

Software Engineering Notes

Contributor: Tanya Garg
KMV CS(H)

Computer Science Notes

Download FREE Computer Science Notes, Programs, Projects, Books for any university student of BCA, MCA, B.Sc, M.Sc, B.Tech CSE, M.Tech at
<https://www.tutorialsduniya.com>

Please Share these Notes with your Friends as well

facebook



2-01-18

| | |
|----------|--|
| Page No. | |
| Date | |

INTRODUCTION

- provides theoretical background and some procedures in order to find end product.

Software → application made for a specific task & is user interactive.
↓

program that may or include a user interface and may have a database.

Engineering → to develop software.
↳ defines some procedures.

(end product)

• A computer software is a product that software professional built and support over a long term.
It includes programs that execute within a machine of any size, architecture and content.

✓ Software Engineering encompasses a process, a collection of methods and an array of tools that allows a programmer to build high quality software.

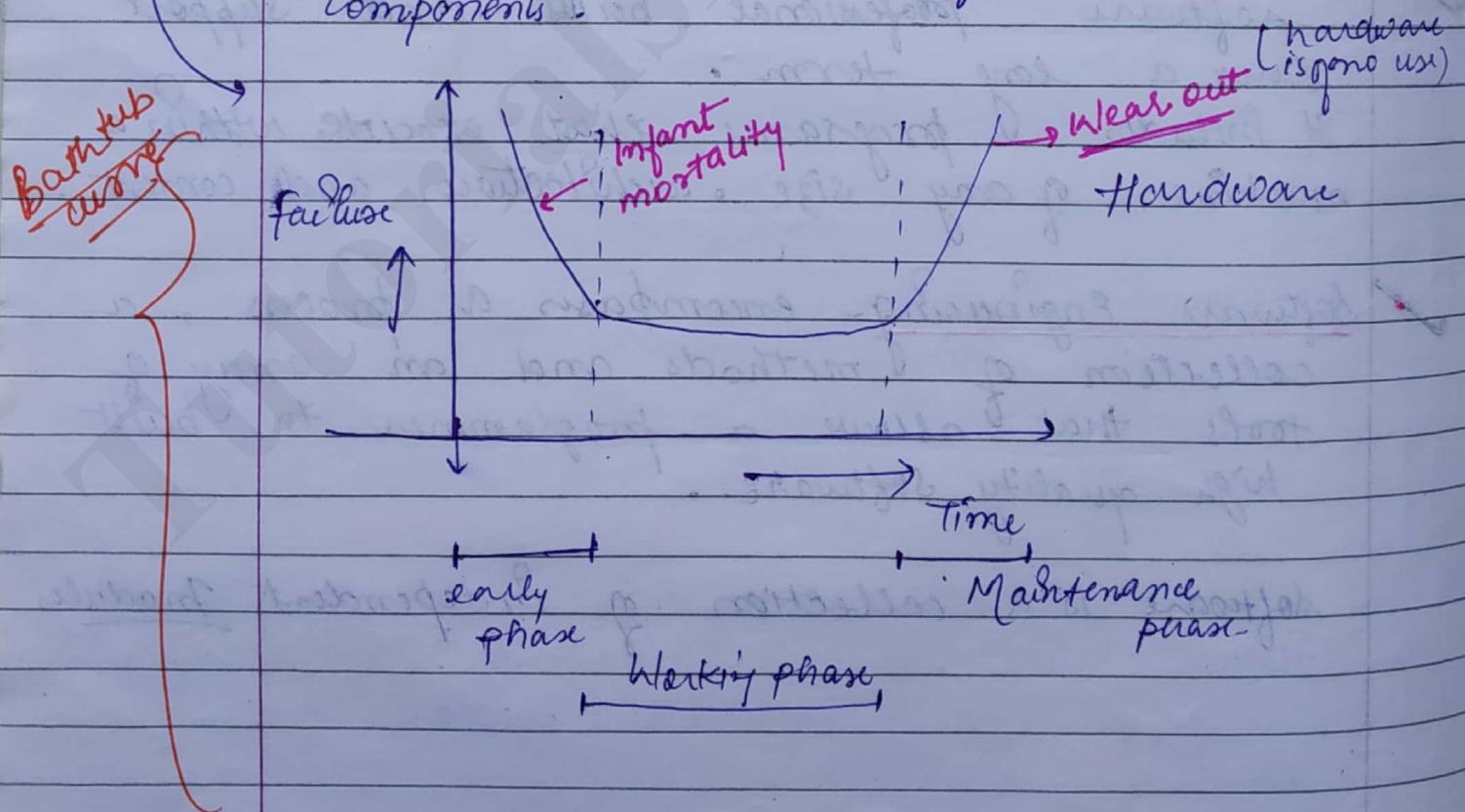
Software is a collection of independent modules

⇒ The Nature of Software

- Software takes on a dual role.
It is a product and at the same time the vehicle for delivering a product.

⇒ Characteristics of Software

- 1) software is logical rather than physical system element.
- 2) software is developed or engineered. It is not manufactured.
- 3) software doesn't wear out.
- 4) A software is composed of several reusable components.



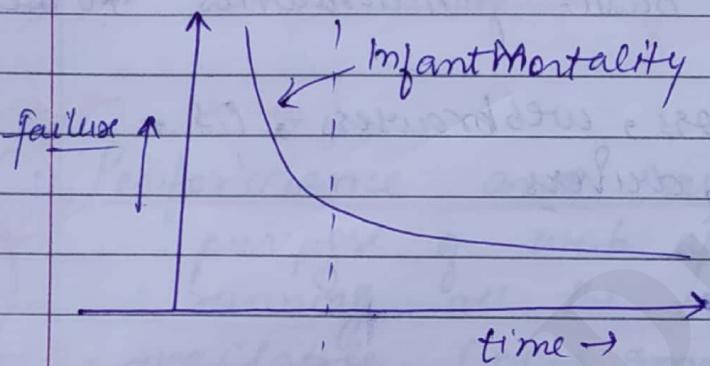
- Infant Mortality →

The product fails in the very initial state/phase

In working phase, possibility of failure is least.

Software Curve

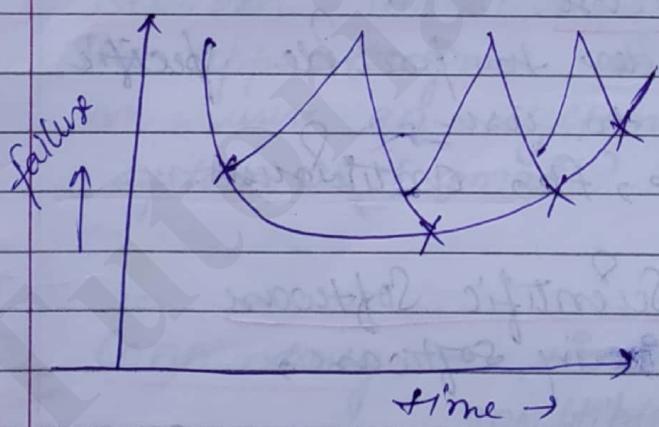
e.g. → windows xp. (Theoretical curve)



It might happen that during its initial / deployment phase, it may fail.

Idealized Curve

→ But if a software is made



(Practical Software Curve)
(actual)

Theoretical Curve isn't possible as of changes in technology and change in needs. Thus we always need to modify or ~~change in~~ update our existing software.

As soon as change in software is done, it is

possible that it might fail, resulting in high failure.

Changes in a software are limited as it leads to increase in cost & time.

→ Software Application Domain

1) System software.

which provide basic functionalities to the system.

Ex → text editors, web browsers, OS, compilers, drivers.

directly interact
with hardware
& system

2) Application Software

designed in order to provide specific functionalities to user.

eg → WhatsApp, fb, antiviruses.

3) Engineering or Scientific Software

aeronautical engineering software,

4) Embedded Software

software which are embedded into hardware device.

eg → AC, speedometer

- 5) Web Applications → Browser apps, gmail, fb, orcut, netbanking apps
- 6) AI software → software used in pattern recognition robotics, etc. Basically used in artificial engineering.

⇒ The unique nature of Web applications completely rely to internet activities.

[Design issues]

- 1) Performance and availability.
purpose of web appln must be solved and running all the time, that is must be available (e.g. → gmail). These apps have a wide nature that is desirable.
- 2) Network Intensiveness (heterogeneity)
The apps designed in web can be used by an user on system irrespective of hardware system OS (gmail).
- 3) Concurrency → simultaneously access to multiple user. e.g. → gmail access by multiple users.
- 4) Unpredictable load → the no. of user accessing a particular app is are changing.
primarily with webapps



A problem of concurrency.

① InConsistency

The data must remain consistent while multiple access web apps.

② fault tolerant.

⑤

Data driving

They transfer data from one user to another where data can be text, images, etc. Any type of data can be transported

⑥ Content Sensitive

since networking is involved, thus content may leak by any intruder or appn, thus content becomes sensitive. Thus, data to be communicated must be secure.

⑦ Continuous Evaluation

All the web apps (stand alone or targeted) must be timely updated/evaluated to meet the market needs. If not updated the web page becomes less popular & thus fails.
eg → Orkut is now obsolete.

⑧ Security

All apps must be secure

(9) Immediacy

software must be launched in market quickly otherwise some other app. The target for its releasing is very short.

Web apps are called "short duration apps".

(10) Aesthetics

This is associated with ~~local screen~~ look and feel. The GUI (Interface) of a web app is of prime concern.

The services must be consistent.

⇒ SOFTWARE ENGINEERING

• By Fritz Bauer, [Nov 69]

According to Fritz,

"Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reusable and works efficiently on real machines" (for a long duration of time)

economic in terms of time, human effort & cost.

Effective

gives out correct output

Efficient

gives out correct output with less time
Its output stands on all software

This definition tells reliability but not Technical aspects.

tells about qualitative but not a quantitative approach.

It doesn't take above some parameters.

→ This defn is modified in

• By IEEE [IEEE 93a]

has developed the more comprehensive definition which states that →

Software Engineering is

* ① The application of systematic, disciplined, quantifiable approach to the development, operation & maintenance of software i.e the application of Engineering to software.

* ② The study of approaches has as in (1)

Disciplined in the sense that the software must lie within well defined boundary

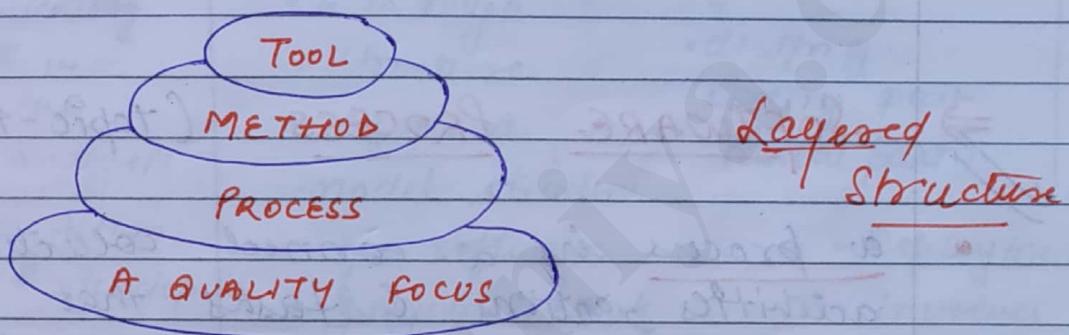
Quantifiable in the sense that its performance can be calculated / measured to meet customer's needs.

Development - refers to initial stage design and development

Operation → It is the stage when we operate our software.

Maintenance → continuous process ^{until} with software goes out.

Software Engineering as Layered Technology (Topic 3)



- Product that is to be made must be a quality product meeting user requirements.
- Process that binds or ~~how to~~ defines how to build tools for a software and methods available to provide a quality product.
Acts as a medium b/w upper & lower layer & lower layer.
- defines a framework with a set of activities
Methods are the activities
- We have particular methods at every step of process.
Testing is a process
& unit testing, alpha testing are methods

→ Tools

They are automated software or that are software ~~itself~~ itself.

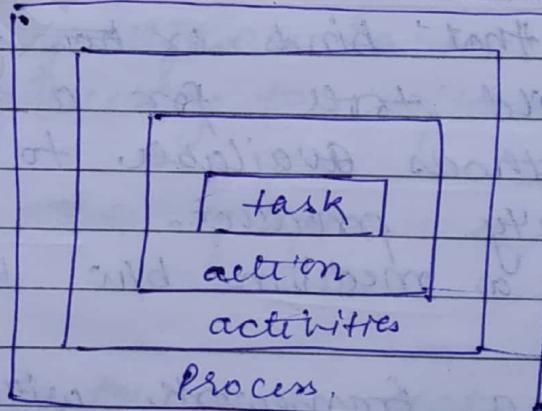
{ eg → Android Studio for Android)

(Inbuilt software)



SOFTWARE PROCESS (Topic-7)

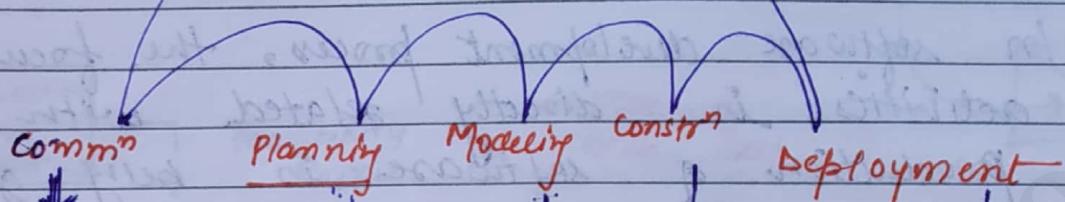
- A process is a connected collection of activities, actions & tasks that are performed when some work product is to be created.
- A process defines who is doing what, when and how to reach a certain goal.



- A process framework establishes the foundation for complete software engineering process by identifying a small number of framework activities that are applicable to all software projects irrespective of their size and complexity. These framework activities are called as Umbrella activities.

5

Umbrella activities



- developer is communicating with user & vice versa. It is the requirement analysis
 - provides interaction b/w client & developer
 - It is a necessary & an integral step.

 - defines how to carry out a development of software
 - associated with reliable design of software
 - Understanding of comp. & designing a plan.
 - logical work
- It is rough structure
 - define a model structure of a software
 - physical work
 - sketches / diagram
- design of coding part (actual part)
 - Deployment involves maintenance also.
 - last step where we deploy our Software.

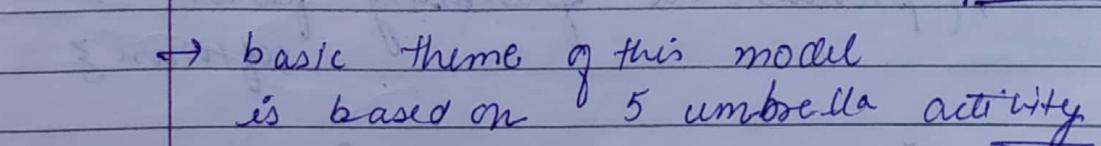
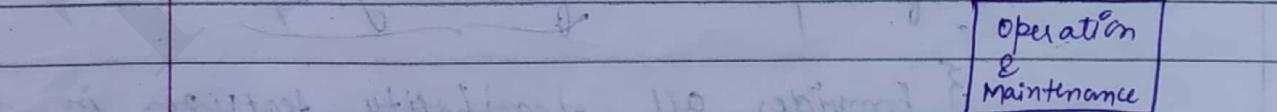
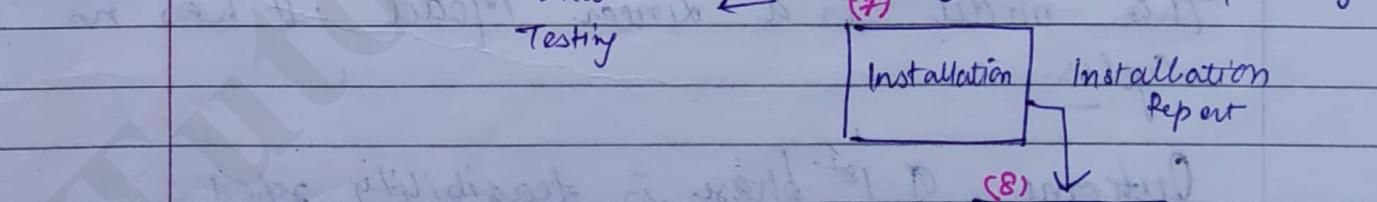
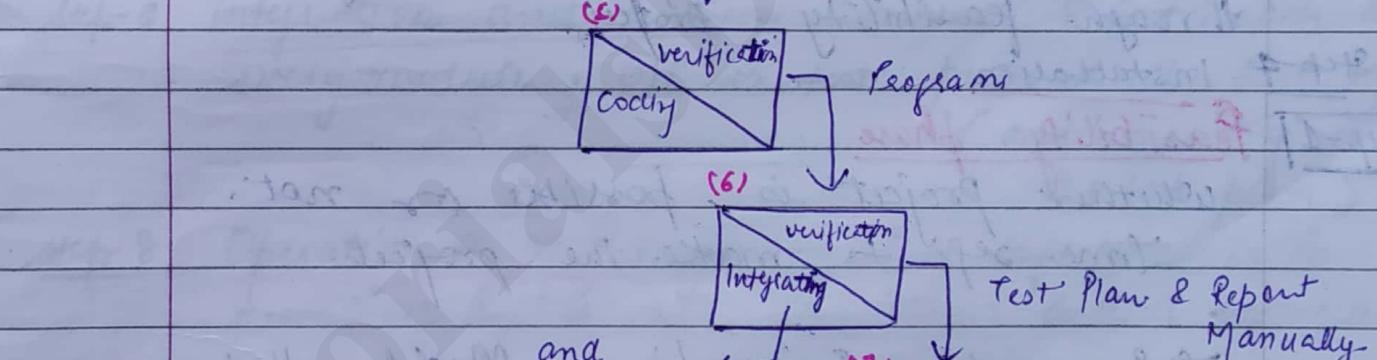
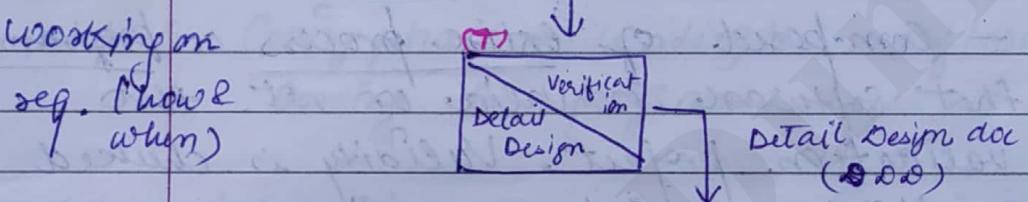
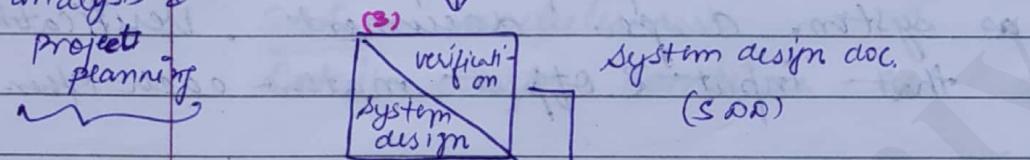
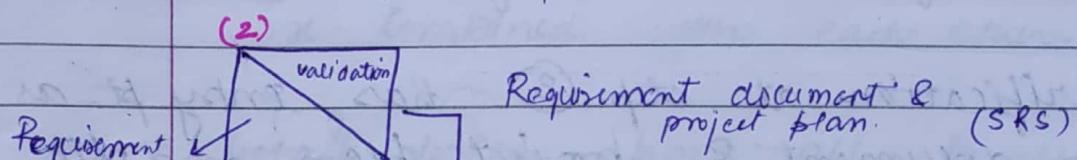
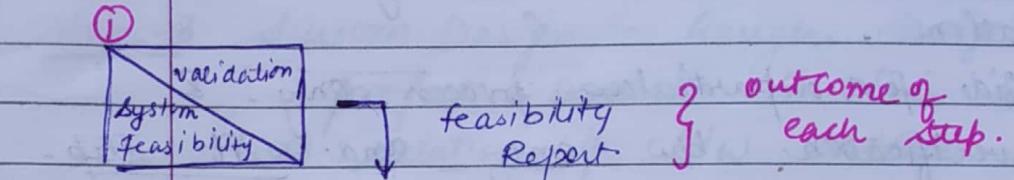
⇒ All the process models follows these 5 activities

- Software Development Process Model

In software development process, the focus on activities is directly related with production of software is being given.

As the development process specifies the major development's and quality control activities that needs to be performed in the project and for this development process various process models have been proposed each with its strength, weakness and applications

1) Waterfall Model



→ basic theme of this model is based on 5 umbrella activity

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 

⇒ Verification

is a small process or small entity in comparison with validation.

It is valid for a particular process only. & we have verification b/w entry and exit step.

e.g. → verification at step ③ has entry pt as Req. document & project plan and exit phase as system design document. Verification ensures that input & output match each other.

- Validity - composed of entire process or program. we check that software is valid or not. Also called as validation project. Validity is checked through feasibility project

Step-1 Installation

Step-1 Feasibility phase

whether project is ⁽¹⁾ possible or not.

Time sep. to make the project.

This model is a → Linear Model. It has no cycles.

Outcome ⁽²⁾ of 1st phase is feasibility report

{ provides all feasibility features in software }

Step-2

Req. Analysis - what we need & what features can be included in ~~req~~ project.

project plan - small planning of the project.

Step-3 System Design - Rough design about the project. Doesn't involve any coding and has relationship b/w entities. How features are combined with each other.

Step-4 Detail design - how all processes are related.
SOD - modules
Detail design combines all modules.

Step-5 Coding phase - writing the software actually. O/P of Coding is programs.

Step-6 Integration and Testing - Combine & test each & every module. O/P is test plan and test report (outcomes of all the test plan)

Step 8 Operation & Maintenance - The software is in execution & is at customer and (end-user)

⇒ Advantages

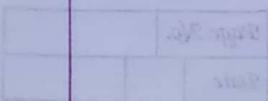
- 1) simplicity
 - 2) clearly divided phases. (clearly divided)
 - 3) easy to execute and implement
 - 4) Intuitive and logical. { each and every step is logically defined }.
- no phase conflict

⇒ Limitation

- 1) It assumes that requirements of a system can be frozen.
- 2) ~~process~~ ~~process~~ freezing the requirement before solution of hardware at early stage which is not preferred in this era q. technology change.
- 3) It follows the big bang approach.
- 4) It is a document driven process.
- 5) All ~~for~~ or nothing approach
- 6) Disallow changes.
- 7) Cycle time is too long.
- 8) User feedback is not allowed.

- We must know all the requirements from the very first step. that is programmer should be clear about his work.

→ Big-Bang - The user comes to know abt the project after coming on 7th or 8th stage. The whole project gets to know in one step, can't be



| | |
|----------|--|
| Page No. | |
| Date | |

known abt. steps (not continuous interaction with the user)

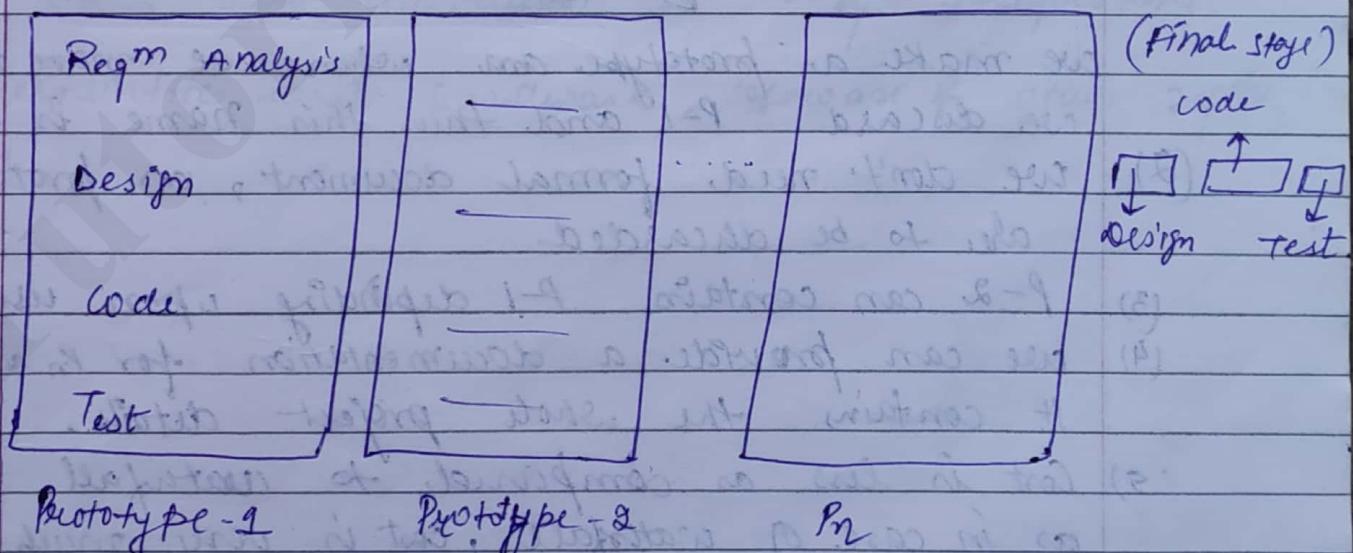
→ All or nothing Approach → project is either accepted or rejected.

⇒ Applications of feedback Waterfall Model

- 1) for well understood problem.
- 2) short duration project.
- 3) Automation of existing manual system.

→ Project made with waterfall model is clear & known beforehand. It is only possible when we enhance an existing product. It is used for short duration, as even if the project fails, the investment would be compensated. It is only used when all resources and req. are known.

⇒ PROTOTYPE MODEL



(Small section of
whole process)

- This model overcomes all limitations of waterfall model. Here, we provide a small product of the entire software.
- Design → designing of given seg^{ms} giving each prototype to user, we can remove old features & keep on adding new features. here customer is regularly involved & thus customer can't deny about the made project. we can take continuous amt. from user according to prototypes.

⇒ Requirements

- (1) Before proceeding, the requirement of prototype is that it must be a short duration prototype. Req^m included must be such that they can be delivered in a timely manner.

Known as "Quick And Dirty Approach".

- we make a prototype and when we move to P₂, we discard P₁ and thus this name is given.
- (2) we don't need formal document, as prototypes are to be discarded.
- (3) P₂ can contain P₁ depending upon user's seg^m.
- (4) we can provide a documentation for P_n as it contains the whole project details.
- 5) Cost is less as compared to waterfall model as in case of waterfall, cost is very much higher at later stages.

⇒ Advantages

- 1) Helps in requirement elicitation (Gathering)
- 2) Reduce risk.
- 3) leads to a better system.
- 4) Minimal documentation.
- 5) Cost effective.

⇒ Disadvantages

(We need to keep working on every prototype)

- 1) front heavy process
- 2) Possibly higher cost due to continuous changes.
- 3) It is a continuous and complicated process.

⇒ Applications

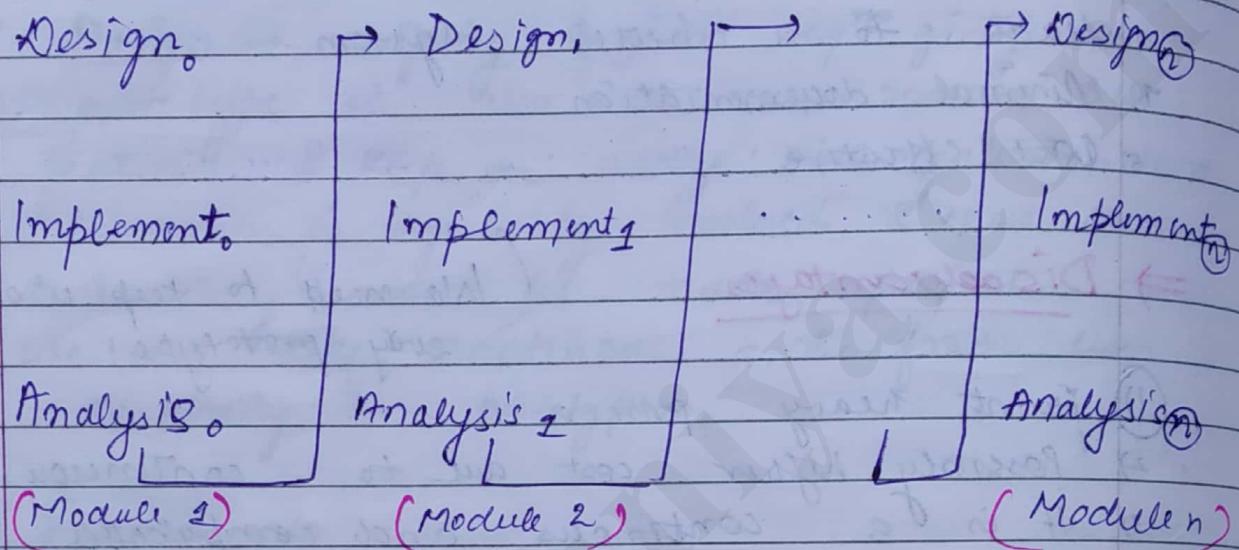
→ (new / beginner users)

- 1) System with novice users.
- 2) when uncertainty exists in requirements
- 3) when user interaction is very important.

uncertainties w.r.t. hardware, software & other segms.

Incremental
→
↓

Iterative Enhancement Model



- Combines the approach of waterfall as well as prototype model

- controls a project control list (PCL)

↓
(contains overall functionality of process)

Advantages

- It counters the third limitation (Big Bang) of waterfall model & combine the benefits of both Prototype & waterfall model.

- The Basic Idea is software should be developed in increments.

- A PCL is created that contains of all the tasks that must be performed to obtain final implementation.

- Redesign of the system is only allowed in initial phases of software designing.

Note →

If we change something at later stages, it gets more costly like waterfall Model.

~~Difference b/w Verification & Validation in S.E~~

Verification

- 1) Is the process of evaluating software to determine whether the products of a given development phase satisfies the conditions imposed at the start of that phase.

- 2) is a static practice of verifying documents, design, code and programs.

- 3) It generally doesn't involve executing the code.

Validation

- 1) The process of evaluating software in beginning, during or at the end of the development process to determine whether the complete software meets the customer's expectation and requirements.

- 2) is a dynamic mechanism of validating the (and ← testing) actual product.

- 3) It always involve executing the code.

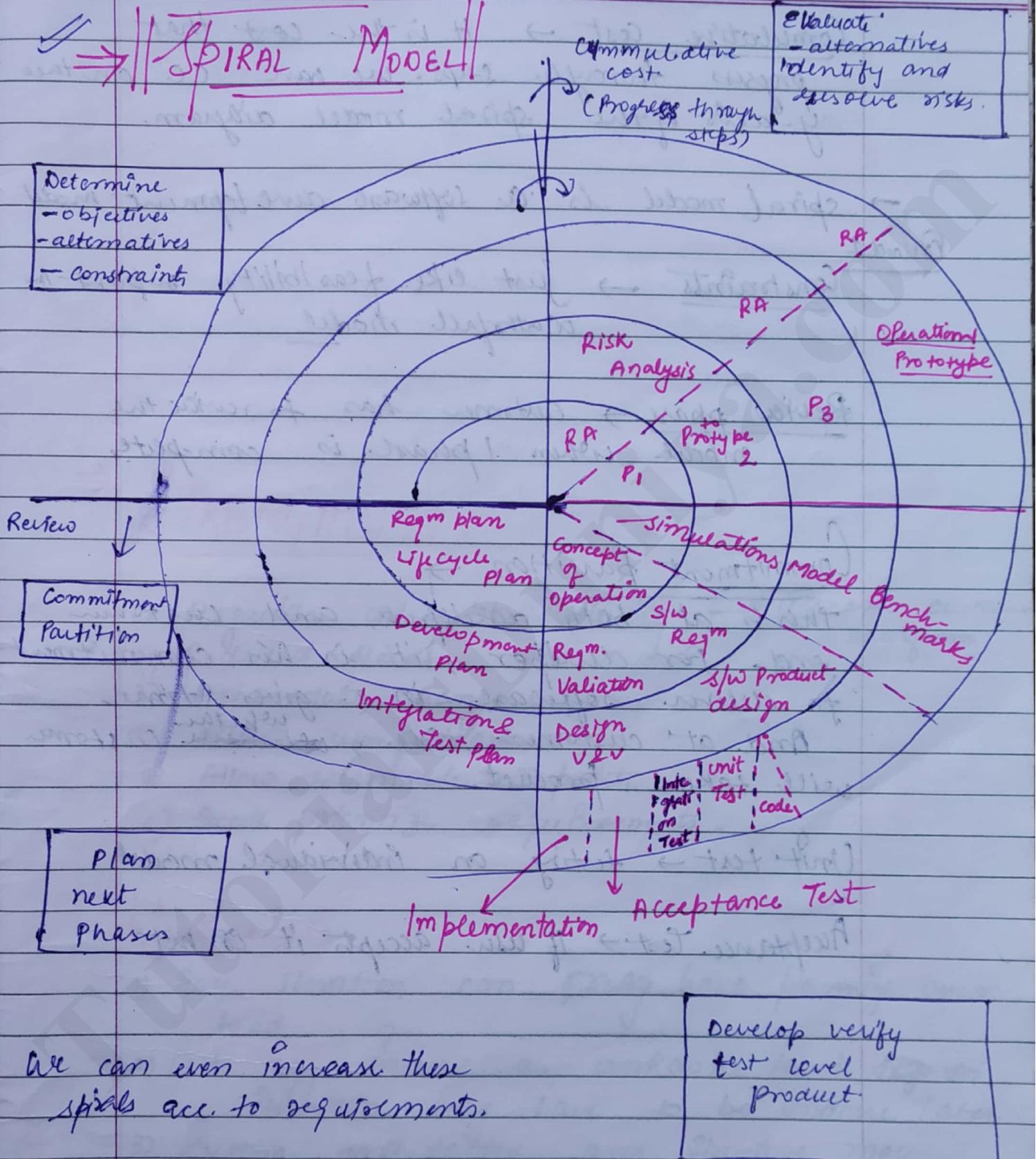
Verification

- 4) It uses methods like inspection, reviews, walk throughs and desk checking etc.
- 5) Is a low level exercise.
- 6) Is to check whether the software conforms to specifications.
- 7) Is used to build high quality software, which is well engineered & error free.

Validation

- 4) uses Methods like functional and system testing.
- 5) Is a high level exercise.
- 6) Is to check whether software meets customer expectation & requirement.
- 7) Is to make the end product effective.

→ ||-SPIRAL MODEL||



We can even increase these spirals acc. to requirements.



| | |
|----------|--|
| Page No. | |
| Date | |

Cumulative cost → It is the cost that progress through steps. we have CC on the y-axis of the spiral model diagram.

→ spiral model is a software development mode

Quadrant 1

Constraints → Just like feasibility report in waterfall model.

Review phase → customer has to review the model when 1 phase is complete

Commitment partition → :

This is at both developer and customer end. For developer, it is his commitment to deliver software in a given time.

And at customer side, ~~whether~~ customer will take product.

Unit test → testing on individual module

Acceptance test → If user accepts it or not.

⇒ STRENGTHS

- 1) Regular and quick delivery. → (increases customer satisfaction)
- 2) Reduced Risks.
- 3) Accommodate changes.
- 4) Allow user feedback.
- 5) Allow reasonable exit points.
- 6) Helps prioritize requirements.

Weaknesses

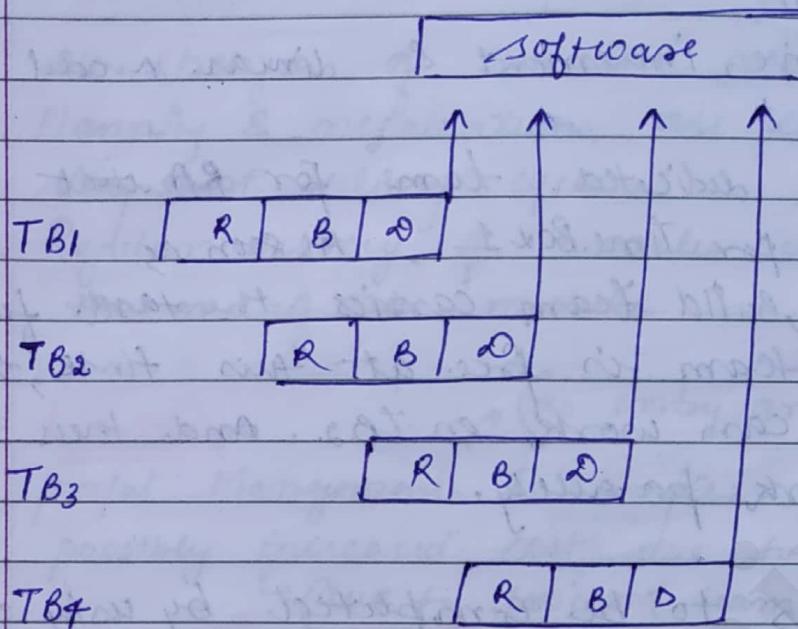
- 1) Each iteration can ~~poorly~~ have planning over head
- 2) Cost may increase as work done in 1 ~~iteration~~ iteration may have to be undone later.
- 3) System architecture and structure may suffer as frequent changes are made.

Both Iterative Enhancement & spiral models
are Iterative models.

Applications

1. for business, where time is of ~~less~~ essence.
2. where ~~size~~ ~~risk~~ of very project can not be taken.
3. where ~~segs~~ are not known and will be available only with time.

⇒ Time-Boxing Model } Pipeline Model }



(Time Box)

| Requirement Team | RA for TB ₁ | RA for TB ₂ | RA for TB ₃ | RA for TB ₄ |
|------------------|------------------------|------------------------|------------------------|------------------------|
|------------------|------------------------|------------------------|------------------------|------------------------|

| Build Team | Build for TB ₁ | Build for TB ₂ | Build for TB ₃ | Build for TB ₄ |
|------------|---------------------------|---------------------------|---------------------------|---------------------------|
|------------|---------------------------|---------------------------|---------------------------|---------------------------|

| Deploy Team | Deploy for TB ₁ | Deploy for TB ₂ | Deploy for TB ₃ | Deploy for TB ₄ |
|-------------|----------------------------|----------------------------|----------------------------|----------------------------|
|-------------|----------------------------|----------------------------|----------------------------|----------------------------|

R → Requirement
B → Build
D → Deployment

RA → Requirement Analysis

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 

- parallel Model
- uses iterative, increment & linear model
- There is a dedicated team for RA that starts works for time box 1. As soon as dev. finishes, build team carries the task further. Rep. team is free at this time, thus Rep. team can work on TB2. And then all teams work parallelly.

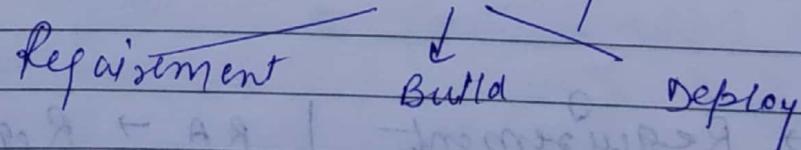
Product has to be completed by using -

- 1) duration of time box (overall)
- 2) duration of each phase of time box, for max. utilization.

① To speed up the development, parallelism b/w different iterations can be employed which uses the pipelining of different iterations within a time box with stages of equal duration.

② The basic unit of development is time box where there is a dedicated team for each process in a time box.

③ It is divided into 3 process



Strength

1. All strengths of iterative model,
2. Planning & negotiations are easier.
3. Very short delivery cycle.
4. Regular delivery of modules with one module at the end of each module.

Weaknesses

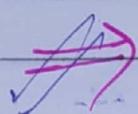
1. Project Management is complex.
2. Possibly increased cost due to large team size.
↳ (due to dedicated team for each process)

Applications

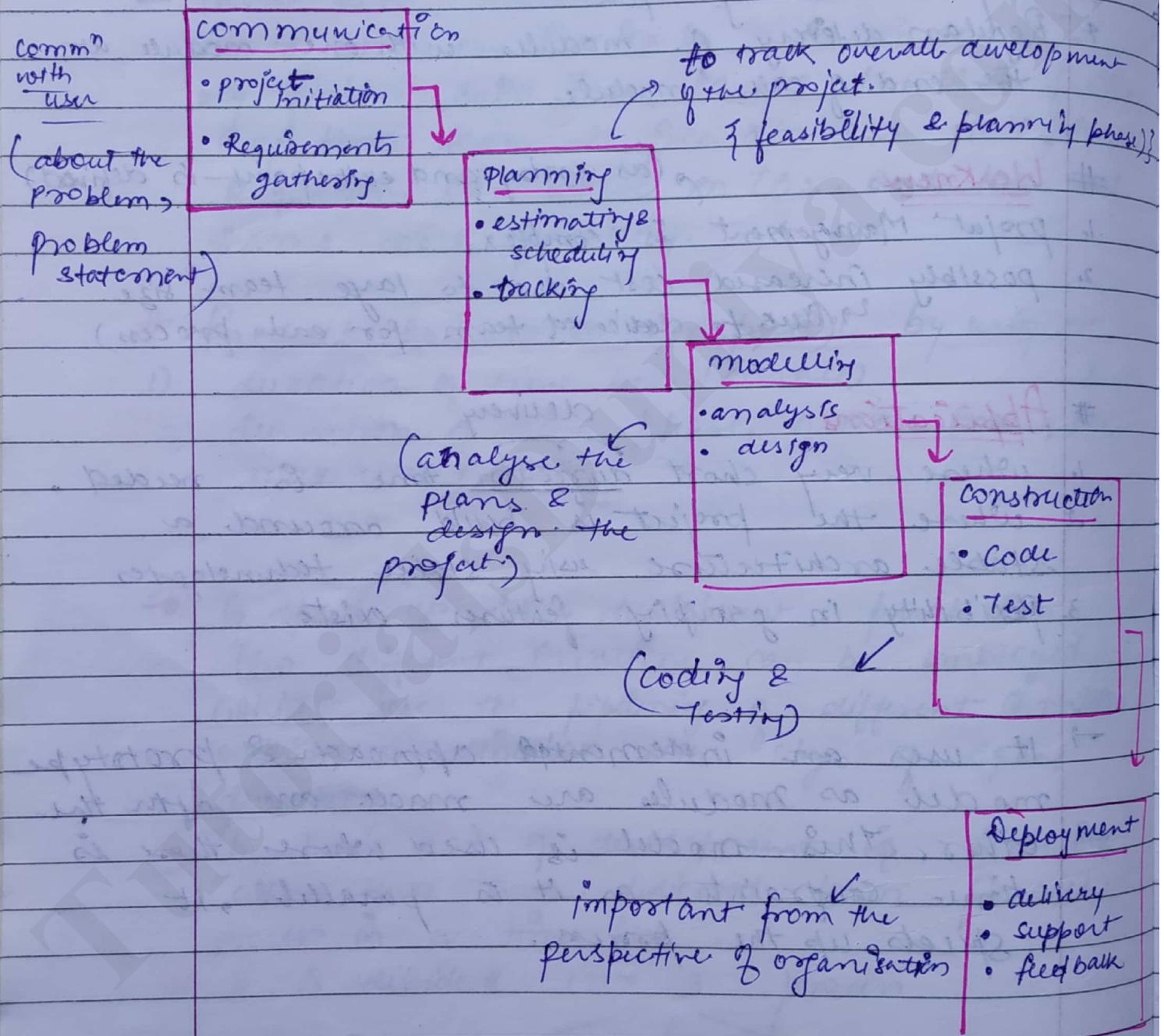
Delivery

1. Where very short duration time is needed.
2. Where the project is build around a stable architecture using stable technologies.
3. Flexibility in grouping features exists.

→ It uses an incremental approach & prototype model as modules are made one after the other. This model is used where there is time constraints as it is parallel, it speeds up the process.

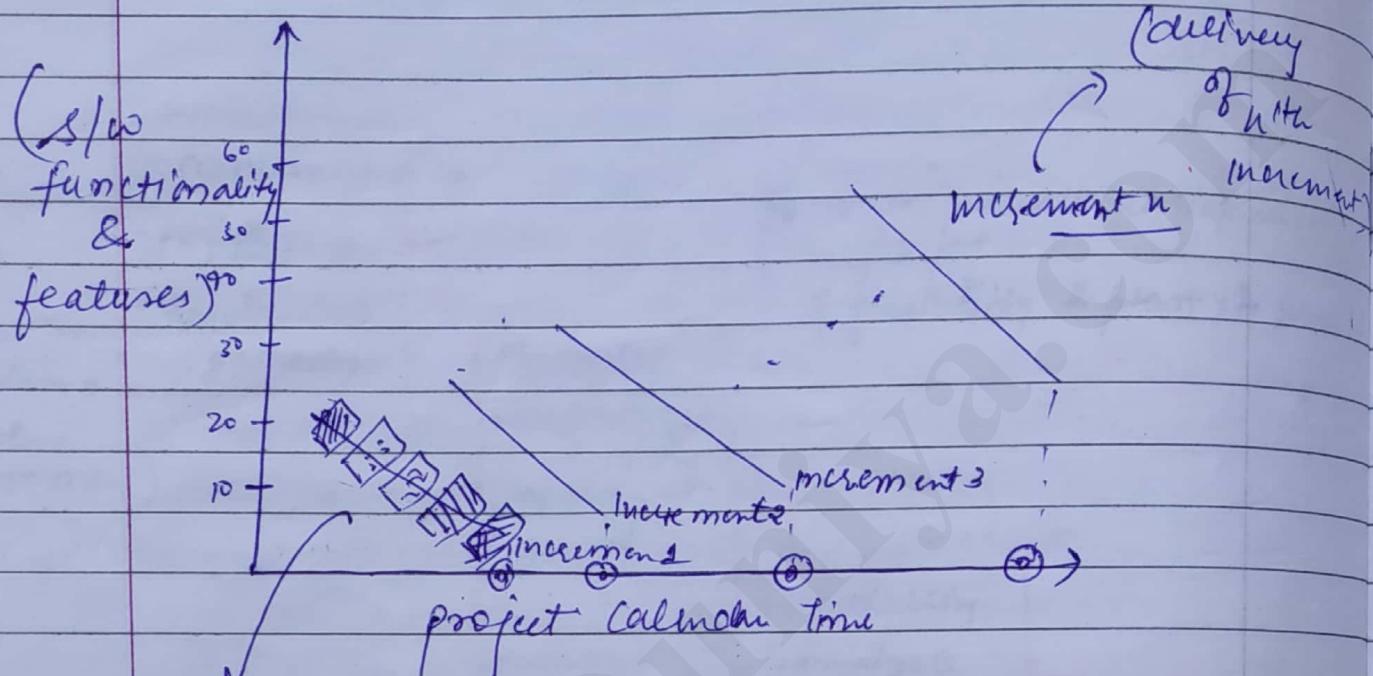


Waterfall Model {according to Pressman}



⇒ Incremental process Model

{ Iterative enhancement }



As we keep on increasing the features, more time is required to complete the project.

||||| communication

... planning

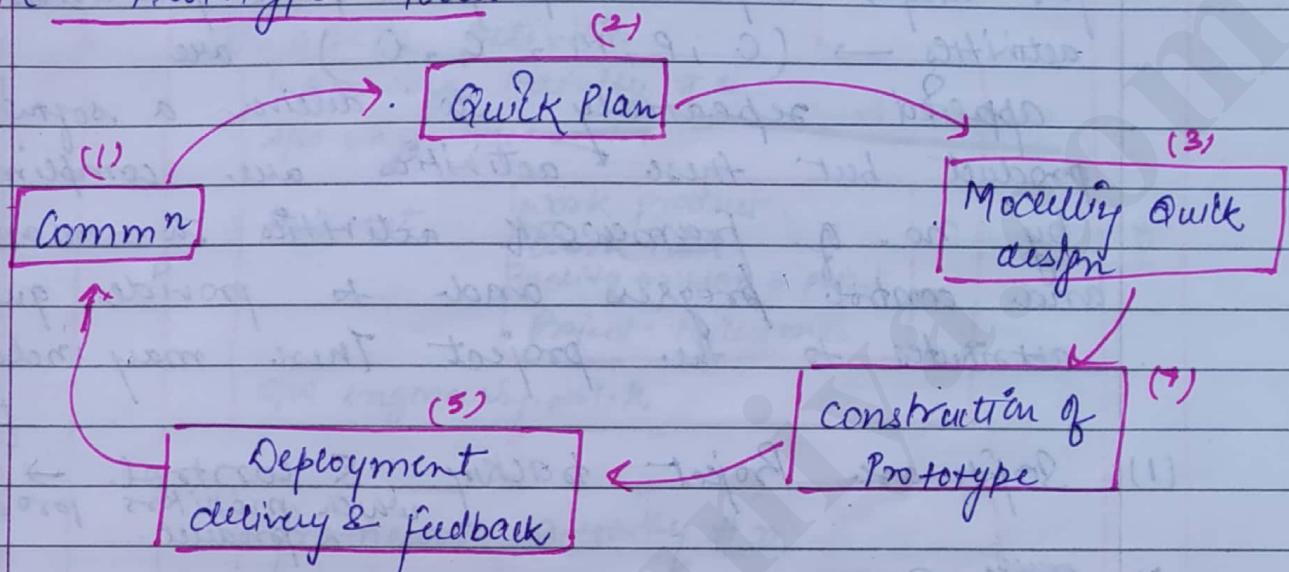
~ modeling

|||| construction

diagonal lines deployment

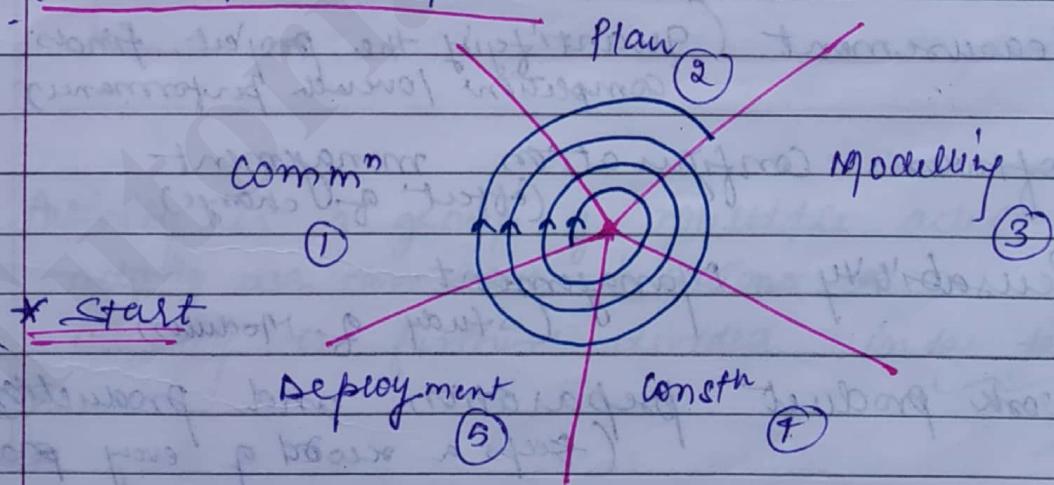
→ Evolutionary Process Model ←

① (1) Prototype Model



It is a cyclic product until a product is delivered.

② (2) Spiral Model

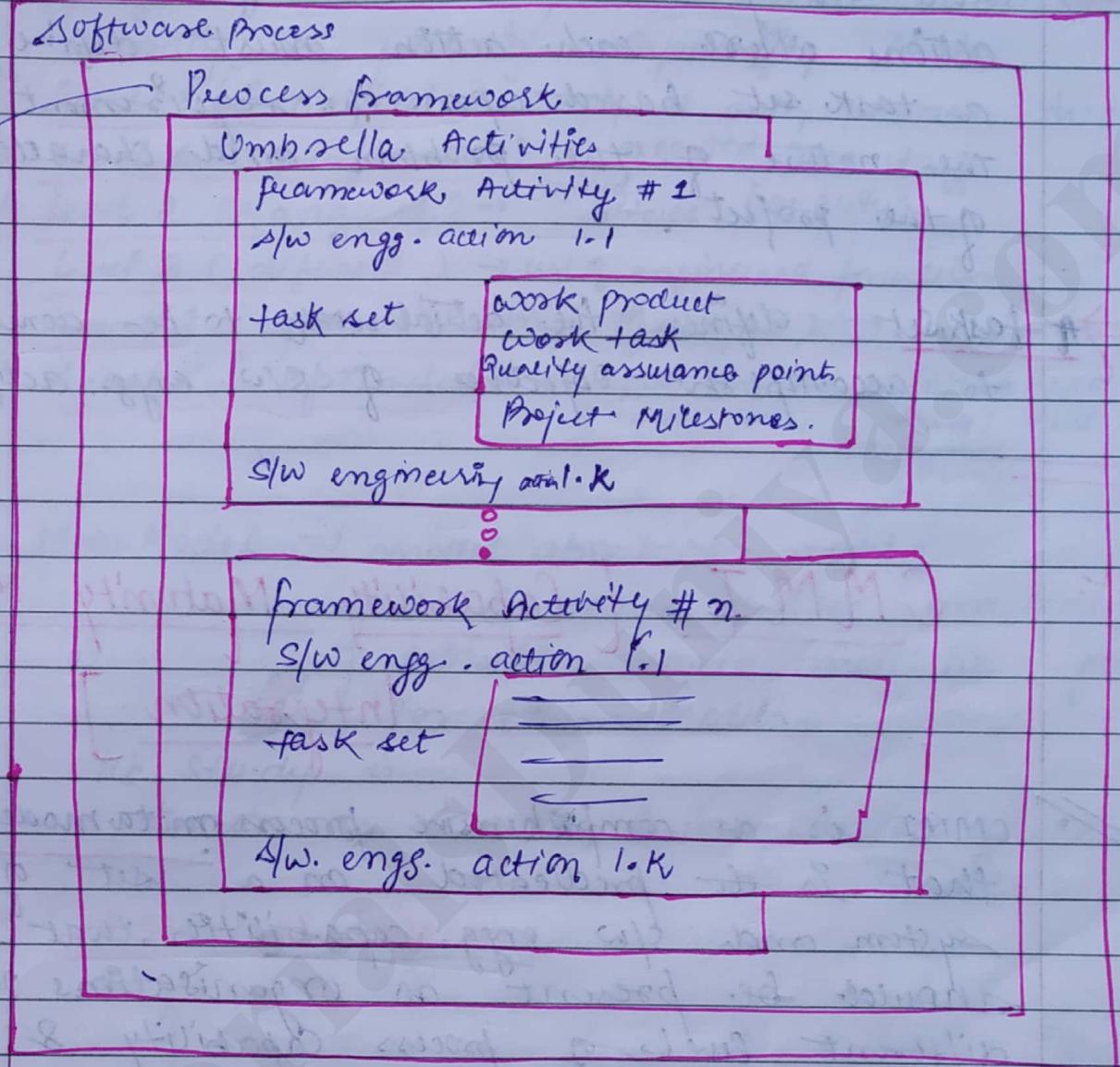


→ Defining a framework activity

for major software projects, the five umbrella activities → (C, P, M, C, O) are

applied repeatedly to deliver a software product but these activities are complemented by no. of framework activities to manage and control progress and to provide quality attributes to the project. These may include

- (1) Software Project tracking & control. → ~~team which monitors process of a product.~~
- (2) ~~Risk~~ Risk Management. (team that assess risks)
- (3) Software Quality Assurance (adding on quality features)
- (4) Technical reviews
- (5) Measurement (quantify the project. finds competition / overall performance)
- (6) Software Configuration management (effect of change)
- (7) Reusability Management. (study of Modules)
- (8) Work product preparation and production. (keeps a record of every ~~phase~~ outcome of each phase)



- Activity is a group of multiple actions - In an activity we can have ~~n~~ actions. ~~a~~
Activity is further divided into task set.
max. of ~~n~~ tasks
- a task set is smallest unit of Execution.

⇒ Each activity is divided into a number of actions where each action must define a task set based upon the requirement giving the nature of the problem and characteristic of the project.

⇒ A task set defines the actual work to be done to accomplish objectives of s/w engg. actions.

CMMI [Capability - Maturity Model Integration]

✓ CMMI is a comprehensive process meta-model that is predicated on a set of system and s/w engg. capabilities that should be present as organisations reach different levels of process capability & maturity.

→ It represents a meta-model in 2 different ways:

1st Model
(1)

As a continuous model which describes a person in two dimensions →
Capability level & process area.

Each process area, is formally assessed against

specific goals and practices and is rated according to following capability levels.

level 0 (incomplete) → obj. at entry pt are incomplete.

level 1 (performed) → executed acc. to J.

Level 2 (Managed) → defined policies (quality).

Level 3 (defined) → use of engineered practices.

Level 4 (quantitatively Managed) →

Level 5 (optimised) → Optimising the statistical need to meet the customer's need.

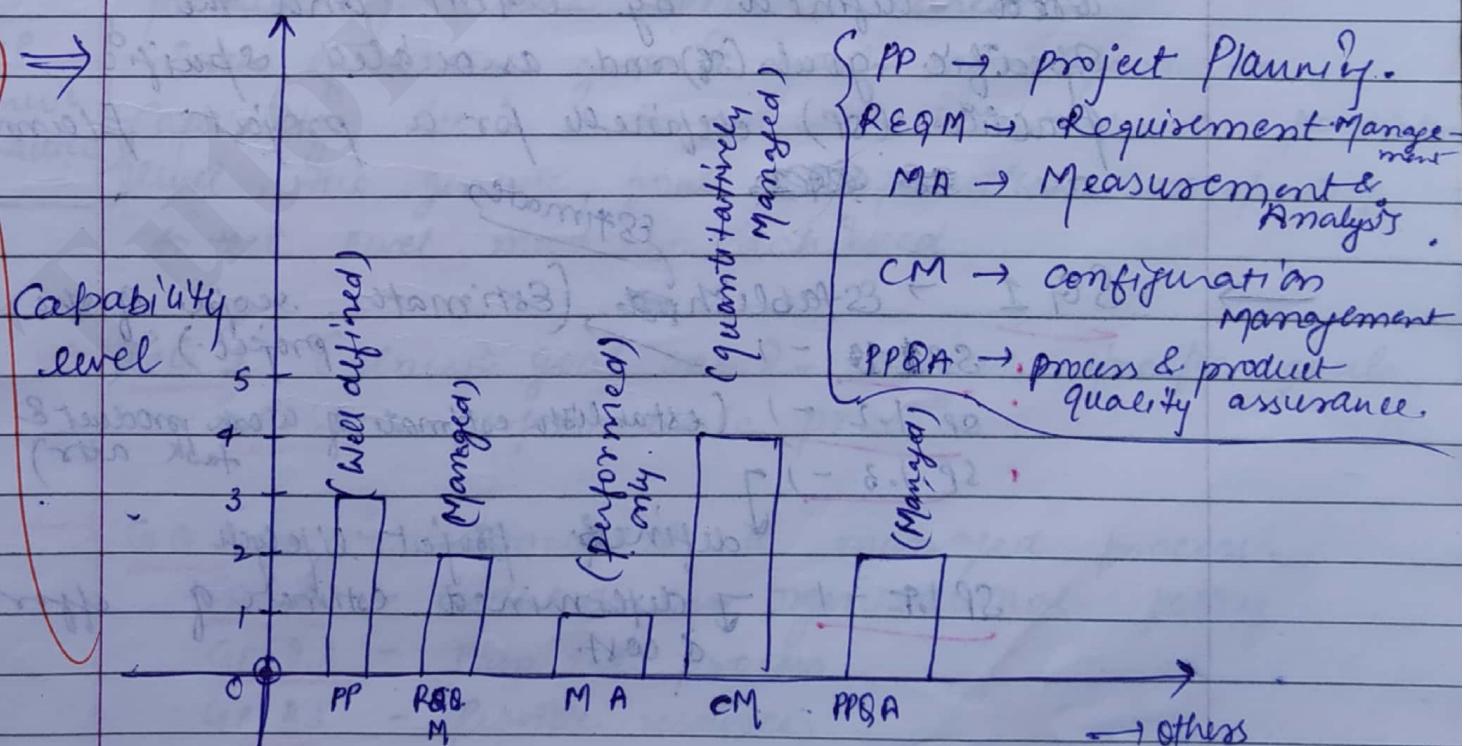
(users)

Meta Model → model about the model

It is some specific practices / principles that must be followed for other models

Its study is known as CMMI. (integration of all other models)

CMMI Level Diagram



CM → changes are completely and accurately managed ..

book

~~Page → CG~~

Page → (797 - 801)

~~X~~ → The CMMI defines each process area in terms of specific goals and specific practices required to achieve these goals. Specific goals establish the characteristics that must exist if the activities adopted by a process area is effective and specific practices define a goal into set of process related activities.

for example,

project planning is one of the process area defined by CMMI and the specific goals (SG) and associated specific practices (SP) defined for a project planning are ~~as follows~~.



SG 1 → Establishes ^{estimates} (Estimates scope of the project.)

• SP 1.1 - 1

• SP 1.2 - 1 (Establish estimate of work product & task attr.)

• SP 1.3 - 1

• SP 1.4 - 1

↓ defines project life cycle

↓ determined estimate of effort & cost

→ SG 2 - develop a project plan

- SP 2.1-1 establish the budget & schedule
- SP 2.2-1 identify project risks.
- SP 2.3-1 planned for data management
- SP 2.4-1 plan for project resources
- SP 2.5-1 plan for needed knowledge & skills
- SP 2.6-1 plan for stakeholder involvement
- SP 2.7-1 establish the project plan.

→ SG 3 - obtain commitment to the plan.

- SP 3.1-1 review plans that affect the project
- SP 3.2-1 reconcile work and resource level
- SP 3.3 obtain plan commitment.

⇒ In addition to SG and SP, CMMI also defines generic goals & practices for each process area.

G.P.
(generic
practices)

Each of these generic goals corresponds to capability levels. In order to achieve a particular level, the generic goal & practice corresponding to that level must be achieved.

→ GG 1 (generic goal one.) - achieve specific goals.

- G.P. 1.1 - perform basic practices.

→ GG 2 - institutionalize a managed process.

- G.P. 2.1 - establish an organisational policy.
- G.P. 2.2 - plan the process
- G.P. 2.3 - provide resources.

- GP 2.4 - assign responsibility
- GP 2.5 - train persons.
- GP 2.6 - manage configuration
- GP 2.7 - identify and involve relevant stakeholders
- GP 2.8 - Monitor and control the Process.
- GP 2.9 - Objectively evaluate adherence
- GP 2.10 - review status with higher level management

→ G1 G1 - 3 - Institutionalize a defined process

- GP 3.1 - establish a defined process.
- GP 3.2 - collect improvement information.

→ G1 G1 - 4 - Institutionalize a quantitatively managed process.

- GP 4.1 - establish quantitatively objectives for the process.
- GP 4.2 - stabilize subprocess performance

→ G1 G1 - 5 - Institutionalize & optimizing process.

- GP 5.1 - ensure continuous process improvement
- GP 5.2 - correct root causes of problems.

Q3

(ii) The Staged Model

2nd Model

defines the same process area, goals & practices with a difference that it defines 5 maturity levels rather than 5 capability levels.

To achieve a maturity level, the specific goals and practices associated with a given process area must be achieved. The relationship b/w maturity level & process area may be given as →

| Level | Focus | Process Area |
|---------------------------|--------------------------------|--|
| • Optimising (last level) | continuous process improvement | 1) organisational innovation & deployment 2) causal analysis & resolution. |
| • Quantitatively Managed | quantitative management | 1) organisational process performance 2) quantitative project management |
| • Defined | process standardisation | 1) Reg'm development 2) technical sol'n 3) product integration 4) verification 5) validation 6) organisational process focus. 7) organisational process definition. 8) organisational training. |

Level

Focus

Process Area

- 9) Integrated project management
- 10) Integrated supplier management
- 11) Risk management
- 12) Decision analysis & resolution
- 13) Organisation env. for integration &
- 14) Integrated learning.

• Managed

Basic project Management

- a) Reg. management
- project planning.
- b) project monitoring & control
- c) Supplier agreement management.
- d) Measurement & Analysis
- e) Process & product quality assurance
- f) Configuration management

• performed

basic project development

- a) CPMCA → umbrella activities
- b) project development & deployment

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 

18-01-18

| | |
|----------|--|
| Page No. | |
| Date | |

→ Requirement Engineering

- 1) It is a condition or capability needed by a user to solve a problem or achieve an objective.
- 2) It is a condition or a capability that must be met or processed by a system to satisfy a contract, standard, specification or other formally imposed document.

Need for SRS → (Software req → specification)

- (i) An SRS establishes the base for agreement b/w client and the supplier on what the software product will do.
- (ii) It provides a reference for validation of the final product.
- (iii) A high quality SRS is a pre-requisite for high quality software.
- (iv) A high quality SRS reduces the development cost.

Pol
Rn
Mat
Mn
Upp

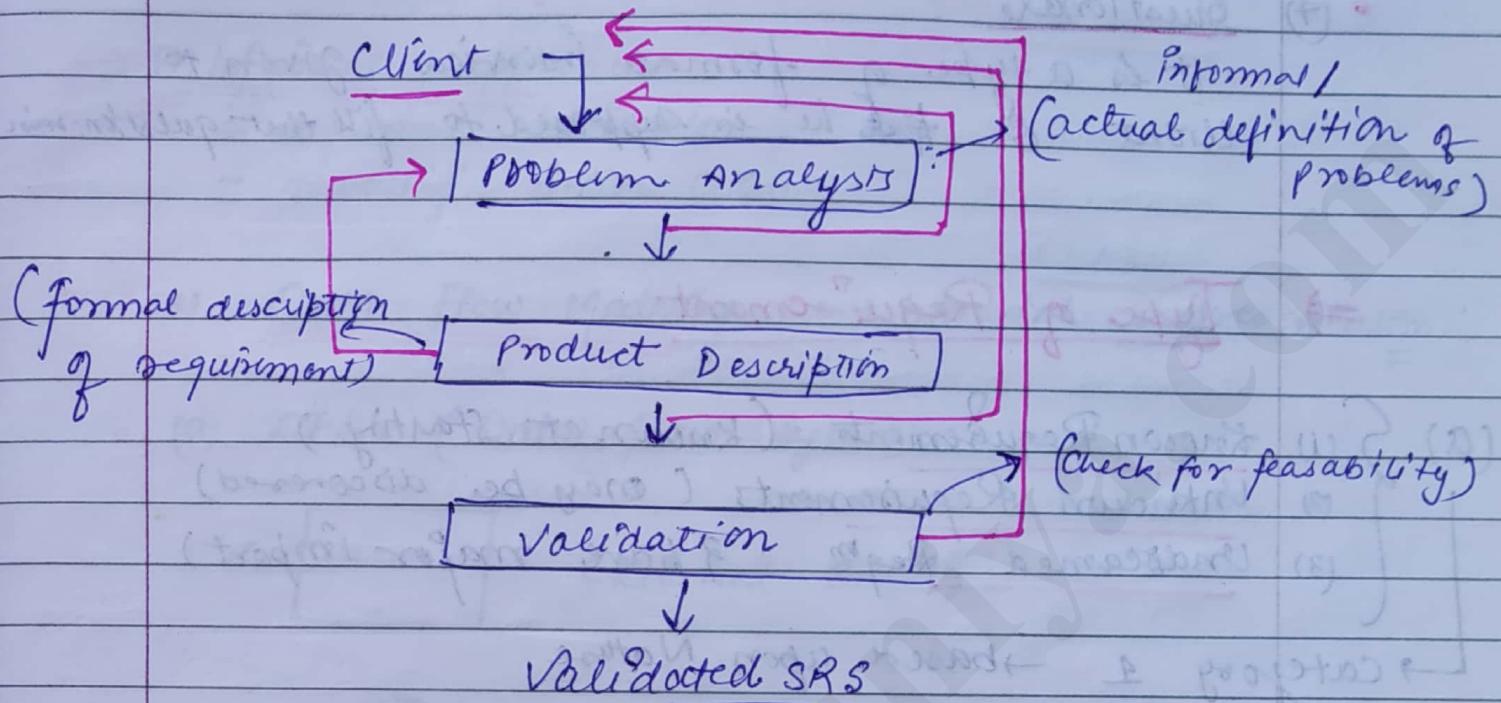
| Phase | Cost (person-hrs) |
|-------------|-------------------|
| Req'm | 2 |
| design | 5 |
| Coding | 15 |
| Testing | 50 |
| Maintenance | 150 |

① eg → If we have 50 new segm only at segm phase
 $\therefore \text{cost} = 50 \times 2 = 100 \text{ person-hrs}$

② • If new segms are not defined at Req'm phase & instead at other levels \rightarrow 65% design, 2% coding, 36% at testing & 3% at maintenance.
 then, $(65\% \text{ of } 50 \times 5) + (2\% \text{ of } 50 \times 15) + (36\% \text{ of } 50 \times 50) + (3\% \text{ of } 50 \times 150)$
 $= \underline{\underline{1152.5 \text{ person hrs}}}$

Introducing changes at later stages can result in more increased no. of person-hrs. as compared to changes in only segm phase as cost at later stages is increasing exponentially.

~~Q-2~~ Requirement process



⇒ This is not a linear process. It is a cyclic process

~~Q-3~~ Techniques for reqm' Elicitation

• (1) Interview

To gather / analyse info. from the client.

• (2) Brainstorming

It is a group process / discussion where there are all teams at developer and client end.

Reqm can be defined in a better way. One problem / reqm' can have n' no. of scopes & n no. of clients.

• (3) fast 3 facilitated application specification Technique } Requirement elicitation

- (7) Questionnaire

It is a type of formal interview given to client & he is supposed to fill that questionnaire.

⇒ Type of Requirement

- (a) {
(1) Known Requirements (known at starting)
(2) Unknown Requirements (may be discovered)
(3) Undesired Requirements (have major impact)
- Category 1 → based upon Nature

- (b) Category 2 { (based on functionality) }
(1) functional Regm feature
(stated already that must be included)
(2) Non functional Regm
additional qualities to be included in the project.
- availability, testing, usability, maintainability.

(c) Category 3

- (1) User Regm & (2) System Regm.
specified by end user & client - inherent regm that exists in system.

(d) Category 4

- (1) Interface requirement
H/w S/w - Interface API
- (Actual view of project)

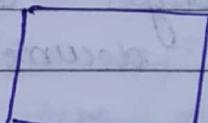
⇒ Problem Analysis

→ (1) Informal approach

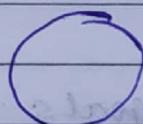
- informal discussion
- meeting with client

→ (2) Data-flow Modeling → pictorial representation

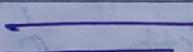
• (a) DFD's (Data flow Diagram)



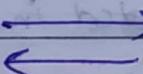
External entity



Process



Data store



Data flow

→ Level 0: contextual Diagram

In Level 1 → we elaborate the processes in more detailed manner.

22-01-18

Software Requirement Specifications (SRS)

~~definition
of
SRS~~

the final output of Repm Engineering is software Requirement specification Document (SRS)

It is a formal document with a ~~fixed~~ pre-defined template which converts the outcome of analysis space into a well formed specified document.

Characteristics of SRS

To properly satisfy the basic goals SRS should have certain properties.

Some desirable characteristics of SRS are-

~~Mostly~~

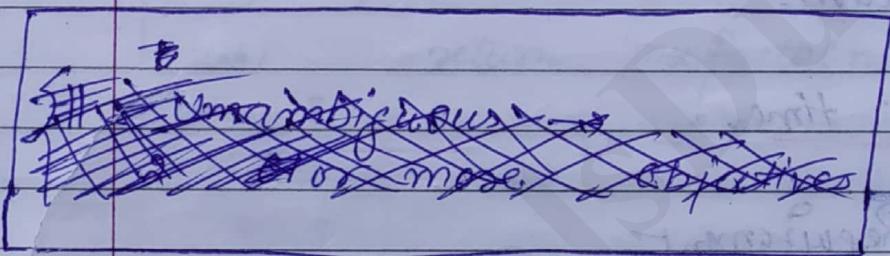
- (1) Complete → reqms should be stated initially.
- (2) Correct
- (3) Unambiguous → conflicts b/w reqms. should not be
- (4) Verifiable → Cost effective Measure must exist. (reqm must be consistent)
- (5) Consistent → with respect to each other.
- (6) Rigorous for Importance or Stability →
- (7) Modifiable → SRS should be changeable provided it ensures (Ranking) all other properties properly
- (8) Traceable →
Should be mapped with some features.) Categorize the reqms on the basis of their importance & stability

what is correct SRS .

Correct

- When reqms are valid & possible (feasible)
- functionalities should be properly defined,
- whatever is desired at end or by the software must be properly defined in SRS.

Note: Maintaining complete is almost impossible or difficult because we need to write down all the reqms at the start of the project that are going to be used later for project.



COMPONENTS OF SRS → Design Issues of SRS

SRS should specify some basic system properties and the issues that may exist while specifying SRS are -

(1) functional Requirements

(2) Performance Requirement

(3) External Interface Regm.

(4) Designed constraints

Software Project Planning

After the finalisation of SRS, a process to estimate size, cost and development time of the project is carried out & this process is called Software Project Planning.

It includes following activities.

- 1) Size Estimation : LOC overall size of project
- 2) Cost Estimation.
- 3) Development time.
- 4) Resource Requirement
- 5) Project scheduling

LOC → Lines of Code or (KLOC)
(Kilo)

→ Size Estimation

1) Lines of code

a line of code is any line of program that is not a comment or blank line regardless of no. of statements or fragments of statements on the line.

for ex. →

II. factorial function.

(1) int facto (int n) ,

(2) {

(3) if (n == 1)

(4) return 1 ;

(5) else,

(6) return n * facto(n - 1) ;

(7) } // main function.

(8) int main ()

(9) {

(10) int num;

(11) cout << " enter a number" ;

(12) cin >> num;

(13) cout << " factorial is" << facto(num) ;

(14) return 0 ;

(15) }

// end of program.

This program have LOC count of 15.

↓
except comments &
blank lines

(Numerical)

~~#~~ function point Analysis (FPA)

this technique is used for size measurement which decomposes a system into 5 functional units.

(E I)

(1) Input → information entering into the system.

(2) Output ^(EO) → Information leaving the system.

(3) Enquiries ^(EQ) → Information Request for instant access to information.

(Maybe an input or an output)
Combination
of Input & Output

(4) Resources maintained in the system

(5) System used by our system
(part of other system)

Internal logical file → (ILF) ~~help~~ help within the system.

External interface file (EIF) → inf n held by other system that is used by the system being analysed.

⇒ OFP → unadjusted FP

Function point

Name: Code:

functional weighting factors.

| unit | low | Mett. Avg | high |
|------|-----|-----------|------|
| EI | 3 | 4 | 6 |
| EO | 4 | 5 | 7 |
| EQ | 3 | 4 | 6 |
| ILF | 7 | 10 | 15 |
| EIF | 5 | 7 | 10 |

- Q) consider a project with following functional units
 (1) no. of user input = 50
 (2) no. of user output = 40
 (3) no. of user enquiries = 35.
 (4) no. of user files = 6.
 (5) no. of external interface = 7.

Compute unadjusted function point, considering EI, EO, ~~EQ~~ as low , EQ as average & ILF, EIF as high .

$$\begin{aligned}
 & \text{SOLN} \\
 & \frac{50 * 3}{\text{low}} + \frac{40 * 4}{\text{avg}} + \frac{35 * 7}{\text{avg}} + \frac{6 * 15}{\text{high}} + \frac{7 * 10}{\text{high}} \\
 & = 150 + 160 + 140 + 90 + 40 \\
 & = 580 \rightarrow \underline{\underline{UFP}}
 \end{aligned}$$

25-01-18



Q → 8 An application has a following → 10 low EI,
12 high EO, 20 low ILF, 15 high EIF
2 12 avg EQ. Calculate OEP. on

$$\begin{aligned}
 & 3 \times 10 + 7 \times 12 + 20 \times 7 + 15 \times 10 + \\
 & 12 \times 7 \\
 = & 30 + 84 + 140 + 150 + 84 \\
 = & \cancel{30} \quad \cancel{84} \quad \cancel{140} \quad \cancel{150} \\
 = & \cancel{84} \quad 452
 \end{aligned}$$

Q → 3

| | | | |
|------------------|----------|-------|---|
| EI → 10 with low | = 10 × 3 | = 30 | } |
| 18 with avg | = 15 × 7 | = 60 | |
| 17 with high | = 17 × 6 | = 102 | |

$$\begin{aligned}
 EO \rightarrow L &= 6 &= 24 \\
 \rightarrow A &= 0 &= 0 \\
 \rightarrow H &= 13 &= 91
 \end{aligned}$$

$$\begin{aligned}
 EQ \rightarrow L &= 3 &= 9 \\
 A &= 7 &= 16 \\
 H &= 2 &= 12
 \end{aligned}$$

$$\begin{aligned}
 ILF \rightarrow L &= 0 &= 0 \\
 A &= 2 &= 20 \\
 H &= 1 &= 15
 \end{aligned}$$

$$\begin{aligned}
 EIF \rightarrow L &= 9 &= 45 \\
 A &= 0 &= 0 \\
 F &= 0 &= 0
 \end{aligned}$$

General formula

$$UFP = \sum_{i=1}^5 \sum_{j=1}^3 Z_{ij} * w_{ij}$$

↓
Total entities available

weighting factor

Z is count of no. of functional units of type IT

w_{ij} is the associated way correspond to i th row & j th column.

⇒ function point is calculated using the following formula

$$\boxed{F.P. = UFP * CAF}$$

Complexity adjustment factor.

$$CAF = \left(0.65 + 0.01 \sum_{i=1}^{17} f_i \right)$$

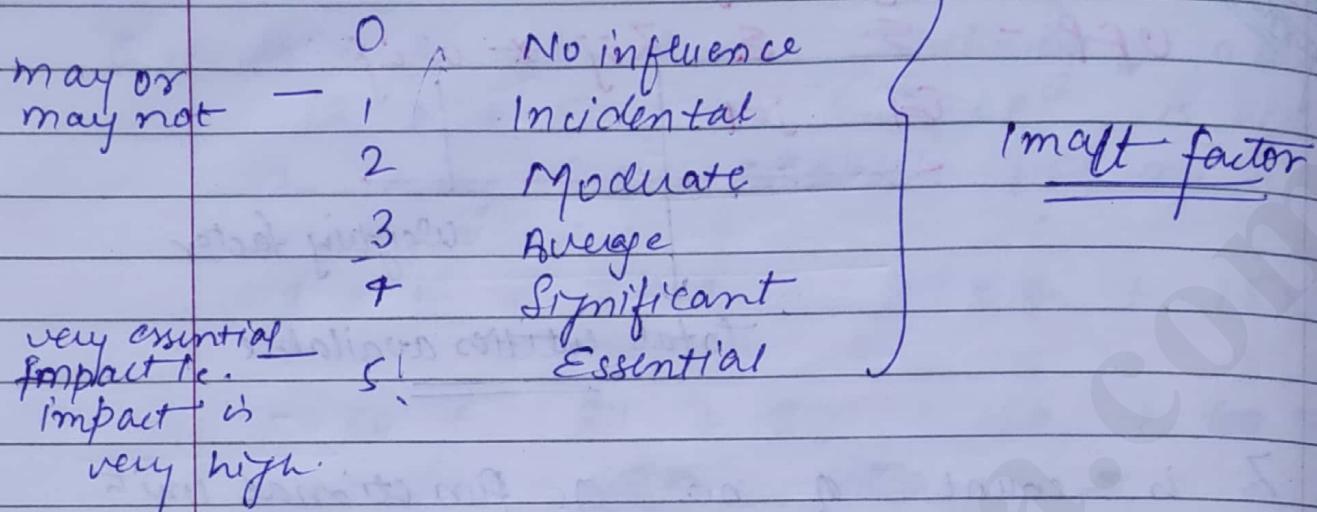
for $i = 1$ to 17.

where f_i is rated based upon degree of influence ranging from 0 to 5 with following set of 17 Questions.

<https://www.tutorialsduniya.com>

Given an input factor & correspondingly we may find some impact on question.

| | |
|----------|--|
| Page No. | |
| Date | |



~~Ques~~ Questions with respect to these 6 Impact factors are

Q-1 Does the system require reliable back up & recovery ?

Q-2 Is data communication required. ?

Q-3 Are ~~they are~~ ^{these} distributed processing sys.

Q-4 Is performance critical. ?

Q-5 Will the system run in an existing heavily utilized operational environment?

Q-6 Does the system require on line data entry ?

Q-7 Does the online date entry require input transaction to be build over multiple screen & operation ?

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 

Q8 Are the master files updated online ?

Q9 Is the input, output, files or enquiries complex?

Q10 Is the internal processing complex ?

Q11 Is the code design to be reusable ?

Q12 Are the conversion & installation included in the design ?

Q13 Is the system designed for multiple installations in different organisations ?

Q14 Is the application design to facilitate to change & ease of use by the user

Example - 1

Assume all complexity factor as average compute function point.

$$\underline{\text{OFP}} = \underline{580} \quad (\text{from 1st example})$$

that means

$$\begin{aligned} \text{CAF} &= 0.65 + 0.01(17 \times 3) \\ &= 0.65 + 0.01 \times 42 \\ &= \underline{1.07} \end{aligned}$$

(3)
from
Impact factor

Now,

$$\begin{aligned} F.P &= UFP * CAF \\ &= 580 * 1.07 \\ \boxed{F.P} &= 620.6 \end{aligned}$$

Example - 2

Given $CAF = 1.10$
compute \underline{UFP} & \underline{FP} .

↓

452 (from Q-2)

$$FP = UFP * CAF$$

$$= 452 * 1.10$$

$$= 497.2$$

Example 3

- In addition to Q3, the system requires
- (1) significant I/O data communication
 - (2) performance is very critical.
 - (3) designed code may be moderately reusable.
 - (4) system is not designed for multiple installation in different organization.
 - (5) Other CAF are average.

$$CAF = 0.65 + \left[\frac{0.0410 \times 3 + 4 + 5 + 2 + 0}{10} \right] * 0.01$$

$$= 0.65 + 0.01 * 41 \\ = 1.06$$

$$FP = 449.49$$

COCOMO - I

(constructive cost model)

| Tables for screen | no. of sources of data tables | | | |
|----------------------|-------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|
| | no. of views contained. | total < 4 server < 2 client < 3 | total < 8 server 2-3 client 3-5 | Server > 3 total > 8 client > 5 |
| 1-3 | Simple | Simple | Simple | Medium |
| 3-7 | Simple | Medium | Medium | Difficult |
| 7-8 | Medium | Difficult | Difficult | Difficult |

for reports

| Tables for reports | no. of sources of data tables | | | |
|-----------------------|-------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|
| | no. of sections contained | total < 4 server < 2 client < 3 | total < 8 server 2-3 client 3-5 | Server > 3 total > 8 client > 5 |
| 0 or 1 | Simple | Simple | Simple | Medium |
| 2 or 3 | Simple | Medium | Medium | Difficult |
| 4+ | Medium | Difficult | Difficult | Difficult |

| Object type | Simple | Medium | Difficult |
|---------------|--------|--------|-----------|
| Screen | 1 | 2 | 3 |
| report | 2 | 5. | 8. |
| GUI component | — | — | 10 |

~~Q8~~ Consider a database project with following clues →

- 1) The application has 7 screens with 4 views each and 7 data tables for 3 servers & 7 clients.
- 2) The ~~app~~ application may generate 2 reports of 6 sections for 8 database tables, 3 servers & 5 clients.
- 3) There is 10% reuse of object points.
- 4) The developers experience & capability in similar environments is low & maturity of environment in terms of capability is also low calculate object point count, new object point & effort to develop this project.

~~Answer~~

→ 7 screens - 7 views each / 7 tables → 3 servers + 7 clients
 for this, complexity is medium (in acc. to table).

2 reports - 6 sections / 8 tables 3 screens
5 clients.
complexity is Δ (Difficult)

(1) To calculate Object point count

$$\begin{aligned} OP = & \text{no. of screen * weight} + \\ & \text{no. of report * weight} - \\ & = 7 * 2 + 2 * 8 + 0 * 10 \\ & = 27 \end{aligned}$$

\leftarrow (Complexity of difficult level from table)

We took $(0 * 10)$ as there was nothing mentioned about the 3GL factor in the question.

NOTE

- Screen represents view of project (in terms of webpages)
- Reports represents tables to be used in project.
- 3GL components are used for graphic components.

\swarrow defined during project planning

→ to normalize project point

$$\Rightarrow \boxed{\text{NOP}} = \frac{O.P \times (100 - \% \text{ reuse})}{100}$$

(New Object Point)

$$= \frac{24 \times (100 - 10)}{100} = \frac{24 \times 90}{100} = 21.6$$

$$\boxed{\text{Effort} = \frac{\text{NOP}}{\text{PROD}}} \quad \text{LOCOMOT}$$

~~Developer's Maturity / cap.~~

~~Table 7~~

| Developers Maturity / cap | very low | low | nominal | high | very high |
|---------------------------|----------|-----|---------|------|-----------|
| Env. maturity / cap. | very low | low | nominal | high | very high |
| PROD | 4 | 7 | 13 | 25 | 50 |

standard unit for effort
↳ person month

$$\text{Effort } E = \frac{\text{NOP}}{\text{PROD}} = \frac{21.6}{7}$$

$$E = 3.086 \text{ P.M}$$

↳ (Person Month)

Note → If one is low and one is high, we consider low one as prod yes and thus effort less. Thus, always consider the lower cost.

| | |
|----------|--|
| Page No. | |
| Date | |

Batein :-

⇒ COCOMO is software estimation model and stands for Constructive Cost Model.

~~###~~ It is one of the most widely used cost estimation model which addresses the following areas.

• (1) Application Composition Model

It is used during early stages of software engineering when prototyping of user interfaces, consideration of various interactions, and assessment of performance and evaluation of technology is considered.

• (2) Early design stage Model

(SRS has been validated & submitted)
It may be used once requirements have been stabilized and basic software architecture is established.

• (3) Post architecture Stage Model

It is used during construction of the software.

~~###~~ COCOMO-II Model requires sizing information such as object point, function point & lines of code.

It uses the following complexity table to associate weight to calculated object points based upon no. of screens, reports & 3GL components (Third Table)

Once complexity is determined, object point is calculated by multiplying the number of object types with their corresponding weights. & to normalise the calculated object point % of reuse is estimated and object point is adjusted to new object point (NOP) with the following relation →

$$\text{NOP} = \frac{\text{OP} \times (100 - \% \text{ reuse})}{100}$$

To derive an estimate of effort based on the computed NOP value, a productivity rate must be derived, which is the ratio of developer's experience and envi. maturity / capability rated on a scale from very low to very high as given in Table 7.

$$\boxed{\text{Effort (E)} = \frac{\text{NOP}}{\text{PRD}}}$$

$$\left. \begin{array}{l} \text{Effort (E)} = 1.4(L)^{0.93} \\ \text{Documentation (DOC)} = 30.97(L)^{0.90} \\ \text{Duration (D)} = 7.6(L)^{0.26} \end{array} \right\}$$

Basic CoComp

Organic

semi
detached

embedded

$$\text{effort} E = a_b (KLOC)^{b_b}$$

a_b , b_b , c_b , d_b } 4 factors

$$D = c_b (E)^{d_b}$$

Development time

| S/W 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 | 0.32 |

Organic

2-50 KLOC

Innovative factor

little

• experienced developers

• familiar with environment

Semi
detached

50-300 KLOC

• medium

• Average experience

Embedded

300 + KLOC

• significant

• real-time sys.

• no previous experience

• new environment.

Q-1

→ A project cost estimated to be 900 Kloc calculate the effort and development time.

$$E = a_b (KWC)^{b_b}$$

$$= 3.6 \times (900)^{1.20}$$

$$= 4772.8137 \text{ pm}$$

$$D = C_b (E)^{d_b}$$

$$= 2.5 (E)^{0.32}$$

$$= 2.5 (4772.8137)^{0.32}$$

$$= 38.07 \text{ months}$$

Q-2

= A project size of 200 Kloc. calculate the effort & D=3, avg staff size, & prod of the project.

Average Staff Size = $\frac{E}{D}$ persons

$$\text{PROD} = \frac{KLOC}{E}$$

(no. of lines of
code made
by person
in a
month
given time)

$$E = 3 \times (200)^{1.12} = 1133.12 \text{ p-m}$$

$$D = 2.5 \times (E)^{0.35} = 2.5 \times (1133.12)^{0.35} \\ = \underline{29.3 \text{ months}}$$

$$SS = \frac{E}{D} = \frac{1133.12}{29.3} = 38.67 \text{ person} \\ \underline{\underline{= 39}}$$

$$PROD = \frac{200}{1133.12} = 0.17650 \text{ kloc/p-m} \\ = \underline{\underline{176.5 \text{ loc/p-m}}}$$

⇒ Software Egn Model

It is a dynamic multi-variable model that assumes a specific distribution of effort for software development. It is derived from productivity data collected over 4000 software projects. Based upon these data, the estimation model is given by the egn.

$$E = \frac{\text{Loc} * B^{0.333}}{P^3} * \frac{1}{t^7}$$

where E = effort in p-m, t = project duration in months.
 B = special skill factor.

| Type of project | KLOC | B |
|-----------------|------|------|
| small | 5-15 | 0.16 |
| large | >70 | 0.39 |

P is the productivity parameter that reflects ~~overall~~ overall process maturity & management practices & is given as.

| Type of project | P |
|--------------------------------|-------|
| real time embedded S/w. | 2000 |
| telecomm & sys/ S/w | 10000 |
| business system application | 28000 |

→ To simplify the estimation process, the software eqn model is converted to another estimation model which defines minimum development time & effort as →

$$t_{min} = 8.14 \frac{LOC}{P^{0.43}} * \text{monthly}$$

$$E = 180 B t^3 \quad P-m, E \geq 20.$$

| | |
|----------|--|
| Page No. | |
| Date | |

Q → for a CAO software the recommended value of P is 12000 with an estimate of 33.2 kloc

$$P = 12000$$

$$LOC = 33.2 \text{ kloc}$$

find min time & effort required for this project. Given the value of B is 0.28.
 $t = ?$

$$t_{\min} = \frac{8.14 \times 33.2 \times 1000}{(12000)^{0.43}}$$

$$= 4761.17977 \quad 56.760$$

Time taken is 56.760

⇒ ~~OS~~ PROJECT SCHEDULING

Scheduling of a project is necessary to provide timeline & to apply check points to evaluate progress of any software project.

Some common techniques for project scheduling →

1. Gantt chart

is a timeline chart which is developed to fix a schedule for a given project. The horizontal axis is time where each task is assigned its own horizontal band of calendar duration and on the vertical axis, all the activities are listed (distinct) and tasks every identified and grouped into categories for a given duration. These

The advantages of a gantt chart are

- 1) The time is explicit & linear
- 2) All tasks are clearly visible
- 3) Deadlines are shown.
- 4) Progress is indicated by filling the task boxes.

It may have following shortcomings:

- 1) Task might not be associated with people.
- 2) Person hours are not indicated.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 

- 3) dependencies are not clearly indicated
- 4) critical paths are not given

Example

work task

1. Identify needs & Benefits

1.1

1.2

1.3

1.4

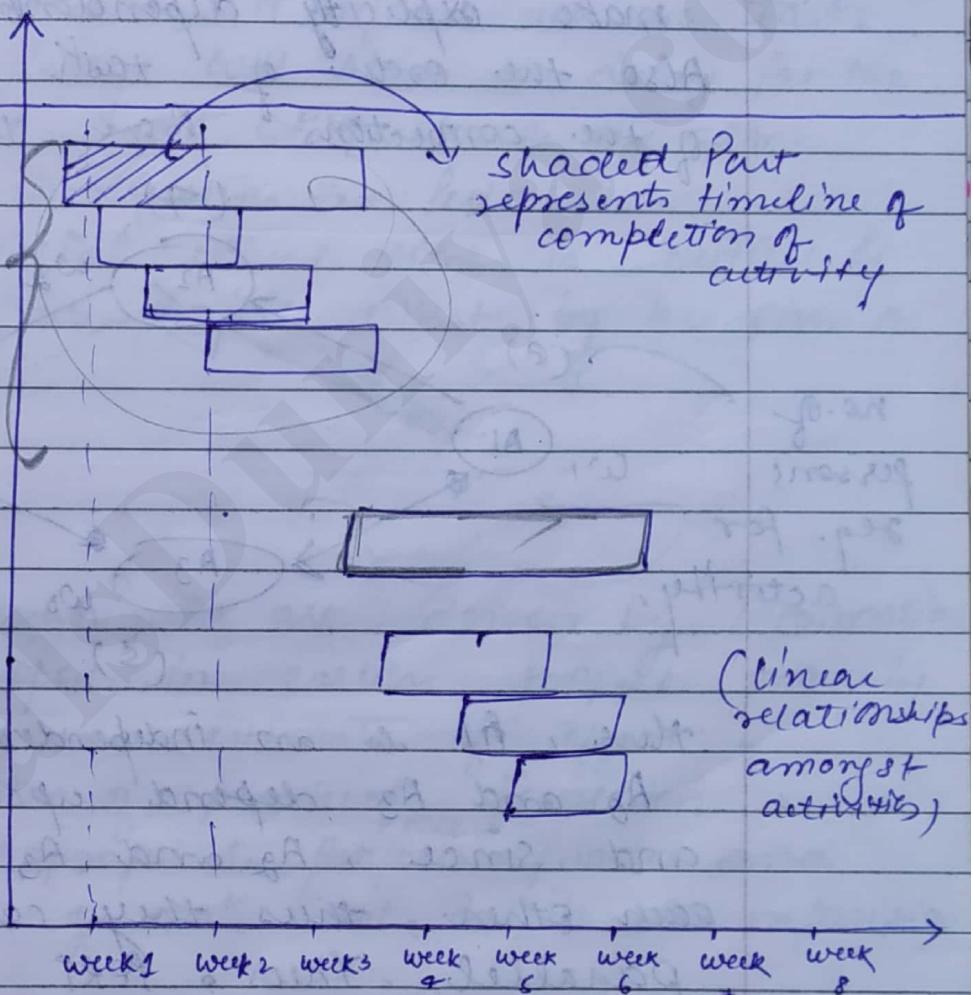
2. Design desired I/O, control

2.1

2.2

2.3

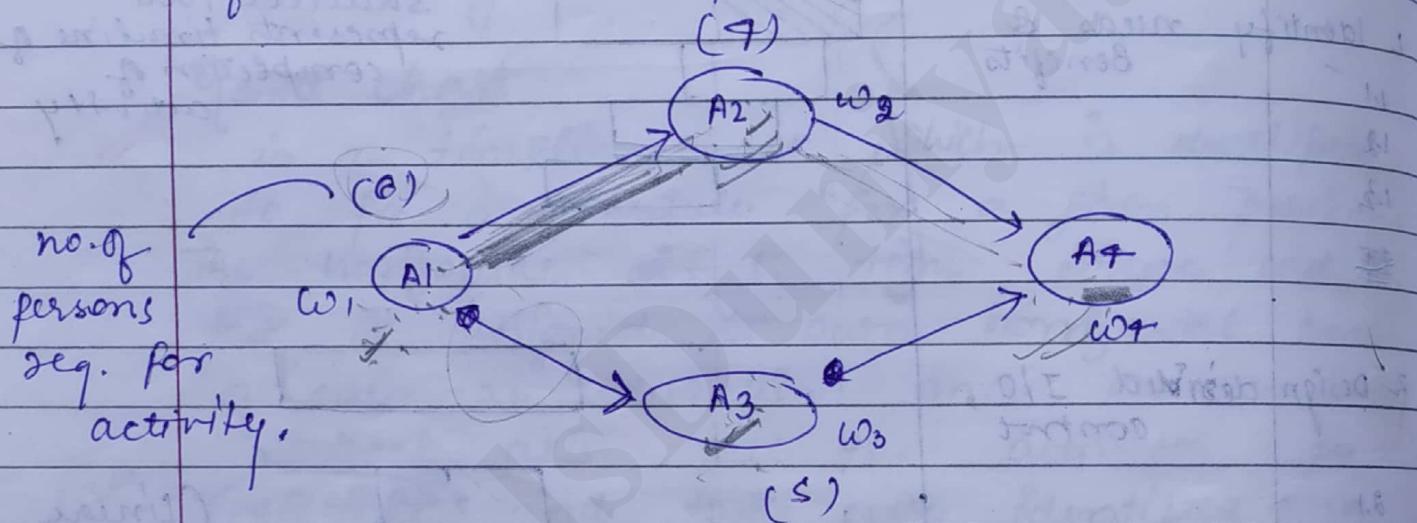
we can extend such activities



~~2~~ PERT

(Program Evaluation & Review technique)
in which the shortcoming of gantt chart are resolved.

The Pert uses a collected series of nodes to make explicitly dependencies b/w tasks.
Also the order of task is given by flow of the connections from left to right.



Here, A₁ is an independent activity but A₂ and A₃ depend upon outcome of A₁. and since, A₂ and A₃ are independent of each other, thus they can be executed in parallel. Thus, Pert represents both linear & non-linear relationship.

Page No. _____
Date _____

orig int = n;

while ($711 \neq 0$) {

rem int = ~~711~~ % 10;

revint = revint * 10 + rem;

$n = n / 10;$

3. CPM (Critical Path Method)

is similar to Pert chart but it includes explicit indication of critical path that a sequence of tasks which defines the minimum amount of time required for the project.

In other words, it defines all the tasks where a delay will lead to delay for the entire project. CPM shares the same strengths & weaknesses as Pert.

CPM is crucial where time is critical & delivery of the project must be given on time.

⇒ All the techniques are driven by information already developed in earlier project planning activities such as estimate of effort, decomposition of product function, selection of appropriate ^{process} project model and identification of task set & its decomposition.

- Interdependencies among the tasks are defined using work breakdown structures which may be defined for the overall project or for individual modules.

The objective of project scheduling is to enable a project manager to define work task, establish dependencies, estimate the resource required & developed some graph, chart or table that helps to track & control the progress of SW project.

(WBS)

- 1) First Step is to specify work breakdown structure,
- 2) assign start & end date & attach human resource & attach other data to each task.

Contingency → a future event or circumstance which is possible but cannot be predicted with certainty.

⇒ Risk Management

proactive strategy

Reactive strategy.

↓
Identify all the risks before & identify strategies for handling risks.

{ early - approach }
first allow risk to come then solve, drawback → project failure

Drawback → It is not possible to predict all risks.

⇒ Risk analysis and management are actions that help a software team to understand & manage uncertainty.

A Risk is a potential problem which may or may not occur but it's a good practice to identify the risk, assess its probability of occurrence, estimate the impact and establish a contingency plan to develop a successful project. Risk Management is a group of activities that involves →

- (1) Risk identifications.
- (2) Risk specifications.
- (3) Risk analysis. [ranked for probability & Impact]
- (4) Risk Management Plan.

→ In the context of s/w engineering risk is concerned with future happenings and it may involve some unpredictable change or choice. With respect to future concern, we have to identify what risk might cost the s/w project to change. With respect to change & choice, the concern is to identify change in customer requirement, development technologies, target environment & methods & tools available and adopted to develop a project.

⇒ Types of Risk Management Strategies

1. Reactive Strategy

This strategy is based on the concept that never take any action about the problem until they appear.

Majority of s/w teams rely on reactive risk strategies which monitors the project for likely risk and resources are set aside to deal with them. More commonly it is observed that software team does nothing about the risk until something goes wrong.

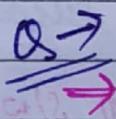
~~Syllabus~~
2

Pro-active Strategy

A proactive strategy begins long before technical work is initiated. Potential risks are identified, their probability & impact are assessed & they are ranked by their importance.

The S/W team establishes the plan for managing the risks.

The primary objective is to avoid risk but bcoz not all the risk can be avoided or identified the team develops a contingency plan that enable respond in controlled and effective manner to minimize/avoid risk.



Characteristics of S/W risk

1. Uncertainty

The first characteristic of S/W is uncertainty. i.e. a risk may or may not happen and there is no 100% probable risk associated with a given project and if they exist they must be considered as constraints during requirement analysis phase.

2. Loss

If the risk becomes a reality, unwanted consequences or losses will occur. Thus, it is required to quantify the level of uncertainty.

and degree of loss associated with each risk.

⇒ Effect of S/w Risks.

1. project risk threatened the project plan. i.e. if the project risk becomes real, it is likely that project scheduled will slip and time & cost will increase.
2. Technical risk threatened the quality & timeliness of s/w to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible
3. Business risk threatened the viability of s/w and it may put the project in danger - some common business risks are:
 - a) Building a product that no one really wants
 - b) Building a product that no longer requires.
 - c) Building a product that sales team does not understand how to sell.
 - d) Loosing the support or charge of focus of people in management.
 - e) Loosing budget.

~~Q2~~ Categories of Risk

1. Known Risk

These are the risks that can be uncovered after careful evaluation of project plan, the business & technical environment in which project is developed and other associated resources such as unrealistic delivery date, poor documentation or poor development environment.

2. Predictable Risk } depends on history }

that are identified from past project experience, such as staff turnover, poor communication & inadequate effort for a given project.

3. Unpredictable Risk

These are the risks which are extremely difficult to identify in advance.

(Security threat)

13-02-18

| | |
|----------|--|
| Page No. | |
| Date | |

⇒ 7 Principles of Risk Management

- 1. Maintain a global perspective
- 2. Take a forward looking view.
- 3. Encourage open ~~con~~ communication.
- 4. Integrate risk into software process.
- 5. Emphasize a continuous process.
- 6. Develop a shared product vision among stakeholders.
- 7. Encourage teamwork.

→ Global perspective.

Identify all the risks involved in project, constant all organization & business. { entities related

- As implementation of project proceeds, to project keep on analyzing risks
- Software involves team work not a single person therefore communicate with involvement every team member of project.
↳ (stakeholders)

Identify overall impact factors probability

- Analysis of risk should be a continuous process.
- ⑥ → Isolation should not be there, all the tasks must be properly performed in the vision of all stakeholders.

→ First Process of Risk Management

(1) Risk Identification

Risk Ident'n is a systematic effort to specify threats to the project plan. By identifying known and predictable risks the project manager may take necessary steps to avoid and control all these risks.

All the identified risks may be broadly classified into two categories .

- 1) Generic Risks

these are the potential threat to every s/w project

- 2) Project Specific Risk

it can be identified with clear understanding of technology , people & environment that is specific to the software project that is to be built.

One common Method to identify risk is to create a risk item checklist . All the predictable & known risks are analysed under following subcategories →

- (1) product size (p.s)
- (2) business impact (Bu)
- (3) Stakeholder characteristics (sc)
- (4) Process definition (pd)
- (5) Development environment (de)

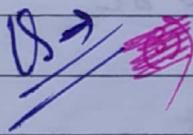
- (E) Technology (Te)
- (F) Staff size and experience (ST)
- (G) Cost risks (CU)

15-02-18



Assessing Overall Project Risk.

we have to answer several questions that are formed by a detailed survey of similar type of project, and based upon the answers received, the risk associated with the project mainly may be identified. If the questions are answered negatively then some steps should be adopted to manage the risk and the degree to which the project is at risk is directly proportional to no. of negative responses received.



Risk components & drivers.

The risk components are identified in the following categories.

- (1) Performance risk

The degree of uncertainty that a product will meet its requirements.

- (2) Cost risk

The degree of uncertainty that a product budget will be maintained.

- (3) Schedule Support Risk

The degree of uncertainty that the project is easy to correct, adapt & enhance.

- (7) Scheduled Risk

The degree of uncertainty that the project will be delivered on time.

The impact of each work deliver is divided into one of the 4 impact categories

- Q → 1) Negligible 2) Critical
 3) Marginal 4) Catastrophic

| Component Category | Performance | Support | Cost | Schedule |
|--------------------------------|---|--|----------------------|--|
| (due to (high) sudden changes) | ① Perform meet reqm | ★ | \$ 500K | ↑ cost, delayed delivery, significant financial crisis |
| catastrophic | ② Limited/no technical feat. | ③ non responsive failure to meet segm lead to degradation of performance | ↑ cost | unachievable control list. |
| wrongly interpreted | reduction in technical features | minor delay in modification | shortage of funds | slipping of Ioc |
| critical | failure to meet reqm & degradation in secondary objectives. | | (1K to 500K) \$ cost | |
| 3. marginal | min/agan in tech. performance | responsive s/w support | sufficient funds | realistic schedule |
| 4. negligible | no agan in performance | easily supportive | budget overrun | early delivery |

Ioc → Unachievable control list.

④ Failure

→ when cost exceeds

$\left\{ \begin{array}{l} \$ 500K \rightarrow \text{catastrophic} \\ \$ 100K - 500K \rightarrow \text{critical} \\ \$ 1K - 100K \rightarrow \text{marginal} \\ \text{below } \$ 1K \rightarrow \text{negligible} \end{array} \right.$

19-01-18

Risk projection

| | |
|----------|--|
| Page No. | |
| Date | |

Developing a Risk Table

Q-7

| Risk | Category | Prob. | Impact | RMM |
|------------------------------|----------|-------|--------|-----|
| 1. Size estimate is sig. low | PS | 60 | 2 | |
| 2. Late no. of user | PS | 30 | 3 | |
| 3. Less Reuse | PS | 70 | 2 | |
| 4. Delivery deadline not met | BU | 50 | 3 | |
| 5. cost exceeds | CU | 40 | 1 | |
| 6. Custom Regm changed | PS | 30 | 2 | |
| 7. Funding lost | BU | 90 | 3 | |
| 8. Lack of planning | TE | 50 | 2 | |
| 9. Staff inexperienced | ST | 40 | 3 | |

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 

- A risk table is a simpler technique for ~~other~~ risk projection.
The first column list all the risks that are identified.
They are categorised and probability and impact of each risk is listed in the next column. Once the risk table is completed
→ It is sorted by probability and by impact.

A cut-off line which is drawn horizontally at some point in table implies that only risks that are above the cut-off line will be given further attention.

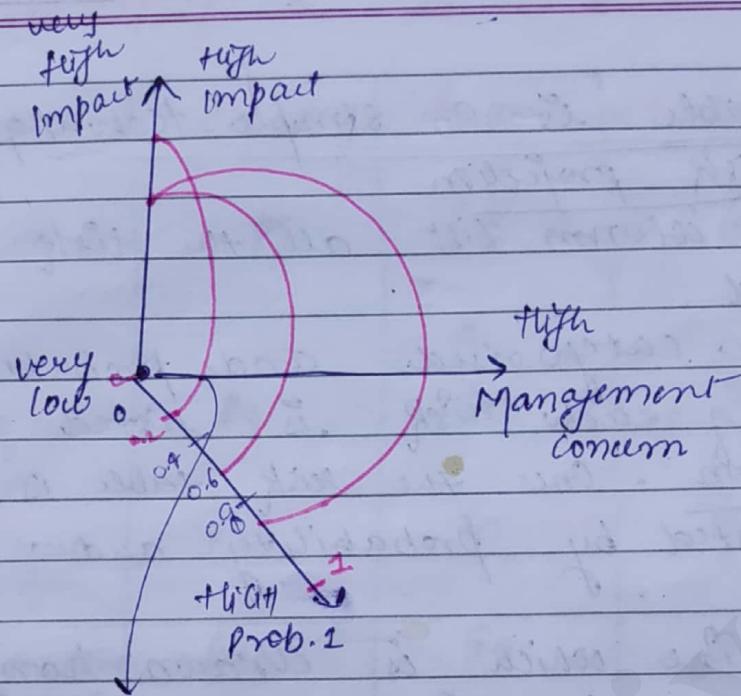
Risks that falls below the line are re-evaluated to accomplish second order prioritisation.

Risk Impact & Probability have a distinct influence on management concern.

→ A Risk factor with high impact & low probability should not absorb significant amount of ~~risk~~ management time.

However, high impact risk with moderate to high probability and low impact risk with high probability should be carried forward.

Risk with high probability should be ~~concerned~~ first concerned.



A risk with very high impact & with very low probability have very low management concern.

⇒ Assessing the Risk Impact (Estimate)

Three factors that affects the consequences that are likely if a risk occurs.
nature, scope & timing.

- Nature of the risk indicate the problems that are likely if it occurs - for ex → poorly defined external interface will lead to system integration problems
- Scope of a risk combines the ~~new~~ severity with its overall distribution. for ex → how much the project will be affected if the given risk appears.

- Timing of a risk concerns when & how far the impact will be felt.

Numerical Example

Risk = Only 70% of software components for reuse will be integrated into applications. The remaining functionality will have to be custom developed. Calculate the risk exposure given that risk probability is 80%. No of reusable components is 60. Average component size is 100 LOC & cost is 19\$.

- reuse
- we can ~~use~~ of 70% & rest 30% is custom developed

60 reusable components

70% → 70% = 18 reusable components will be custom developed

30% → 30% = 12 reusable components will be custom developed

$P = 80\%$

(Component size) $E = 100 \text{ LOC}$

$C = 19\$$

$$\boxed{\text{Risk Exposure (RE)} = P * C} \rightarrow \begin{array}{l} \text{(cost to the project if the} \\ \text{risk occurs)} \\ \text{(probability)} \end{array}$$

$$\begin{aligned}C &= 18 * 100 * 14 \\&= 25200\end{aligned}$$

$$P = 80\% = \underline{\underline{0.8}}$$

$$\Rightarrow R.E = 0.80 * 25200 \\= \underline{\underline{20160}}$$

\Rightarrow Risk Refinement

During early stages of project planning a risk may be stated quite generally and it is possible to refine the risk into a set of more detailed risk which is easier to mitigate, monitor & manage.

One way for refine a risk is to represent it in CTC format i.e. Condition, Transition, Consequence.

A general CTC takes the following form.

Given that Condition > then there is a concern that (possibly) consequence >

for eg →

given that < all reusable software components may not conform to specific design standards > then there is a concern that (70%) < of the reusable modules may be integrated into built-in systems resulting remaining 30% components to be custom engineered >

→ The risk may be defined in the following manner.

1. Certain reusable components were developed by a third party, with no knowledge of internal design standards.
2. The design standards has not been solidified and may not conform to existing reusable components.
3. Implementation language is not supported on the target environment..

*(new
sub-variants
may
increase &
w/t
project)*

Imp

RISK MITIGATION, MONITORING & MANAGEMENT

R M M M

~~Do~~

(to reduce
effect of
risk)

(to continually
sense)

(adapt a
policy
to manage
risk)

- A risk management strategy is included in the software project plan and it can be organised into separate risk mitigation, monitoring & management.

Risk mitigation is the risk avoidance activity and in a proactive approach risk avoidance is always the best strategy. This is achieved by developing a plan for risk mitigation. Example →

Eg →

high staff turnover is noted as project risk
eg, and based upon the past history the
likelihood is 70% and impact is
critical.

Develop a strategy to mitigate this risk.

Risk Monitoring is a project tracking activity
which proceeds after the commencement
of project. The primary objectives of project
monitoring are →

- 1. To assess whether predicted risk occur
and to analyse its effects.
- 2. To ensure that risk aversion steps
defined for the risk are being properly
applied a
- 3. To collect the information that can be used
for future risk analysis.

Example →

In case of high staff turnover, the
monitoring activity includes measurement of
project pressure is to calculate the degree of
turnover to identify interpersonal relationships &
potential problems and to carry out market
survey for availability of jobs

Risk Management & contingency plan assumes that mitigation efforts have failed and the crisis becomes a reality. Continuity with example, the project is well underway and a no. of people announce that they will be leaving! If the mitigation strategy has been followed, then backup is available, information is documented & knowledge has been dispersed across the team. For this, a risk information sheet is been prepared and the format for RIMM sheet is as follows.

from
Book

Risk Information Sheet

| Risk ID | Date | Prob. | Impact |
|---------|------|-------|--------|
|---------|------|-------|--------|

Description:

Refinement:

subcond "1"

Mitigation/
monitoring

1.

2.

3.

Risk Mng/
contingency plan/trigger

Current state:

estimated:

Unit - 5

Software And Quality Management

software quality can be defined as an effective software process applies in a manner that creates a useful product & produces a measurable value for those who produce it & those who use it.

Software quality Attributes.

- (1) Performance & feature quality
- (2) Reliability → provides services without error
- (3) Conformance
- (4) Durability & serviceability
- (5) User Perception and Build Perception
- (6) Correctness → concerns to specific specifications
- (7) Efficiency → min. resources → max. output
- (8) Usability
- (9) Maintainability → used after product Released
- (10) flexibility
- (11) Portability
- (12) Reusability
- (13) Inter operability.
- (14) Robustness
- (15) High Availability / Durability .

system will work efficiently irrespective of no. of errors (e.g. -gmail)

The degree to which
(16) Richness { Product is rich in its features }

⇒ Achieving software quality { e.g. good management practices & sound principles }

It is the result of good project management and solid s/w engineering practices.

Management and practices are applied within the context of following activities.

1. S/w Engineering Methods

To build a high quality s/w we must completely understand the problem to be solved. Capable of creating s/w design that conforms to the said problem & standards & that includes characteristics that lead to s/w that exhibits several quality attributes.

2. Project Management Techniques -

several project management technique may be included to develop a high quality software . some of them are -

- Project estimation - estimate cost of project in advance
- Risk Planning -
- Establish Dependencies. - (establish relation b/w modules , ram)

• Project Planning

3) Quality Control & testing, feedback

Quality control encompasses a set of s/w engg. actions that helps to ensure that each work product meet its quality goals. Models are reviewed and inspected to ensure correctness & consistency of the project. The goal is to uncover & correct errors before testing. A series of testing steps is applied to make the software error free. A combination of measurement & feedback allows a software team to tune the process when any of these work product fails to meet quality goals.

4) Quality Assurance (auditing)

[Combines & establishes the infrastructure that supports all pts stated above] It establishes the infrastructure that supports solid s/w engineering principles and methods, rational project management & quality control actions to build high quality s/w.

It consists of set of auditing and reporting functions that assess the effectiveness & completeness of quality control actions. The goal of quality assurance is to provide management & technical staff with data necessary about the product quality.

• Software Reviews

They are a filter for s/w process which are applied at various pts during s/w engg that aims to uncover errors and defects.

that can be resolved removed. The need for s/w review is correction and adaptation of any changes that may generate during software designing process. Software Reviews are software engg. work products which are purified at every phase of s/w development.

→ Many diff. types of reviews can be conducted as part of s/w engineering.

- (1) Informal Meeting. → (not cont.; checkpoint based)
- (2) formal presentation of s/w architecture to customer management or technical staff

(3) Technical Review - It is the formal & most effective filter for quality control that may include peer reviews, casual reviews, walk-throughs & inspections.

(group activities)
(can be done by anyone)

↳ more formal and done by a dedicated team

⇒ Cost impact of Software Defects

A s/w product may include defect or fault that implies a quality problem that is discovered after the software has been released to end users. [It is different from error which depicts a quality problem that is discovered by s/w engineers before the software is released to end users]. The

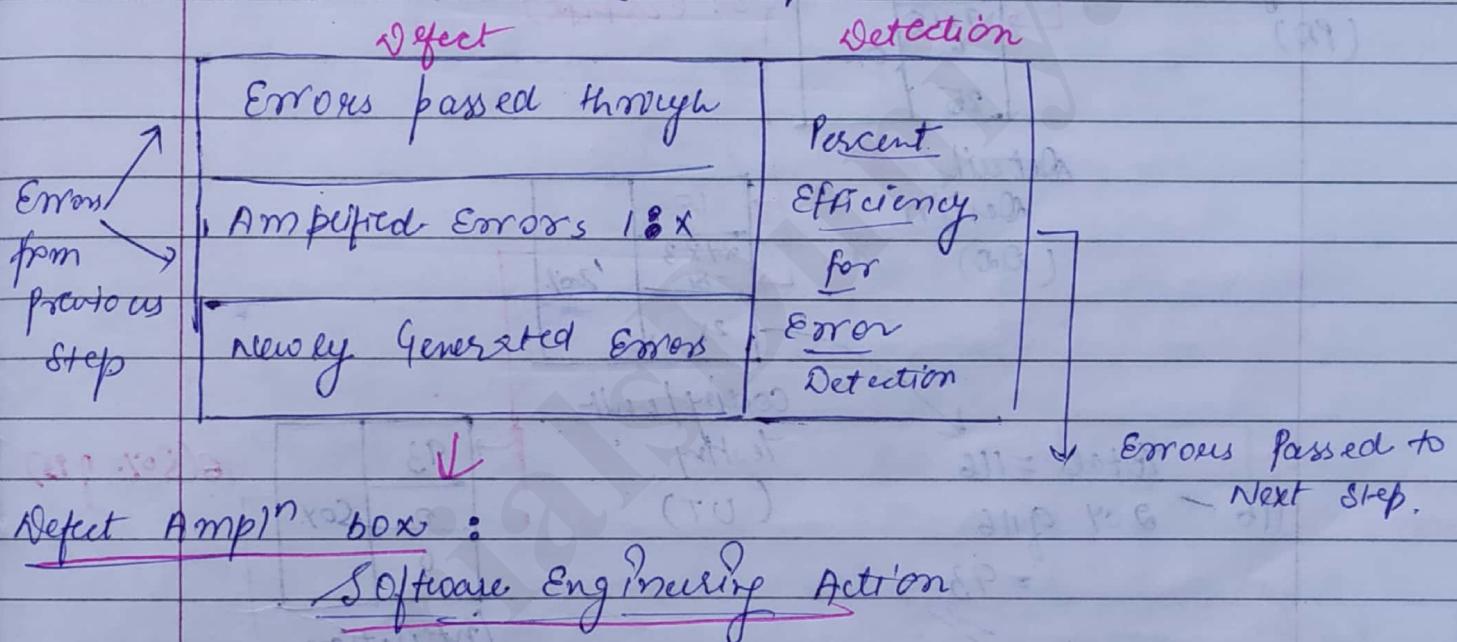
~~s/w quality problem at user end~~
primary objective of technical reviews is to find errors during the s/w development process so that they will not become defects after release of the software]. Early discovery of more

is required so that they do not propagate
to defects in the next step in software
process

→ 8-02-18

Defect Amplification & Removal

A defect Amplⁿ model is used to illustrate the generation and detection of errors during design and code generation phase of software process. The model uses a box to represent software engineering action. This model is depicted as



→ Example for Defect Amplification Model without reviews.

| | | |
|----|---|----|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 10 | 0 | 10 |

Preliminary
Design
(PD)

| | |
|----------------------------|---|
| 6 | 0 |
| $\frac{4 \times 1.5}{= 6}$ | 0 |
| 25 | 0 |

$$37 \cdot (25 + 6 + 6)$$

Detail
Design
(DD)

| | |
|----------------------------|---|
| 10 | 0 |
| $\frac{27 \times 3}{= 81}$ | 0 |
| 25 | 0 |

20%

Coding/unit
Testing
(UT)

| | |
|----|---|
| 93 | 0 |
| 0 | 0 |
| 0 | 0 |

46 (50% of 93)

Integration
Testing
(IT)

| | |
|----|---|
| 46 | 0 |
| 0 | 0 |
| 0 | 0 |

23 (50%)

Validation
Testing
(Vt)

| | |
|----|---|
| 23 | 0 |
| 0 | 0 |
| 0 | 0 |

Latency

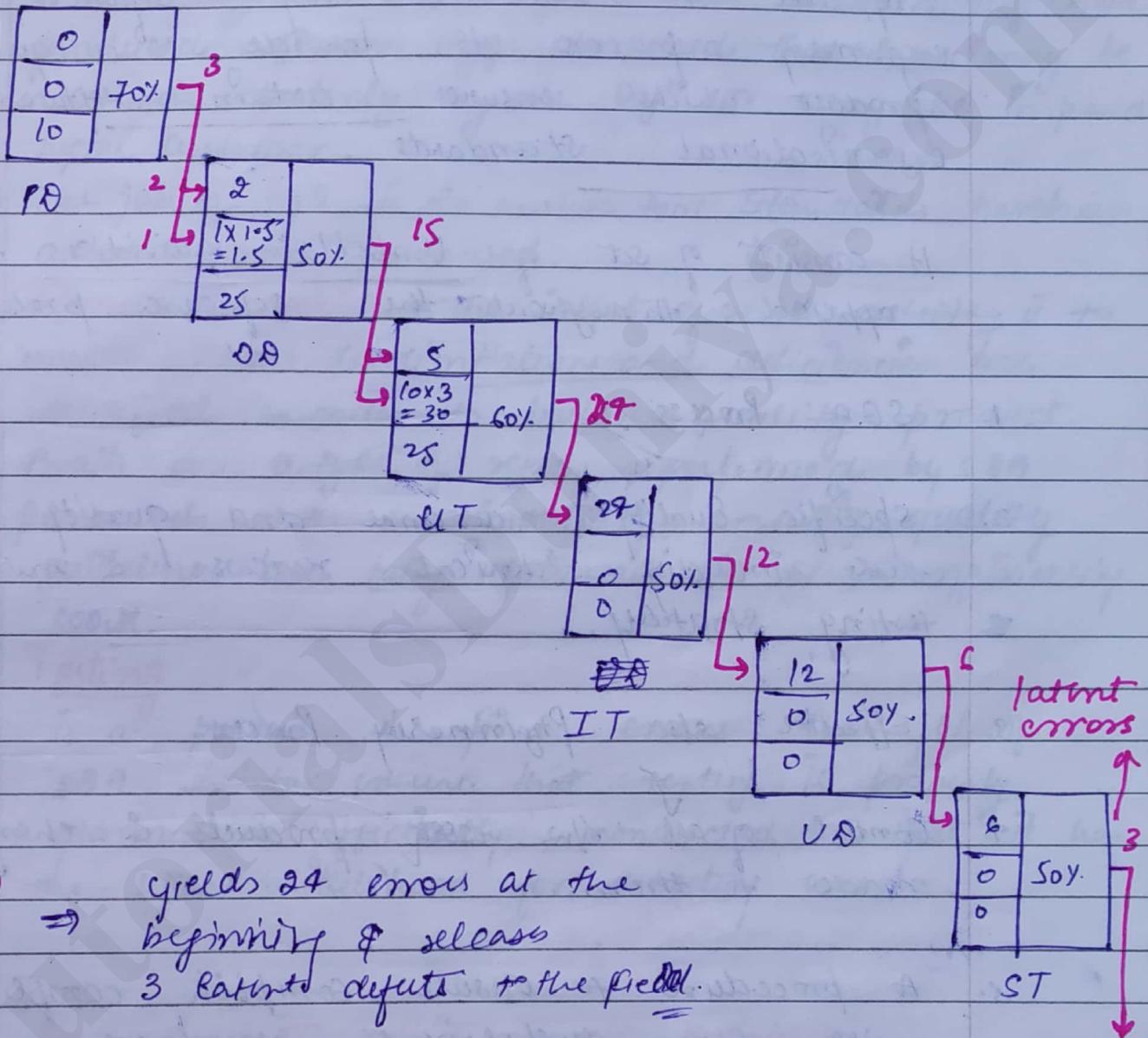
System
Testing
(ST)

yields 94 errors at the
beginning of testing &
Releases 18 latent defect to
the field.

(defects)

After release of
software there are 12
errors because of defects.

⇒ Example for defect amplification model with review at each step.



(56.7% \approx)

⇒ yields 24 errors at the beginning & release
3 latent defects to the field.

latent errors
(12)

⇒ A cost analysis indicates that the process with no reviews produce 3 times (approx) more than process with reviews, taking the cost of correcting latent defects into account

Software Quality Assurance (SQA)

SQA consists of set of monitoring activities applied to each & every software engineering process to provide a quality product which ensures certain software & organisational standards.

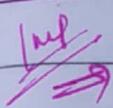
It consists of set of Umbrella activities that is applied throughout the software process.

1. SQA Process
2. specific quality assurance and quality controlled tasks including technical reviews & multi-tier testing strategy.
3. Effective s/w Engineering Process.
4. Control of all s/w work products & changes made to them.
5. A procedure to ensure complete compliance with s/w development standards.
6. Measurement & reporting mechanism.

21-03-18

WEEK
20
2018

| | |
|----------|--|
| Page No. | |
| Date | |



Elements of SQA

- (1) **Standards** → IIEEE, ISO and other standard org'n have produced software engg. standards that have / may be adopted voluntarily by the org'n or may be imposed by the customer.
- The job of SQA is to ensure that standards have been adopted and followed.

- (2) **Review & Audits** → The intent of review activity is to uncover errors. The limitations and deficiencies are identified in order to build a quality product.
- Audits are a type of review performed by SQA personnel with the intent of ensuring quality guidelines are being followed for s/w engineering work.

3. Testing

is a process to find errors. The job of SQA is to ensure that testing is properly planned & efficiently conducted so that it has the highest likelihood for detecting errors.

4. Errors / defect collection and Analysis.

SQA collects and analyse errors to better understand how errors are introduced and what actions should be taken to eliminate them with the intent to restrict error to become a defect.

5. Change Management

change is one of the most disruptive aspect of software process. If it is not properly managed it can lead to confusion and may lead to poor quality. SQA ensures that adequate change management practices have been adopted.

6. Education

for software improvements education of software engineers, managers & other stakeholders is an integral software engineering practice. The SQA is the key member to initiate and develop such kind of educational program to enhance technical skills of the project staff.

7. Vendor Management.

The job of SQA is to ensure that high quality SW results by suggesting specific quality practices that the vendor should follow & incorporating quality mandates as part of any contract with an external vendor. Based upon this 3 categories of softwares are identified.

- (1) Shrink wrapped Packages { ms office }
- (2) Tailored project { component of above category }
- (3) Customised project { specifically tailored/engineered } as per the requirement

8. Security Management

With the increase in cyber crime & several security threats every SW organisation should adopt policies

that protect data at all levels. SQA ensures that appropriate process and technology is used to achieve security.

9. Safety & Risk Management

SQA is responsible to assess the impact of software failure and to initiate the steps required to reduce the risk - SQA ensures that the Risk Mng. activities are properly conducted and risk related contingency plan have been established.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

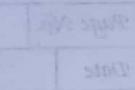
facebook

WhatsApp 

twitter 

Telegram 

19-03-18



| | |
|----------|--|
| Page No. | |
| Date | |

→ Product Metrics

- Measurement is the process by which numbers of symbols are assigned to the attributes of entities in the real world according to the clearly defined rules.
- Product Metrics helps SW eng. to gain insight into the design & constn of SW by focusing on specific measurable attributes of SW eng. work products.
- To carry out qualitative assessment is not enough product metrics provides a basis from which the development steps are conducted more objectively and accessed more quantitatively.

→ Difference between Measures, Metrics & Indicators

individual quantity (degree of affinity)
↑ (degree of affinity)
in measure

(relation b/w 2 measures)

- Within the context of software engineering, a measure provides a quantitative indication of the extent, amount, dimension, capacity or size of some attribute of a product or process.
- Measurement is the act of determining a measure.
- Metric is a quantitative measure of the degree to which a system, component or process possesses a given attribute. Metric establishes relation b/w 2 measures. or It is a degm of defining measures.
↳ SW when a single datapoint has been collected.

if - no. of errors uncovered, a measure has been established. Measurement is the collection of one or more these data points and software metric relates those individual data points by establishing some formula for example → average no. of errors per review.

highlight

or
assessment
of a
measure-
ment

• Indicator is a metric or combination of metrics that provides insight into s/w projects process, a s/w project or the product itself. An indicator provides enables s/w engineers to adjust s/w process and to improve s/w products - for example, measurement dashboards are used to monitor progress and initiate change when required.

Metrics for Requirement Model

Read Pg → 619 to 623

Process and Project metrics

S/w process and project metrics are quantitative measures to gain insight the efficiency of s/w process and projects that are conducted using the process as a framework. The first

~~first~~ step is to collect data for quality and productivity. ~~second~~ These data are analysed in order to find shortcomings compared against past averages and assessed to determine whether improvements have occurred. ~~third~~ Metrics are used to pin-point problem areas, so that remedies can be developed and s/w process can be improved.

⇒ Metrics in process and project domain

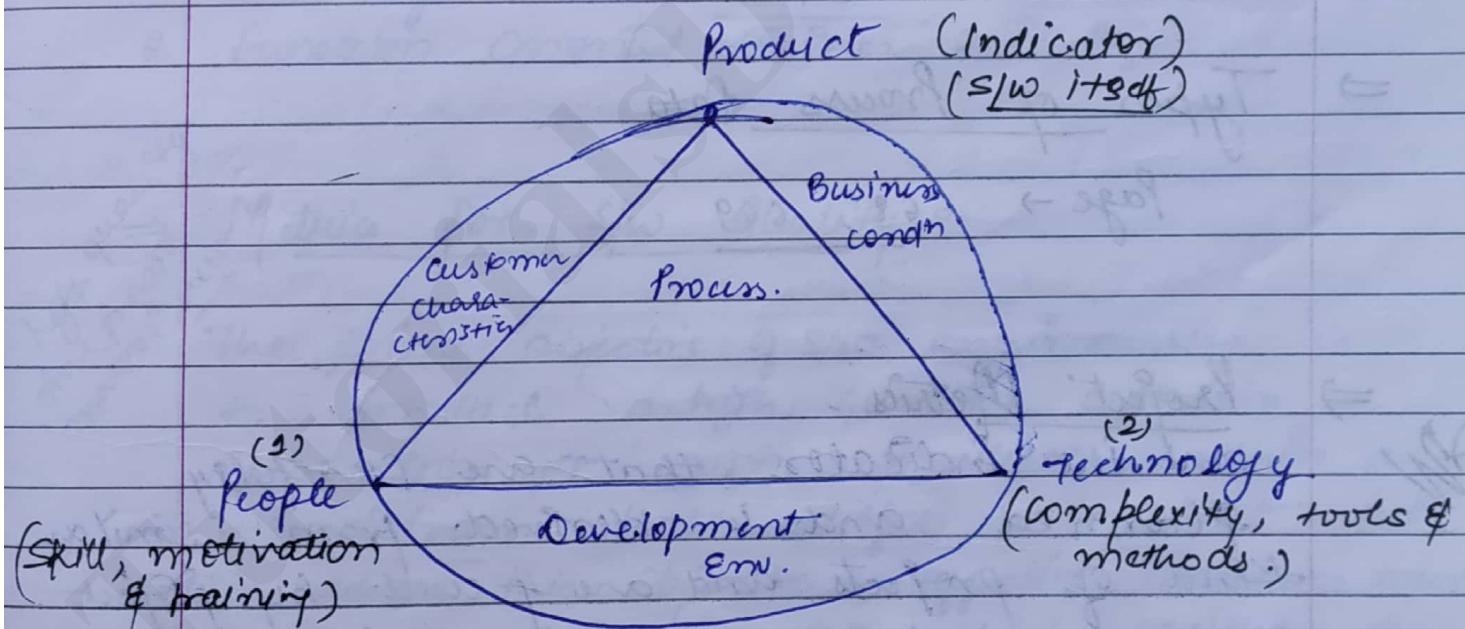


- ① Process metrics are collected app across all the projects and over long periods of time. The intent is to provide ~~and~~ of specific process indicators for s/w process improvement.
- ② Project metric enables a s/w project manager to →
 - 1) Assess the status of ongoing process/project.
 - 2) Track potential risks.
 - 3) Uncover problem areas.
 - 4) Adjust work flow or fast. {adjust timeline chart}
 - 5) Evaluate the Project team availability to control work product quality.
 {continuous evaluations}

⇒ factors affecting process Metric & milestones {

- To improve the measurement process ; some specific attributes of the process are identified , a set of meaningful metrics based upon these attributes is developed and then indicators are provided to assess the quality and productivity and are analysed for improvement .

⇒ Determinants / factors .



- The factors that have a performed influence on S/w quality and organisational performance are people , technology and product .
- The skill and motivation of people is the key factor that influence quality & performance

- The complexity of the product can have a substantial impact on quality and team performance.
- The technology which includes methods and tools that populates the process has an impact on productivity and quality.
- In addition, a given process executes within a circle of environmental conditions. ex → integrated s/w tools, business condition (deadlines) and customer characteristics (ease of communication.)

⇒ Types of Process Data

Page → 668, 669

⇒ Project Metrics -

~~QW~~ all the indicators that are carefully planned and is derived from similar kind of projects and are used by project managers and s/w team to adopt project work flow and technical activities.

- Metrics collected from past project are used as a basis from which effort & time estimate are made for current s/w work. As the project proceeds, measures of effort & time are

compared to original estimates, and the project manager uses this analysis to monitor & control progress. These project metrics are used to minimize development schedule by making adjustments to avoid delay and mitigate potential problems and risks. These are used to assess product quality on an ongoing basis and activities may be modified when necessary.

⇒ Software Management

1. size Oriented Metrics. { LOC }
Page → 674 LOC per FP.
2. Function Oriented Metrics { FPF }

~~High quality~~ ~~Time frame~~ ~~market need~~ Metrics for s/w Quality

The prime objective of s/w engineering is to produce a high quality system's application or work product within a time frame that satisfies a market need. To achieve this objective, effective measures must be covered with modern tools within the context of mature s/w process, that results into a high quality software.

The quality of a system is as good as the requirements that describe the problem, the design that models the sol'n, the code that

The module must be loosely coupled & highly cohesive

| | |
|----------|--|
| Page No. | |
| Date | |

leads to executable programs and test cases that exercise the software to uncover the errors.

Attributes for Measuring quality

(1) Correctness -

It is a degree to which the SW performs its required functions. The most common measure is defects per kloc, where defect are those problems reported by the user after the program has been released for use.

(2) Maintainability -

It is the ease with which a program can be corrected if an error is encountered, adapted if environment changes or enhance if requirements change. There is no direct way to measure maintainability but some indirect measures may be used.

for ex -

Mean Time To change (MTTC) - It is the time to analyse the change request, design an appropriate modification, to implement the change, tested & test it & distribute the change to all the user.

26-03-18

⇒ function Oriented Design

The design activity begins when the requirement document for the s/w is available and its architecture as we design & define.

It gives the blueprint or plan for a solution in terms of components that are combined to become a system. That is, the design exercise determines the module structure of each component in a architecture.

The design process is divided into two levels. At the first level, the focus is to determine which modules are needed for the system, the specification of these modules & how the modules should be interconnected. This level is called system design or top-level design. In the second level, the internal design of the modules and how the specifications of the module can be satisfied is decided. This is called as Detailed Design or logic Design.

In function Oriented design approach, a system is viewed as a transformation function, transforming the inputs to the desired output.

⇒ Design Principles

- { what are the basic principles of system designing phase }
- 1. correctness.
 - 2. verifiable
 - 3. complete
 - 4. traceable
 - 5. efficient { use least resources }
 - 6. simple and understandable.
 - 7. Maintainable
 - 8. Effective

⇒ Problem Partitioning and hierarchy.

The complexity of SW & the implementation of large problems is avoided bcoz it results into a complicated design which is poor in maintainability. for solving large problems , the basic principle of divide and conquer is adopted, where the complete problem is divided into smaller pieces and each piece can be conquered separately .

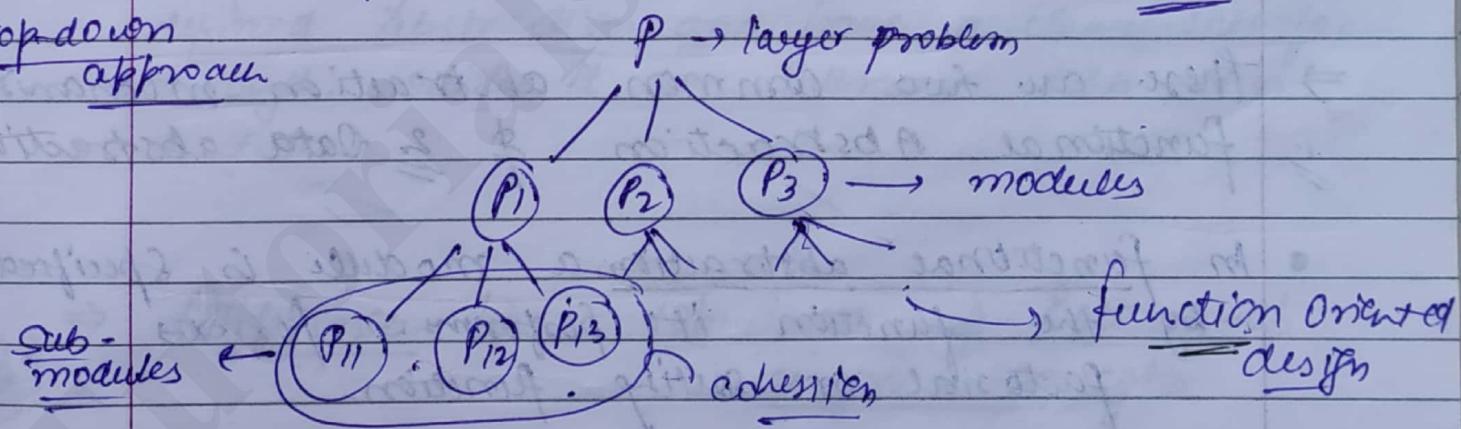
for SW design , the goal is to divide the problem into manageable small pieces that can be solved separately . The basic idea is to reduce the cost and complexity of solving the entire problem as compared with the problem when it is divided into pieces. The different pieces may not be entirely independent of each other & they have to cooperate & communicate to solve larger problem.

This ~~compl~~ communication adds complexity which increases as no. of components and their dependency increases - So the design is aimed to identify the partitioning point where further partitions have to be stopped..

If a module can be modified separately, it is called as independent module and it is desired that we must have more independent modules in our system. Total independence of modules is not possible but the design process should support as much independence as possible..

The design produced by problem partitioning can be represented as hierarchy of components & whose relationship between these elements can be depicted.

Top down approach



⇒ Abstraction

An abstraction of a component describes the external behaviour of that component without bothering with the internal details that produce the behaviour.

Abstraction is an indispensable part of the design process & essential for problem partitioning. Problem partitioning results into components which may interact with other components. If the external behaviour of other components is known to the designer, it reduces the complexity to understand the details of those components, which allows the designer to concentrate on one component and abstraction of other components.

During the design process, abstractions are used in reverse manner. & the components that do not exist, their design is carried out.. & for other components abstract specification are used.

- ⇒ There are two common abstraction mechanism
 - 1. functional Abstraction & 2. Data abstraction
 - In functional abstraction, a module is specified by the function it performs. for ex → factorial or sorting function.
 - The second unit of abstraction is Data Abstrⁿ, The s/w is modelled upon the real world entities which is represented as by certain data structures and set of operations performed on these data structures. from outside an object the internal of the object are hidden, only

Operations ^{on} of the object are visible. Data abstraction forms the basis for object-oriented design.
for example. → banking transactions.

→ Modularity

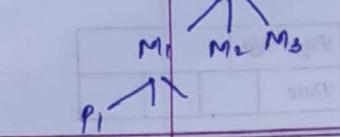
A system is considered modular, if it consists of discrete components so that each component can be implemented separately and once changed to one component as minimal impact on other components. Modularity helps in system debugging, in system repair & in system building. A slow system cannot be made modular by chopping it into pieces. For modularity, each module needs to support a well-defined abstraction and have a clear interface through which it can interact with other modules.

81-PO-8

⇒ Difference b/w Top-down & Bottom-up design strategies

Top-down approach starts from highest level component and proceeds through lower levels.

Bottom-up approach starts with a lowest level component and proceeds through progressively higher levels to the top level component.



Top-down

- Identification of major components & decomposing into lower level components is done until desired partitioning level is achieved.

- Stepwise refinement is adopted.

- Top-down approach is suitable if specification of systems are clearly known and the system development is started from scratch.

Bottom-up

- It starts with designing the most basic or primitive components & proceeds to higher level.

- Bottom-up method works with layers of abstraction.

- If a system is to be built from an existing system, a bottom-up approach is used.

3-07-18

Module Level Concept -

Coupling & Cohesion → { max }

(must be min)

→ A process module is a logically separable part of the program. It is a program unit that is discrete & identifiable with respect to compilation & loading.

Ex → A macro, function, process, or a package.

→ Coupling & cohesion are used to identify degree of modularity & level of partitioning. and it is expected that the coupling is minimum and cohesion is maximum.

⇒ Coupling (minimum)

• coupling between modules is the strength of interconnection b/w modules. It is considered as measure of independence among modules. Two modules are independent if they are modifiable separately. Highly coupled modules are joined by strong interconnections while loosely coupled modules have weak interconnections. The choice of module decides the coupling between modules. It is decided during the system design phase.

The coupling is affected by several factors which includes interface complexity, types of connection & type of communication.

Interaction: Coupling increases with complexity & uncertainty of interface b/w modules. To obtain low coupling, we could minimise the number of interfaces per module & complexity of each interface within a given module & interface of a module is used to pass information to and from other modules. Coupling is reduced by passing information

through parameters & it will increase by the use of shared variables.

→ Complexity of the interface affects the degree of coupling & which increases with increase in complexity. Interfaces uses communication of objects b/w different modules & coupling is reduced if we transmit data selectively as required.

for ex → A field of a record is needed by the procedure & to reduce coupling just pass on the required field.

→ The type of info flow is the next factor that affects coupling. Information is divided into 3 categories.

1. Control Info
2. Data Info
3. Hybrid Info (control + data)

passing or receiving control inform' means that the action of the module will depend on the control information which makes it difficult to understand and to provide abstraction. Transfer of data means that a module passes some data as input & gets some data as output. Interface with only data communication result in lower level of coupling followed by the control data but it is considered highest if the info is hybrid i.e. both data & control.

⇒ Cohesion

~~Types~~
Cohesion of a module represents how tightly bound the internal elements of the module are to one another. Cohesion is the concept that establishes the intramodule dependency. With cohesion it is evaluated that how closely the elements of a module are related to each other. The greater the cohesion of each module in the system, the lower is coupling b/w different modules. There are several levels of cohesion.

→ 1. Co-Incidental Cohesion. (Weakest)

- It occurs where there is no meaningful relationship among the elements of a module.
- [It occurs if an existing program is modularised by chopping it into pieces & by making each piece as a module.]
For ex :-
 - If a module is created to save duplicate code by combining some part of the code that occurs at many different places. The module is likely to have co-incidental cohesion.]
- The resulting modules may contain random collection of functions.

S-09-18

Note Co-incidental is the weakest level & functional is the strongest level.

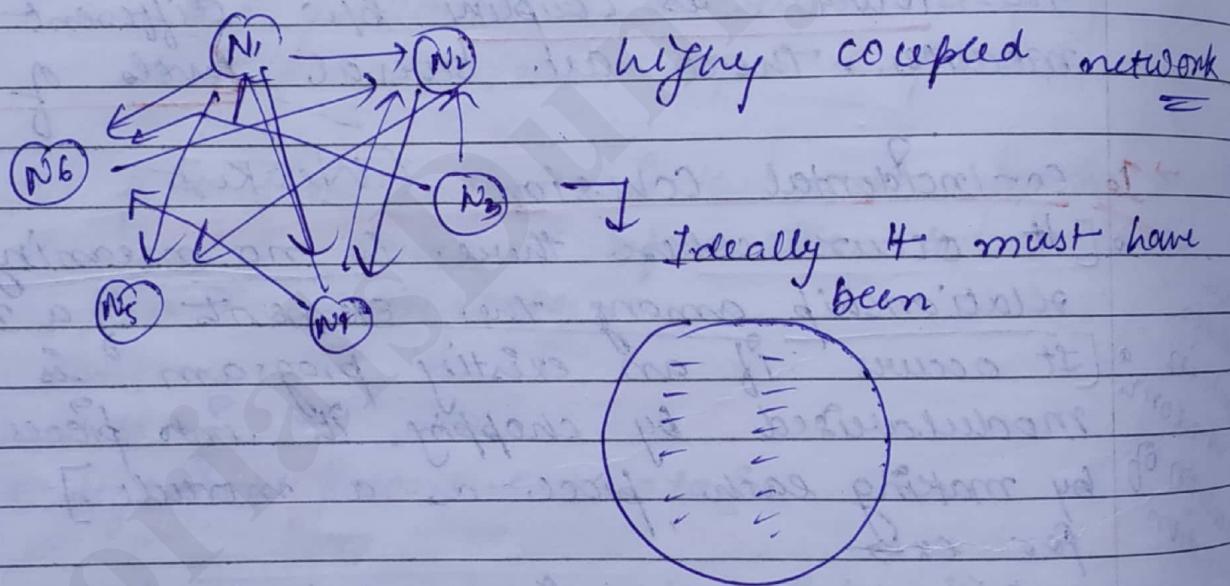
2

Logical Cohesion

a module has logical cohesion if there is some logical relationship b/w elements of a module & the module perform functions that fall in the same logical class.

e.g. Data handling, data I/P, data O/P etc.

A set of print functions generating an output report is arranged in a single module.



3.

Temporal Cohesion

same as logical cohesion except that the elements also related in time & are executed together. Modules that perform activities like initialisation, clean-up & termination are usually temporal bound.

The degree of temporal cohesion is higher than logical cohesion as the elements are logically related and orderly executed.

This avoids the problem of passing together,

4 Procedural cohesion

It contains the elements that belongs to a common procedural unit. In which a certain sequence of steps has to be carried out in a predefined order for achieving an objective.

for eg → A loop or a sequence of decisive statements in a module may be combined to form a common separate module

- The algorithm for decoding a module

5. Communicational cohesion

a module with communicational cohesion has elements that are related by a reference to the same I/p and O/p data. In a communication bound module the elements are together because it refers to adopt operations on same data structures.

for eg → • print & punch record. or
• set of defined operations on a stack.

No coupling increases

(module)

Push

Pop

{ this coupling increases as both push & pop depends on data in module

Hence, coupling communication overhead increases and cohesion reduces

(6)

Sequential Cohesion

When the elements are together in a module because the O/P of 1 forms the I/P to other, we get sequential cohesion.

If the elements of the module from different parts are executed in sequence. A sequence bounded module may contain several functions or part of different functions.

(7)

functional Cohesion

It is the strongest cohesion. In a functional bound module all the elements of the module are related by performing a single function.

for eg → Computing the square root or sorting the array

→ By single function

we not mean

simple mathematical

function but those modules which have single goal function like

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

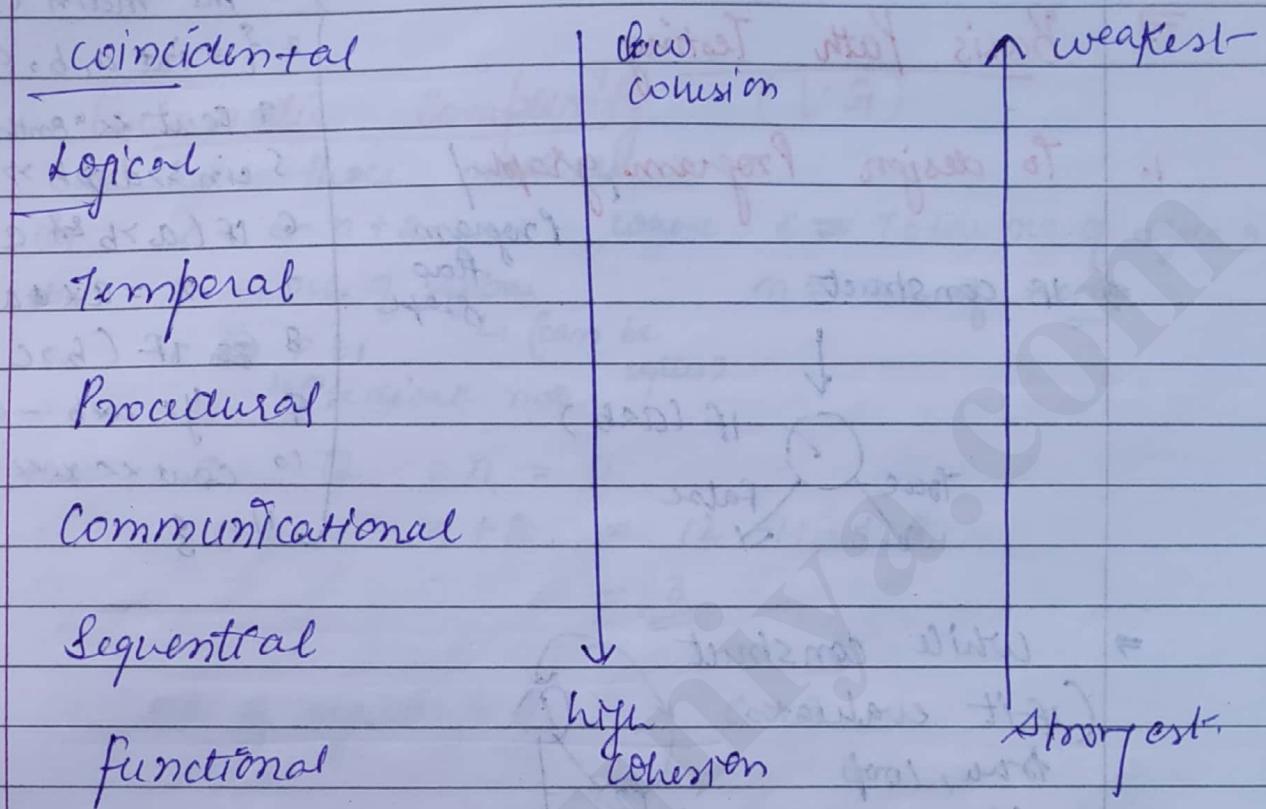
Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 



Coupling

vs

Cohesion

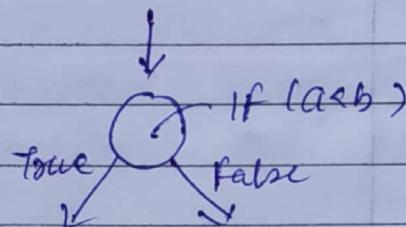
- | | |
|---|--|
| <ul style="list-style-type: none">• While designing you should strive for <u>low coupling</u> i.e. dependency between modules should be less• Is <u>Intra-module</u> concept | <ul style="list-style-type: none">• While designing you should strive for <u>high cohesion</u> i.e. a cohesive component/module focus on a single task with little interaction with other modules of the system• Is <u>Intra-module</u> concept |
|---|--|

9-04-18

⇒ Basis Path Testing

1. To design Program graph /

⇒ IF construct



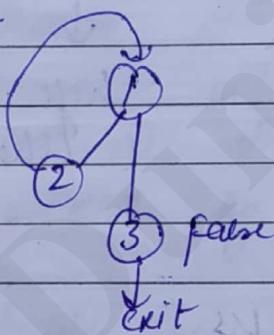
Program flow graph

```

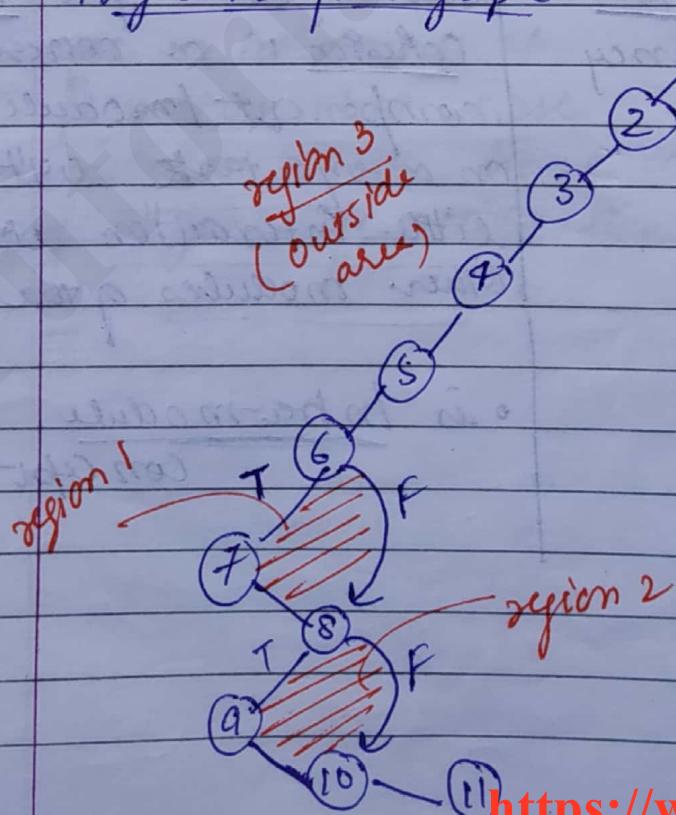
1 int main ()
2 {
3     int a, b, c, x=0;
4     cout << "Enter 3 no. ";
5     cin >> a >> b >> c;
6     if (a > b & a > c)
7         x = a * a + b * b;
8     else if (b > c)
9         y = a * b - a * c;
10    cout << x << y;
11 }
  
```

⇒ While construct

(if it evaluates
true, loop
back to initial
node)



Program flow graph



2.

Cyclomatic Complexity (VG)

We have those methods.

- $C = e - n + 2$ where $e =$ Total no. of edges,
- $C = \text{no. of regions}$ $n = \text{no. of nodes}$,
- $C = P + 1$ ↗ (can be cycles)
↳ (Predicate nodes)

thus $e = 12, n = 11$

$$C = e - n + 2 = 12 - 11 + 2 \\ = \underline{\underline{3}}$$

$$\text{no. of regions} = 3 \Rightarrow C = 3$$

Predicates nodes are those nodes which have atleast 2 outgoing edges.

$$P = 2 \quad (\text{node } 6 \text{ & } 8)$$

$$C = P + 1 = 2 + 1 = \underline{\underline{3}}$$

3.

To find no. of independent Paths

count of (independent path) is cyclomatic complexity
 \therefore It is 3.

4.

To determine Independent path.

compare from
it is not
Independent
path

- $1 - 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 1 - 11$ {Linear path}
- $1 - 6 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 1 - 6, 8 - 11$
- $1 - 6 \rightarrow 8 \rightarrow 10 \rightarrow 11 \rightarrow 1 - 6, 8, 10, 11$.
- $1 - 8 \rightarrow 10, 11$

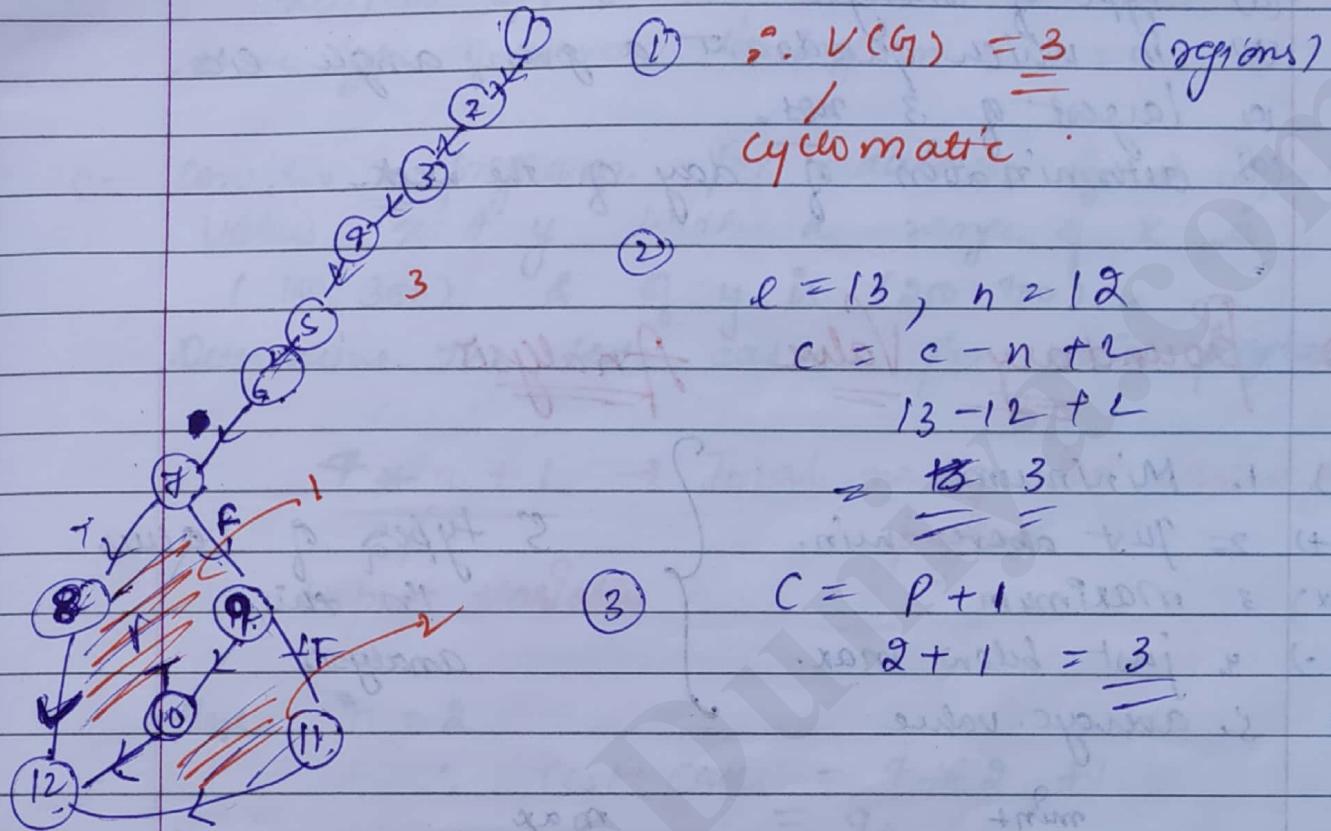
NOTE

Minimum test cases must be equal to no. of independent paths for overall & efficient testing of project

Q Consider a program for determination of a division of a student based upon the marks in 3 subjects. write the program, draw the program graph, cyclomatic complexity & find the independent paths

1. int main ()
2. {
3. int a , b , c ;
4. cout << "Enter the marks " ;
5. cin >> a >> b >> c ;
6. int avg = $\frac{a+b+c}{3}$;
7. if (avg > 60)
8. cout << "First division " ;
9. else if (avg <= 60 && avg >= 40)
10. cout << "Second division " ;
11. else
12. cout << "Third division " ;
13. }

→ program flow graph



No. of Independent Paths $\rightarrow 3 = C$

Independent Paths \Rightarrow

(a) 1-7-8-9-12

(b) 1-7, 9, 11, 12

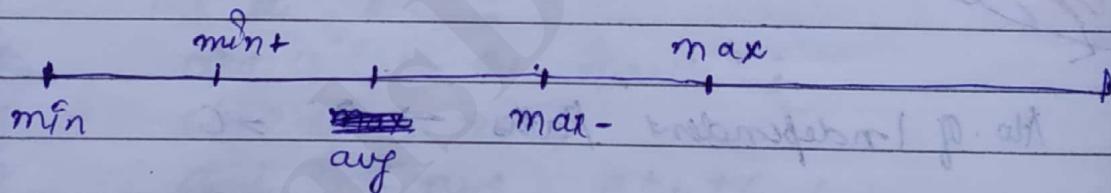
(c) 1-7, 9, 10, 12

Q → Perform basic path Testing.

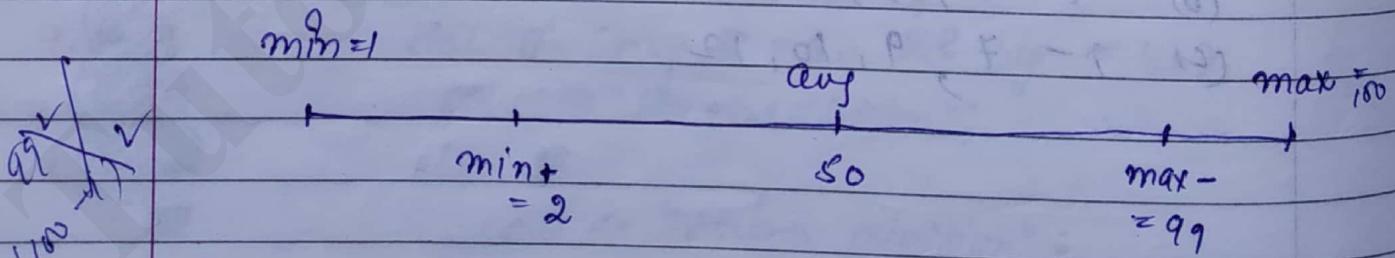
- (a) type of triangle
- (b) in which quadrant a given angle lies.
- (c) largest of 3 nos.
- (d) determination of day of the week.

→ Boundary Value Analysis

- (min) 1. Minimum
- (min+) 2. just above min.
- (max) 3. maximum
- (max-) 4. just below max.
- (avg) 5. average value
- } 5 types of values
in this analysis



Q → determine test cases to find square of a given number x within the range 1 to 100.



| | <u>test cases</u> | <u>Expected output</u> | <u>Received opp</u> |
|----|-------------------|------------------------|---------------------|
| 1. | 1 | 1 | |
| 2. | 2 | 4 | |
| 3. | 50 | 2500 | |
| 4. | 99 | 9801 | |
| 5. | 100 | 10000 | |

NOTE

If there is a deviation b/w expected & required o/p, this means that there is some type of error in program.

- Q- Consider a program for addition of 2 input values x & y with a range of x is $(100, 300)$ & of y is $(200, 400)$. Determine the test cases for this program.

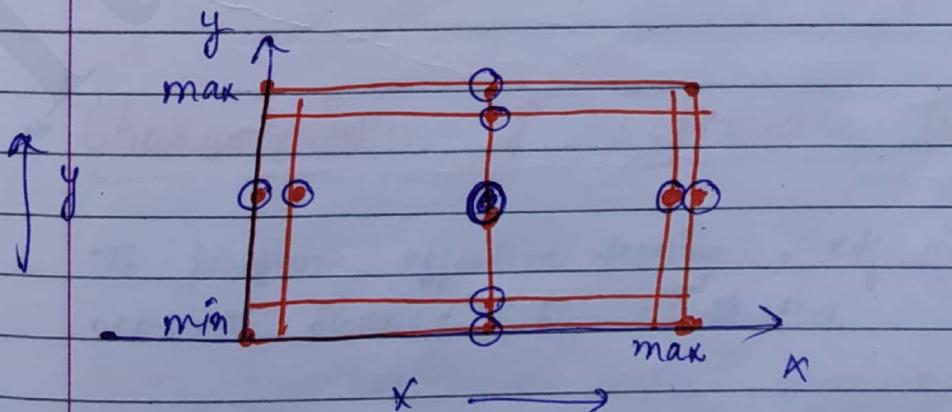
$\frac{4 \times n + 1}{\text{Input Variables}} \rightarrow \text{Total no. of test cases for program}$

Here $n = 2$

$$\therefore \text{Test cases} = 4 \times 2 + 1 \\ = 9 //$$

| $x \rightarrow \min$ = 100 | $\min + 1$ 101 | avg 200 | $\max - 1$ 299 | \max 300 |
|-------------------------------|-------------------|------------|-------------------|---------------|
| $y \rightarrow \min$ = 200 | 201 | 300 | 399 | 400 |

for this, we have 25 test cases but we won't explore all of them



Top
Bottom

S-09-18

Page No.
Date

| | | |
|----------|--|--|
| Page No. | | |
| Date | | |

Q → Perform basic path testing

1. Prime no
2. factorial
3. leap year
4. GCD (division)

9-04-18

Unit-7 Testing

- Testing is a set of activities that can be planned in advance and conduct systematically.
- Testing is a process to uncover errors that were made inadvertently when it was designed and constructed.
- It includes a set of steps into which specific test case design techniques and testing methods should be defined for a complete software process.
- Test begins with a focus on single component or set of related components and applies test cases to uncover errors after components are tested they must be integrated until complete system is constructed.
- A series of high order test are executed to meet customer requirements.
- Test specification plan documents is constructed by project manager, software engineer or testing specialist that describes an overall strategy and procedure for specific testing steps and types of test that will be conducted.
same as SRS, RMM

Characteristics of software Testing Process

1. To perform effective testing ; effective techniques should be conducted.

| | | |
|----------|--|--|
| Page No. | | |
| Date | | |

2. Testing begins at the component levels and work outward towards the integration of entire computer based system or software
3. Different testing techniques are adopted for different software engineering approaches & are applied at different point of time
4. Testing is conducted by the developer, project manager or an Independent test group (ITG)
5. Testing and debugging are different activity but debugging must be accommodated in any testing strategy.
(discovering compile time error / run time error)

⇒ Verification & Validation

Page no → 951

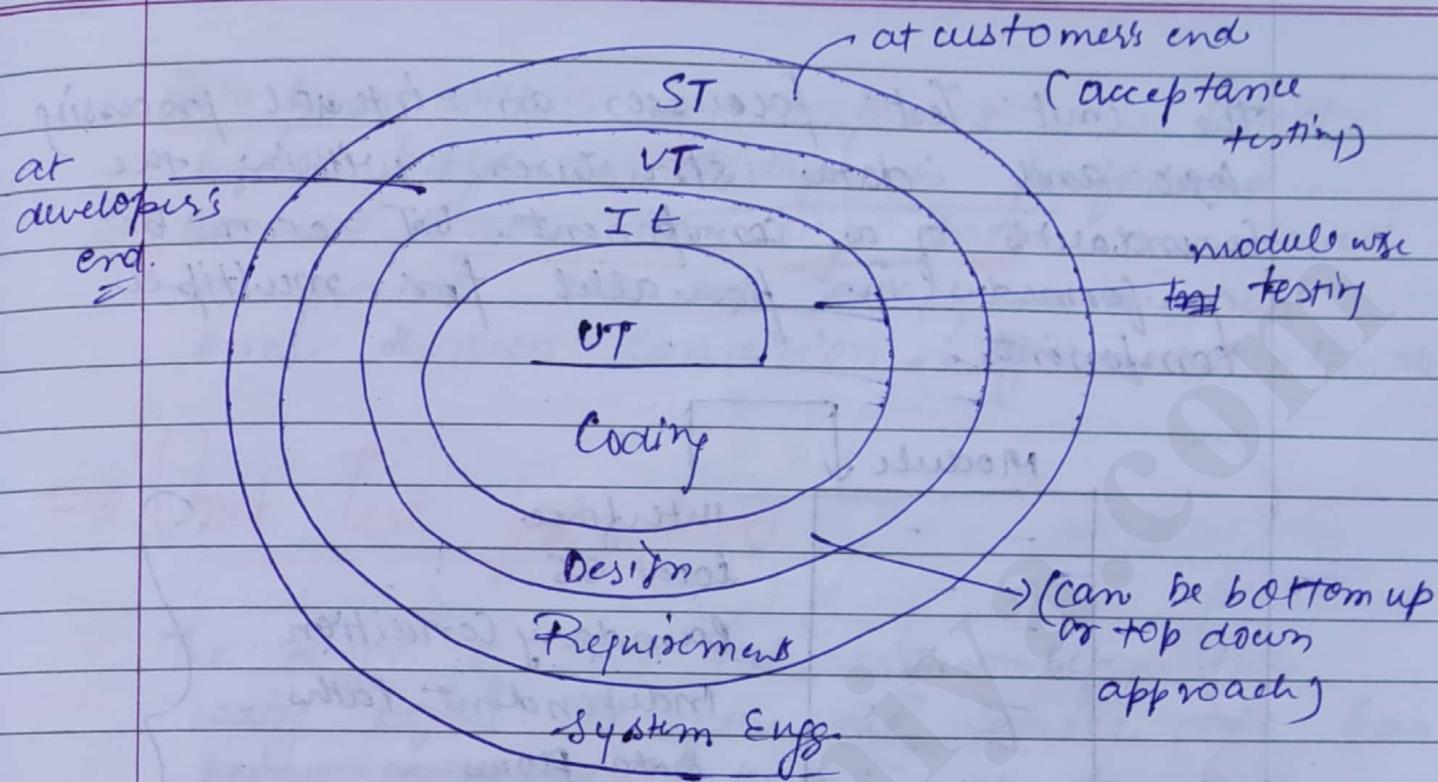
⇒ Software testing Strategies

UT - Unit Testing

IT - Integration Testing

VT - Validation testing

ST - System testing



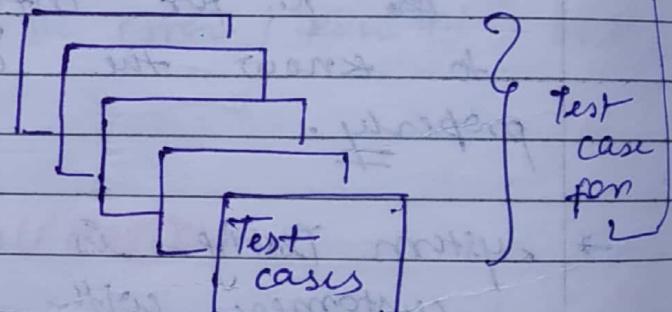
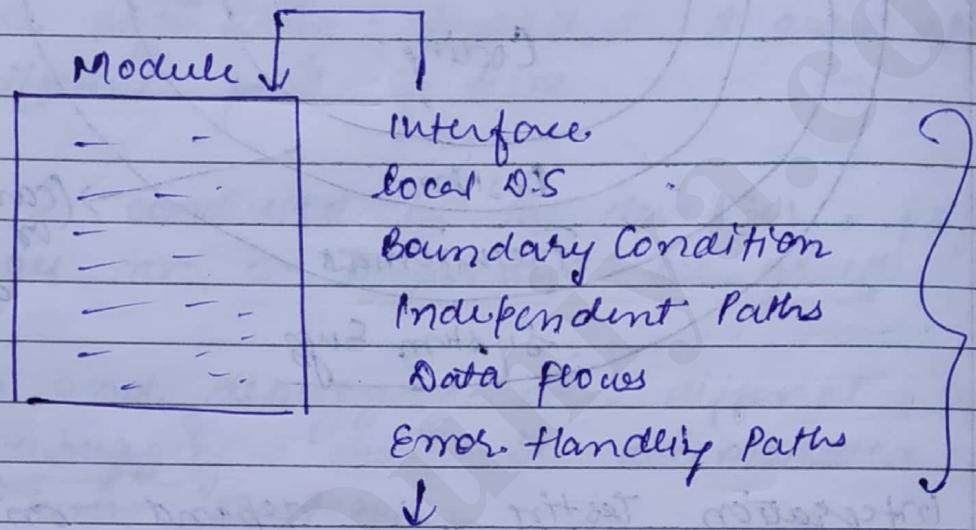
- Integration Testing is depend on design.
For Integration testing, we need to know the design of the system properly.
- System Testing is done at the end of customer with customer's perspective. It is testing of whole process.

Unit Testing

(1) Unit Testing → white box testing approach

[focusses on verification effort on smallest unit of software design i.e. the software component or module.] [Using the component level design, controlled paths are tested to uncover error within the boundary of the module.]

The unit Test focusses on internal processing logic and data structures within the boundaries of a component. UT can be performed in parallel for multiple components.



The module interface is tested to ensure Input / output information flow. Local Data structures are examined for Integrity & correctness during Program execution. for all Independent paths are executed at least once to provide completeness of all statements in a module. Boundary conditions are tested as it is assumed that software often fails at its boundary.

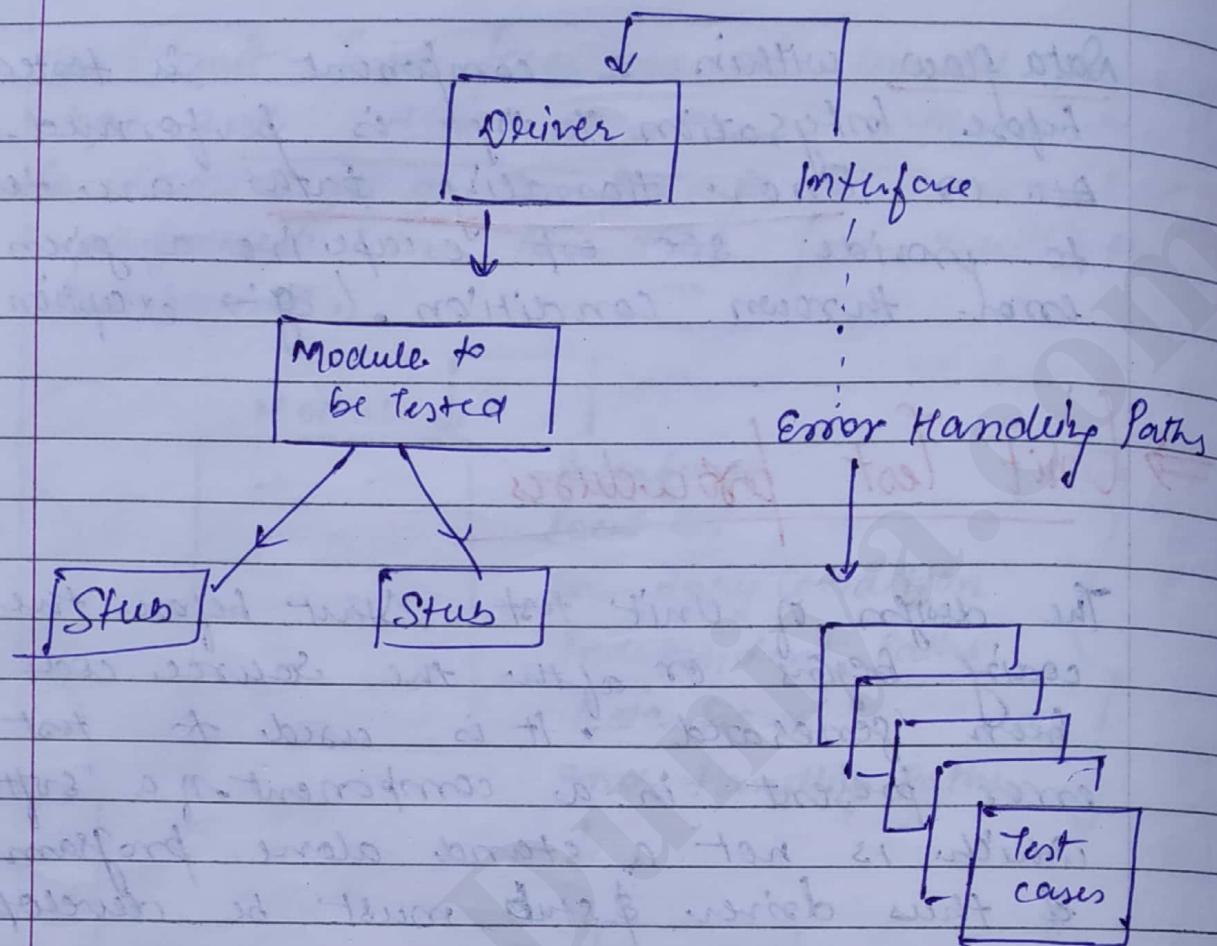
Data flow within a component is tested before Integration Testing is performed. All or Error handling paths are tested to provide ~~see esp~~ escape from a given error thrown condition. (e.g. exception handling)

⇒ Unit Test procedures

The design of Unit test starts before the coding begins or after the source code has been generated. It is used to test error present in a component of a software which is not a stand alone program & thus driver & stub must be developed for each unit test.

Driver is a main program that accepts test case data passes this data to the component to be tested and prints the relevant results.

Stub serve to replace the module that are invoked by the component to be tested. It is also called the dummy sub program which uses the subordinate module. Interface may do minimal manipulation, prints verification of entries & returns control to the module undergoing testing.



INTEGRATION TESTING

problem with UT is that it will not take into consideration interface b/w diff. modules.

Interface results in loss of data & many potential problems.

→ So, Even if the units of SW are working fine individually, there is a need to find out if units integrated together will also work or not.

This type of testing is known as Integration testing.

→ Top down Integration

- is an incremental approach to construction of software architecture.
- Modules are integrated by moving downwards to the control hierarchy, beginning with the main control ~~com~~ module to subordinate module upto the last module.
- uses a common hierarchical OF integration

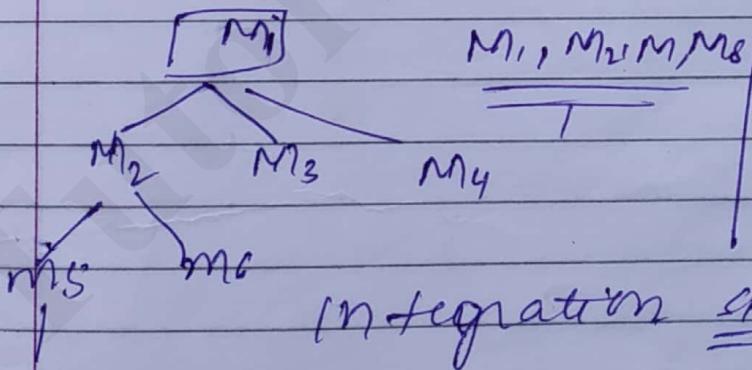
B·F Integration

(V) Integrates all components on a major control path of the program structure from main module to last module until a given depth.

(I) It incorporates all components directly subordinate at each level moving along a horizontal structure.

M_1, M_2, M_3, M_4 first.

modules
only those can be collected together who have same parents



Integration steps

- ① the main control Module is used as a test driver & subordinate modules as stubs.

⇒ Bottom - up Integration

- this testing begins construction & testing with components at lowest level in program structure

Implemented with following steps.

- ① Low level components are combined into clusters that form a specific software subfunction.
- ② A driver is written to coordinate test case & output
- ③ A cluster is tested.
- ④ Drivers are removed & combined moving upward in the program structure.

| M₈ - M₅ - M₆] cluster

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at <https://www.tutorialsduniya.com>

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

facebook

WhatsApp 

twitter 

Telegram 