suggests an approach that addresses project objectives, milestones and responsibilities, management and technical approaches and required resources, it is called W5HH principle.

- **Why** is the system being developed?
  - Assesses the validity of business reasons and justifications
- **What** will be done?
  - Establishes the task set required for the project
- **When** will it be done?
  - Establishes a project schedule
- **Who** is responsible for a function?
  - Defines the role and responsibility of each team member
- **Where** are they organizationally located?
  - Notes the organizational location of team members, customers, and other stakeholders
- **How** will the job be done technically and managerially?
  - Establishes the management and technical strategy for the project
- **How** much of each resource is needed?
  - Establishes estimates based on the answers to the previous questions.

****

Process and Project Metrics

Q) What are process and project metrics?

A 'Metric' is quantitative measure of the degree to which a system,component or process possesses an attribute. The metrics are quantitative measurement that enable you to gain insight into the efficiency of the software process and project.

Process Metrics:

Process metrics are collected across all projects for a long a period of time.Process matrics have long term impact, their intent is to improve the process itself. Grady says they are classified as private and public.

Private Metrics

Private metrics are collected on individual bases. Examples of private metrics are include defect rates(by individual),defect rates(by component) etc.

Public Metrics :

Public metrics generally a public information that originally was private to individuals and teams, Examples of public metrics are defect rates(by team) etc.

Project Metrics:

A project metrics are tactical. That is project metrics and indicators derived from them are used by a software manager and a software team to adapt project workflow and technical activities.Project metrics are used to avoid development schedule delay, to asses the quality of the product on an on-going basis.Every project should measure its inputs (resources), outputs (deliverables), and results (effectiveness of deliverables).

****

Q) Explain software measurement?

A 'measure' is quantitative indicator of an attribute. A 'measurement' is an act of measure. Software measurements can be classified as

1.Direct Measurement :

The direct measurement of the software process include cost and effort applied. Direct measures of the products includes LOC(Lines Of Code), memory size, speed etc.

2.Indirect Measurement :

The indirect measurement of the software process include functionality quality, complexity, etc.

Types of Metricies :

Size-Oriented Metrics:

These are used to minimize the development time schedule, analyzing the errors and manpower. Consider the project \'Alpha' which has 1200 LOC, Cost $165,errors 134, people worked on the project is 5 and 356 pages of document. The project \'Beta' which has 2700 LOC, Cost $440,errors 321, people worked on the project is 5 and 1224 pages of document. The project \'Gamma ' which has 2200 LOC, Cost $314,errors256, people worked on the project is 6 and 1050 pages of document.

Project	LOC	$(00.00)	pp.DOC	ERRORS	PEOPLE
Alpha	1200	165	356	134	5
Beta	2700	440	1224	321	5
Gamma	2200	314	1050	256	6

Function-Oriented Metrics:

Function oriented metrics are software metrics use a measure of the functionality delivered by the application, since the functionality cannot be measured directly, it must be derived indirectly using other measures called the FP(Function Point) are derived using an empirical relationship based on countable(direct) measurers of software information domain. That includes the following parameters.

Parameter	count		simple	avg	complex
No. of user inpus		x	3	4	5
No. of user outputs		x	4	5	7
No. of user Inquiries		x	3	4	6
No. of Files		x	7	10	15
No. of external interfaces		x	5	7	10

Count=total

To compute FP(Function Point) the following relationship is used.

FP=count-total x (0.65+0.01x Fi)

Where Fi(i=1 to 14) are complexity Adjustment values based on(numerical) responses to questions posed on the software product ,process, technologies etc.

Extended Function Point Metrics:

A number of extensions to the basic function-point measure have been proposed to make it more adequate for systems engineering applications.

The feature-point measure is a superset of the basic function-point measure especially for application requiring complex software.

****

8. What are the metrics for software quality?

A. The goal of software engineering is to produce a high quality system, application or product. To achieve this goal software engineers must apply effective methods with modern tools.

The quality factors:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- portability

The overview of factors that affect quality: The set of quality factors assess software from three distinct points of view:

1. Product operation
2. Product revision
3. Product transition.

Measuring quality: The following are the some of the measures of quality

- *Correctness*: Correctness is the degree of which the software performs its required function. The measure for correctness is *defects per KLOC*, where defect is the lack of conformance to requirements.
- *Maintainability*: It is the ease with which a program can be corrected if an error is encountered, adapted if its environment.
- *Integrity*: It is very important in the age of hackers and firewalls. To measure this 2 additional attributes required:
  - **Threat:** Is the probability that an attack occurs within a given time.
  - **Security:** Is the probability that an attack will be repelled.
- *Usability*: It is an attempt to express *user friendliness* and can be measured in terms of four characteristics: (1) The physical skill required to learn the system (2) The time required to become efficient in the use of the system (3) The net increase in productivity measured when the system is used by someone who is efficient (4) The subjective assessment of users attitudes toward the system.

Defect Removal Efficiency: The **defect removal efficiency** (DRE) gives a measure of the development team ability to **remove defects** prior to release. It is calculated as a ratio of **defects** resolved to total number of **defects** found.

Calculation:

To be able to calculate that metric, it is important that in your defect tracking system you track:

affected version, version of software in which this defect was found.

release date, date when version was released

DRE = Number of defects resolved by the development team / total number of defects at the moment of measurement.

DRE is typically measured at the moment of version release, the best visualization is just to show current value of DRE as a number.

Example:For example, suppose that 100 defects were found during QA/testing stage and 84 defects were resolved by the development team at the moment of measurement. The DRE would be calculated as 84 divided by 100 = 84%

****

Software Estimation

9. What is software project estimation? (Or) What is software estimation?

Now a days, software is most expensive, a large cost estimation error can bring the difference between profit and loss. Software cost and effort estimation involves too many variables – human, technical, environmental - can affect the ultimate cost of software. To achieve reliable cost and effort estimates, a number of options arise:

1. Delay estimation until late in the project.
2. Base estimates on similar projects that have already been completed.
3. Use simple decomposition techniques to generate project cost and effort estimation.
4. Use one or more empirical models for software cost and effort estimation.

a. The first option is attractive but not practical. Cost estimates must be provided "up front".

b. The second option works well, but past experiences has not always good indicator of future results.The remaining options are viable approaches for software estimation. *Decomposition techniques* take "divide and conquer" approach. *Empirical models* can be used to complement decomposition techniques.

10. Explain decomposition techniques.

Software project estimation is a form of problem solving, to make it easy, we decompose the problem as a set of similar problems. These are classified as

Software Sizing: Before an estimate can be made, the project planner must understand the scope of the software to be built and generate an estimate of its "size.

In the context of project planning, size refers to a quantifiable outcome of the software project. If a direct approach is taken, size can be measured in LOC. If an

indirect approach is chosen, size is represented as FP. There are four different approaches to the sizing problem:

Fuzzy logic" sizing: This approach uses the approximate reasoning techniques that are the important concepts of fuzzy logic.

Function point sizing: The planner develops estimates of the information domain characteristics.

Standard component sizing: The approach uses different standard components that are generic to a particular application area.

Change sizing: This approach is used when a project must be modified in some way as a part of project.

Problem based Estimation:

Lines of code and function points were described as measures from which productivity metrics can be computed. LOC and FP data are used in two ways during software project estimation: (1) as an estimation variable to "size" each element of the software and (2) as baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections LOC and FP estimation are distinct estimation techniques.

Yet both have a number of characteristics in common. The project planner begins with a bounded statement of software scope and from this statement attempts to decompose software into problem functions that can each be estimated individually. LOC or FP (the estimation variable) is then estimated for each function.

Process based Estimation:

The most common technique for estimating a project is to base the estimate on the process that will be used. That is, the process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated. Like the problem-based techniques, process-based estimation begins with a delineation of software functions obtained from the project scope.

A series of software process activities must be performed for each function. Functions and related software process activities may be represented as part of a table similar to the one presented. Once problem functions and process activities are melded, the planner estimates the effort (e.g., person-months) that will be required to accomplish each software process activity for each software function. Senior staff heavily involved in early activities is generally more expensive than junior staff involved in later design tasks, code generation.

****

Empirical Estimation Models

11. Explain empirical estimation models.

A. An *estimation model* for computer software uses empirically derived formulas to predict effort as a function of LOC and FP. The empirical data that support most estimation models are derived from a limited sample of projects.

The structure of estimation models: A typical estimation model is derived using regression analysis on data collected from past projects. The structure of such models takes the form

$$E = A + B \times (ev)^{C}$$

Where A, B and C are empirically derived constants, E is the effort in person-months, and ev is the estimation variable.

LOC oriented models

$E = 5.2 * (KLOC)^{0.91}$	Walston-Felix model
$E = 5.5 + 0.73 * (KLOC)^{1.16}$	Bailey-Basili model
$E = 3.2 * (KLOC)^{1.05}$	Boehm simple model
$E = 5.288 * (KLOC)^{1.047}$	Doty model

FP oriented models

$E = -13.39 + 0.0545 * FP$	Albrecht and Gaffney model
$E = 60.62 * 7.728 * 10^{-8} * FP^{3}$	Kemerer model
$E = 585.7 + 15.12\ FP$	Matson, Barnett, Mellichamp

The COCOMO model: The Constructive Cost **Model** (**COCOMO**) is a procedural software cost estimation **model** developed by Barry W. Boehm. The **model** parameters are derived from fitting a regression formula using data from historical projects.

The hierarchy of COCOMO models takes the following form:

Model 1. The Basic COCOMO model is a static, single-valued model that computes software development effort (and cost) as a function of program size expressed in estimated lines of code (LOC).

Model 2. The Intermediate COCOMO model computes software development effort as a function of program size and a set of "cost drivers" that include subjective assessments of product, hardware, personnel and project attributes.
Model 3. The Advanced COCOMO model incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

The COCOMO models are defined for three classes of software projects. Using Boehm's terminology these are:

Organic mode–relatively small, simple software project in which small teams with good application experience.

Semi-detached mode –an intermediate software project in which teams mixed experience levels.

Embedded mode –a software project that must be developed within a set of tight hardware, software and operational constraints.

The Basic COCOMO equations take the form:

$$E = a_b \, KLOC^{\,b_b}$$

$$D = c_b \, E^{\,d_b}$$

where E is the effort applied in person-months, D is the development time in chronological months and KLOC is the estimated number of delivered lines of code for the project (express in thousands). The coefficients a_b and c_b and the exponents b_b and d_b.

The coefficients a_b and c_b, and exponents b_b and d_b are given by

Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.33

The intermediate COCOMO model takes the form:

$$E = a_i \, KLOC^{\,b_i} \times EAF$$

where E is the effort applied in person-months and *KLOC* is the estimated number of delivered lines of code for the project. The coefficient a_i and the exponent b_i.th

Planning

12. What are the objectives of project planning?

Planning:

The objective of software project planning is to provide a framework that enables the manager to make reasonable estimates of resources, cost and schedule. The planning objective is achieved through a process of information discovery that leads to reasonable estimates. The activities that are associated with software project planning.

Tasks sets for project planning :

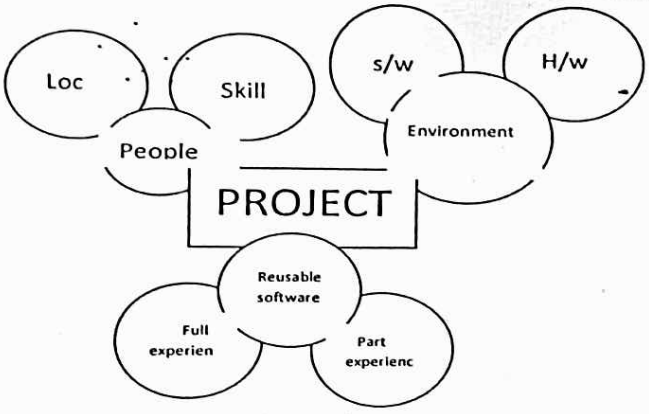
Software scope and feasibility:

The first activity in the software project planning is the determination of software scope. Software scope describes the functions and features that are to be delivered to end users. Scope is defined using one of two techniques.

- A narrative description of software scope is developed after communication with all stakeholders.
- A set of use cases are developed by users.

Feasibility: Once scope has been identified, it is reasonable to ask: "Can we build software to meet this scope? Is the project feasible?"

Human Resources: The second software planning task is estimation of the resources required to accomplish the software development effort. There are three types of resources:



Human Resources: The planner begins by evaluating scope and selecting the skills required to complete development. Organizational position and specialty are specified. The number of people required for a software project can be determined only after an estimate of development effort is made.

Reusable Software Resources: Software engineering emphasizes reusability, i.e the creation and reuse of software building blocks. These building blocks often called as *components*.

Environmental Resources: The environment that supports the software project, often called the *software engineering environment* (SEE), incorporates hardware and software.

****

Risk Analysis

13. What is the difference between proactive and reactive risk management?

A. We will look at the differences between the two risk management approaches.

Definition of Proactive and Reactive Risk Management

Reactive: "A response based risk management approach, which is dependent on accident evaluation and audit based findings."

Proactive: "Adaptive, closed loop feedback control strategy based on measurement, observation of the present safety level and planned explicit target safety level with a creative intellectuality."

Purpose of Proactive and Reactive Risk Management

Reactive risk management: Reactive risk management attempts to reduce the tendency of the same or similar accidents which happened in past being repeated in future.

Proactive risk management: Proactive risk management attempts to reduce the tendency of any accident happening in future by identifying the boundaries of activities, where a breach of the boundary can lead to an accident.

Features of Proactive and Reactive Risk Management

Timeframe

Reactive risk management: Reactive risk management solely depends on past accidental analysis and response.

Proactive risk management: Proactive risk management combines a mixed method of past, present and future prediction before finding solutions to avoid risks.

Flexibility

Reactive risk management: Reactive risk management does not accommodate prediction, creativity, and problem-solving ability of humans in its approach which makes it less flexible to changes and challenges.

Proactive risk management: Proactive risk management includes creative thinking, prediction. Further, it principally depends on the accident source to reduce the accident which is a human attribute. So, this lets it be very adaptive to changing environment.

****

14. What are the types of Software Risks?

A. Software risk encompasses the probability of occurrence for uncertain events and their potential for loss within an organization. The following are some of the risks related to project, product, and business risks.

- **Project Risks: Project risk** is an uncertain event or condition that, if it occurs, has an effect on at least one **project** objective. **Risk** management focuses on identifying and assessing the **risks** to the **project** and managing those **risks** to minimize the impact on the **project**.
- **Technical Risks:** The probability of loss incurred through the execution of a **technical** process in which the outcome is uncertain. Untested engineering, technological or manufacturing procedures entail some level **technical risk** that can result in the loss of time, resources, and possibly harm to individuals and facilities.
- **Business Risks: Business risk** is the possibilities a **company** will have lower than anticipated profits or experience a loss rather than taking a profit. ... A **company** with a higher **business risk** should choose a capital structure that has a lower debt ratio to ensure it can meet its financial obligations at all times.
- **Known Risks:** Known risks are those that can be uncovered after careful evaluation of the project, the plan, the business and technical environment in which the project is being developed.
- **Predictable Risks: Predictable** risks are extrapolated from past project experience.
- **Unpredictable Risks:** These are the risks, which are extremely difficult to identify in advance.

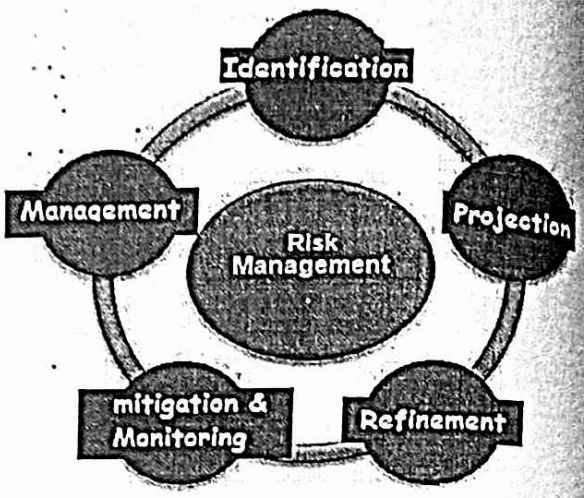
****

15. Explain Risk Management Process.

A. Risk management is the identification, projection, refinement and management of **risks.**

Risk Identification: Risk identification is a systematic attempt to specify threats to the project plan. By identifying known and predictable risks, the project manager takes a first step toward avoiding them. One method for identifying risks is to create a *risk item checklist*. It can be used for risk identification and focuses on some subset of known and predictable risks in the following categories:

- *Product Size*
- *Business Impact*
- *Customer Characteristics*
- *Process Definition*
- *Development Environment*
- *Technology to be build*
- *Staff size and experience*



Risk Projection: Risk projection, also called risk estimation, attempts to rate each risk. The project planner, along with other managers and technical staff, performs four risk projection activities:

- Establishes a scale that understands the probability of the risk
- Describes the consequences of the risk
- Estimate the impact of the risk on the project and on the product
- Identifies the overall accuracy of the risk projection.

Risk Refinement: Risk refinement is the Process of restating the **risks** as a set of more detailed **risks** that will be easier to mitigate, monitor, and manage. **CTC (condition-transition-consequence)** format may be a good representation for the detailed **risks.**

Risk Mitigation, Monitoring and Management: Risk analysis activities are used to assist the project team in developing a strategy for dealing with risk. An effective strategy must consider three issues:

- Risk Avoidance – Leads to mitigation

- Risk Monitoring – The project manager monitors factors those indicate whether the risk is being more or less likely.
- Risk Management – It assumes that mitigation efforts have failed and the risk has become a reality.

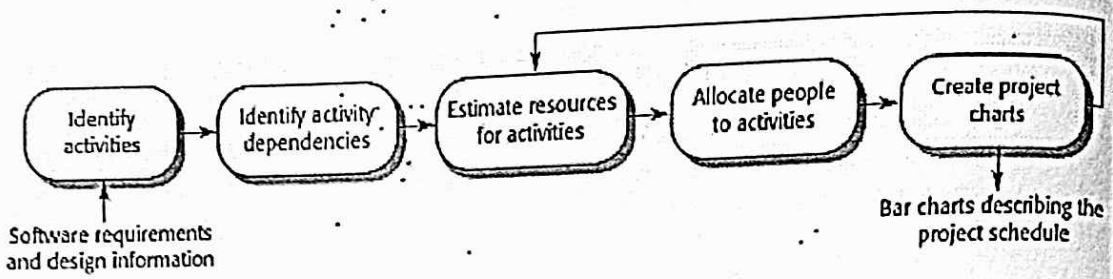
****

Software Project Scheduling

16. Explain Project scheduling. (Or) Explain Software Project Scheduling.

A. Project scheduling is one of the most difficult process for project manager. Managers estimate the time and resources required to complete activities and organize them into coherent (systematic) sequence. Schedules must be continually updated as better progress.

Project scheduling involves separating the total work into separate activities and judging the time required to complete these activities. Some of these activities are carried out in parallel.



****

17. What is Project scheduling? Explain different techniques for project scheduling.

A. Project Scheduling

Project scheduling is concerned with the techniques that can be employed to manage the activities that need to be undertaken during the development of a project.

Scheduling is carried out in advance of the project commencing and involves:

- identifying the tasks that need to be carried out;
- estimating how long they will take;
- allocating resources (mainly personnel);
- scheduling when the tasks will occur.

Once the project is underway control needs to be exerted to ensure that the plan continues to represent the best prediction of what will occur in the future:

- based on what occurs during the development;

- often necessitates revision of the plan.

Effective project planning will help to ensure that the systems are delivered:

- within cost;
- within the time constraint;
- to a specific standard of quality.

Two project scheduling techniques will be presented, the Milestone Chart (or Gantt Chart) and the Activity Network.

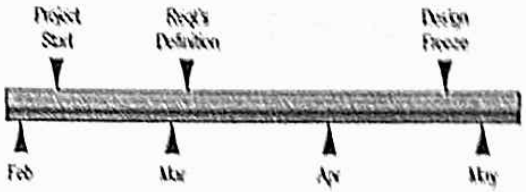
Milestone Charts:

Milestones mark significant events in the life of a project, usually critical activities which must be achieved on time to avoid delay in the project.

Milestones should be truely significant and be reasonable in terms of deadlines (avoid using intermediate stages).

Examples include:

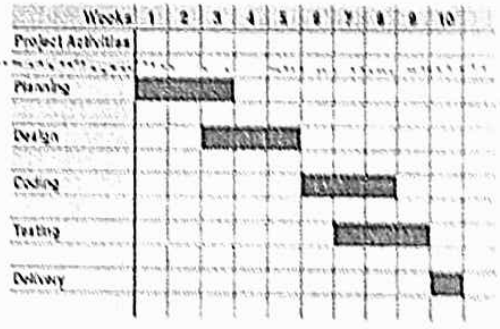
- installation of equipment;
- completion of phases;
- file conversion;
- cutover to the new system



Gantt Charts:

A Gantt chart is a horizontal bar or line chart which will commonly include the following features:

- activities identified on the left hand side;
- time scale is drawn on the top (or bottom) of the chart;

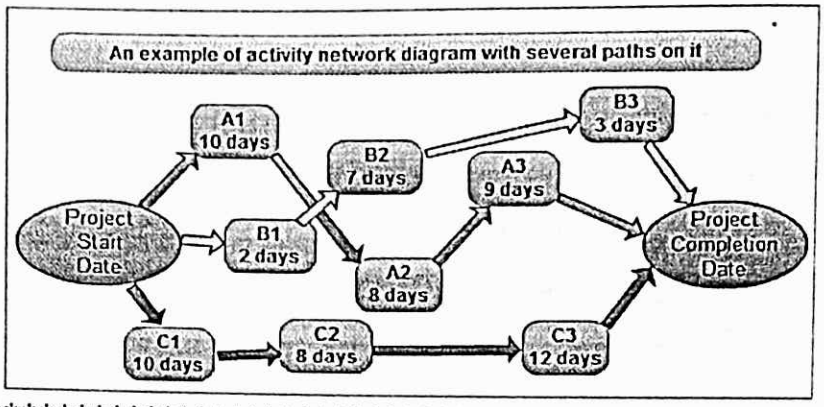


Activity Networks:

It is a technique for evaluating the performance of large development projects, which became known as PERT - *Project Evaluation and Review Technique*. Other variations

of the same approach are known as the critical path method (CPM) or critical path analysis (CPA).

The heart of any PERT chart is a network of tasks needed to complete a project, showing the order in which the tasks need to be completed and the dependencies between them. This is represented graphically:



****