Course Outcomes

CO1 : Demonstrate Software Process Models
CO2 : Illustrate Requirement Engineering process.
CO3 : Discuss Software architecture and Design
CO4 : Apply coding principles and Testing techniques
CO5 : Discuss Software Estimation and Maintenance
CO6 : Describe Quality Management and Metrics

Scanned by CamScanner

UNIT-I

Software :

Software is defined as a collection of

1) code instructions
2) data that operates on that instruction
3) Documentation (comment lines, EULA, Activation key, user manual)

Software Engineering

Software Engineering is the application of software principles & practices for optimizing the time, cost and effort factors of software.

Nature of software (legacy software)

Software engineering helps in achieving a good quality software catering to the needs of developers, customers and management people

⇒ poor software engineering may lead to

* loss of life
* loss of the organization's reputation
* budget overdue
* time overdue
* customer disatisfaction
* poor quality software
* wastage of resources.

Scanned by CamScanner

Software evaluation

→ law of continuing change (1974)
→ law of increasing complexity (1974)
→ law of self regulation (1974)
→ law of conservation of organisational stability (1980)
→ law of continuing familiarity (1980)
→ law of declining quality (1996)
→ The feedback system (1996)

Software Application Domains

System software

This software provides support to application software.

Ex: Operating system, Audio drivers etc.

Application Software

This software is directly visible or interactive with the end users

⇒ Generally, these are developed for stand-alone applications.

Engineering / Scientific software

This software generally consists of number crunching algorithms that are designed for systems of high processing capabilities.

Ex:- Super computers, Space research program systems

Scanned by CamScanner

Embedded Software

This software resides within another software

⇒ This software has limited capabilities

⇒ It provides additional functionalities only to the actual software.

Ex:- Microwave oven, Microprocessor systems

product line software

It is a specific purpose software intended for large number of consumer base

Ex: Broadcasting system, e-retile business etc

Web applications

Web app software supports a wide range of application from stand-alone industrial level services

Ex:- html (hypertext markup language)

XML (extended), DHTML (dynamic HTML)

Artificial intelligence software

This software makes use of non-numerical algorithms to solve complex problems where straight forward analysis is difficult to make

⇒ This software is useful in prediction based decisions

Open source software

This software is freely available to use along with source code visibility

Ex: Browser softwares like internet explorer, firefox, navigator

Scanned by CamScanner

Software Process

⇒ software process is the collection of various engineering tasks or activities to complete a software project.

⇒ choosing a correct software process ensures in success of the project

Software engineering - A layered Technology

Software Myths

These are the misconceptions of various people involved in the development of the software project.

⇒ **Management Myths**

Management people focus on non-technical issues like project administration, project mapping, project tracking resource

Myth-1 :

We already have a book that is full of Standards and procedures for building software won't provide my people with everything they need to know.

Scanned by CamScanner

Reality: Although this book serves as a reference for some issues, it doesn't address situations like

a) New projects

b) New technologies

c) dynamic / new bugs in the software

myth-2 :

If we get behind schedule, we can add more programmers and catch up (sometimes called the Mongolian horde concept)

Reality: Adding new programmers consume time for training purposes. This could lead to much more delay in project completion

myth-3 :

If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

Reality: If the third party lacks proper clarity and efficient communication, while developing the project, the software project may fail.

Scanned by CamScanner

⇒ Customer Myths

A customer is the enduser or client or both with respect to a software.

Myth 1:

A general statement of objectives is sufficient to begin writing programs we can fill in the details later

Reality: Requirement gathering serves as the 1st input in software development lifestyle cycle However, ambiguous and incomplete requirement may lead to project failure.

Myth 2:

Project Requirements continually change, but change can be easily accomodated because software is flexible

Reality: Errors discovered at the later states of SDLC are very hard and costly fix

Software development lifecycle :-

requirements → design → Arch → coding → testing → Deploying → maintenance

Scanned by CamScanner

Partitioner

⇒ Practitioner's Myths

Practitioner's or coders or developers are the people responsible for developing the code for the software

Myth 1:

Once we write the program and get it to work our job is done

Reality: About 60-80% of the effort spent on the software is actually after delivering the software to the customer for the first time

Myth 2:

Until I get the program running I have no way of assessing its quality

Reality: Even without running the code, we can assess the quality of the software with the help of formal technical issues

Myth 3:

The only deliverable work product for a successful project is the working software

Reality: Documentation & production support are also deliverable work products

Myth 4:

Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down

Scanned by CamScanner

Reality:

Maintaining documentation improves the quality of the software, which is a necessity

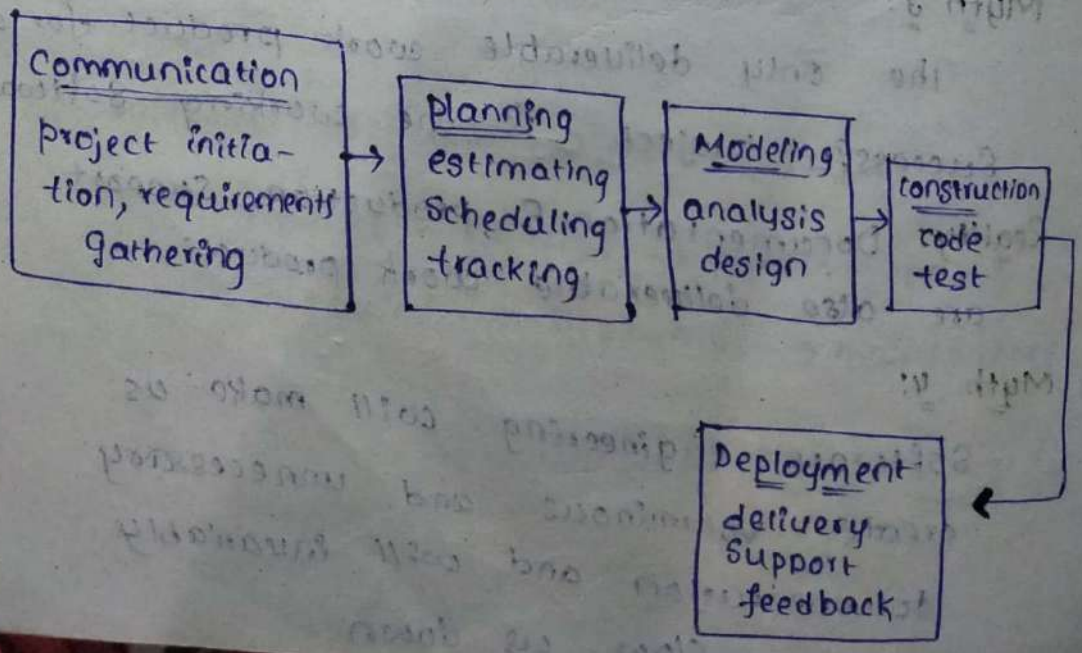
Software process Models

Software process models is the set of software project development activity.

⇒ A software development process models is diuided into 3 categories

1) Prescriptive (project plan is fixed)
2) Iterative (ceterin/all the steps in the project can be repeated), also supports backward propagation)
3) evalutionary (This models have rapid iteration)
4) Agile approach

Waterfall Model



Scanned by CamScanner

⇒ Waterfall model is proposed by Royce,
⇒ In the early 1950's
⇒ Because of its unidirectional flow it got the name
⇒ This model doesn't supports backtracking
⇒ All the stages have predefined subtasks
⇒ The output of a stages serves as the Ip to the next stage.

Advantages

* Only suited for small & automation projects
* less complex

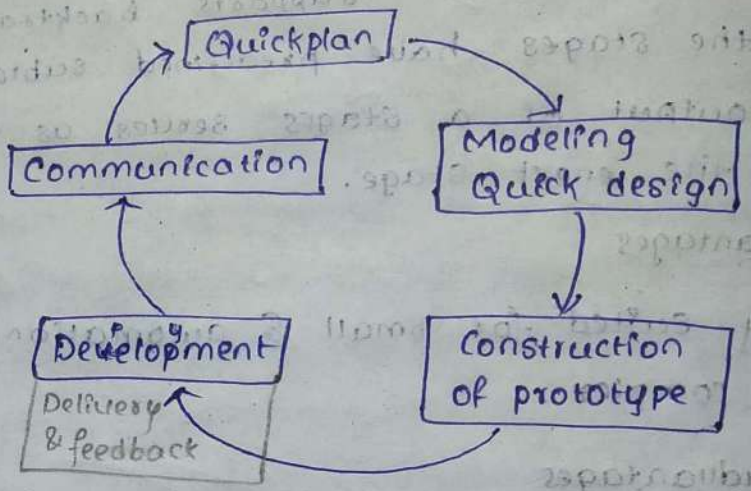
Disadvantages

* Requirements are fixed before the start of the project (Bigbang)
* It leads to requirements bloating (unnecessary requirements)
* Customer can't assess the quality before and during the project and feedback
* Doesn't support iteration
* Hardware complexity

Scanned by CamScanner

Evolutionary Process Model

Prototype model



Iterative & prototype models

⇒ Both the models allows iteration in software development

⇒ We can start the project with the known requirements

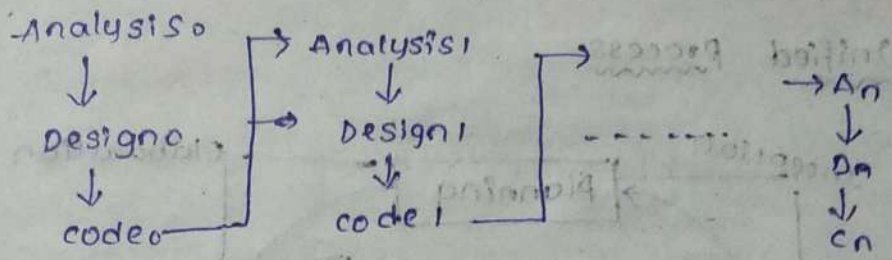
With them, the initial working version of the software is developed

⇒ changes are made after evaluating this version by the customer

⇒ Feedback for the customer is taken for necessary modifications

⇒ A checklist is maintained and verified for each version until the final software is ready

Scanned by CamScanner



Iterative model

Advantages

⇒ It is not necessary to fix all the requirement before the start of the project.

⇒ Requirements bloating is avoided

⇒ If the client and the developer are uncertain about the project environment, this method is more suitable

⇒ Increased customer feedback

⇒ When compared with waterfall model, the quality of the software is high.

Disadvantages:

⇒ Not suited for Automation projects

⇒ Not recommended for old & existed Software

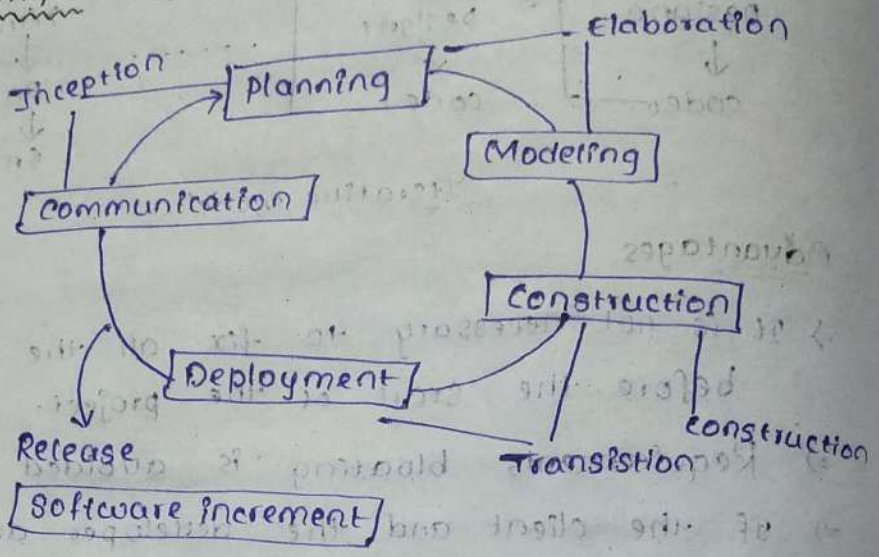
⇒ sometimes, higher interaction of customers or clients is not possible

⇒ possible increase in time, cost & effort when compared with waterfall model

⇒ Generally, not recommended for large troubles

Scanned by CamScanner

Unified Process:



Inception phase

- Vision document
- Initial usecase model
- Initial project glossary
- Initial business case
- Initial risk assegment
- Project plan
  - Phases & iterations
- Business model, if necessary
- One or more prototypes

Elaboration phase:-

- Usecase model
- Supplementary requirements including nonfunctional
- Analysis model
- Software Architecture Description
- Executable architectural Prototype
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adopted workflows
  - milestones
  - technical work products
- Preliminary user manual

Scanned by CamScanner

Construction phase :-

Design model

coding { Software components

Integrated software increment

testing { Test plan & produce Test cases

documentation { support documentation User manuals installation manuals description of current increment

Transition phase :-

Delivered software increment

Beta test reports

General user feedback

⇒ Unified process model is generally adopted for object oriented project environments.

⇒ The increments and iterations are increased over prototype and waterfall models.

Inception phase :-

The objectives & scope of the project are well defined in this phase

Elaboration phase :-

It is ~~as~~ the requirements gathered in the previous phase are planned in detail, covering the design & architectural issues of the software.

Feasibility of the requirements is checked (practical possibility)

Scanned by CamScanner

Construction phase :-

It involves the software coding & testing parts. Necessary documentation is also maintained.

Transition phase :-

The developed software is thoroughly checked for errors (especially critical & moderate errors) B-testing is done & the feedback is obtained After evaluating these parameters, the software is either released or withheld

Agile process model

Agility :-

Agility means supporting higher frequency of changes in software

⇒ The Agile approach is developed in the early 1990's

Agile process

⇒ It is difficult to predict in aduance which software requirements will persist and which will change.

⇒ For many types of software, design and construction are interleaved.

⇒ Analysis, design, construction and testing are not as predictable as we might like.

Agile princ

* Change
high.
* The client
* Rather u
small
* phase to
ou

Extreme

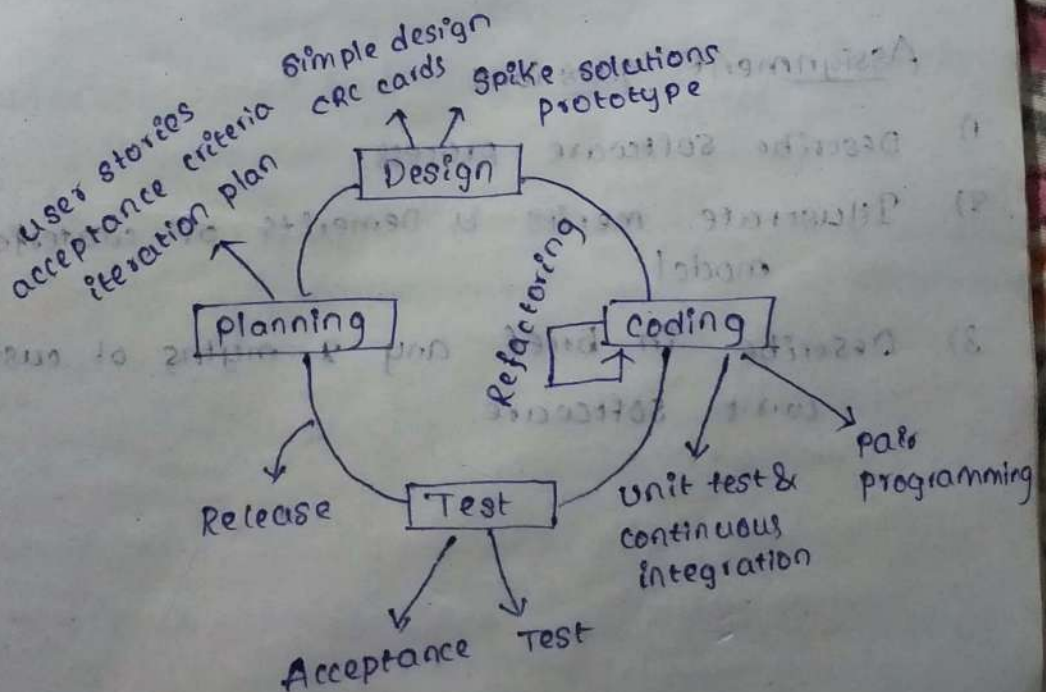
user stori
acceptance
itera

Scanned by CamScanner

Agile principles

* Change frequency of the software is very high.
* The client interactions are heavy
* Rather versions, software is developed in small increments.
* Phase to face communication is preffered over documentation

Extreme programming (XP)



Scanned by CamScanner

Pair programming

For a particular or a set of modules, the no.of programmers are increased from 1 to 2.

⇒ while one prog. person codes, the other person reviews the code and vice versa

⇒ This technique reduce more bugs in unit testing

Note:

levels of testing

1) Unit testing (programmers) (module/at a time)
2) Integration (2 or more mods) } tester, programmers (2 or more modules)
3) System (all mods) }
4) Acceptance → α, β } customer

Assignment questions

1) Describe software process
2) Illustrate merits & Demerits of waterfall model
3) Describe, in brief any 2, myths of customer with software

De-merits

Waterfall

* Requireme are fixed bloated

* Unidirec

* User feed after so delivery

* documen driven

* Small & Autom

* oldest

De-merits

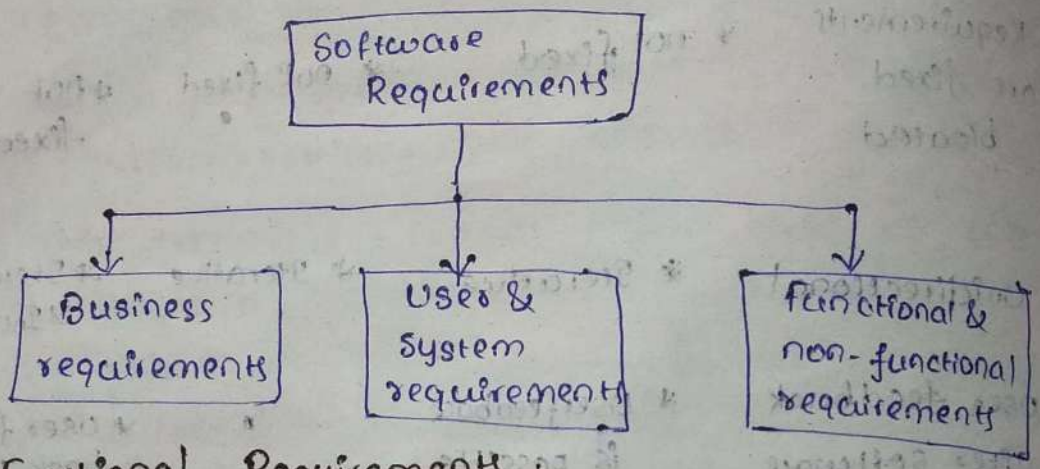
Waterfall	Iterative & Prototype	Unified process	Agile
* Requirements are fixed, bloated	* not fixed	* not fixed	* Not fixed
* Unidirectional	* Iterative	* Iterative	* Iterative & Increment
* user feedback after software delivery	* User feedback is possible		* user feedback high frequency
* document driven process	* flexible	* more flexible	* High flexible (face to face)
* Small & Automation	* Small to moderate	* Moderate to big	* large & complex
* oldest	* somewhat recent	* object oriented Software	* latest
	* Unclear enviroment	* recent	* Unclear environment

Scanned by CamScanner

UNIT-II

SOFTWARE REQUIREMENTS

*



Functional Requirements:

Functional requirements are directly related or associated with the user choices.

Non-functional requirements

These provide additional support to the functional requirements.

Ex:- for ATM kiosk system.

⇒ The functional requirements are

i) Transaction service request - Delay

ii) Genuine card etc., (Authentication & Validation)

Note:

1) functional requirements r correspond to the task modules in the software.

2) There could be multiple dependencies possible among the functional & non-functional requirements.

3) Modifying
modify th
(design

⇒ Money
requirem
Non-fun

1) Availabili

2) Authentic
(ch

3) Securit

4) Error

5) perform

User

The us
limitati

Ex:-

User -

a) lan

b) Ease

c) grap
cod

Scanned by CamScanner

3) Modifying one (more requirements prompts to modify the dependent (or) related requirements (design issue).

⇒ Money withdrawl is one of the functional requirements.

Non-functional requirements are :-

1) Availability (Resource availability through 24x7)
2) Authenticating & validity (checking for genuinity) (checking for correctness of IIP)
3) Security (maintaining the confidentiality of user data)
4) Error recovery (atleast an error should be displayed)
5) performance (the through put of system should be minimal).

User requirements

The user choices and priorities, the limitations from the user requirements.

Ex:- User interface design.

User-interface design

a) language Support
b) Ease of use → naive
→ Moderate
→ professionals
c) graphical user interface: It includes colour coding spells, grammar checks & alignments.

Scanned by CamScanner

System Requirements

⇒ These requirements pertain to the operational environment of the software

⇒ These are the technical issues not dictated by the user (in general)

Examples

Sample hardware requirements :-

RAM : 1 GB (or) above
HDD : 40 GB

Sample software requirements :-

OS : Windows 7 or above
TOOL : MS-Access V 10.1

Business Requirements

Business requirements define the project goal & the profits after delivering the software

⇒ It focusses on the market segment of the software

Scanned by CamScanner

Software Requirement specification (SRS) document

An SRS document is the legal agreement between all the stake holders (representatives)

⇒ It seals the 1st phase of SDLC

purpose of SRS

⇒ It is useful to clear ambiguities among stake holders

⇒ It provides basic for later of software development

⇒ It acts as a reference arguments

⇒ ensure high quality software product

⇒ reduces development efforts.

Characteristics of SRS

⇒ Correctness

⇒

⇒ Unambiguity

⇒ completeness

⇒ Consistency

⇒ should be ranked for importance and/or stability

⇒ Verifiability

⇒ Modifiability

⇒ Testability

Scanned by CamScanner

⇒ Validity

⇒ Traceability

Structure of SRS Document

1. Introduction
   - 1.1 ⇒ purpose
   - 1.2 ⇒ Scope
   - 1.3 ⇒ Definitions, acronyms and abbreviations
   - 1.4 ⇒ References
   - 1.5 ⇒ Document Overview
2. General Description
   - 2.1 ⇒ production perspective
   - 2.2 ⇒ Product functions
   - 2.3 ⇒ User characteristics
   - 2.4 ⇒ General constraints
   - 2.5 ⇒ Assumptions and dependencies
3. Specific requirements
   - 3.1 ⇒ Functional requirements
     - 3.11 → Functional requirement 1
       - 3.111 * Introduction
       - 3.112 * Input
       - 3.113 * processing
       - 3.114 * output
       - ⋮
     - 3.1N → * Functional requirement M

3.2 ⇒ Extreme
3.3 ⇒ Perfor
3.4 ⇒ Design
3.5 ⇒ Secur
3.6 ⇒ Main
3.7 ⇒ Reali
3.8 ⇒ Avai
3.9 ⇒ Data
3.10 ⇒ Docu
3.11 ⇒ Sa
3.12 ⇒ Op
3.13 ⇒ Site

Consider

Soft

⇒ After
are
require
Requi
steps.

Scanned by CamScanner

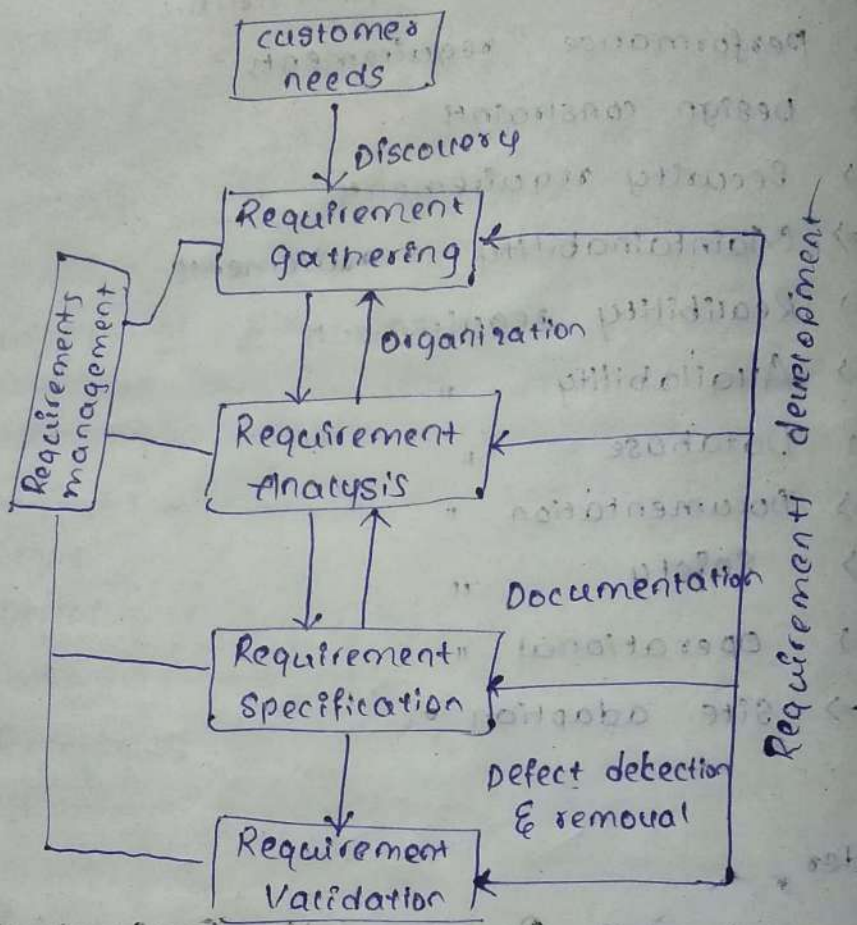
3.2 ⇒ Extreme interface requirements
3.3 ⇒ Performance requirements
3.4 ⇒ Design constraints
3.5 ⇒ Security requirements
3.6 ⇒ Maintainability requirements
3.7 ⇒ Realibility requirements
3.8 ⇒ Availability "
3.9 ⇒ Database "
3.10 ⇒ Documentation "
3.11 ⇒ Safety "
3.12 ⇒ Operational "
3.13 ⇒ Site adaption

Consider

Software Requirements Engineering process

⇒ After gathering all the requirements, they are subjected to feasiability study, requirements elycitation and analysis, Requirements validation, requirements management steps. In order to frame SRS document

Scanned by CamScanner



Requirements are classified into

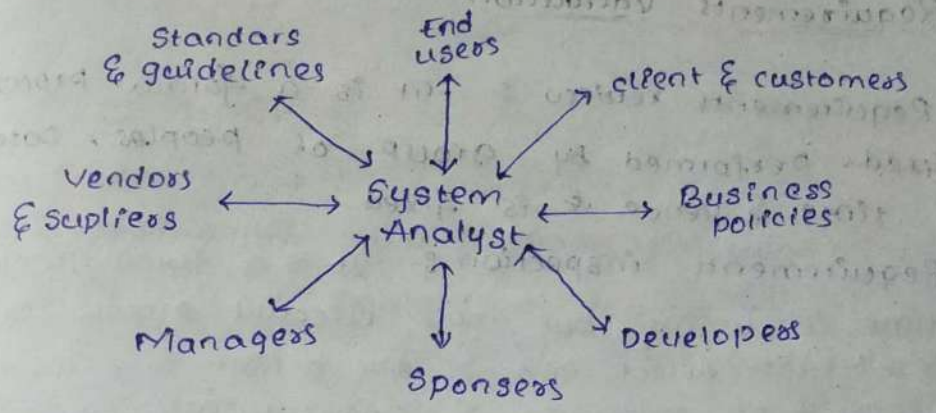
- critical / Major
- Moderate
- Minor

Requirements gathering

⇒ A formal meeting with all the stake holders is held before the start of the project.

⇒ The system analyst is the person who is responsible for recording or documenting all the stake holders feedback.

Scanned by CamScanner



Fact-finding Techniques :-

1) Interviewing :- formal meeting/technique, face to face communication, gather facts costly & timetaken

2) Questionaires :-
informal technique but effective & less constly than interviewing.

3) Joint applications Development (JAD) :-
group (reviewity) group meeting (formal & informal)

4) Onsite observation :- The system analyst checks the website/application for any issues

5) prototyping :- The included requirements in an executable form are checked for errors.

6) view points :- The view points of various stake holders are gathered.

7) Review records :-
Available documentation for related projects or past versions of the software are refferred. for taking suitable decisions

Scanned by CamScanner

Requirements Validation

1) Requirements review :- It is a formal process performed by group of peoples. Date, time & venue is fixed

2) Requirement inspection :- It is a formal, costly, time consuming, but more effective process. It is possible to detect 90% of the deffect. This technique consists of moderator & inspector team.

3) Testcase generation

4) Reading

5) Prototyping

⇒ Requirement validation is useful for checking errors that arise because of different stake holders' views

⇒ (2) Inspector inspects artefacts and notifies if errors are found to the moderator, until & unless the error is checked, this process continues (Re-checking)

⇒ (3) Testcase :-
A testcase represents a set of test inputs, expected & actual outcomes and decision based on those outcomes

Scanned by CamScanner

if $tc_{expected} = tc_{actual}$ then testcase pass

$tc_{expected} != tc_{actual}$ ⇒ status = fail

G-mail login

→ Inputs : 2 ⟶ id, password

→ condition → valid id, valid password } login

Invalid id, Invalid pass
Blank id, Blank password } ✗ not login

→ output ⟶ 1) Inbox opening

2) Error msg displaying reason

S.No.	Inputs	expected o/p	A-o/p	Result
1	valid id valid password	login ✓	login ✗	error
2	✓ id ✗ pwd	login ✗	login ✓	fail
3	✓ id ✓ pwd	login	login	pass

⇒ the project manager & the tester identifies high priority requirements and for these requirements, testcases are generated, executed and analysed for errors.

⇒ this technique reduces more errors than previous 2 techniques.

Scanned by CamScanner

4⇒ Reading

1) Ambiguity
2) Understability
3) Completeness
4) Redundancy
5) Testability
6) Modifability
7) Traceability

⇒ In this technique, the reader is a reviewer who checks the system for errors either by adhoc based reading or by check-list based reading.

5⇒ Prototyping

An executable model of the proposed system is developed and checked for errors.

⇒ This is developed in incremental manner

Requirements management :-

Management activities are apply before, during and even after delivery of the software product to the client

⇒ It is a non-technical activity that monitors the entire project periodically.

Scanned by CamScanner

Some of the activities in requirement managem

1. planning for the project requirements
2. Focussing on the requirements identification process
3. managing the requirements changes
4. controlling and tracking the changes
5. Agreeing on the requirements among stakeholders

⇒ changes in requirements can have a negative impact on the project success

⇒ Requirement tracebility and requirement change management process helps in project tracking.

⇒ Certain requirements management tools are used to understand and analyse the requirement changes

Tool :-

A tool is a mechanism which saves human evaluation/computing effort by running the inputs fed to it and generates results.

⇒ It is necessary to maintain an audit trail of changes, mechanism to authenticate change approval etc.

Scanned by CamScanner

UNIT-3

SOFTWARE ARCHITECTURE AND DESIGN

Software Architecture :

It follows software design in which details represents how the final system is implement.

⇒ Sometimes, software Architecture is reffered as "blueprint" of the system

Role of software Architecture :-

1) Understanding and communication :- A common view among the stake holders has to be obtained

2) Reuse :- Architecture helps in understanding the fixed & variable products in the software

3) Construction and Evalution :- The internal system components & their dependencies are constracted & evaluted before making decisions.

4) Analysis :-

It is important to understand the behavioural characteristics of the system before being built. Also suitable alternatives have to be suggested in case of any issue.

⇒ Reference documents also help in analysing of the software

Scanned by CamScanner

Architecture Views

Views represents a system as the collection of elements and the relationship among them.

⇒ Architecture views are categorised into 3 types.

1) Module view :-

It is code-based view in which the system is implemented as collection of code units, each unit implementing some system functionality.

Ex:-

packages, classes, functions, sub-routines etc

2) Component and connector view :-

A component is a runtime entity, a system can be viewed as the collection of components.

⇒ connectors are means for interactions among components.

Ex:-

Objects, process etc. (components)
pipes in Unix environment, Bus topology (connectors)

3) Allocation view :-

⇒ It focusses on how the different software units are allocated to resources like hardware, people, file system etc.

⇒ It signifies the operating environment of the application / system.

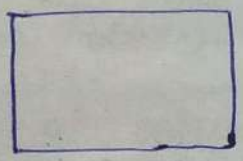
Scanned by CamScanner

- Component & connector view :-

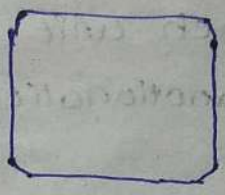
Components are generally computational elements or data stores

Components :-

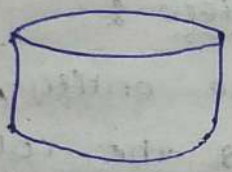
client



Server

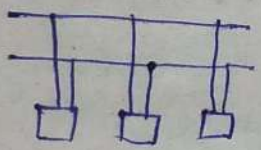


Application

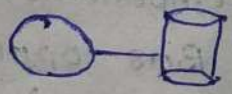


Database

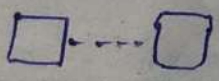
Connectors :-



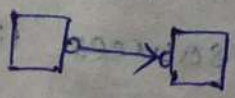
Bus type connector (1-many)



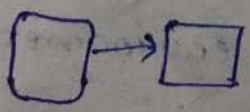
Database access



Request reply



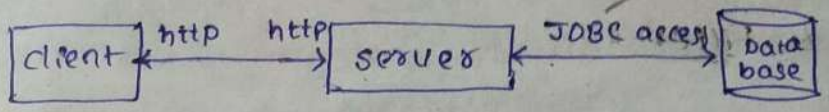
pipe (1-1)



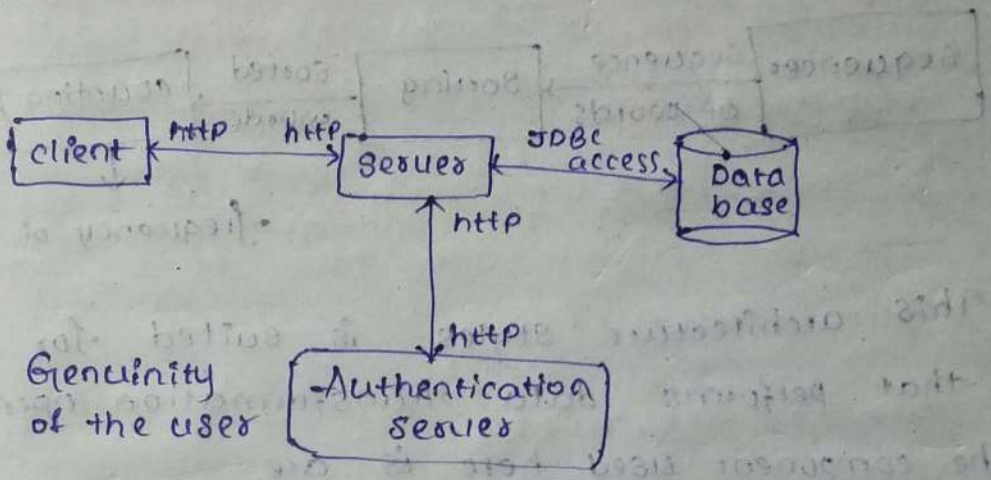
RPC (Remote procedure call)

Scanned by CamScanner

Example :- (1)



Example :- (2)



proxy server :-

- Dictionary based
- Content based

tube

Youtube ✗

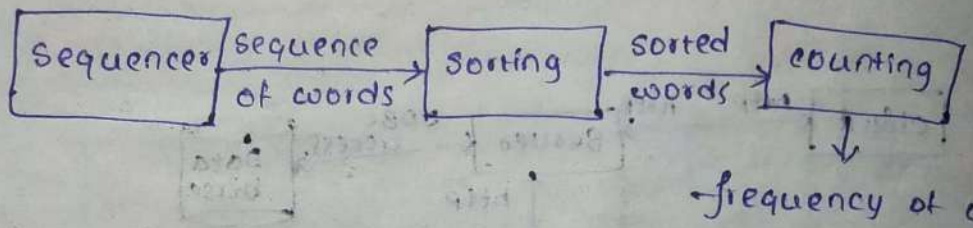
gaana → music / entertainment

Middleware
- proxy
- Authe
- 1 more server

Scanned by CamScanner

Architecture styles for C & C view :-

1. pipe & Filter :-



This architecture styles is suited for systems that performs data transformation operation

⇒ The component used here is are

1) Sequencer
2) sorting
3) counting

⇒ A filter receives the data from some defined Ip types performs the data transformation and then sends the olp data to other filters on the defined olp pipes

⇒ A filter can be independent & asynchronous

NOTE:

Even if the data transmission is asynchronous, care should be taken by the system to avoid dataloss, longer delays.

⇒ The pip
channel
to an
⇒ This st
buffer

Constro

1. The
know
2. The p
a fi
filt

2. Shared

⇒ Shared
and
⇒ Data
all
⇒ Data
the
⇒ The

Scanned by CamScanner

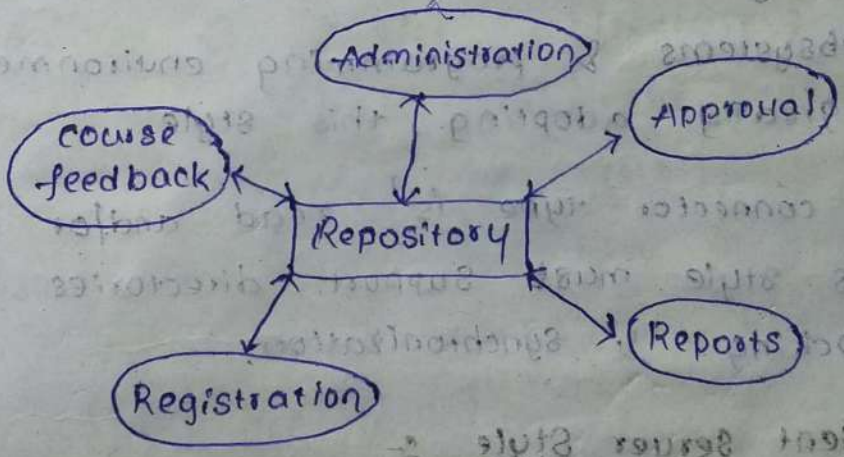
⇒ The pipe connector is a unidirectional channel which transfers data from one location to another

⇒ This shout system should also support buffering & synchronization

Constraints :-

1. The filter should work without the knowledge of the producer / consumer
2. The pipe must connect to an o/p port code of a filter to the I/p port code of another filter

2. Shared data style :-



⇒ Shared data style consists of data accessors and data repositories.

⇒ Data repositories are the data stores (they store all the common data)

⇒ Data accessors retrieve whole or part of the common data from the repositories

⇒ The data accessors directly don't communicate with each other.

Scanned by CamScanner

⇒ Two styles are possible here

1. Blackboard style
2. Repository style

Blackboard style :-

If some data is posted on the repository, all the accessors know the details

Repository style :-

In repository style, data repository supports permanent storage and has control over the accessors

⇒ Database applications use this architecture style

⇒ Websystems & programming environments also greatly adopting this style

⇒ The connector type is read and/or write

⇒ This style must support directories buffering, locking & synchronization.

3. Client server Style :-

Client server style is bested suited for distributed computing

⇒ Components are of 2 types

1) client
2) server

⇒ The constraint is the client can communicate with the server but not with other clients

Scanned by CamScanner

⇒ The communication starts with the client request to the server.

⇒ If the server is not busy and the request doesn't involve any compromaisational factor, then the server connects to the client

⇒ After successful connection is made, the server processes the client request and sends back a reply

⇒ The connector is of type request / reply

⇒ This connection is asymmetric

⇒ This architecture style follows n-tyle n-tier architecture.

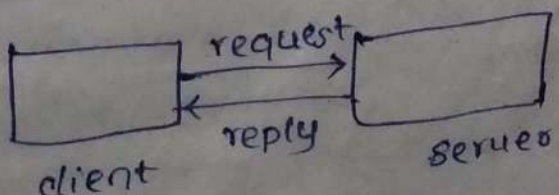
Examples for 3-tier architecture :-

i, Client tier :- client make the request & receive the request.

ii, Business tier :- Data submitted by the client is processed & suitable business rules are applied.

iii, Database tier :- The entire data resides in database tier, and the business tier communicates with the database tier.

Client server (2-tier) :-



Scanned by CamScanner

Documenting Architecture Design :-

1. System & Architecture context
2. Description of Architecture views
3. Across view documentation

Software Design :-

Software Design explains what the final Software intends to do

It is characterited into

1) Low level design (LLD)
2) High level design (HLD)

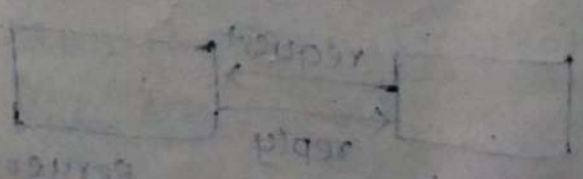
Open closed principle

⇒ The principle is stated by "Bertrand Meyer as "A software entity should be open for extension but closed for modification"

⇒ It means new demands can be accomodated but existing code should not be modified.

Ex:

A client printer system:



Detailed design :-

⇒ This design activity may / may not be performed formally.

⇒ The detailed design is helpful in making the software code consistent in design working.

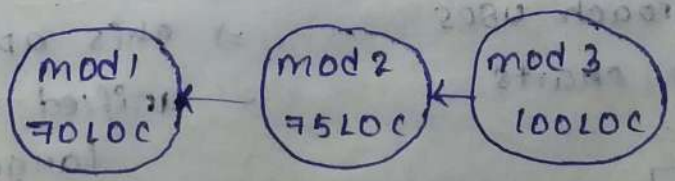
⇒ logic or Algorithm design :-

Basic goal is specifying the logic for different modules of the system. It requires framing an algorithm for a design.

⇒ It also involves evaluating some metrics

metric :-

Software size :- The no. of modules formed for a system form the Inputs



Software size is = sum is 250 LOC = 0.25 KLOC

NOTE:

Depending upon the no. of modules, each & every module size, & interactions among the modules, the software size complexity may increase or decrease.

Scanned by CamScanner

Metric 2 :-

Network metric

(metric for function oriented design)

⇒ This metric explains how good a structure chart is

⇒ Graph impurity = n-e-1

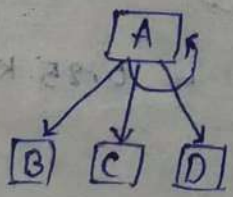
where n = no. of nodes of structured chart
e = no. of edges of " "

Functional Oriented design

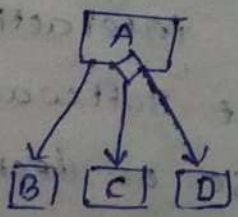
⇒ This design approach is useful for functional / procedural / Structural programming constructs

Ex:- C

⇒ This approach uses structure charts



Iteration



Decision

Object oriented design

⇒ It is a recent approach i.e., well suited for object oriented constructs.

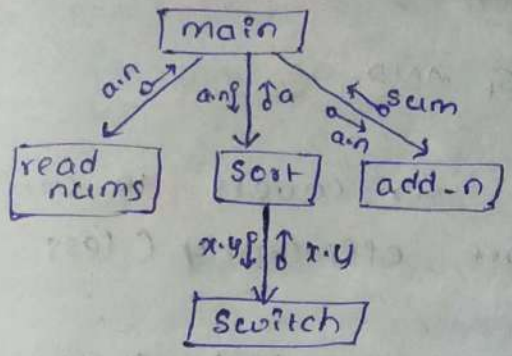
Ex:- C++, Java

⇒ This approach uses unified modelling language (UML) diagrams.

UML

→ class diag
→ sequence diag
→ collaboration diag
→ state chart
→ Interaction

Scanned by CamScanner



Structure chart

UNIT-4

CODING AND TESTING

⇒ It is very important for coders to write code quickly & more efficiently (less errors)

⇒ For this purpose, all reputed software organizations follows basic programming principles & guidelines.

Programming principles :-

⇒ structured programming / go-to less programming

⇒ Information hiding

⇒ some programming practices

⇒ coding standards

Go-to less programming :-

⇒ It is developed as a principle, in order to avoid code skipping (especially forward go to statement)

⇒ This feature restricts the certain operations can only be performed on certain information This is called data hiding or information hiding

It reduces module coupling and achieve system maintainance.

⇒ some programm

a) Control Ab

b) Gotos

c) Informati

d) User defin

e) Trusted

f) given

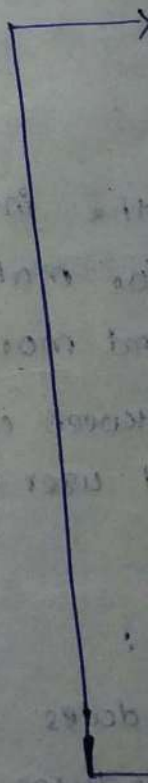
d) User define

a) Namin

b) files

c) stater

d) comm



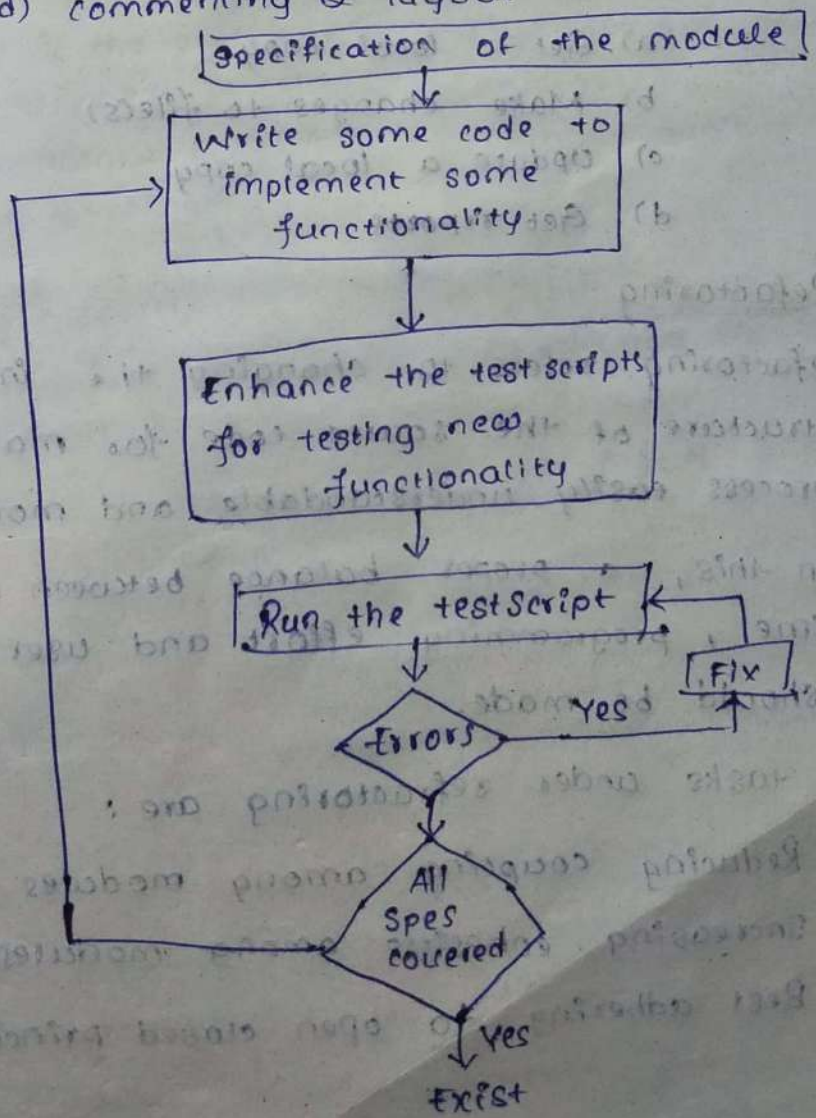
Scanned by CamScanner

⇒ some programming practices

a) Control Abstraction
b) Gotos
c) Information hiding
d) User defined types
e) Trusted data sources
f) given importance to exceptions

d) User defined types :-

a) Naming convention
b) files
c) statements
d) commenting & layout



Scanned by CamScanner

Managing En . code :

It is very important to store, organize and generating reports for different versions of software source code.

(i) Source code control and build :-

Different tools are available to control source code modification.

Ex:- cvs tool for unix (www.cvshome.org) for more information

Sum of the commands used in cvs are related to the following operations.

a). Get a local copy.
b). Make changes to file(s)
c) Update a local copy.
d) Get reports

(ii) Refactoring

Refactoring refers to changing the internal structure of the source code for making coding process easily understandable and more efficient.

=> In this, a proper balance between compilation time, programming effort and user understability should be made.

=> The tasks under refractoring are :

a) Reducing coupling among modules
b) Increasing cohesive among modules
c) Best adhering to open closed principle.

Testing Co

1. Testing :

=> Testing is
the soft
cations
=> It helps
and also
=> Testing i

Tes

=> Testing is
done to
the softw
as the sp

=> Testing sup
pre-requ
debu

2. Test case :

ins

Scanned by CamScanner

Testing Concepts

1. Testing :-

⇒ Testing is the process done to check whether the software is working as per the specifications or not.

⇒ It helps in exposing the errors in the software and also supports debugging.

⇒ Testing is also a measure for software quality.

Testing	Debugging
⇒ Testing is the process done to check whether the software is working as the specification	⇒ Debugging is the process to correcting the error. (location, analyze, correct)
⇒ Testing supports/ pre-requisite the debugging	⇒ Debugging depends on testing results.

2. Test case : It consists of set of inputs & education instructions.

Scanned by CamScanner

Test suit

The collection of related testcases forms a test suit.

Test hardness

Test hardness, it helps in automating the testing process, also known as testing framework.

Manual	Automation
Testers:	Tester:
→ write testcases	→ design
→ Execute testcases	→ decision
→ Generate results	Tool:
→ Take some decision	→ Executes testcases
	→ generates results

Examples of tools:

Java, .Net → QTP (Quick Test Professional)

(freeware) Selenium → Java

Perfect testing → load Runner

Error :-

Error is the reference to the difference among the expected & actual outcomes of a test step.

⇒ It is possible because of an human action.

Fault:

It is a condition that causes a system to fail in performing its required function (also known as Bug or defect).

⇒ possible reason could be a software null function.

Failure:

⇒ It is a
to perform
its funct
⇒ until and
reasons
genera
levels of

1, Unit testi
which
their

2, Integrati
combine

3, System
syste

4, Accep
so

α- tes

β- te

Regression

Test

co

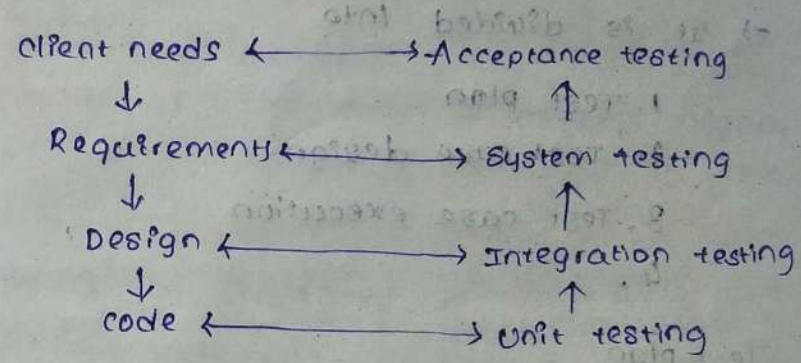
Scanned by CamScanner

Failure:

⇒ It is a in-ability of a system (or) component to perform its functionality according to its functionalities.

⇒ until and unless the failure occurs, the reasons for possible errors & faults are generally not traced out.

levels of testing



1, Unit testing :- It the 1st level of testing in which programmers (or) developers test their own code for errors.

2, Integration Testing :- Two or more modules are combined tested for any interface errors.

3, System Testing: All the modules of the system are tested for errors.

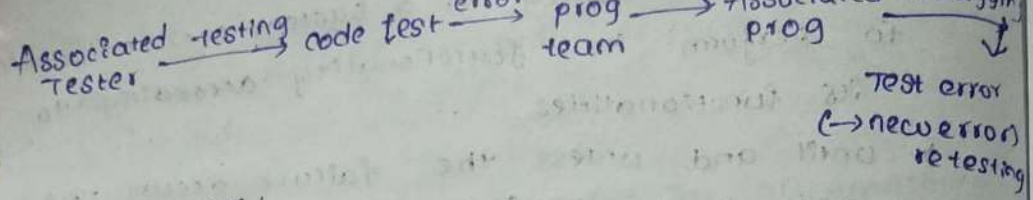
4, Acceptance testing : The end user test the software for errors.

α- testing: Developer environment

β- testing: client environment.

Regression Testing :-

Testing is done in order to check whether corrected code has new errors or not



Testing process:

It is, the overall task of designing, planning executing & analyzing the test reports.

⇒ It is divided into

1. Test plan
2. Test case design
3. Test case execution
4. .

Test plan

It consists of following steps:

Step 1: planning the test activities

Step 2: Test environment, Report references

Step 3: Testing techniques involve

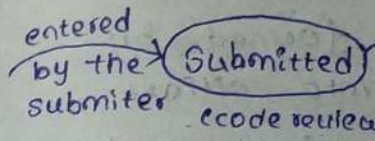
Step 4: Expected format of test outcome

Step 5: Team size fixation & allocation

Testcase Design

The primary task of a texture has to design easily understandble and more efficient test case.

Testcase Exec



(Lifecycle of

Testing Tech

The efficien
depends
projects.

⇒ These are

1. Black
2. White
3. greybo

Black bo

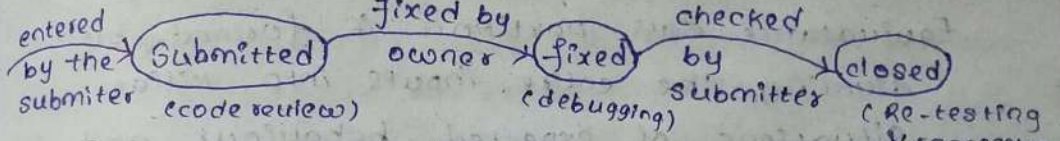
* This testing
trys to and
the function
a progra

* Coding knoc
not requir

*

Scanned by CamScanner

Testcase Execution



(Lifecycle of a defect)

Testing Techniques :-

The efficiency of the testing process highly depends on technique(s) involved in the projects.

⇒ These are $ of 3 types

1. Blackbox technique
2. Whitebox "
3. greybox "

Black box	White box
* this testing technique trys to understand the functionality of a program only.	* this testing technique trys to understand the program's internal structure
* Coding knowledge is not required	* code should be visible for testing
*	

Scanned by CamScanner

Blackbox testing technique

1. Equivalence class partition :-

Equivalance class partition is formed by dividing the entire inputs into either situations of expected behaviour and errorious conditions

⇒ In order to avoid exhaustive testing / 100% testing, whitebox testing techniques are developed.

Ex:-

To create a password, the characters allowed are alphabets (both upper & lowercase), numbers from 0 to 9 and special symbols like @, ! etc

Valid	Invalid class
→ alphabets (lower, uppercase) special characters, numbers	→ all alphabets → all numbers. → alphanumeric numbers → all alphanumeric numbers & special symbols

Ex-2:

The minimum no. of characters allowed is 8 & maximum no. is 255. Draw equivalence class partition table for this

Valid class	Invalid class
→ Greater than 8 characters & less than 255 characters	→ Empty → less than 8 characters → more than 255 characters

Boundary Valid

This techniqu
fact that the
are at and
proved to

Ex:-

The minimum
and maxima

Note:

The general
are return
and, x

1) Control flow

In this te
program be
area are

Statement

Each & e
be exec

Branch

Each a
stat

Scanned by CamScanner

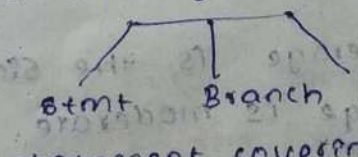
Boundary Valid Analysis

This technique is developed based on the fact that the values of the testcases which are at and nearby boundary values are proved to be more effective.

Ex:-
The minimum no.of characters allowed is 8 (x) and maximum is 255 (y)

Note:
The general values for which the testcases are returned are x, $x-1$, $x+1$, y, $y-1$, $y+1$ and $\frac{x+y}{2}$

1) Control flow based testing :-
In this technique, the execution order of the program segment and all coverage area are tested in a control flow



statement covering
(or)
All nodes criteria

Statement covering (or) All nodes criteria :-
Each & every decorative statements must be executed atleast once.

Branch covering / all edges criteria :-
Each and every possible conditional statements must be executed.

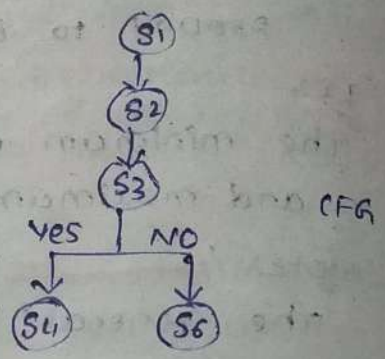
Scanned by CamScanner

Path coverage:-

This issue handles maximum efficient possible paths of loop statement.

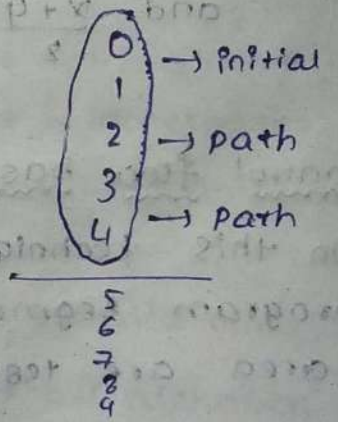
```
Stmt { S-1   pf("enter age");
     { S-2   sf("%d", &age);

Branch { S-3  if(age>=18)
       { S-4  pf("Major");
       S-5    else
       S-6    pf("Minor");
```



CFG

```
S1 for(i=0;i<5;i++)
S2 {
S3      pf("Hello");
S4 }
S5 ←
```



Note:

Statement coverage is the simplest Branch coverage is moderate. path coverage is strongest in catching statements.

⇒ All the three coverage issues are applied in most of the program segments.

2) Mutation testing

If the tester intentionally modifies the program source code it is known as mutation testing.

⇒ The modified code is known as mutant.

Scanned by CamScanner

⇒ If the outcome of original code & the modified code are different then it is known as live mutant. else dead mutant.

⇒ This technique also helps in code improvement along with checking of errors.

⇒ Mutation is not restricted to single line / statement

Eg:

Original code	Live Mutant	Dead mutant

for(i=0;i<3;i++)
pf("Hai");
 	
o/p:- Hai
Hai
Hai
	pf("In Hai");
	pf("In Hai");
	
	o/p: Hai
	Hai
		pf("In Hai");
		pf("In Hai");
		pf("In Hai");
		
		o/p: Hai
		Hai
		Hai

⇒ It can be applied to multiple line, multiple conditions, (or) loop extension conditions.

Risk:

A risk is a probabilistic event which has a negative impact on the slw project. The reason to be human; technical, non-technical, management client, legal, social, environmental, global, market, ---etc.

Scanned by CamScanner

Reactive Vs Proactive risk Strategies

Proactive risk strategy

⇒ It begins long before technical work is initiated

⇒ potential risks are identified, their probability and impacts are assessed & ranked for importance.

⇒ The main principle here is a contengency plan delevelopment in a more efficient & controlled manner.

Reactive risk strategy

⇒ It begins with monitoring for likely risks

⇒ The counter measures are applied only when something goes wrong

⇒ The software team tries to correct the problem rapidly, if it fails crisis management task takes over

⇒ It is also known as fire-fighting mode.

Software Risks

Any software risk has 2 characteristics

1) Uncertainity (not possible of 100% occurance) and 2) loss (If the risk is materialized, there would be a definite negative impact on project)

categories

1) project
2) Technical
3) Business
4) known
5) Predicted
6) Unpredicted

Scanned by CamScanner

Project risks:

⇒ This risks occurs, will hault the entire software project

⇒ It is given high priority among all the 6 risks

Ex:-
Project cost, project schedule, resource, stakeholders, requirements problem etc.

Technical risks :-

⇒ They threaten the quality & time to produce the software

Ex:- Design flaws, interface errors, verification & maintainance problems

Business risks :

⇒ They threaten the viability of the sw

1) Ex:- Market business risk :-
Building an excellent system / product no one

2) Sales risk :-
Building a product that the sales force doesn't understand how to ~~sale~~ sell.

known risks :-

⇒ After careful evaluation of the project plan, the business & technical environment in which the project is being developed, some risks can be identified & listed

Predictable risks :

⇒ These are obtained from experience of the past projects (personal)

Scanned by CamScanner

Unpredictable risks

They can and do occur but are extremely difficult to identify in advance..

* Risk Management basically involves 4 activities
  1) Risk identification
  2) Risk assessment
  3) Risk prioritization
  4) Risk Mitigation

1. Risk identification

It is a systematic attempt to specify threats to the project plan.

Ex:-

Estimates schedule, resource loading etc.

=> The project manager takes the first step of identifying known & predictable risks

=> Risk identification is classified into
  1. generic risks
  2. product-specific risks.

1. Generic risks :-

A potential threat to every software project

2. product-specific risks :-

Only identified with a clear understanding of technology & the people & the sw specific environment

methods:-

=> Maintaing the check-list of all possible risks which include the following

Scanned by CamScanner

* product size
* business impact
* customer characteristics
* process definition
* development environment
* Technology to be built
* staff size & experience

Relavent questions on these issues are discussed and solutions are taken into account for assessments

Method-2

Risk components & the drivers list maintainance

2. Risk Assessment

It is proposed by US Air force for guidelines of slw risk identification

The risk components are

* performance risk
* cost risk
* support risk
* schedule risk

The risk categories are

1, catastrophic :-

This risk is capable of haulting the slw project immediately after its occurance

2, critical :-

this risk threatens the project success

3. Marginal :-

It has a significant impact on the project

4. Negligable :
It is the least priority risk category
(sometimes these risks are postponed to
next versions of the s/w).

Risk Assessment table developed by US Air force

category	performance/support	cost/shedule
catastiophic	Failure to meet the requirement would result in mission failure	Failure result in cost increase & delay in scheduling
critical	Failure to meet the requirements would degrade system performance to point where mission success in questionable	Failure results in operational delays, and/or increased costs
Marginal	Failure to meet the requirement would result in degradation of secondary mission	costs, impacts &/ or recoverable. Schedule slips with marginal expected values
Negligible	Failure to meet the requirement would create inconvienience or non-operational impact	Error results in minor cost and/or schedule impact with very less expected value

3. Risk P
In th
⇒ the
rea
⇒ Con
th
⇒ Ar
Risk
* Estal
lik
* Del
* Es
* No
By
⇒

Scanned by CamScanner

3. Risk projection / Risk Estimation :

In this step,

⇒ the likelyhood or probability, that the risk is real and

⇒ Consequences of the problem associated with the risk, if occurs

⇒ Are verified

Risk Projection steps

* Establish a scale that reflects the perceived likelihood of a risk.
* Delineate the risk consequences
* Estimate the risk impact on the project & product
* Note the overall accuracy of the risk projection so that there will be misunderstanding

By prioritizing risks, the team can allocate resources where they will most impact.

⇒ developing a risk table

This table contain

1. Description of the risk
2. Risk category
3. probability of the occurance of the risk
4. If the risk occurs, the impact factor of the risk on the project.
5. RMMM a pointer to RMMM Plan.

Scanned by CamScanner

Risk	category	probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Funding will be lost	CU	40%	1	
large no. of users than planned	PS	30%	3	
less reuse than planned	PS	70%	2	
lack of training of tools	DE	80%	3	
Technology will not meet expectations	TE	30%	1	
Staff turn over will be high	ST	60%	2	

Scanned by CamScanner

RMMM

Impact Values:

1. Catastrophic
2. critical
3. Marginal
4. Negligible

Risks:

PS : Project Size risk
BU : Business risk
CU : Cost effective risk
ST : Staff Turn over risk
DE : Technology related risk.

a) Sample risk table prior to Sorting

In the aboue table write it docun Sorting order by the Impact values.

⇒ After Sorting out the risks, the project mangger fixes a cutoff line, which means only the risks aboue the cutoff line will be given a first-order prioritization else second-order prioritization is given.

b) Assessing the resource Risk impact

To determine the overall risk consequences the following steps are recommended by the US airforce,

1- Determine the average probability of occurance value for each risk component.

2. Using the impact Assessment diagram, determine the impact for each component based on the criteria shown

Scanned by CamScanner

3. Complete the risk table and analyze the results as described below.

Overall risk Exposure (RE)

$RE = P \times C$

where P = Risk probability occurance

C = cost to the project when risk occurs

C is given by the product of avg component development, avg loc required for developing a component & cost of developing the component

Example:

Suppose we have a system of 60 components in which 70% are useful which means 42 components are useful remaing 18 components needs some work to be done

⇒ let us consider the avg code required for developing a component = 100 LOC, avg cost to develop a component = 14

$C = \text{no.of component risking} \times \text{avg LOC code} \times \text{cost}$

$C = 18 \times 100 \times 14$

$C = 25200$

⇒ let us assume the likelihood / probability be 0.8 or 80%

Now $RE = P \times C$

$= 0.8 \times 25200$

$RE = 20160$

Scanned by CamScanner

Risk refinement :-

Over the progression of time with experience, it is possible to define the risk into a a set of more detailed risks.

⇒ This helps in making the risks easier to mitigate, monitor & manage.

⇒ One way to refine a risk is to maintain is CTC format.

Condition transistion consequence (CTC) format :-

given that <condition> that there is concern that (possibility) <consequence>

⇒ The weightage of

This general condition can be refined as

Subcondition 1 :-

Certain reasable components were deveioped by a third party with no knowledge of internal design standards.

Subcondition 2 :

certain reusable components have been implemented in a language that is not supported on the target environment.

Main condition

The weightage of less usage of components in a project.

Scanned by CamScanner

RMMM plan :-

Risk mitigation monitoring & management.

⇒ The primary responsibilities of a project manager are

1. Risk avoidance
2. Risk monitoring
3. Risk management & contingency planning.

Risk avoidance :-

An identified risk has not to be repeated in the current project.

Risk information sheet :-

Risk ID: P-230	Date: 3/9/20	prob: 80%	Impact: high
Description: Only 70% of the S/W components scheduled for reuse will infact, be integrated into the application. the remaining functionality will have to be custom developed.			
Refinement / context: Subcondition 1 Subcondition 2			
Mitigation / monitoring: 1. Contact 3rd party to determine conformance with design standards 2. press for interface standards completion; consider component structure when deciding on interface protocol			

Scanned by CamScanner

Mitigation steps initiated on 6/2/2020	
originated by	Initiated by

this is the risk information sheet.

⇒ Based on the information in the risk assessment sheets, the project manager derives a cutoff line for the entire risks

UNIT-5

⇒ Risks above the cutoff line will be given high priority.

UNIT-5

SOFTWARE PROJECT ESTIMATION

⇒ Software project Estimates are required to efficiently plan to complete the software project without any hurdles

Examples:

→ No.of users
→ Investment cost
→ Completion time
→ Design estimates
→ effort estimates
→ Risk estimates

Decomposition Techniques

⇒ The decomposition Technique helps in bifurcating / dividing the complex problem of project estimation into smaller components.
so, that it will be easy to derive each individual estimate and integrate them finally.

⇒ Putnam & Myers proposed 4 approaches based on slw sizing problem

* Fuzzy logic sizing
* Function point sizing
* Standard component sizing
* change sizing

Scanned by CamScanner

⇒ An example of LOC-Based Estimation :-

Functions	Estimated LOC
* User interface & control facilities (UICF), Two-dimensional geometric analysis (2DGA)	3,300 5,300
* Three-dimensional geometric analysis (3DGA)	6,800
* Database management (DBM)	3,350
* computer graphics display facilities (CGDF)	4,950
* pheripheral control function (CPCF)	2,100
* Design Analysis Modues (DAM)	8,400
Estimated lines of code	33,200

* ~~LOC based Estimation~~ :-

⇒ LOC based Estimation decomposes entire source code into different functionalities.

⇒ The LOC required for each functionality is estimated and finally they are integrated to estimate the overall LOC required for the S/W.

* Function point based Estimation (FP) :-

⇒ This approach works on the basis of the domain or the range of I/P values.

Scanned by CamScanner

Information domain value	opt	likely	pess	Est count	weight	FP count
No. of external i/p's	20	24	30	24	4	97
No. of external o/p's	12	15	22	16	5	78
No. of external enquires	16	22	28	22	5	88
No. of internal logic files	4	4	5	4	10	42
No. of external interface files	2	2	3	2	7	15
count total						320

Factor	value
(1) Backup and Recovery	4
(2) Data communications	2
(3) Distributed processing	0
(4) Performance critical	4
(5) Existing Operating environmental	3
(6) One-line data entry	4
(7) I/p transaction over multiple sweens	5
(8) ILF's updated online	3
(9) Information domain values	5
(10) Internal processing complex	5
(11) code designed for reuse	4
(12) conversion/installation in design	3

Scanned by CamScanner

(13) Multiple installations	5
(14) Application designed for change	5
Value adjustment factor	1.17

$FP_{estimated}$ = count-total $\times [0.65 + 0.01 \times \Sigma(F_i)]$

$= 14 \times [0.65 + 0.01 \times 1.17]$

$= 375$

Usecase based Estimation

⇒ Here, system is decomposed into subsystems.

	Usecases	Scenarious	Pages
User interface subsystem	6	10	6
Engineering subsystem group	10	20	8
Infrastructure Subsystem group	5	6	5

Total LOC estimate - 42568

Scenarious	pages	LOC	LOC estimate
12	5	560	3,365
16	8	3100	31,233
10	6	1650	7,970

Scanned by CamScanner

Usecase represents the external view of sl/w & different levels of abstraction. They can't explain the complex behaviour among different functions.

$$LOC\ estimate = N \times LOC_{avg} + \sum (S_a | S_{n-1}) + (P_a | P_{n-1}) \times LOC_{adjust}$$

Process based Estimation Approach

This approach is based on various tasks involve in software engineering & its related issues.

Activity	CC	planning	Risk analysis	Engineering		Construction release		CE	Total
Task →				Analysis	Design	code	Test		
Function									
UICF				0.50	2.50	0.40	5.00	N/a	8.40
2DGA				0.75	4.00	0.60	2.00	n/a	7.35
3DGA				0.50	3.00	1.00	3.00	n/a	8.50
CGDF				0.50	3.00	1.00	1.50	n/a	6.00
DBM				0.50	3.00	0.75	1.50	n/a	5.75
PCF				0.25	2.00	0.50	1.50	n/a	4.25
DAM				0.50	2.00	0.50	2.00	n/a	5.00
Totals	0.25	0.25	0.25	3.50	20.50	4.50	16.50		46.00
% effort	1%	1%	1%	8%	45%	10%	36%		

Scanned by CamScanner

UNIT-VI METRICS FOR PROCESS & PRODUCTS

Software Metrics
- Process
- Project
- Product

⇒ Metrics means attribute / characteristic of the slw system.

Process metrics	Product metrics	Project metrics
⇒ Response time	⇒ size	⇒ no.of slw developers
⇒ Execution speed	⇒ complexity	⇒ no.of teams/ persons
⇒ To fix & bugs & errors	⇒ Design features	⇒ cost
⇒ To findout & removal of defects in process	⇒ performance	⇒ scheduling/ planning
	⇒ quality level	⇒ estimation

Software Measurements

It is measured based upon the 5 categories

1. size oriented
2. Function oriented
3. object oriented
4. Usecase oriented
5. Web Engineering

* Software measurement
  - Direct → process, product
  - Indirect → product

* Software measurements can be classified in 2 ways
  1. Direct measures
  2. Indirect measures

* Direct measures of the s/w process include cost & effort applied

* Direct measures of the product include lines of code produced, execution speed, memory size & defects reported over some period of time

* Indirect measures of the product include functionality, quality, complexity, efficiency, reliability, maintainability & many other abilities

* s/w measurements can be defined by the following ways :-

  ⇒ Size oriented metrics
  ⇒ Function oriented metrics
  ⇒ Reconciling LOC & fp metrics
  ⇒ Object oriented metrics
  ⇒ Usecase oriented metrics
  ⇒ web-app project metrics.

Size oriented metrics

Size oriented s/w metrics are derived by normalising quality & productivity measuring by considering the size of the s/w that has been produced.

⇒ A set of simple size oriented metrics can be developed for each project.

Scanned by CamScanner

1. Errors for KLOC (i.e., 1000 lines of LOC).
2. Defects for KLOC
3. cost of KLOC
4. phases of documentation for KLOC
5. In addition other interesting metrics can be calculated, error for person of month, KLOC for person of month, cost of for page of documentation.

project	LOC	effort	$(1000)	P.Pdoc	Errors	defect	People
alpha	12,100	24	168	365	134	29	3
beta	27,200	62	440	1224	321	86	5
gamma	20,100	43	314	1050	256	64	6

function oriented metrics

Function oriented slw metric use a measure of the functionality delivered by the application as a normalisation value.

⇒ The most widely used function oriented metric is the function point (FP)

⇒ Computation of the function point is based on characteristics of the softwares information domain and complexity.

Reconciling LOC & functional point metrics:

The relationship blw LOC & function points depends upon the programming language, that is used to implement the software & the quality of the design.

Scanned by CamScanner

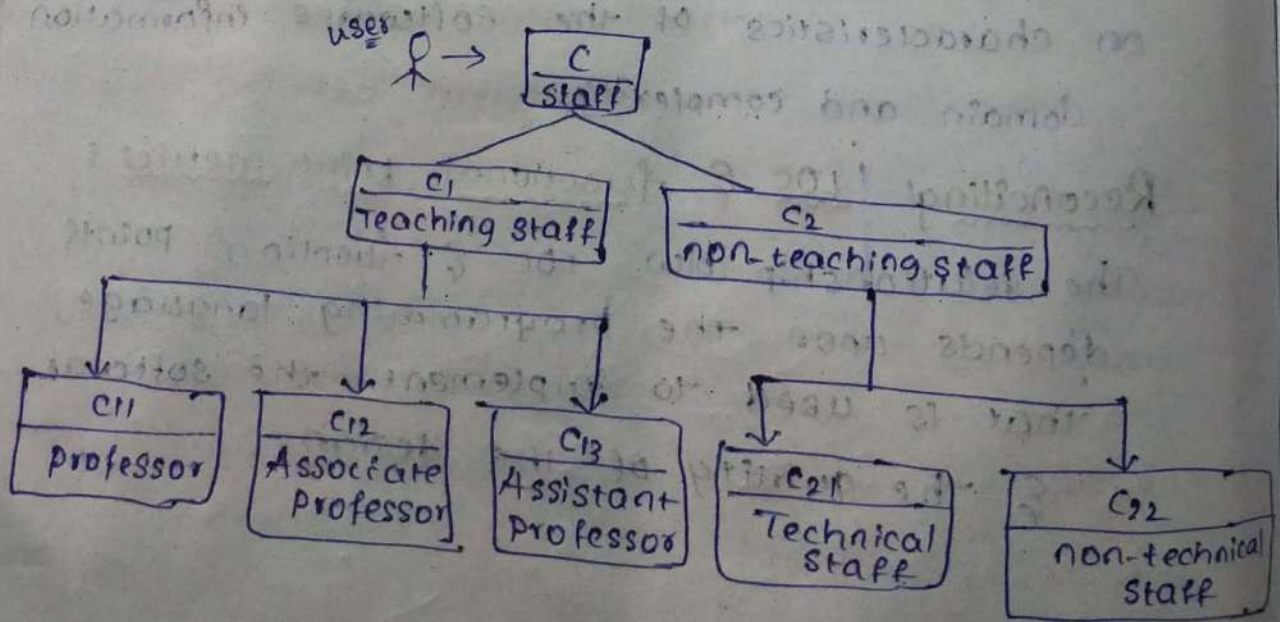
programming language	LOC per fp			
	Average	Medium	low	high
Access	35	38	15	47
Ada	154	–	104	205
C++	66	53	29	178
C#	162	109	33	704
–	–	–	–	–
–	–	–	–	–

Object oriented Metrics

Conventional s/w project metrics LOC & function point can be used to estimate Object oriented s/w projects

⇒ The following are the set of metrics for Object oriented projects.

1. No. of Scenario Scripts
2. No. of key classes
3. No. of Support classes
4. avg no. of support classes for key class
5. No. of subsystems.

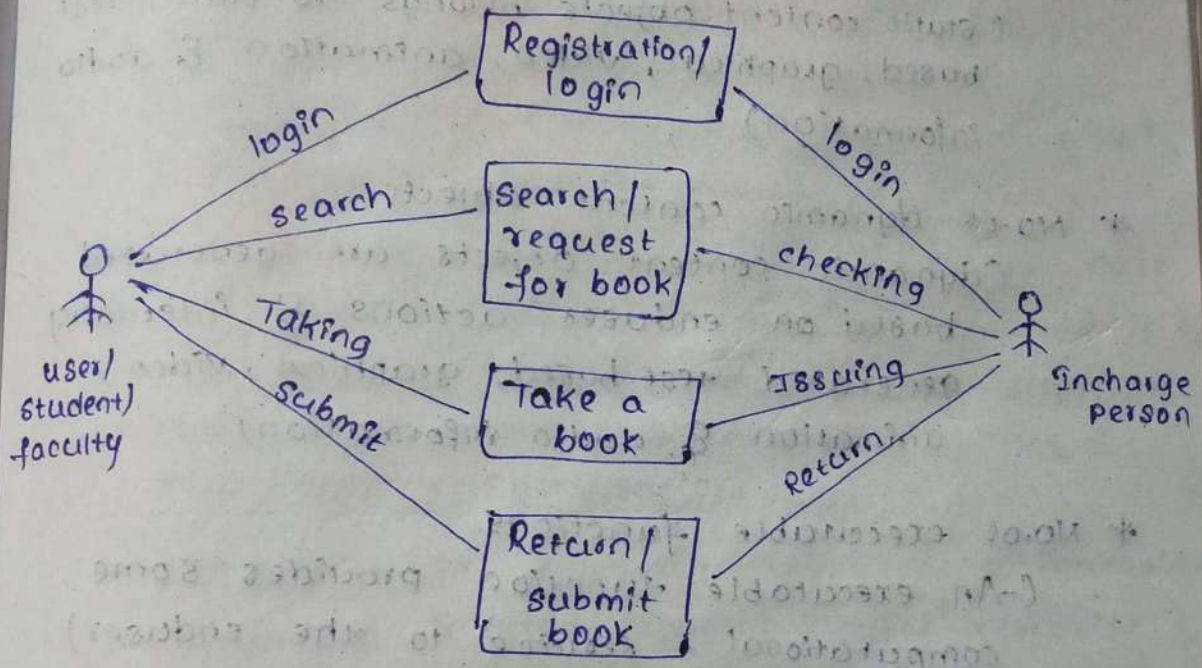


Scanned by CamScanner

Usecase oriented Metrics

Usecases are used widely as a method for describing customer level or business domain requirement that imply s/w features & functions.

Library system



Web-application project metrics

The objective of all web-application projects is to deliver a combination of content & functionality to the end-user

⇒ The following are the set of metrics for web application projects

* No.of static webpages
  (webpages with static content)
* No.of dyanamic webpages
  (webpages with dyanamic content)
* No.of internal page-links
  (internal page-links are pointers that provide a hyperlink to some other webpages within the web application)

Scanned by CamScanner

* No.of data objects
(one/more data objects (database or datafile(s)) may be accessed by a webapplication)

* No.of external system interface
(web applications must interface with business ~~interfaces~~ applications)

* No.of static content objects
(static content objects belongs to static text based, graphical, video, animation & audio information)

* No.of dynamic content objects
(dynamic content objects are generated based on enduser actions & internally generated text based, graphical, video, animation & audio information)

* No.of executable functions
(An executable function provides some computational service to the enduser)

customization index, $$C = \frac{N_{dp}}{N_{dp} + N_{sp}}$$

where N_{dp} = No.of dynamic web pages
N_{sp} = No.of static web pages

Scanned by CamScanner

Metrics for SLW quality

Software Engineering is to produce a high quality system, application & product within a time frame that satisfies a user requirements.

Measuring quality

There are many measures of SLW quality. They are:-

1. **correctness** :- A program must operate correctly & it provides a little value to its user.

2. **Maintainability** :- Maintainability is the ease with which a program can be corrected if an error is encountered, adapted different environment changes & change in the requirement specification

3. **Integrity** :- SLW integrity has become increasingly important in the age of cyber terrorist & hackers. To measure integrity, 2 additional attributes must be defined.
   * threadt
   * Security

$$\text{Integrity} = \sum [1 - (\text{threat} \times (1 - \text{security}))]$$

4. **Usability** :- Usability is an attempt to quantify ease of use & can be measured in terms of the characteristics of the SLW product.

Scanned by CamScanner

Defect Removal efficiency

A quality metric that provides benefit & both the project & process level is defect removal efficiency (DRE)

⇒ DRE is defined in the following manner.

$$DRE = \frac{E}{E + D}$$

$$DRE_i = \frac{E_i}{E_i + E_{i+1}}$$

where E = no. of errors found before delivery of the s/w to the enduser

D = no. of defects found after delivery

E_i = no. of errors found during s/w engineering action (i)

E_{i+1} = no. of errors found during s/w engg action (i+1)

QUALITY MANAGEMENT

Quality Concepts

Quality is a complex & multifactor concept that can be described from 5 different points of view

1, The transcendental view

Quality is something that you immediately recognize but can't explicitely defined.

2, The user view :-

Quality in terms of an end user specific goals.

Scanned by CamScanner

3) The Manufacture view :-
Define Quality in terms of the original specifications of the product.

4) The product view :-
Quality described by the characteristics of the product (functions & features of the product)

5) The value based view measures :-
Quality based on how much a customer is willing to pay for a product.

Quality

⇒ Quality has a characteristic or attribute of something

⇒ Quality of design refers to the characteristics that designed as specify for a product

⇒ S/W development quality of design defines the degree to which the design meets the function and features, specified in the requirement model.

User satisfaction = Complaint free product + good quality + Delivery within budget & schedule

Software Quality

An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.

Scanned by CamScanner

⇒ SW Quality enhances the 3 important points

* An effective software process establish the infrastructure that supports any effective building a high quality software product.
* A useful product delivers the concept, functions & features that are end users desires.
* By adding value for both the product and user of a software product, high quality software provide benefits for the SW organisation and end user community.

ISO 1926, Software Quality factors

The ISO 1926 standard was developed in an attempt to identify the key quality attributes for computer software.

⇒ The standard identifies 6 key quality attributes:

1. Functionality :-

It identifies the suitability, accuracy & security of the product.

2. Realiability :-

It defines the maturity, recoverability & Performance of the product.

3. Usability :-

Software is easy to use as indicated by the understandability & operatability of the product.

Scanned by CamScanner

4. Effecency :-

Software makes optimal use of system resources as indicated by the time behaviour & resource behaviour

5. Maintainability :-

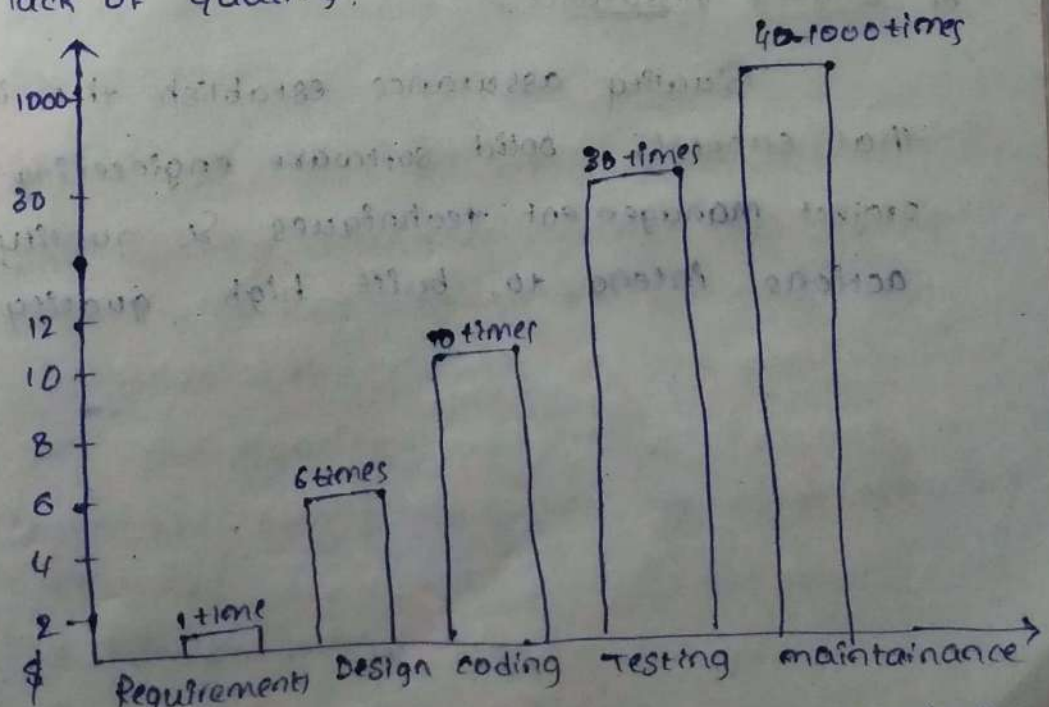
The ease with which repair may be made to the slw as indicated by the analysis, stability changability & testability.

6. Portability :-

The ease with which the slw can be transported from one environment to another as indicated by the adaptability, installation of the product & replacement of the product.

Cost of Quality

The cost of quality includes all costs in the persuit of quality (or) in performing quality related activities and the downstream cost of lack of quality.



Relative cost of correcting errors & defects.

Scanned by CamScanner

Achieving slw quality

The contest of 4 activities that help a slw team achieve high quality software.

i. slw engineering methods :-

If you expect to built high quality slw, you must understand the problem to be solved by applying the suitable methods.

ii. project management techniques :-

A project manager uses estimation to verify the delivery are achievable.

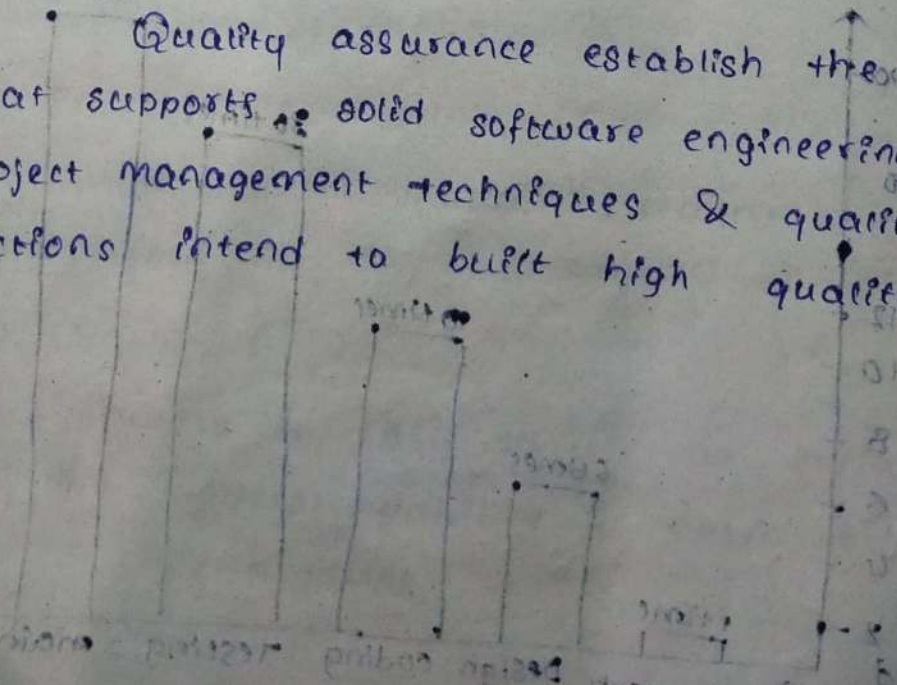
[schedule planning
risk planning]

iii. Quality control :-

Quality control defines a set of slw engineering actions that help to ensure that each work products meet its quality goods.

iv. Quality Assurance :

Quality assurance establish the infrastructure that supports solid software engineering methods, project management techniques & quality control actions intend to built high quality software



Scanned by CamScanner

Software Quality Assurance (SQA)

SQA defines the following things :-

1. software quality assurance process
2. Specific quality assurance & quality control tasks (technical reviews & testing strategies)
3. Effective slw engg practice (Methods & tools)
4. Control of all software work products
5. A procedure to ensure compliance with slw development standards
6. Measurement and reporting mechansims.

Background Issues :

⇒ The first format quality assurance & control functions was introduced in Bell lab's in 1960's

⇒ During the 1940's more format quality control was introduced (described)

⇒ These based on the measurement & continue process improvement as key of quality management.

Elements of slw Quality Assurance

⇒ slw quality assurance elements defined by the a board range of concerns and activities that focus on the management of slw quality

⇒ These can be summarized in the following manner:

1. Standards
2. Reviews & auditing

Scanned by CamScanner

3. Testing
4. Error & defect correction & analysis
5. change management
6. Education
7. Vendor management
8. Security management
9. Safety
10. Risk management

SQA tasks or Activities:-

SQA is composed of a varieties of tasks associated with two different categories

1. software engineers who do technical work &
2. SQA group that has responsibility for quality assurance planning, record keeping, analysis & reporting

2. SW engineers address quality by applying solid technical methods & measures, conducting technical reviews & performing well planned SW testing

⇒ The following are the SQA tasks :-

* prepares & SQA plan for a project
* participants in the development of the SW projects SW process description
* reviews SW engineering activities to verify compilance with the defined SW process.
* Audits designed SW work products to verify compilance with those defined as the part of the software process.

Scanned by CamScanner

* Ensure that deviations in software work & work products are documented & handled according to a documented procedure
* Records any non compliance & reports to Senior management.

Software Reviews

Software reviews are a filter for the software process.

i.e., reviews are applied at various points during software engineering and to uncover errors and defects that can be removed

⇒ Software reviews purify s/w engg workproducts including requirement & design models, coding & testing data

⇒ Many different types of reviews can be conducted as part of software engineering (informal reviews & formal reviews)

⇒ s/w engineering mainly focus on technical or peer reviews

⇒ these are done by the casual reviews, inspections &

⇒ A technical review (TR) is the most effective filter from a quality control stand point conducted by the s/w engineers.

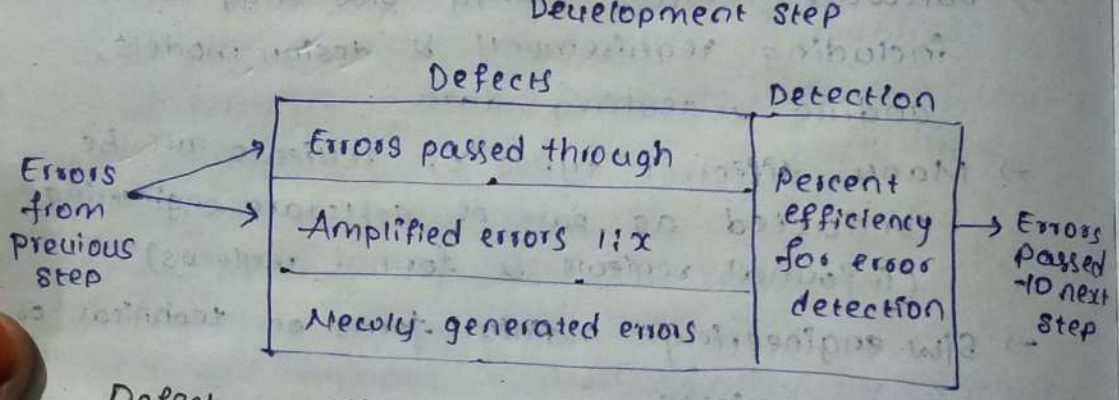
Scanned by CamScanner

Cost Impact of software defects

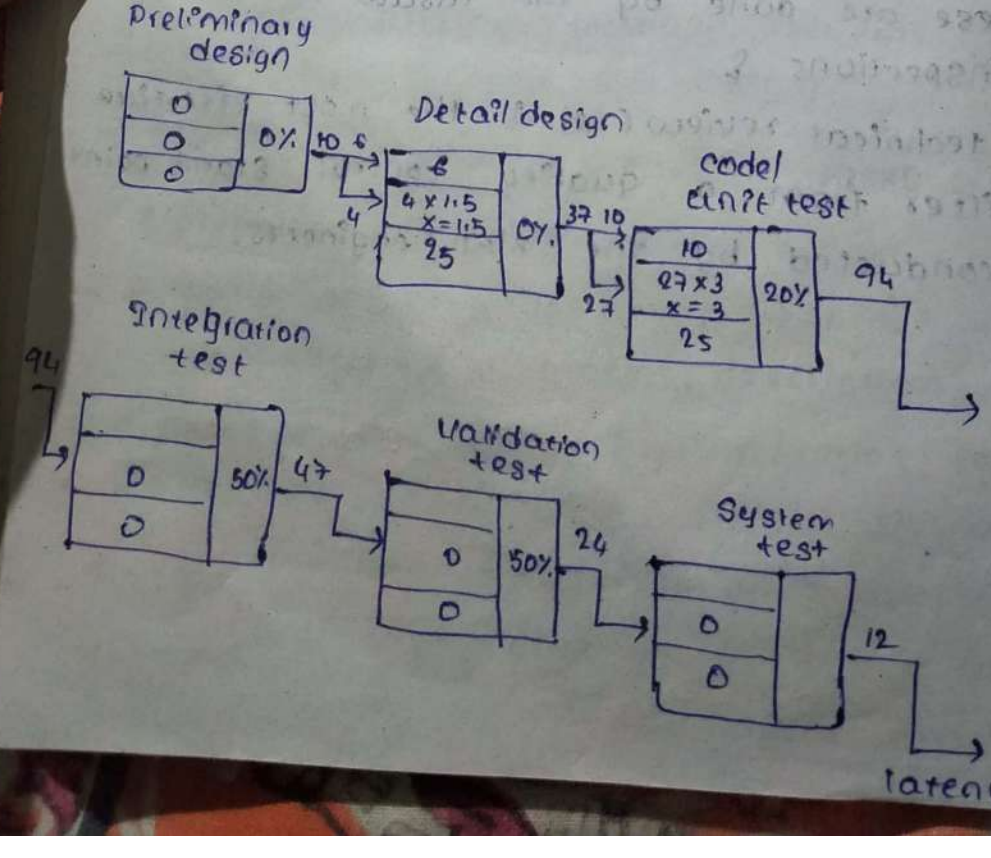
The primary objective of technical reviews is to find errors during the process so that they don't become defects after release of the software.

Defect amplification & removal method

A defect amplification model can be used to illustrate the generation & detection of errors during the design & code generation actions of a software process.

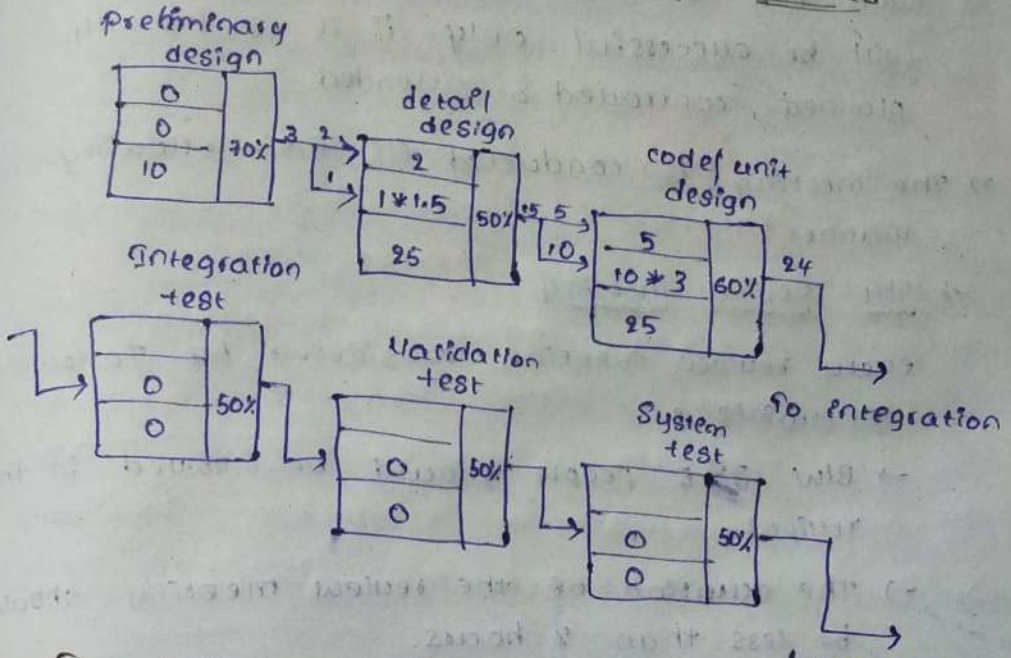


Defect amplification, no reviews



Scanned by CamScanner

Defect amplification, reviews conducted



Formal Technical Reviews (FTR) :-

A formal technical review is a SW quality control activity performed by SW engineers.

⇒ The objectives of FTR are

* To uncover errors in functions, logic (or) implementation for any representation of the software.
* To verify that the SW under review meets its requirements
* To ensure that the SW has been represented according to predefined standards
* To achieve software that is developed in a uniform manner.
* To make projects more managable.

⇒ FTR is actually a class of reviews that includes casual review, inspections & walk throughs.

Scanned by CamScanner

⇒ Each FTR is conducted as a meeting & will be successful only if it is properly planned, controlled & attended.

⇒ The meeting is conducted in the following manner:

* The Review meeting

every review meeting considered by following constraints:

→ Blw 3 & 5 people should be involved in the review.

→ The duration of the review meeting should be less than 2 hours.

→ Intimating the time before conducting the meeting.

* Review reporting & record keeping

A formal technical review summary report is completed. A review summary report answers 3 questions.

→ What was reviewed?
→ Who reviewed it?
→ What were the findings & conclusions?

* Review guidelines

The following represents a minimum set of guidelines for formal technical reviews.

→ review the product
→ set an agenda & maintain it
→ limit debate & group discussion
→ identify problem areas
→ take ~~no~~ written notes
→ limit the no. of participants in the review

Scanned by CamScanner

→ Develop a checklist for each product i.e., likely to be reduced
→ Allocate resources & schedule time for FTR
→ conduct meaningful training for all reviewers

* Sample driven reviews :-

To accomplish this, the following steps are suggested :-

⇒ Inspection a fraction A_i of each s/w work product; &
⇒ Record the no.of parts f_i found within A_i;
⇒ Develop a gross estimate of the no.of calls within work product; by multipling f_i by $\frac{1}{A_i}$
⇒ Sort the work products in ascending/ deseending order.
⇒ Focus available review resources

Statistical S/W Quality Assurance (SQA)

SSQA implies the following steps.

1. Information about s/w errors & defects is collected & categorized
2. An attempt is made to trace each error & defect to its underlying cause
3. Once the cause have been identified move to correct the problems that have caused the errors & defects.
4. Although 100's of different problems are uncovered, all can be dragged

Scanned by CamScanner

i. IES :- Incomplete or erroneous specifications
ii. MCC :- Miss interpretation of customer communication
iii. IDS :- Intentional deviation from specifications
iv. VPS :- Violation of programming standards
v. EDR :- Error in data representation
vi. ICI :- Inconsistent component Interface
vii. EDL :- Error in design logic
viii. IET :- Incomplete or erronious testing
ix. IID :- Inaccurate or Incomplete documentation
x. PLT :- error in programming language translation of design
xi. HCI :- Inconsistent human computer interface
xii. MIS :- Miscalleneous

	Total		Serious		Moderate		Minor	
Error	NO	%	NO	%	NO	%	NO	%
IES	205	22	34	27	68	18	103	24
MCC	156	17	12	9	68	18	76	17
IDS	68	5	1	1	24	6	23	5
VPS	25	3	0	0	15	4	10	2
EDR	130	14	26	20	68	18	36	8
ICI	58	6	9	7	18	5	31	7
EDL	45	5	14	11	12	3	19	4
IET	95	10	12	9	35	9	48	11
IID	36	4	2	2	20	5	14	3
PLT	60	6	15	12	19	5	26	6
HCI	28	3	3	2	17	4	8	2
MIS	56	6	0	0	15	4	41	9
Total	942	100	128	100	379	100	435	100

Software

Software re
The probab
a compute
ment for

Measures

If we cons
a simple

where,
MTBF = m
MTTF =
MTTR =

⇒ SLM au
a pro
require
is de

availa

Scanned by CamScanner

ecifications
stomer

specifications
standards
n

face

ng
documentation
age translation

ter interface

te	Minor	
	NO	%
	103	24
	76	17
	23	5
	10	2
	36	8
	31	7
	19	4
	48	11
	14	3
	26	6
	8	2
	41	9
	435	100

Software Reliability

Software reliability is defined in statistical terms. The probability of failure free operation of a computer program in a specified environment for a specified time.

Measures of realiability & availability of s/w

If we consider a computer based system, a simple measure of reliability is

$$MTBF = MTTF + MTTR$$

where,
MTBF = meantime between failure
MTTF = meantime to failure
MTTR = meantime to repair

⇒ s/w availability is the probability that a program is operating according to requirement at a given point intime is defined as

$$availability = [MTTF / (MTTF + MTTR)] \times 100\%$$

Scanned by CamScanner