UNIT- IV

CODING & TESTING

Programming principles and Guidelines :-

* Structured Programming
* Information Hiding
* Some programming Practices
* Coding Standards

Structured Programming :-

⇒ A program has a static structure as well as a dynamic structure. The static structure is the structure of the text of the program. The Dynamic structure of the program is the sequence of statements executed during the execution of the program.

⇒ The objective of structured programming is to write programs so that the sequence of statements executed during the execution of a program.

⇒ The assertion about the expected final state of a program is called the post condition of that program, and the assertion about the input condition is called the pre-condition of the program.

$$P\{S\}Q.$$

If assertion P is true before executing S, then assertion Q will be true after executing S, if the execution of S terminated. Assertion P is the pre-condition of the program and Q is the post condition.

⇒ The most commonly used single entry and single exit statements are:

Scanned by CamScanner

Selection : if B Then S1 else S2

if B Then S1

Iteration : While B do S

repeat S until B

Sequencing : S1 ; S2 ; S3 ; ...

⇒ Structured programming leads to programs That are easier to Understand Than Unstructured programs. and That such programs are easier to ~~understande~~ understande ~~Than~~ formally prove.

Information Hiding :-

⇒ Information hiding can reduce the coupling between modules and make The System more maintainable.

⇒ Information hiding is also an effective tool for managing The Complexity of developing software.

⇒ By using Information Hiding The data to produce Some desired results

⇒ Many of the older languages, like Pascal, C and FORTRAN, do not provide mechanisms to support data abstraction.

⇒ Most modern OO languages provide to implement information hiding.

* Some programming Practices :

There are many programming practices That can also help toward That objective. Some of the following rules That have been found to make code easier to read as well as avoid Some of the errors.

✓ Control Constructs
✓ Goto
✓ Information hiding
✓ user-defined Types

Scanned by CamScanner

✓ Nesting
✓ Module Size
✓ Module Interface
✓ Robustness
✓ Switchcase with default
✓ Empty Catch Block
✓ Return from finally Block etc.

Control Constructs : It is desirable that as much as possible single entry, single-exit construct be used.

Gotos : only when the alternative to using goto is more complex should the goto be used.

Information Hiding : only the access functions for the data structures should be made visible while hiding the data structure behind these functions.

User-Defined Types : Modern languages allow users to define types like the enumerated type. when such facilities are available, they should be exploited where applicable.

Nesting : If nesting of if-then-else constructs becomes too deep, then the logic become harder to understand.

Example if C_1 Then S_1
else if C_2 Then S_2
else if C_3 Then S_3
else if C_4 Then S_4 ; etc.

(harder)

(Easier)

If C_1 Then S1;
If C_2 Then S_2;
If C_3 Then S_3;
If C_4 Then S_4;

Module Size : A programmer should carefully examine any function with too many statements (more than 100 LOC). It may not be functionally cohesive.

Scanned by CamScanner

Robustness : A program is robust if it does something planned even for exceptional conditions.

Switch case with Default : If there is no default case in a "Switch" statement, the behavior can be unpredictable if that case arises at some point of time which not predictable at development stage.

Empty Catch Block : An Exception is caught, but if there is no action, it may represent a scenario where some of the operations to be done are not performed.

Return from finally Block : one should not return from finally Block as it can create false beliefs.

Coding Standards.

Coding Standards provide rules and guidelines for some aspects of programming in order to make code easier to read.

=> Most organizations who develop software regularly develop their own standards.

=> In general coding standards provides guidelines for programmers regarding

* Naming Conventions
* File organization
* Statements
* Documentation (commenting)

Naming Conventions : Some of the standard naming conventions that are followed often are:

-> Package names should be in lowercase (e.g. mypackage)

Scanned by CamScanner

→ Type names should be nouns and should start with uppercase (Eg. Day, DateofBirth)

→ Variable names should be nouns starting with lowercase (Eg name, amount).

→ Constant names should be ~~Verbs starting with uppercase~~ (Eg all uppercase (Eg PI, MAX_TEMP).

→ Method names should be Verbs starting with lowercase.

Files organization :-

There are conventions on how files should be named.

→ Java source files should have the extension .java

→ Each file should contain one outer class and the class name should be the same as the file name.

→ Line length should be limited to less than 80 columns and special characters should be avoided.

Statements :- These guidelines are for the declaration and executable statements in the source code.

→ Variables should be initialized where declared, and they should be declared in the smallest possible scope.

→ class variables should never be declared public

→ use only loop control statements in a for loops

→ Loops variables should be initialized immediately before the loop.

→ Avoid the use of break and continue in a loop.

→ Avoid use of do...while construct

Commenting and layout :-

→ Comments are textual statements that are meant for the program reader to aid the understanding of code

→ The purpose of the comments is not to explain in

Scanned by CamScanner

english The logic of the program.

→ providing comments for modules is most useful, as modules form the unit of testing, Compiling, Verification, and modification.

→ Java provides documentation comments That are delimited by "/**...*/", and which could be extracted to HTML files.

Some such Guidnes are:

→ Single line comments for a block of code should be aligned with the code They are meant for.

→ There should be comments for all major variables explaining what They represent.

→ A block of comments should be preceded by a blank Comment line with just "/*" and ended with a line containing just "*/"

Incrementally Developing Code :-

when modules are assigned to developers, They use some process for developing the code.

⊛ An Incremental Coding process

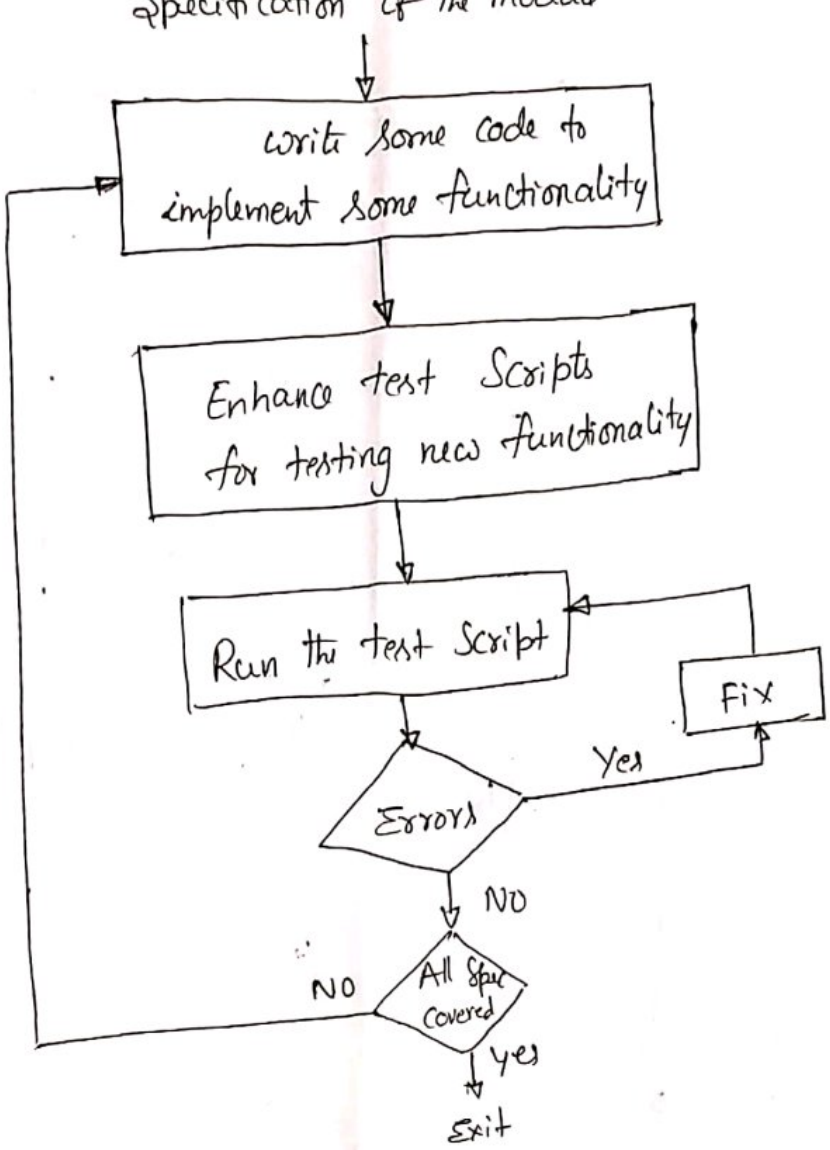
⊛ Test Driven Development. ⊛ pair programming

An Incremental Coding Process :-

⇒ The process followed by many developers is to write the code for the currently assigned module, and when done, perform Unit testing on it and fix the bugs found.

⇒ A better process for coding, which is often followed by experienced developers, is to develop the code incrementally.

Scanned by CamScanner



An Incremental Coding process

⇒ The Basic Advantage of developing code incrementally with testing being done after every round of coding is to facilitate debugging.

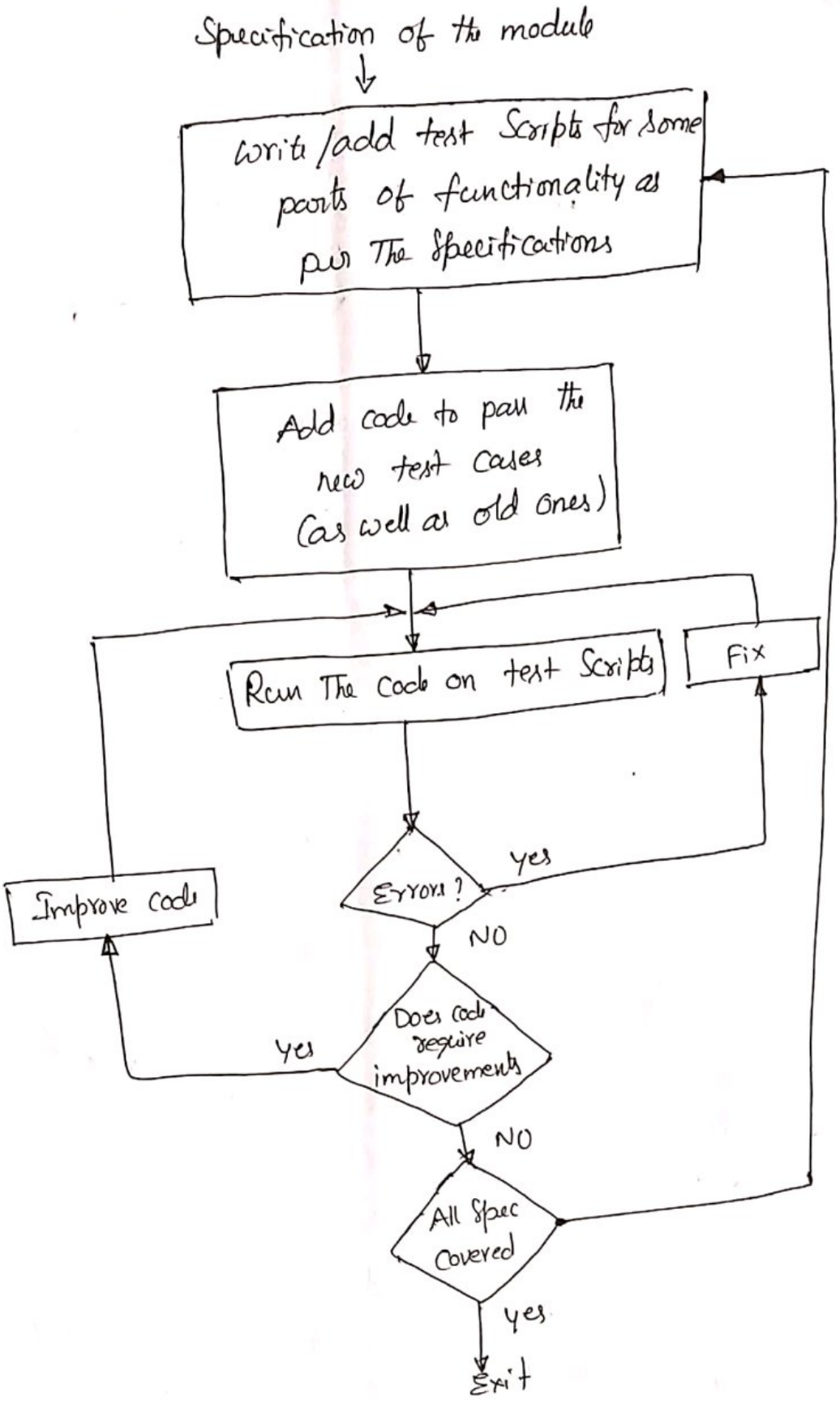
Test-Driven Development :-

→ Test-Driven Development (TDD) is a coding process that turns around the common approach to coding.

→ Instead of writing code and then developing test cases to check the code.

→ This writing of test cases before the code is written makes the development usage-driven.

Scanned by CamScanner



Test-driven development process.

Pair Programming :

⇒ Pair programming is also a coding process that has been proposed as a key technique in extreme programming (XP) methodology.

Scanned by CamScanner

=> In pair programming, code is not written by individual programmers but by a pair of programmers.

=> The basic motivation for pair programming is that as code reading and code reviews have been found to be very effective in detecting defects.

Managing Evolving Code

In such a dynamic scenario, managing evolving code is a challenge: Here we discuss two aspects of this
-> How to manage the different versions that get created.
-> How to maintain code quality under changes

* Source code Control and Build
* Refactoring

Source code Control and Build

=> In a project many different people develop source code. Each programmer creates different source files, which are eventually combined together to create executables.

=> A modern source code control system contains a repository which is essentially a controlled directory structure,

Some of the types of commands that are generally performed by a programmer are:-

* Get a local copy: A programmer in a project works on a local copy of the file. commands are provided to make a local copy from the repository.
  Exapl. cvs checkout <module>
* Make changes to file(s): The changes made to the local file by a programmer remain local until the changes are committed back on the repository.
  Ex: ~~cvs update command~~. by cvs commit < file >

Scanned by CamScanner

Update a local copy :-

Changes committed by project members to the repository are not reflected in the local copies that were made before the changes were committed

Eg:- by cvs update command.

Get reports : Source control tools provide a host of commands to provide different reports on the evolution of the files.

Refactoring : Refactoring is the technique to improve existing code and prevent this design decay with time.

=> Refactoring is part of coding in that it is performed during the coding activity.

=> Refactoring also plays an important role in test-driven development - code improvement step in the TDD process is really doing refactoring.

=> Refactoring is defined as a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.

=> Refactoring generally results in one or more of the following:

1. Reducing coupling
2. Increased Cohesion
3. Better adherence to open-closed principle.

Scanned by CamScanner

Testing

It is a process of find (or) detect the error (or) bug in programm (or) Software.

Testing is defined as check whether expected output and actual output both are same (or) not.

Testing Concepts

* Error
* Fault
* failure
* Test case
* Test Suite
* Test ware
* Test plan
* Test Strategy

Error: Difference of Expected output and actual output.

⇒ Normally arises in Software Error means to change the functionality of the program

Fault: A wrong or mistaken step, process or Data Definition in a computed program

Failure: It is the inability of a Software System or Component to perform its required functions

Test Case: It is a well designed document for develop a good software.

Test case design format

Tid	Input	Expected output	Actual output	Status

Tid → Test case id

Scanned by CamScanner

Test Suite : Collection of Test cases are called Test Suite.

Test wave :

Collection of test plans, test cases, test scripts and any other items needed to design and perform a test.

Test plan. A Test plan is a document describing Software testing Scope and activities. It is the basis for formally testing any Software/product in the project.

Test planning Activities :

→ To determine the Scope
→ Documenting test Strategy
→ Decide Entry and Exit Criteria
→ Evaluating the test estimate

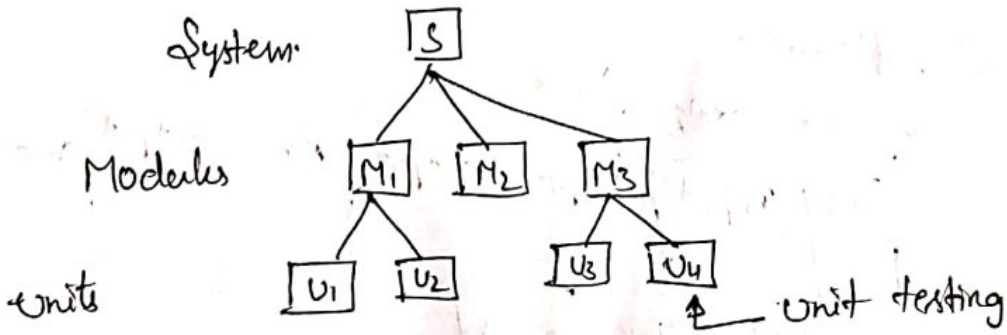
Test Strategy : It is a set of guidelines That explains test design and determines how testing needs to be done.

→ Test Strategies are * White box testing.
* Black box testing.

Levels of Testing

The Basic levels are Unit testing, integration testing, functional testing, System testing, Acceptance testing and Regression testing.

Unit testing : The Smallest Component in Software is called Unit. The first level of testing is called Unit testing



Scanned by CamScanner

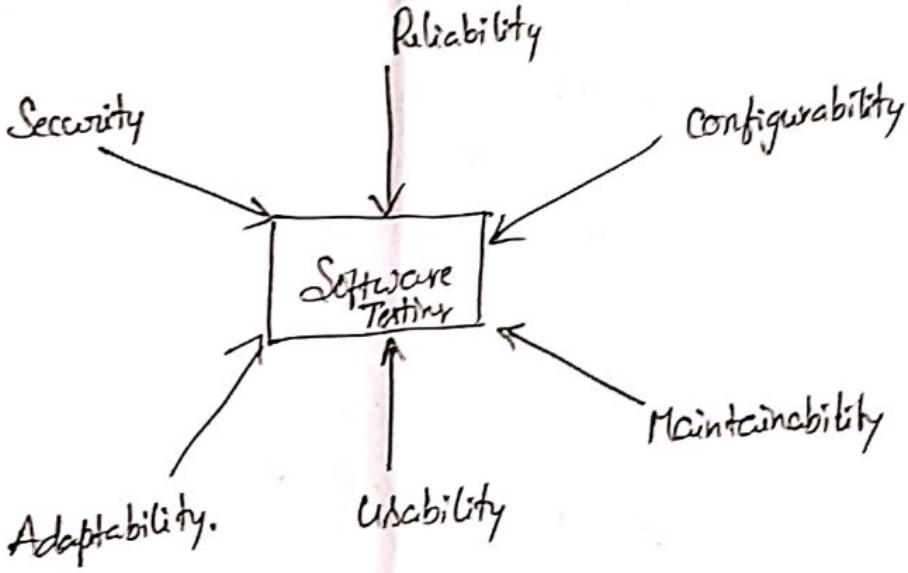
Integration Testing :

The next level of testing is often called integration testing. In this many unit tested modules are combined into subsystems.

System testing

Here the entire software system is tested.

The different categories (or) types of system testings are.



Acceptance Testing :

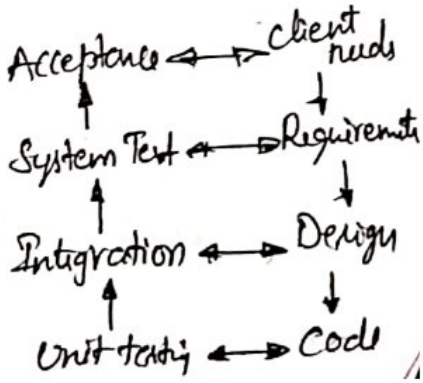
Acceptance testing is often performed with realistic data of the client to demonstrate that the s/w is working satisfactorily.

Acceptance Testing types → Alpha testing, Beta Testing.

Alpha testing : Testing done at developer site.

Beta testing : Testing done at user site.

Regression Testing : It is performed when some changes are made to an existing system.



Scanned by CamScanner

Testing process :

The Testing process for a project consists of ~~high~~ Three high-level tasks - test planning, test case design, and test execution

Test planning A test plan is a general document for the entire project that defines the scope, approach to be taken, and the schedule of testing, as well as identifies the test items for testing and the personnel responsible for the different activities of testing

Test plan Template :

* Test plan identifier
  → provide a unique identifier for the document.

Introduction :-

* provide an overview of test plan
* Specify the goals/objectives
* Specify any constraints

References :

* project plan
* Configuration Management plan

Test Case Design

(Refers the previous page in Testing Concepts)

Test Execution : It is the process of executing the code and comparing the expected and actual results

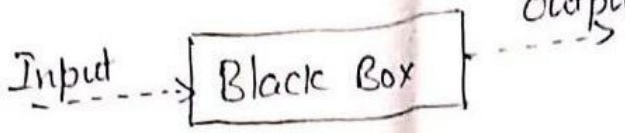
following factors are to be considered for test execution

* Based on risk, Select a test suite
* Executes tests, report bugs, and capture test statistics
* Resolve blocking issues as they arise
* Report test cycle findings and status

Scanned by CamScanner

Black box Testing

It is defined as a testing technique in which functionality of the Application Under test is tested without looking at the internal code structure.



Black box Testing techniques:

* Boundary Value Analysis
* Equivalence class Testing
* Decision Table Testing

Boundary Value Analysis:-

Boundary Value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the System or not.

⇒ It is very useful in reducing the number of test cases

⇒ Boundaries are

Max
Min
NV (Nominal value)

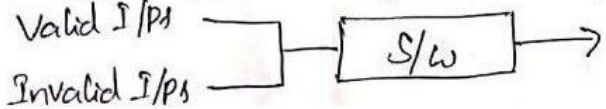
Max^+ min^+
Max^- min^-

for Example A = 1-10

Max - 10 $Max^+ - 11$ $Min^+ = 2$
Min - 1 $Max^- = 9$ $Min^- - 0$
NV = 5

Equivalence class Testing

It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.



Scanned by CamScanner

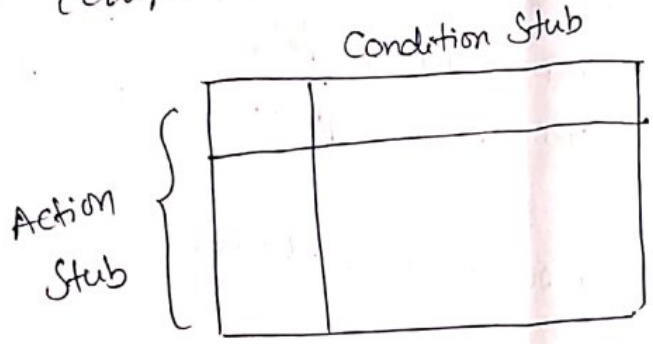
⇒ Equivalence class Testing depending upon

* Completness
* Redundancy
* performance.

Decision Table Testing

Decision table testing is a software testing technique used to test System behavior for different input combinations.

⇒ This is Systematic approach where the different Input Combinations and Their Corresponding System behavior (output) are captured in a tabular form.



Example

How to make Decision Base table for login Screen.

The Condition is Simple if the user provides Correct username and password The user will be redirected to The homepage., If any of The input is wrong, an error message will be displayed.

Condition	Rule 1	Rule 2	Rule 3	Rule 4
Username (T/F)	F	T	F	T
password (T/F)	F	F	T	T
output (E/H)	E	E	E	H

T - correct F - wrong E - Error H - Home Screen

Scanned by CamScanner

White box Testing

It is testing of a Software Solution's internal structure, design, and coding.

⇒ In this type of testing, code is visible to the tester.

⇒ It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability.

White box Testing Techniques

* Basic path Testing
  - ~~CFG~~ (cyclomatic Complexity)
  - CFG (Control flow Graph)
* Branch coverage
* Mutation

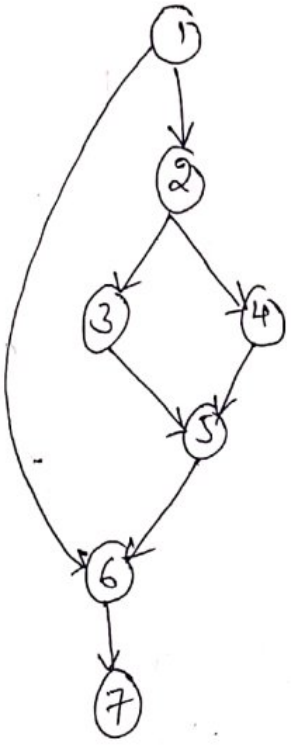
Basic path Testing

The basic/basis path testing is same, but it is based on a white box testing method, that defines test cases based on the flow or logical path that can be taken through the program.

Example

```
if A = 50
Then if B > C
 Then A = B
 else A = C
 endif
 endif
 print A
```

CFG



Cyclomatic Complexity (C) = E - N + 2

⇒ 8 - 7 + 2 ⇒ 1 + 2 = 3

Scanned by CamScanner

⇒ Draw a control flow graph

⇒ Calculate cyclomatic complexity

* It is a metrics to determine the no. of independent path.

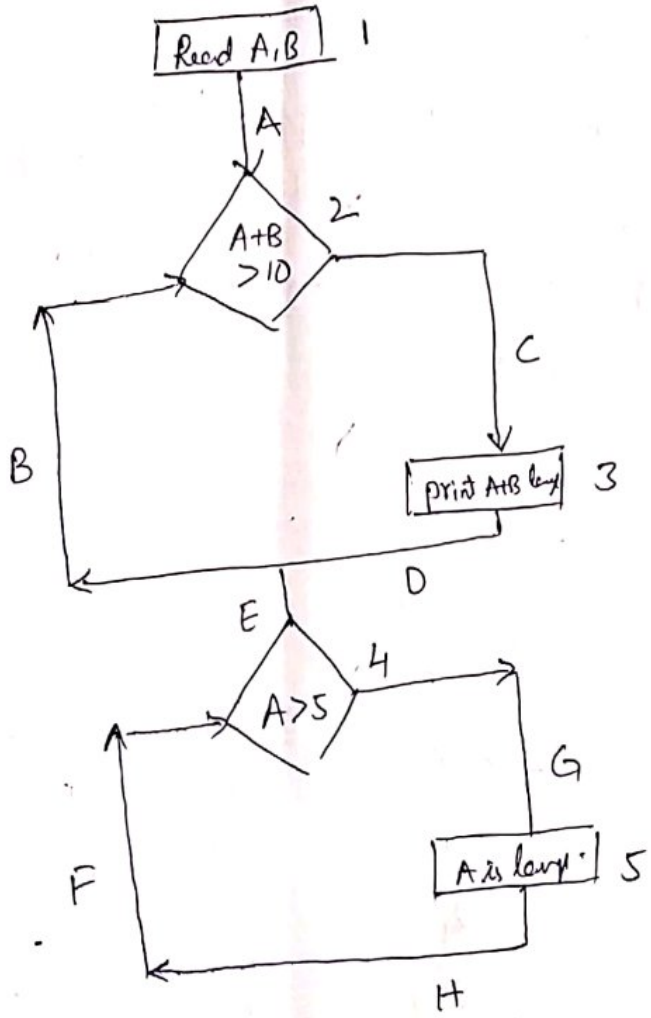
⇒ find a basic set of paths

⇒ Generate test cases to exercise each path

Branch Coverage

Branch coverage (or) Decision coverage is a testing method, which aims to ensure that each one of the possible branch from each decision point is executed.

for example



Branch coverage

1) 1A-2C-3D-E-4G-5H

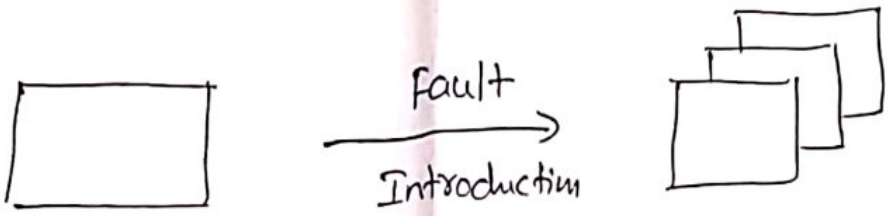
2) 1A-2B-E-4F

Branch coverage is 2

Scanned by CamScanner

Mutation

Mutation Testing is a type of Software testing where we mutate (Change) certain statements in the Source code and check if the test cases are able to find the errors



Original program mutant program.

Example

Original program

```
if (x>4)
print "Hello"
else
  print "Hi"
```

Mutant program

```
if (x<y)
  print "Hello"
  else
  print "Hi"
```

Types of Mutations

1) primary mutation
2) Secondary Mutation.

Example

```
if (a>b)
  c = a-b;
else
  c = b-a;
```

primary mutation

```
c = a+b; // primary
   (or)
c = b/a ; //primary
```

Secondary Mutation

```
if (a=b) // Secondary
```

Scanned by CamScanner

Risk Management

Reactive Vs Proactive Risk Strategies

Reactive: A response based risk management approach, which is dependent on accident evaluation

⇒ Depends on past accidental analysis and response.

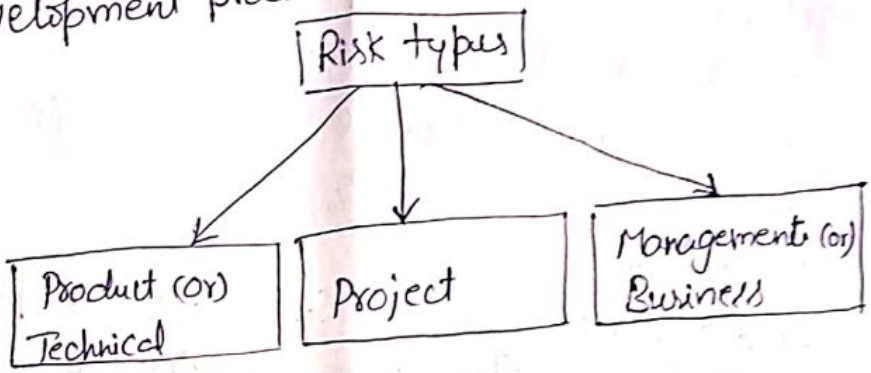
Proactive: Observation of the present safety level and planned explicit target safety level with a creative intellectually

⇒ It combines a mixed method of past, present and future prediction before finding solutions to avoid risks

Risk: It is an expectation of loss, a potential problem that may or may not occur in the future.

⇒ It is generally caused due to lack of information, control or time.

⇒ A possibility of suffering from loss in Software development process is called a Software risk.



Software Risk:

Risk always involves two characteristics.

Uncertainty: The risk may or may not happen: That is, There are no 100% probable risks.

Loss: If the risk becomes a reality, unwanted consequences or losses will occur.

Scanned by CamScanner

When risks are analyzed, it is important to quantify the level of uncertainty in the degree of loss associated with each risk.

Different Categories of risks are considered

* Project risk
* Technical risk
* Business risk

Project risks :- If project risks become real, it is likely that project schedule will slip and that costs will increase.

Technical risks :- If a technical risk becomes a reality, implementation may become difficult or impossible.
Technical risks identify potential design, implementation, interface, Verification, and maintenance problem.

Business risks :-
These are monetary risk which are associated with budget overruns.
→ Improper Budget Estimation
→ Cost overruns due to underutilization of resources
→ Expansion of project scope.

Risk Identification

Risk Identification is a Systematic attempt to specify threats to the project plan. There are two distinct types of risks
1. Generic risks and
2. product-specific risks.

Generic risks :- These are potential threat to every software project.

Product-specific risks :- It can be identified only by those with a clear understanding of the technology.

Scanned by CamScanner

1) Establish a scale that reflects the perceived likelihood of a risk
2) Delineate the consequences of the risk
3) Estimate the impact of the risk on the project and the product and
4) Note the overall accuracy of the risk projection so that there will be no misunderstandings.

Developing a Risk table

Risk	Probability	Impact	RMMM
			Risk Mitigation Monitoring & Management

⇒ A project team begin by listing all risks in the first column of the table.
⇒ Each risk is categorized in Next, the impact of each risk is assessed.
⇒ The categories for each of the four risk components - performance Support, cost and schedule. are averaged to determine an overall impact value.

Assessing Risk Impact

Three factors affects the consequences that are likely if a risk does occur: its nature, its scope, and its timing.

→ The Nature of the risk indicates the problem that are likely if it occurs
→ The Scope of a risk combines the severity with its overall distribution.

Scanned by CamScanner

Generic Subcategories :

* product size
* Business impact
* Customer characteristics
* Development environment
* Technology to be built
* Staff size and experience.

Risk Components and Drivers

The risk components are defined in the following manner:

* performance risk
* Cost risk
* Support risk
* Schedule risk

Performance risk : The Degree of uncertainty That the product will meet it requirements and be fit for its intended use.

Cost risk : The Degree of uncertainty That the project budget will be maintained.

Support risk : The degree of uncertainty that the resultant software will be easy to correct, adapt and enhance.

Schedule risk : The Degree of uncertainty That the project schedule will be maintained and That the product will be delivered on time.

Risk Projection :

It is also called risk estimation, attempts to rate each risk in two ways - The likelihood or probability That the risk is real and the consequences of the problems associated with the risk, should it occur.

The project planner, along with other managers and technical staff, performs four risk projection activities.

Scanned by CamScanner

RISK MITIGATION MONITORING AND MANAGEMENT

An effective strategy must consider three issues:

* Risk avoidance
* Risk monitoring
* Risk Management and Contingency planning

If a Software team adopts a proactive approach to risk, avoidance is always the best strategy. To mitigate this risk, project management must develop a strategy for reducing turnover. Among the possible steps

* Meet with current staff to determine causes for turnover
* Mitigate those cause that are under our control before the project starts.
* Organize project teams so that information about each development activity is widely dispersed

6) The RMMM PLAN

A risk management strategy can be included in the software project plan or the risk management steps can be organize into a separate risk mitigation, Monitoring and management plan.

The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.

Scanned by CamScanner

→ Finally, The timing of a risk considers : when and for how long The impact will be felt.

The overall risk exposure

$$RE = P \times C$$

P – probability of occurence for a risk

C - cost to the project should the risk occur.

Risk Identification : only 70 percent of the Software Components scheduled for reuse will, in fact, be integrated into the application

Risk probability - 80% (likely)

Risk Impact - 60% reusable s/w Components were planned.

Risk exposure $RE = 0.80 \times 25,200 - \$20,200$

Risk Refinement

One way of risk refinement is to ~~forepare~~ represent the risk in condition-transition- consequence (CTC) format.

This general Condition can be refined in the following manner.

Sub Condition-1

Certain reusable Components were developed by a Third party with no knowledge of internal design standards.

Sub Condition-2 : The Design standard for Component interfaces has not been solidified and may not conform to Certain existing reusable components.

Sub Condition-3 : Certain reusable Components have been implemented in language that is not supported on the target environment

Scanned by CamScanner

UNIT-V

Software Project Estimation

⇒ Both before and during a project we need to be able to estimate the resources (people and time) that the project will consume in the future.

⇒ These estimates are used to allocate resources and budget and to plan the schedule for the project.

⇒ Estimates are typically based on historical data (previous projects) and on various rules and heuristics to determine the size and scope of a project from very incomplete information.

⇒ An organization must expand the effort to record accurate cost and time information about its projects if it is going to do serious estimation for future projects.

Estimation Techniques :

Estimation Techniques are two types

1. Decomposition
2. Empirical Models

Decomposition : Divide project into more manageable parts and estimate each part, then sum.

Empirical Models : use historical data from similar projects to estimate project based on a few simple parameters like lines of code or number of modules

Estimation It is an attempt to determine how much money, effort, resources and time it will take to build a specific S/W based System or project

Decomposition Techniques

LOC and FP data are used in two ways during Software project estimation. LOC and FP estimation are distinct estimation techniques

Lines Of Code (LOC)

As the name suggest, LOC count the total number of lines of source code in a project.

The units of LOC are:

1) KLOC - Thousand lines of code
2) NLOC - Non Comment lines of code
3) KDSI - Thousand of delivered Source Instructions

what is a LOC

Declarations, Actual code including logic and Computation

What is NOT LOC?

* Blank lines
* Comments

Blank lines : Included to improve readability of code.

Comments : Included to help in code understanding as well as during maintainance.

⇒ Not include Blank lines & comments, because Do not

contribute any kind of functionality.

⇒ The size is estimated by comparing it with the existing systems of same kind.

⇒ The experts use it to predict the required size of various components of software and then add them to get the total size.

Example

⇒ The organization average productivity for system of this type is 620 LOC/pm

⇒ Based on a burdun labor rate of $8000 per month.

⇒ The cost per line of code is approximately - $13. (8000/620)

Advantages

* Universally Accepted and is used in many models like COCOMO.
* Estimation is closer to developer's perspective.
* Simple to use

Disadvantage

* Different programming languages contains different number of lines.
* No proper industry standard exist for this technique
* It is difficult to estimate the size using this technique in early stages of project.

2) Functional point Analysis (FP) :-

Decomposition of FP-based estimation focuses on

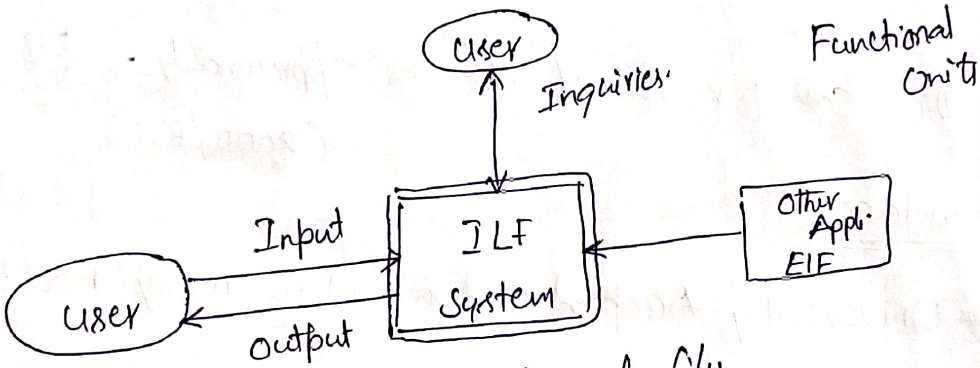
information domain values rather than software ...

⇒ The project planner estimates inputs, outputs, inquiries, files, and external interfaces for a software.

⇒ They are quickly accepted as an industry standard for functional sizing.

⇒ for sizing software based on FP, several recognized standards and/or public specifications have come into existence.

⇒ focuses on what functionality is being delivered



ILF → Internal logical files

EIF → External Interface files

⇒ A System has 5 types of functional units.

1) Internal logical files (ILF) : control info. (or) logical data present within the system

2) External Interface files (EIF) : control data (or) logical data present in other system.

3) External Inputs (EI) :
Data/control info. that comes from outside the system

4) External outputs (EO)
Data that goes out of the system after generation.

5) External Enquires (EQ) :
Combination of I/P - O/P resulting data retrieval

Step-1

=> Each function point is ranked according to complexity There exists pre-defined weights for each F-P in each category.

Functional Units ↓i	j → weighting factors		
	low	Average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	15

How to assign weights (or) rank function points?

→ Dependent on the organization

→ Based on past projects.

Step-2

Calculate unadjustment function point. by multiplying each FP by its corresponding weight factor.

$$UFFP = \sum_{i=1}^{5} \sum_{j=1}^{3} (Z_{ij} * W_{ij})$$

Z_{ij} → Count of no. of functional units of category i classified as complexity.

W_{ij} → weight from above table.

Step-3

Calculate final function points

$$\text{Final FP} = UFP \times CAF$$

CAF → Complexity Adjustment factor

Calculated using 14 aspects of processing complexity.

* 14 questions answered on a scale of 0 to 5.

0 → No Influence

1 → Incidental

2 - Moderate

3 — Average

4 — Significant

5 — Essential.

$$CAF = [0.65 + 0.01 * \Sigma Fi]$$

$$\boxed{FP = UFP \times CAF}$$

Example Given the following values, compute F.P when all complexity adjustment factors and weighting factors are average.

User I/p = 50

User o/p = 40

User Enquiries = 35

User Files = 6

External Interfaces = 4

$$\sum_{i=1}^{14} fi = 14 \times 3 \text{ (Average)}$$

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} Zig \cdot Wig$$

$$UFP = 50 \times 4 + 40 \times 5 + 35 \times 4 + 6 \times 10 + 4 \times 7$$

$$= 200 + 200 + 140 + 60 + 28$$

$$= 628$$

$$CAF = [0.65 + (0.01 \times 42)] \Rightarrow 0.65 + 0.42 \Rightarrow 1.07$$

$$\boxed{FP = UFP \times CAF} \Rightarrow 628 \times 1.07 = 672 \text{ (Approximately)}$$

Advantages

=> Size of software delivered is measured independent langugs and technology & tools

=> FP are directly estimated from requirements, before design & coding.

=> It provides a reliable relationship to effort.

Disadvantages

=> It needs evaluations with a lot of judgment involved

=> Less research data is available on function points as compared to LOC.

=> It is a time consuming method.

COCOMO

The COCOMO model developed by Boehm's.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required.

=> COCOMO consists of hierarchy of Three increasingly detai-led and accurate forms.

These are types of COCOMO model.

1. Basic COCOMO model
2. Intermediate COCOMO model
3. Detailed COCOMO model.

=> The first level, Basic COCOMO can be used for quick and slightly rough calculations of software costs. It is Accuracy is somewhat restricted due to the absence of sufficient factor considerations.

Softwares are 3 modes.

* organic
* Semi Detached
* Embedded

	organic	Semi-detached	Embedded
Size	2-50 KLOC	50-300 KLOC	300 & Above KLOC
Team Size	Small Size	medium Size	large Size
Dead line	Not tight	medium	Tight

Estimation of Effort :-

$$E = a(KLOC)^b$$

The above formula is used for the cost estimation of for the basic cocomo model. It is also called Subsequent model. The constant values a and b for the basic model.

Software projects	A	B
organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

The effort is measured in person-Months and as evident from the formula is dependent on Kilo-Lines of codes.

Intermediate model :-

Based on 15 different parameter - Calculate the Effort Adjustment factor (EAF)

$$E(Effort) = (a(KLOC)^b) * EAF$$

values of a and b in case of the intermediate model are

Software project	A	B
organic	3.2	1.05
Semi Detached	3.0	1.12
Embeddded	2.8	1.20

3. Detailed Model :-

Detailed cocomo incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process.

The six phases of detailed cocomo are.

1. planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. cost Constructive model.

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the Software lifecycle.

Empirical Estimation Techniques

Empirical Estimation Technique are based on the data taken from previous projects and some based on guesses and assumption.

There are many empirical estimation Techniques but most popular are.

* Expert Judgement Techniques
* Delphi cost estimation.

Expert Judgement Techniques

=> An expert makes an educatid guess of the problem size after analyzing the problem Throughly

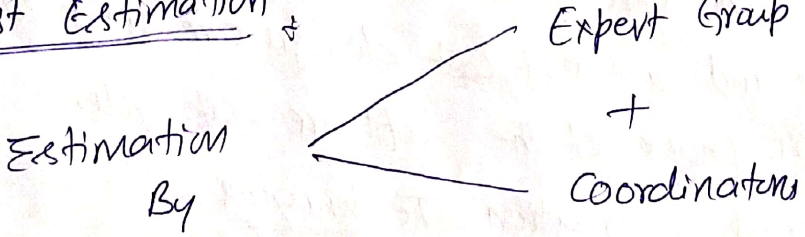
=> Expert estimates the cost of different Components i.e modules and submodules of the System.

Disadvantages

→ Human error
→ Considering Not all factors and aspects of the project
→ Individual bias
→ More chance of failure.

Estimation by group of experts minimizes factors such as individual oversight, lack of familarity with a particular aspect of a project, personal bias, and the desire to win Contract Through overly optimistic Estimate.

Delphi cost Estimation :

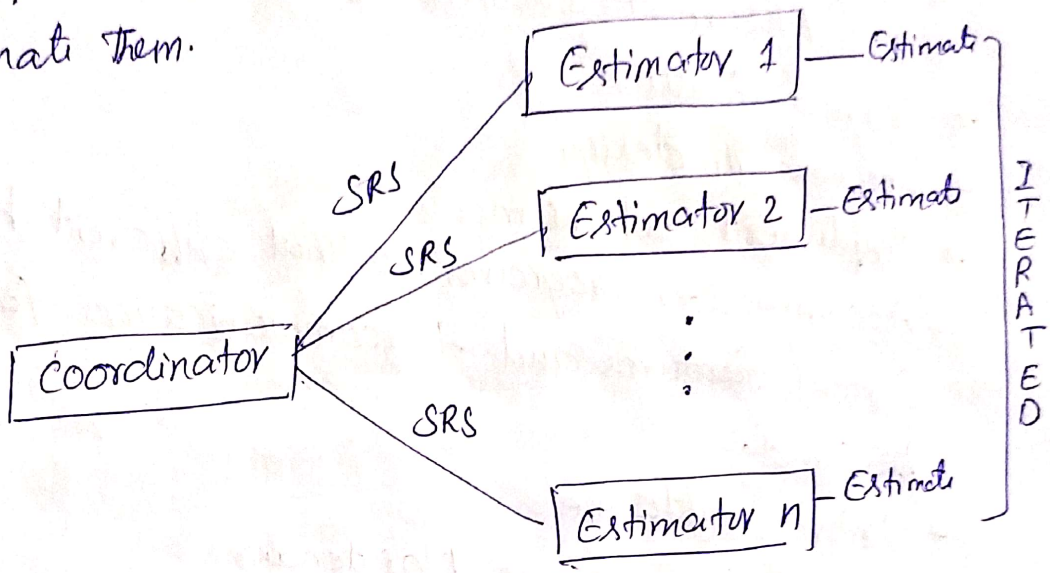


Role of members :

Coordinator provide a copy of software Requirement Specification (SRS) document and a form of Recording his cost estimate to each estimator.

Estimators complete this individual estimate anonymously and submit to the coordinator with mentoring, if any, crucial characteristic of product which has influenced his estimation.

The coordinator prepare and distributes the Summary of the responses to all estimators and they Re-estimate them.



This process is iterated for several rounds.

No discussion is allowed among the estimators during the entire estimation process because then may be many estimators get easily influenced by the rationale of an estimator who may be more experienced

After the completion of several iterations of esti-mations, the coordinator takes the responsibility of com-piling the results and preparing the final estimate

Software Maintenance

Software Maintenance is the process of modifying a Software product after it has been delivered to the customer. The main purpose of Software maintainance is to modify and updat Software application after delivery to correct faults and to improve performance.

Need for Maintenance :

Software Maintainance must be performed in order

* Correct faults
* Improve the design
* Implement enhancements.
* Accommodate programs so that different hardware, S/w System features and telecommunications facilities can be used.
* Retire Software

Categories of Software Maintenance

Maintenance can be divided into the following.

1. Corrective Maintenance
2. Adaptive Maintenance
3. purfective Maintenance
4. preventive Maintenance.

Corrective Maintenance :

Corrective maintenance of a Software product may be essential either to rectify some bugs observed while the System is (or) enhance the performance of the System.

Adaptive Maintenance :

This includes modifications and updations when the customers need the product to run on new platforms, on new operating Systems or new hardware

Perfective Maintenance :-

A Software product needs maintenance to support the new features that the users want or to change different types of functionalities of the System according to the customer demands.

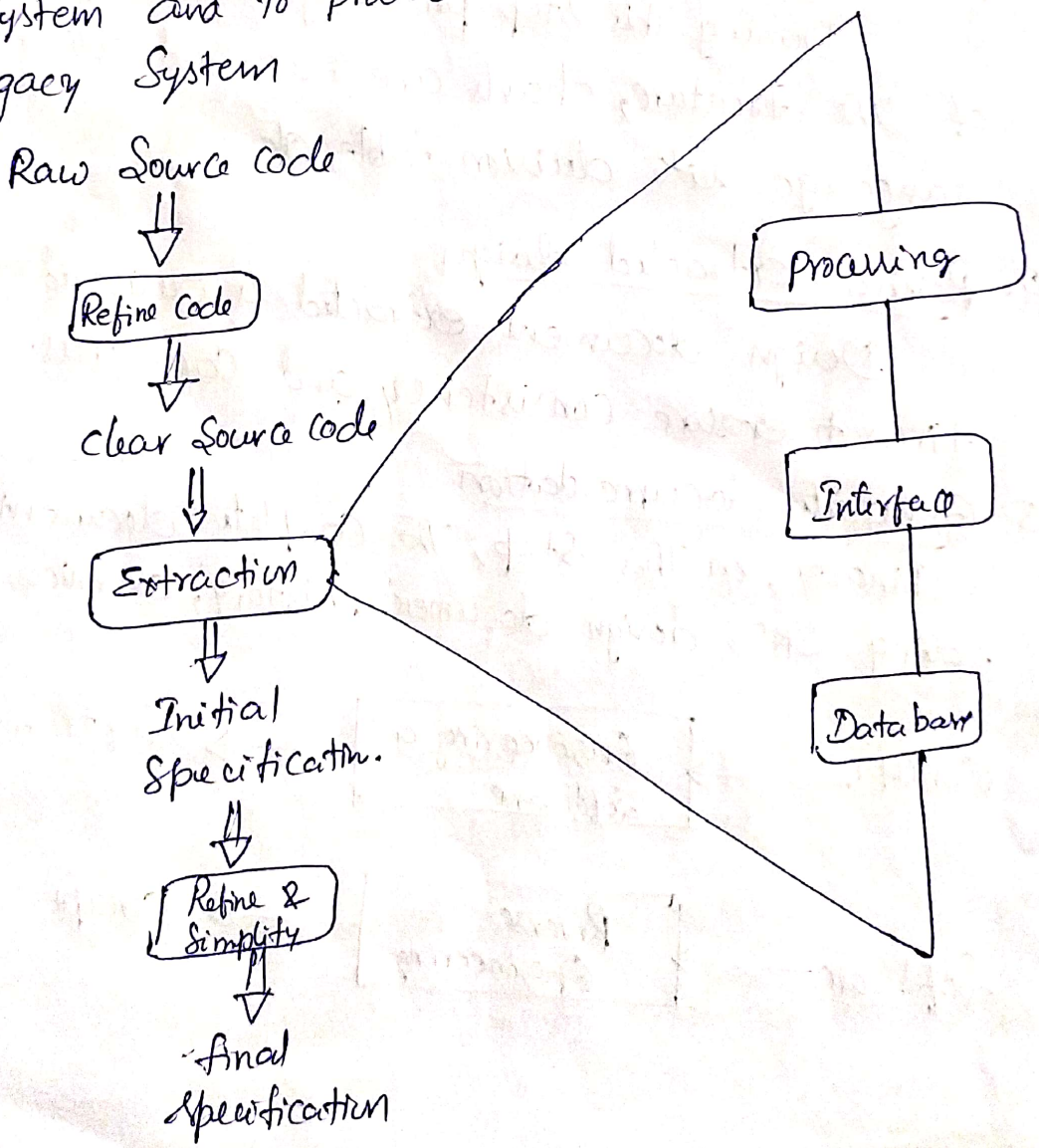
Preventive Maintenance :-

This type of maintenance includes modifications and updations to prevent future problems of the Software.

Reverse Engineering

It is a process of recovering the design, requirement specifications and functions of a product from an analysis of its code.

The purpose of reverse engineering is to facilitate the maintenance work by improving the understandability of a System and to produce the necessary documents for a legacy System



Reverse engineering Goals

* Cope with Complexity
* Recover lost information
* Detect Side effects
* facilitate Reuse
* Synthesize higher abstraction.

Steps of Software Reverse Engineering

1. Collection Information

This Step focuses on collecting all possible information (i.e Source design documents etc.). about the Software.

2. Examining the information

The information collected in Step-1 are studied so as to get familiar with the System.

3. Recording the functionality :

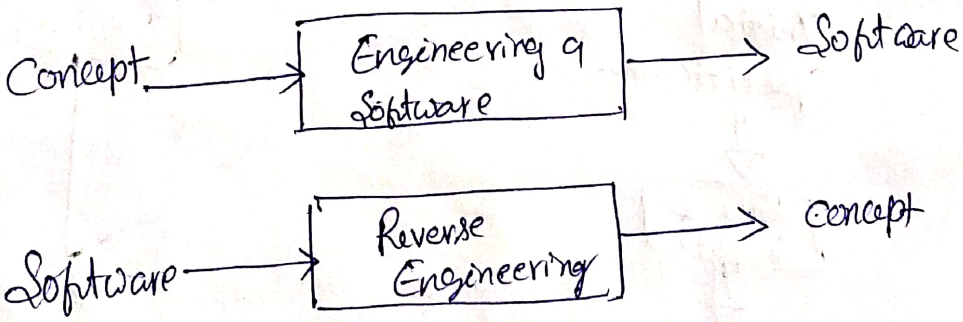
During this step processing details of each module of the Structure, charts are recorded using structured language like decision table, etc.

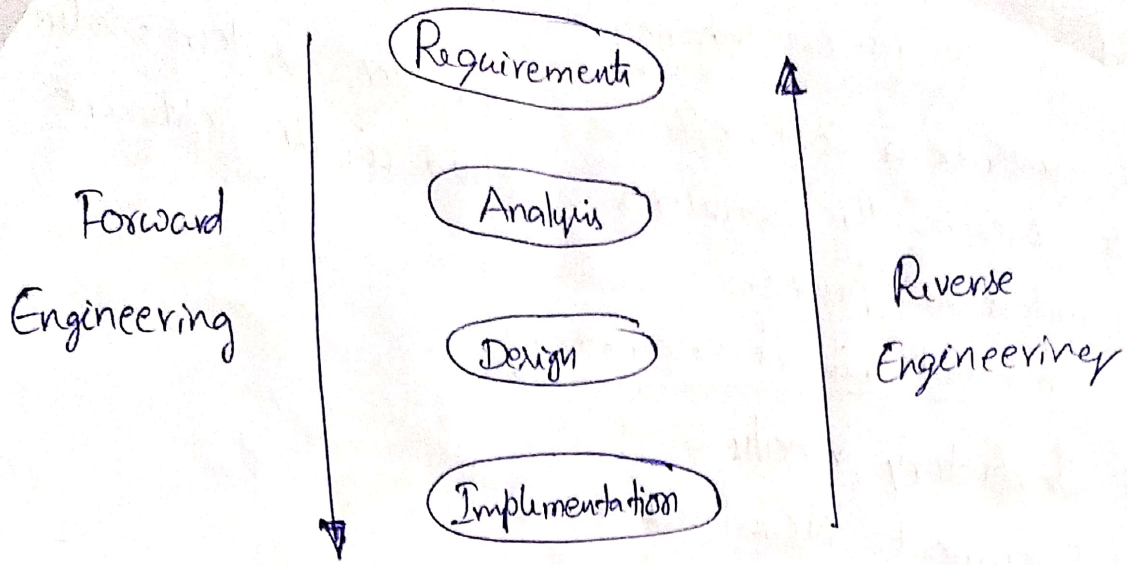
4. Review extracted design

Design document extracted is reviewed several times to ensure consistency and correctness.

5. Generate documentation

Finally, in this Step, the Complete documentation including SRS, design document, history, overview, etc.





Software Engineering

Reverse Engineering Tools

* System Monitoring tools
* Disassemblers
* Debuggers.

System Monitoring tools:

Network activity, file access and register access

Disassemblers

Translate binary code to assembly code

Debuggers: It is used in disassembling mode to set. break points and step Through program execution.

Reengineering

Software Re-engineering is a process of software development which is done to improve the maintainability of a Software System

=> Re-engineering is the examination and alteration of a system to reconstitute it in a new form.

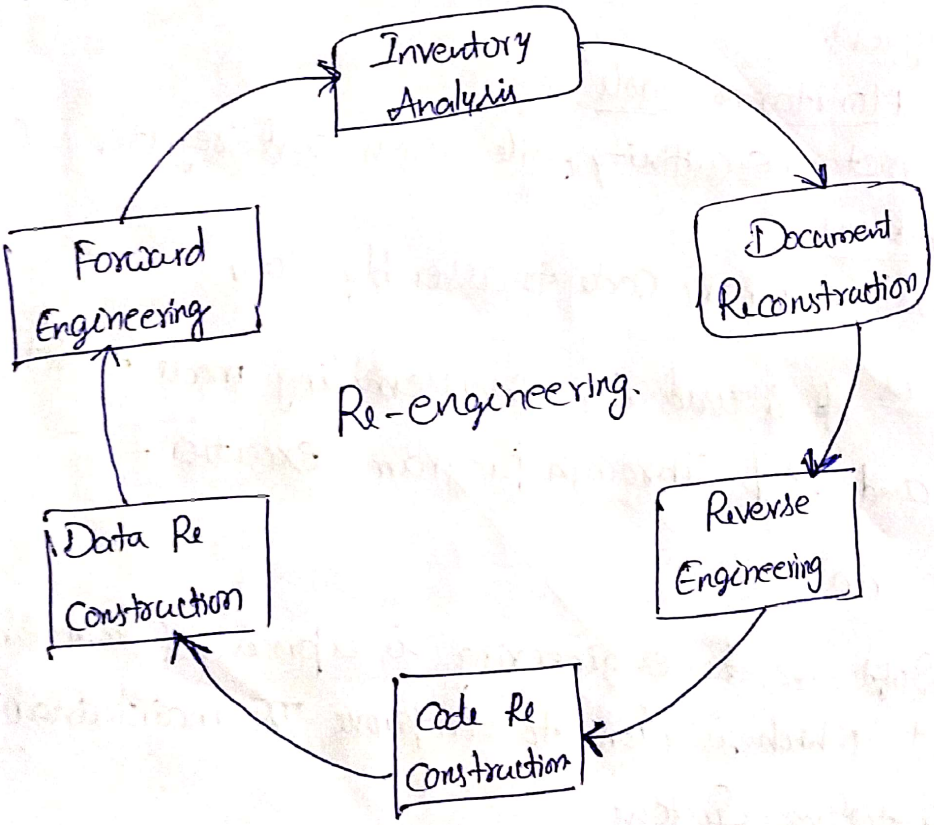
=> This process encompasses a combination of sub-process like reverse engeneering, forward engeneering, reconstructing etc.

Objectives of Re-engineering

* To describe a cost-effective option for System evolution
* To describe the activities involved in the Software maintenance process.

Steps involved in Re-engineering :

1. Inventory Analysis
2. Document Reconstruction
3. Reverse Engineering
4. Code Reconstruction
5. Data Reconstruction
6. Forward Engineering



Advantages

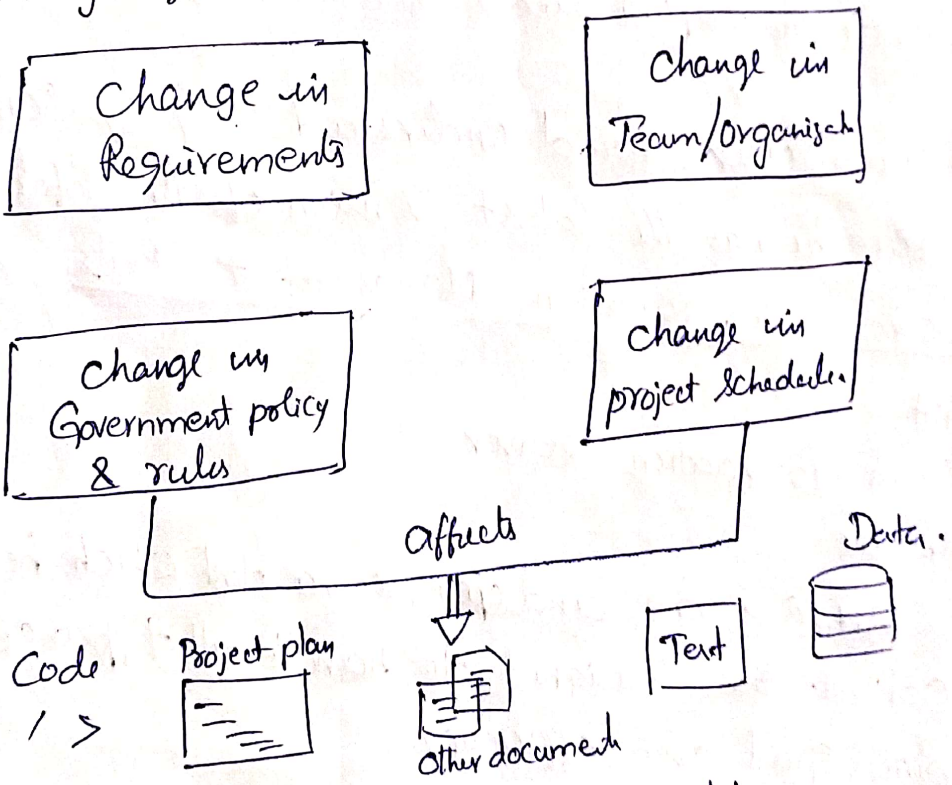
1. Reduce Risk : As the Software is already existing, the risk is less as compared to new Software development.
2. Reduce cost : The cost of re-engineering is less than the costs of developing new Software.

Configuration Management

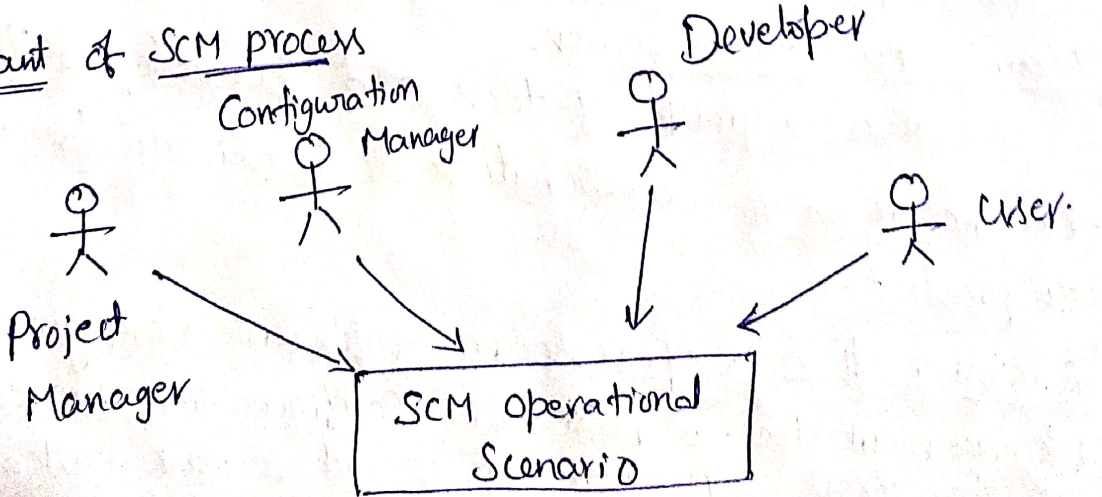
Software Configuration Management is defined as a process to systematically manage, organize, and control the changes in the documents, codes and other entities during the Software Development Life cycle.

Need of Configuration Management

* There are multiple people working on software which is continually updating.
* Changes in user requirement, policy, budget, schedule need to be accommodated.
* Software should able to run on various machines and operating systems.



Participant of SCM process



Configuration Manager:

→ Configuration Manager is the head who is Responsible for identifying configuration items.

→ He/she needs to approve or reject change requests.

Developer

→ The Developer should check the changes and resolves conflicts.

Auditor

→ The Auditor is responsible for SCM audits and reviews

→ Need to ensure the consistency and Completeness of release

Project Manager

→ Ensure That the product is developed within a certain time frame.

→ Generates reports about the status of the Software System.

User

→ The end user should understand they key SCM terms to ensure he has the latest version of the Software

Software Configuration Management Tools:

1. Git
2. Team foundation Server
3. Ansible

Git: Git is a free and open source tool which helps version control. It is designed to handle all types of projects with speed and efficiency.

Team foundation Server:

It is a group of tools and Technologies That enable the team to collaborate and coordinate for building a product.

Ansible: It is a open source Software Configuration Management tool. Apart from Configuration management it also offers application deployment & task automation