s/w quality

- The degree to which a system meets specification and ….
- Two approach:
- 1. defect management approach.
- 2. quality attributes approach.

Defect management approach

- Define defect …..
- This approach is based on counting and managing defects.
- Number of defects is counted and action is taken as per the severity.
- Defect leakage matrices tools

Quality attributes approach

- Quality is maintained.
- Characteristics :

1. 1.functionality : set of functions.
   suitability , accurateness , interoperability , compliance , security.
2. Relaibility: capability of s/w under certain condition
   maturity , recoverability
3. Usability : ease and simple use of function. understandability , learn ability
4. Efficiency : good architecture and coding practices
5. Maintainability : ability to find and fix faults. analyzability , changeability , stability
6. Portability : ability to adopt changes in its environment.
   adaptability , installability,

s/w testing

- Define : a process of executing a program or s/w with the intent of finding errors , validating against requirements.
- Is a process of validating and verifying.
- Case studies :

1. disney's lion king
2. Y2K bug

Goal of s/w testing

- Find bug
- Find bug as early as possible
- Make sure that bug is get fixed.
- Classification:

1. immediate goals
2. Long term goals
3. Post implementation goal

1. Immediate goals : Bug discovery.

a. Bug prevention: consequent action of bug discovery.

2. Long term goals: affect the s/w quality

a. Reliability
b. Risk management?

3. Post implementation goals:

a. Reduce maintenance cost
b. Improved s/w testing process.

Testing terminology

1. Failure : s/w unable to perform function according to the specification.
2 . Error : mistakes made by programmer.
3. Defects : errors in coding or logic
4. bug : logical mistake.

- Bug occur when one or more of the following five rules is true :
  1. s/w doesn't do….but product specification says.
  2. s/w does ….but product spec says it should do.
  3. s/w does….but product spec doesn't mention.
  4. s/w doesn't do ….but product spec doesn't mention but it should do.
  5. s/w is difficult to understand .

Role of tester

- Test lead / manager
- Test architect : leverages the available testing infrastructure
- Test designer
- Test automation engineer
- Test methodologist

What is test cases

- A set of values ,execution precondition and expected for a particular objects.
- Objectives
- verify with rqmts
- Improve quality
- Avoid post deployment risks

Test to pass approach:

- Don't push capabilities
- Straight forward test cases
- Always run test to pass cases first

Test to Fail approach:

- Push capabilities
- Break the s/w

When to start testing

- Entry criteria: minimum set of conditions that should be meet in order to start testing.
- All documents , design , rqmts available
- s/w tools available
- All personnel must be trained
- Exit criteria: minimum set of conditions in order to close a particular project phase.
- Completion of all test case execution
- Completion of code coverage
- No higher priority or severe bugs left

Skills of tester

- They r explorer: like to get a new piece of s/w
- They r trouble shooter
- They r relentless
- They r creative
- They r perfectionists
- They exercise good judgment
- They r persuasive: good in making point of view clear

Quality assurance

- Focus on process used to make product
- It is proactive quality process
- Goal : .....
- Establish a good quality mgmt system
- Verification is an example of QA
- QA is managerial tool

Quality Control

- Ensure quality in product. Focus on finding defects in actual product
- Reactive quality process
- Goal : Identify defects after product is developed but before released.
- Finding and eliminating sources of quality problems through tools
- Validation is an example of QC
- QC is corrective tool

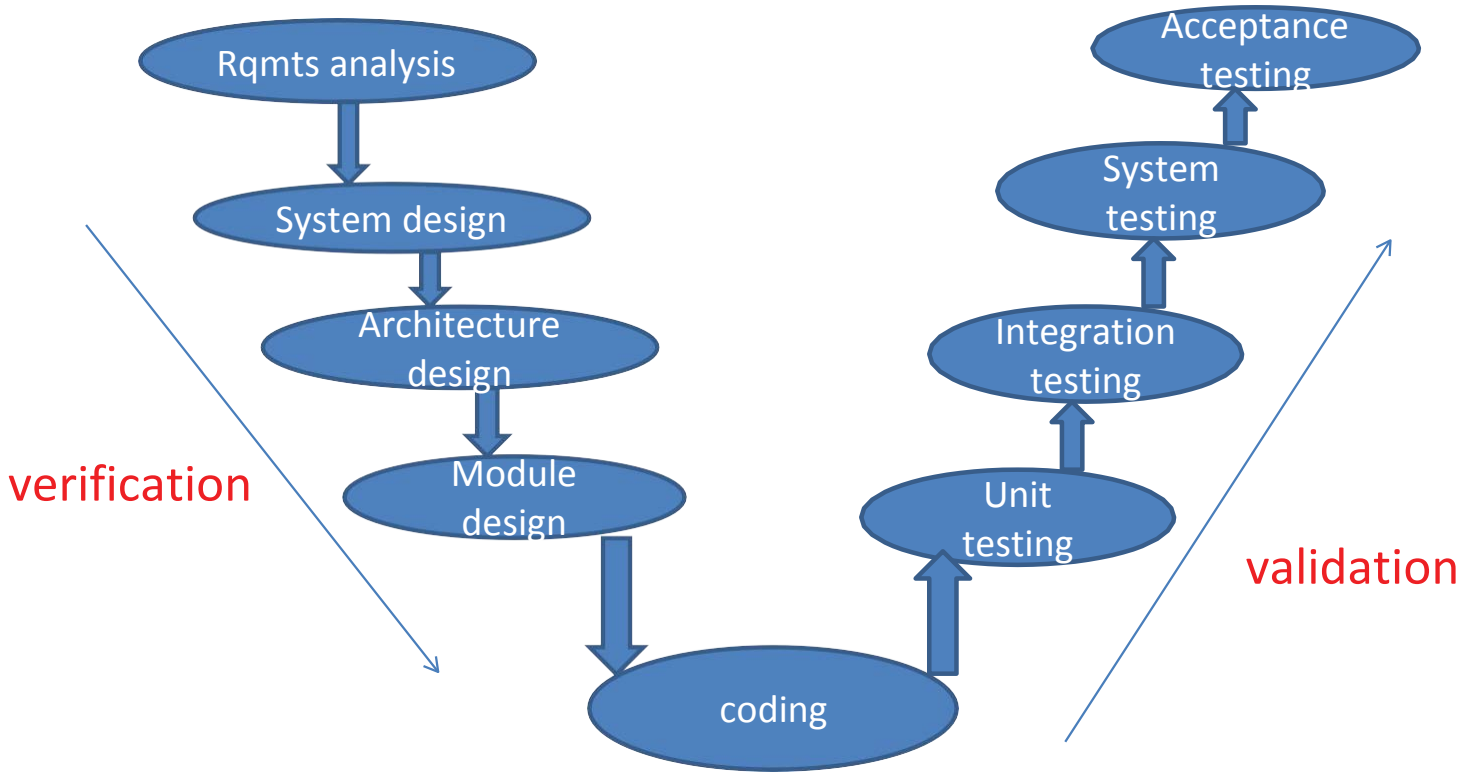
Verification

- Are we building the product right ?
- It doesn't involve executing the code
- Uses methods like : reviews walkthrough , inspection
- Check whether the s/w conforms to specification
- It is low level exercise.
- Done by QA team

Validation

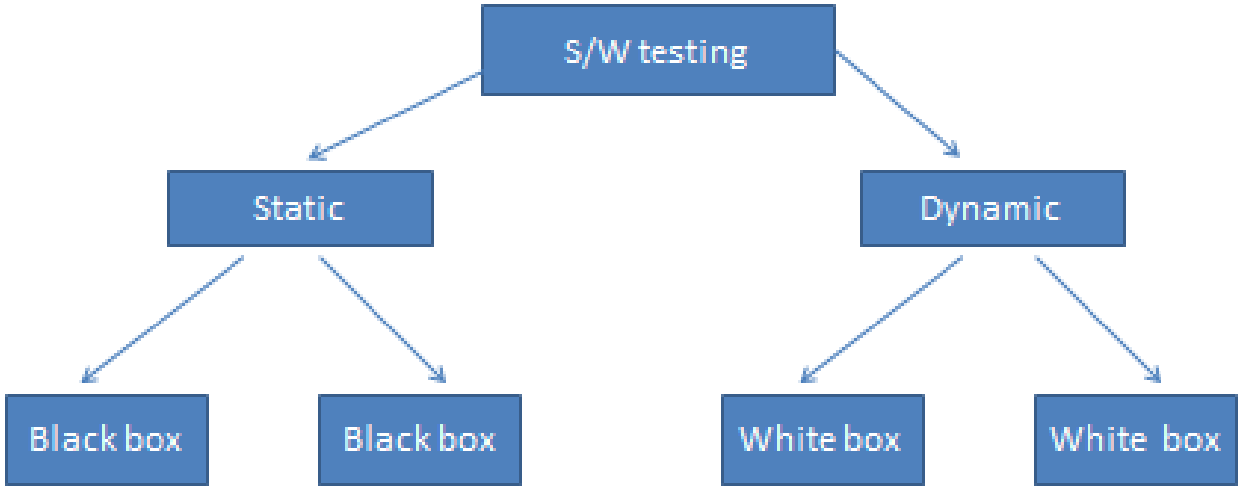
- Are we building the right product?
- It involve executing the code
- Uses methods like : black box , white box testing
- Check whether s/w meets customer rqmts.
- It Is high level exercise
- Done by QC team

V Model



- Also known as verification validation model.
- Is an extension of the waterfall model and is based on association of testing phase for each corresponding development stage.

Chapter 2 : Types of Testing



➢ Static

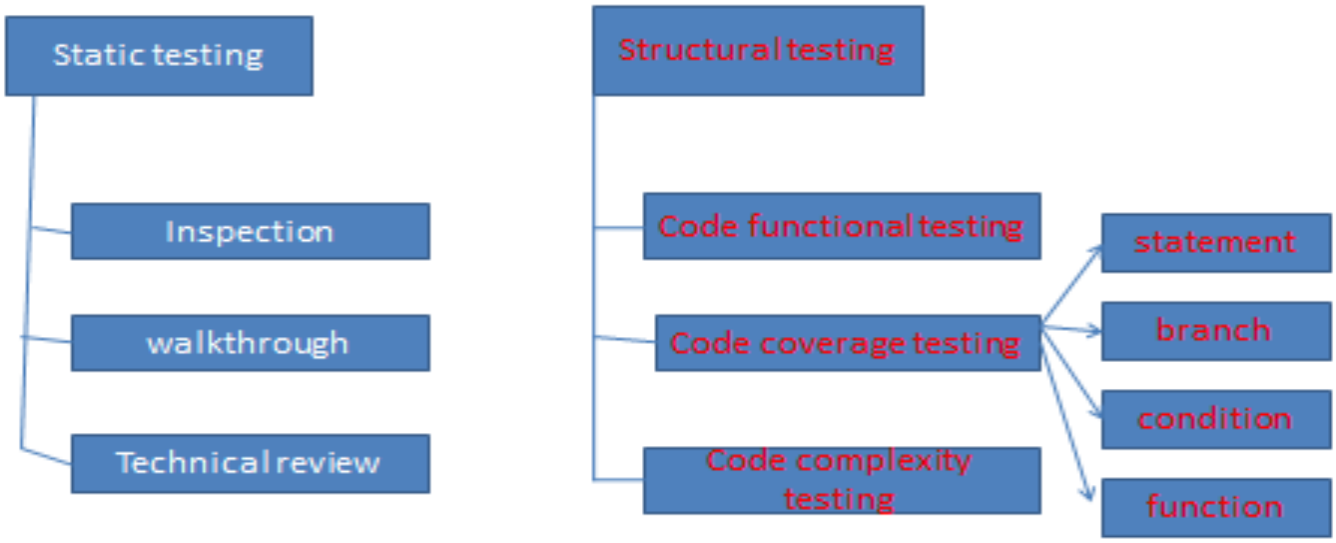
- Testing done without executing program
- Does verification
- Prevention of defects
- Checklist and process to be followed
- Cost of finding and fixing defects is low

➢ Dynamic

- Testing done by executing the program
- Does validation
- Finding and fixing defect
- Involves test cases for execution
- Cost of finding and fixing defects is high

❖White box testing

- Detailed investigation of internal logic and structure of the code.
- Also known as glass box/ structural testing.
- View code.



➢ static testing

- Isa type of testing in which the program source code is tested without running it.
- Tester only review and examine the code.
- SWB
- Eg : review ,inspection , walkthrough

❖1.1.a reviews

- Examining the documents such as rqmts , design , test cases.
- Informal review
- Formal review: simple meetings
- 4 key elements of FR :
- Identify problem.
- Follow rules: like : amount of code , time , different role.
- Prepare : each participant is supposed to be prepare
- Write a report : summarizing the result of the review

❖1.1 b Inspection

- Most formal type of reviews
- Highly structured and require training for each participant
- Person who present the code , isnt the original programmer.
- Other participant is called inspectors.
- Reviewing the code from different perspective such as a user , a tester or a product support person.
- Some inspector called moderator , recorder : assure that the rules are followed
- One inspector review the code backword.
- After inspection , inspector meet again to discuss the defect they found and prepare report.
- It require training

❖1.1 c Walkthrough

- Programmer present the code to a small group.
- Reviewers should receive copies of s/w in advance. ??
- Having at least one senior programmer as reviewer
- Presenter read through the code , reviewer listen and raise question.
- Presenter write a report.

➢Structural / dynamic white box testing

- What is happening inside the system
- Tester are required to have knowledge of internal implementation
- Determine what to test , what not to test.

❖1.2 a code functional testing

- Quick check.
- Before submitting code to code coverage
- Methods :
- Obvious test : knowing i/p and expected o/p
- Debug version : make sure , program is passing through right loops , iterations the right number of times.
- Debugging tools : adding breakpoint in the module to view certain state of variable.

❖1.2.b Code coverage testing

- Is a measure used to describe the degree to which the source code of a program is tested
- High light aspects of the code which may not be adequately tested and which require additional testing.
- Compiler debugger : it is sufficient for small programs.
- Code coverage analyzer:
- Code coverage analyzer : hook into s/w . Run transparently in the background while testing.
- Each time a funcation , line of code , loop is executed , analyzer record the information
- Prepare statistics
- It shows :
- What part of the s/w your test case don't cover.
- Which test cases are redundant.
- What new test cases need to be created for better coverage.

➢ Statement coverage

- It make sure that every statement in the program should execute at least once.
- no.of statement exercised * 100
  total no.of statements
- It covers only true conditions
- Eg : 1: print "hello world"
  2: print " date is :" ; Date$
  3: print " time is :" ; Time$
  4 : End

- Advantage :
- It verify what the written code is expected to do
- Disadvantage:
- It can't test false condition
- It doesn't report that whether the loop reaches its termination condition
- It doesn't understand logical operators

➢ Branch coverage

- It cover all the paths in the s/w
- Ensure that whether a program can jump to all possible destinations
- Also known as decision coverage
- Report whether Boolean expressions tested in control structure evaluated to both true & false.
- It cover switch statement , exception handlers and interrupt handlers.
- Number of decisions outcomes tested * 100
  Total number of decision outcomes
- Eg : 1: print "hello world"
  2: if Date$="01-01-2016 then
  3: print " happy new year"
  4:end if
  5:print " date is: " ; Date$
  6: print " time is " ;Time$
  7:End

Date $	Line #
01-01-2016	1 ,2 ,3 ,4 ,5 ,6 ,7
01-02-2016	1 , 2 , 5 , 6 , 7

- Disadvantage : ignores branches within boolean expression

➢ Condition Coverage

- Also known as predicate coverage
- Each one of the boolean expression have been evaluated for both TRUE FALSE.
- Takes the extra conditions on the branch statement.
- Total decisions exercised * 100
  Total no.of decisions
- Eg : 1: print “hello world”
  2: if Date$=“01-01-2016 AND Time$=“00:00:00 “ then
  3: print “ happy new year”
  4:End if
  5:print “ date is: “ ; Date$
  6: print “ time is “ ;Time$
- 7:End

Date$	Time$	Line #
00-00-2016	00:00:00	1 , 2, 5 , 6, 7
01-01-2016	11:11:11	1 , 2 , 5, 6, 7
00-00-2016	11:11:11	1 , 2 , 5, 6, 7
01-01-2016	00:00:00	1 ,2 ,3 ,4, 5, ,6 ,7

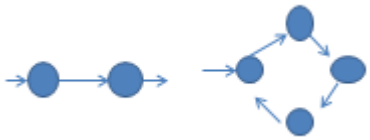
- When consider branch coverage , first 3 conditions would be redundant and could be equivalence partitioned into single test case.
- With condition coverage all 4 cases are important.
- Condition coverage is sufficient for code coverage.

➢ Function coverage

- Identify how many program functions are covered by test cases
- Measure how many times a given function is called

➢ Code complexity testing

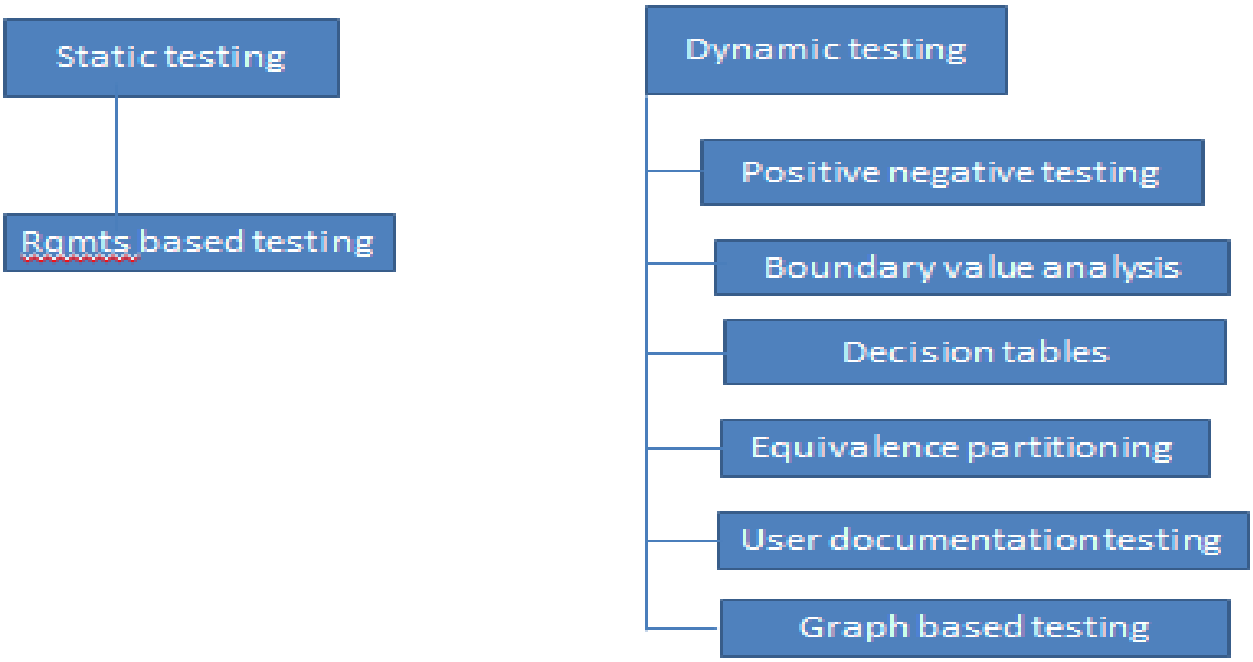
- Cyclomatic complexity is a source code complexity measurement , calculated by developing a control flow graph of the code.
- It measure the amount of decision logic in s/w



- Each circle , called a flow graph node represent statement.
- Areas bounded by edges and nodesare called regions.
- Complexity is computed in one of three way:
- 1. no.of regions of flow graph
- 2. V(G)= E-N+2
- 3. V(G)=P-1 where p : no.of decision points

Black box testing

- Behavioral testing
- Internal implementation of s/w being tested is not known to the tester
- Derives sets of i/p conditions that will fully exercise all functional requirements for a program
- Eg : tester , without knowledge of the internal structure of a website, tests the web pages by using a browser , providing i/p's and verifying o/p's



- Advantage:
- Tester can be nontechnical
- No need to have detailed functional knowledge of system.

- Disadvantage:
- Challenging to design without having clear functional specification.
- Difficult to identify tricky i/p's if test cases not developed based on specification. Eg : website testing

❖1.1.a Requirements based analysis

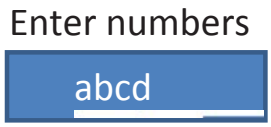
- Detailed review of the requirements specification. Pretend to be a customer.
- Test cases , data , conditions are derived from requirements.
- It includes functional tests : standards , guidelines , review similar s/w .
- non functional attributes such as statements are feasible , consistent , relevant , code free .
- Terminology check list :
- -- if … then missing
- Always , sometime , often , good , fast , efficient

❖1.2 a positive negative testing

- Positive :System validated against valid i/p data.
- It check whether an application behaves as expected with positive i/p.
- Intention is to check whether s/w application not showing error when not supposed to and showing error when supposed to.

Enter only numbers

- Negative : system validated against the invalid i/p data.
- It check whether an application behaves as expected with negative i/p.
- Intention is to check whether s/w application not showing error when supposed to and showing error when not supposed to.
- Goal is to check stability of the s/w against
- incorrect data set.



Positive testing

- Testing System by giving valid data
- Check for only valid set of values
- Verify the known set of test conditions
- Aim: project works as per specifications

Negative testing

- Testing System by giving invalid data
- Check for only invalid set of values
- Done to break the project with unknown set of test conditions
- Aim: break the application by providing invalid set of data.

❖1.2 b Boundary value analysis

- If s/w can operate on the edge of its capabilities , it will almost operate well under normal condition.
- BVA : testing at the boundaries between partitions (valid invalid partition)
- Eg : create 10 element integer array initialize each element to -1

  1: dim data(10) as integer
  2: for i=1 to 10
  3: data(i)=-1
  4: next I

  Eg : password field should accept minimum 8 and maximum 12 character.

Types of boundary condition :

- Numeric -- character -- quantity -- speed
- Characteristics of those types:
- first/last -- min/max -- start/finish
- Eg : field allow 1 -255 characters
- Eg : program read / write to a CD
- Eg : program allow to print multiple pages onto single page

❖1.2 c Decision tables

- Deals with different combination i/p' s with their associated o/p .
- Also called cause effect table.
- Determine test scenario for complex business logic.
- It help tester to search the effect of combination of different i/p 's
- Help us to look at complete combination of conditions.

		R1	R2	R3	R4	R5	R6
conditions	Employee type	S	H	S	H	S	H
	Hours worked	< 40	<40	40	40	>40	>40
Action	Pay base salary	X		X			
	Calculate hourly wages		X		X	X	X
	Calculate over time						X
	Produce absent report		X				

❖1.2 d Equivalence partitioning

- Is a process of methodically reducing the huge set test case into smaller manageable set that still adequately test s/w.
- If s/w behaves in an identical way for a set of value , then the set is termed as equivalence partition.
- i/p data is analyzed and divided into equivalence classes which produces different o/p.

- Two steps :
- -- identifying equivalence partition
- -- picking one value from each partition for the complete coverage.
- Advantages :
- Disadvantage:
- Eg : to copy in calculator: five ways:
- 1. Click copy menu item
- 2. type c
- 3. type C when menu displayed
- 4. press ctrl+c
- 5. press ctrl+shift+c

- Partition these 5 i/p paths into 3
- 1. Click copy menu item
- 2. type c
- 3. press ctrl+c

- Eg : window file name can contain any character except * , ? ,# and filename can have from 1 to 255 characters.
- Equivalence partition:
- Valid characters
- Invalid characters
- Valid length
- Names that are too short
- Names that are too long

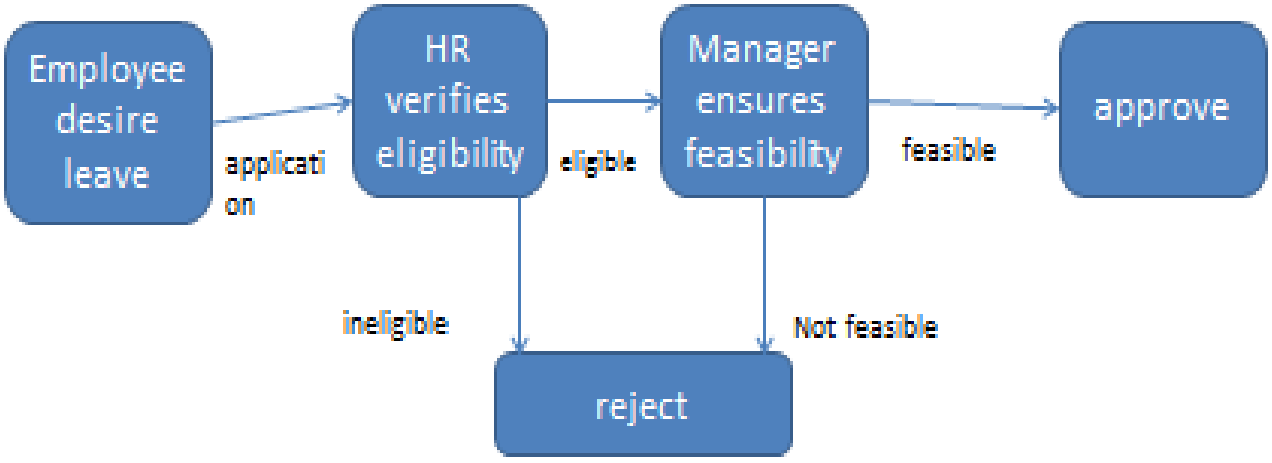
❖1.2 e User documentation testing

- Covers all the manuals , user guide , installation guide, setup guide , read me files online help.

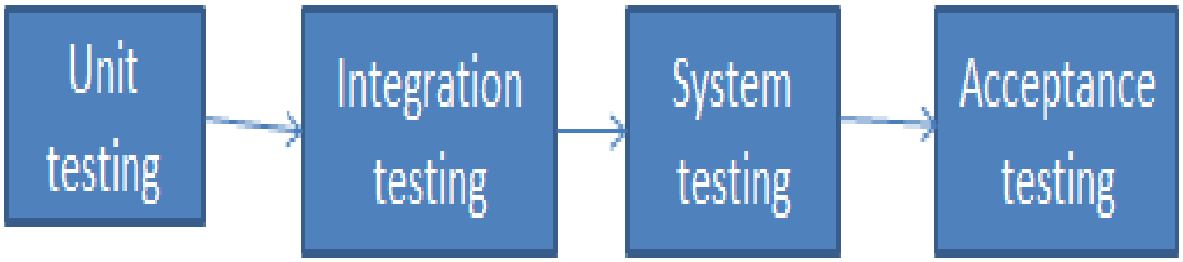
- Two objective :
- --what is stated in document is available in the s/w
- --what is there in the product is explained correctly in the document.
- Advantages :
- --it improve usability
- --it improve reliability
- --it lowers support cost
- list of s/w components :
- --packing text and graphics
- --warranty / registration
- --labels and stickers
- --Installation setup instructions
- --Users manual's
- --EULA

❖1.2 f Graph based testing

- Also called state based testing
- It represent transaction or work flow
- This would support for testing system implementation against system specification



Chap 3 : Levels of testing & special test



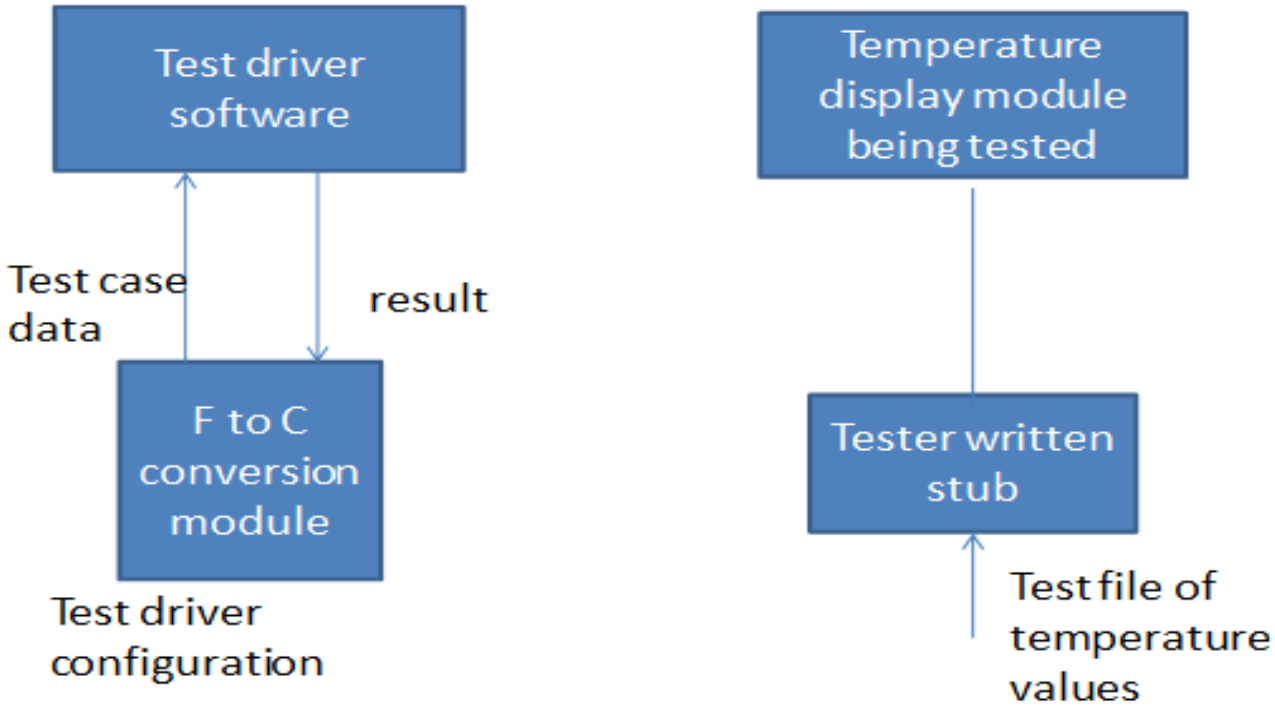
❖Unit testing

- Testing that occurs at lowest level
- Unit testing may be include code file , classes , methods which can be tested individually for correctness.
- Drivers and stubs are written.
- Perform following functions
  1. tests all control paths.
  2. ensure all statements executed at least once.
- Test data structure.
- Check range of i/ps
- Driver:
- simulate calling unit

- Act as main program that accept test cases data , passes such data to the components and print result.
- Used in bottom up approach

- Stubs :
- Simulate called unit
- Act as sub module that are subordinate the components to be tested.

- Dummy subprogram uses subordinate module interface , may do minimal data manipulation , print verification and return control to the tested module
- Used in top down approach

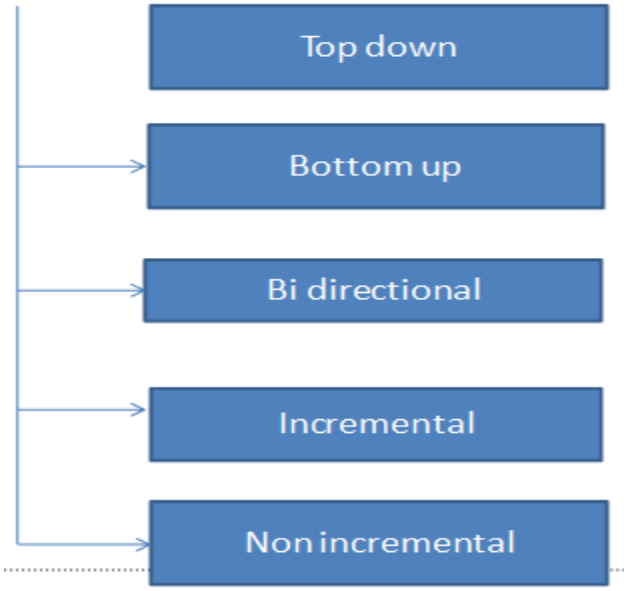


- Advantage of unit test:
- Find problem early in the development cycle
- Basic understanding of units.

- Disadvantage of unit test:
- Initial time required for development
- It will not catch integration errors

❖ Integration testing

- Performed against group of modules
- Objective : take all tested individual modules, integrated them , test them again and develop s/w
- Checks parameters passing between units.
- Ensure that all modules work properly as per user requirements, when they are integrated
- Advantage :
- 1. easy to fix the error compared to system testing



❖ Decomposition based testing

- Decomposition of design into functional modules
- Test interfaces among separately tested modules
- Two methods:
- Integrate all the modules together and then test it
- Integrate modules one by one and test them incrementally

➢ 1.a Top down integration

- Testing begins from top level, progressively adds in lower level module one by one.
- As low level code added , stub will replace with actual components.
- It needs design and implementation of stubs, drivers may not be required

- Advantage :
  1. Feasibility of program can be determine easily at very early stage
  2. Doesn't require drivers.
- Disadvantage:
  1. unit are rarely tested alone before their integration
  2. It became difficult to exercise the top level routines in the desired manner.
  3. Stubs are required

➢ Bottom up integration

- Testing starts at the atomic modules level
- Each subsystem is tested separately and then the full system is tested.
- Steps :
- 1. low level modules are combined into clusters (builds)that perform a specific s/w function.
  2. drivers are written
  3. build is tested
  4. drivers are removed clusters are combined moving upward .

- Advantage:
  1. makes system more robust since each unit is tested first for its correctness.
- Disadvantage:
  1. top level units are most important but tested last , pressure of delivery may cause problem of not completing testing
  2. drivers are required

❖ Bi directional integration

- Follows top down , bottom up approach either simultaneously or one after another.
- Sandwich overcome the
- shortcoming of top down , bottom up approaches.??
- Steps :
  1. bottom up testing start from middle layers and goes upward to the top layer.
  2. top down testing starts from middle layers and goes downward
  3. big bang approach is followed for the middle layers . From this layers , bottom up approach goes upwards and top down approach goes downwards
- Advantage :
  1. suitable for very large project
  2. overcome the short coming of T/D B/U approach.
- Disadvantage:
  1. cant use for project which has huge interdependence between different modules.
  2.High cost.

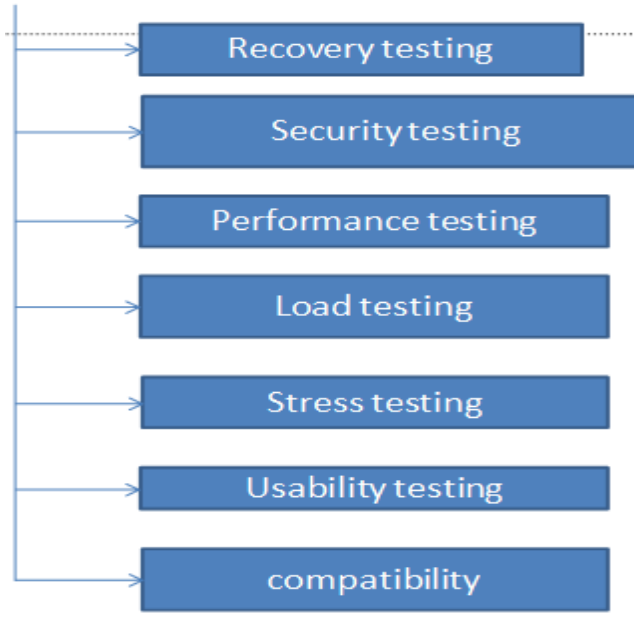
❖ Incremental testing

- Program is tested in small increments by adding a minimum number of components at each interval.
- Errors are easier to isolate and correct and interfaces are more likely to be tested completely.
- Incrementally keep on adding the modules and test the recent environment.
- Advantage :
  1. doesn't require drivers and stubs
  2. easy to localize the errors since modules are combined one by one
  3. Interfacing errors are uncovered earlier
  4. First suspect is the recently added module , thus debugging becomes easy and simple.

❖ Non incremental approach

- Also called big bang testing
- There is no testing of individual units and integration sequence.
- Here all the units are linked at once, resulting in a complete system.
- Testing is done as last phase of development lifecycle.
- Use this method when:
- 1. data flow is very complex
- 2. difficult to identify who is parent and who is child.
- Advantage :
  1. no stub / driver is required.
  2. cost involve is vey low
- Dis Advantage :
  1. defect may not be found easily, 2. hard to debug
  3. difficult to isolate any errors found

❖System testing



- The system is tested against functional / nonfunctional requirements such as accuracy, reliability, speed defined by the s/w system specification.
- IEEE defined system testing as ‘ testing conducted on complete integrated system to evaluate the system’s compliance with its specified requirement
- Final testing done on a s/w system before it is delivered to the customer.
- Advantage :
  1. clearly specify how the application should behave.
- Test system from user point of view.

❖ Recovery testing

- System is forced to fail in different ways to check whether the s/w recovers from failure without any data loss
- Check how fast and better the application can recover against any type of crash
- Example :
  1. while application is running , suddenly restart the computer
  2. while receiving data from n/w , unplug the connecting cable

❖ Security testing

- Attempt to verify that protection mechanisms built into a system will protect it from improper penetration.
- Tester play the role of the individuals who desire to penetrate system.
- It also aim at verifying below principle:
  1. confidentiality 2. integrity
  3. authentication 4. authorization
  5. availability 6.non repudiation

❖ Performance testing

- Non functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload
- Measure quality attribute such as reliability availability , resource usage.
- Factors that governs performance testing:
  1. throughput : capability of a product to handle multiple transaction in given period.
  2. response time : delay between the point of request and first response
- 3. tuning : setting different values to the parameters of the product

❖ Load testing

- Check whether system can perform well for specified load
- You feed it all that it can handle
- Identify maximum operating capacity of an application
- Eg:
  1. downloading a series of large files from the internet.
  2. server that can handle thousands of simultaneous connections , max out the capability.
  3. running multiple applications on a computer or server simultaneously.
  4. reading writing data to and from a hard disk continuously.
- Advantage :
  1. expose memory overflow bugs.
  2. prevent s/w failures
  3. measure performance of internet infrastructure.

❖ Stress testing

- Running s/w under less than ideal conditions— low memory , low disk space , slow cpu
- Limiting s/w to their bare minimum.
- Goal : ensure that s/w doesn't crash in condition of insufficient computational resources.
- Eg : 1. flooding a server with useless email messages
- 2. infect system with viruses , trojans

❖ Usability testing

- Done from an end users perspective to determine if the system is easily usable.
- Whether users feel comfortable with application according to different parameters – the flow , navigation , layout , readability
- Usability testing consist of following main aspects

- - ease of use ?? -- look and feel ?? Speed in interfaces

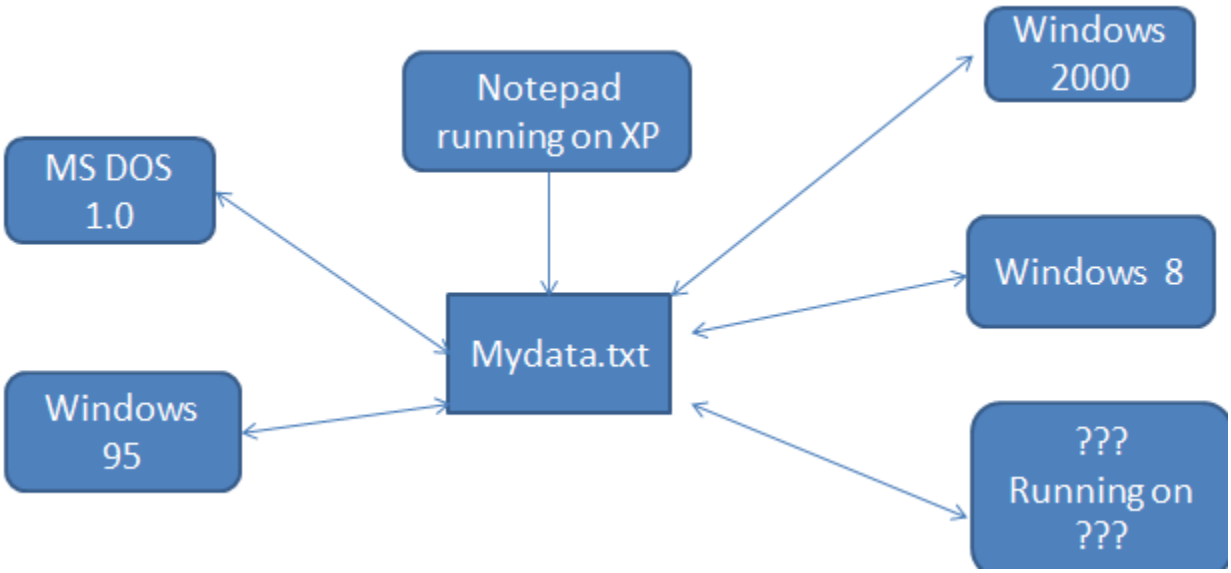
- Example : usability test on website :
- Mistake in web design:
- -- bleeding edge technology
- --scrolling text , marquees running animation
- -- long scrolling pages
- -- non standard link colors
- --outdated information
- -- orphan pages
- -- complex URL
- --lack of navigation suppor

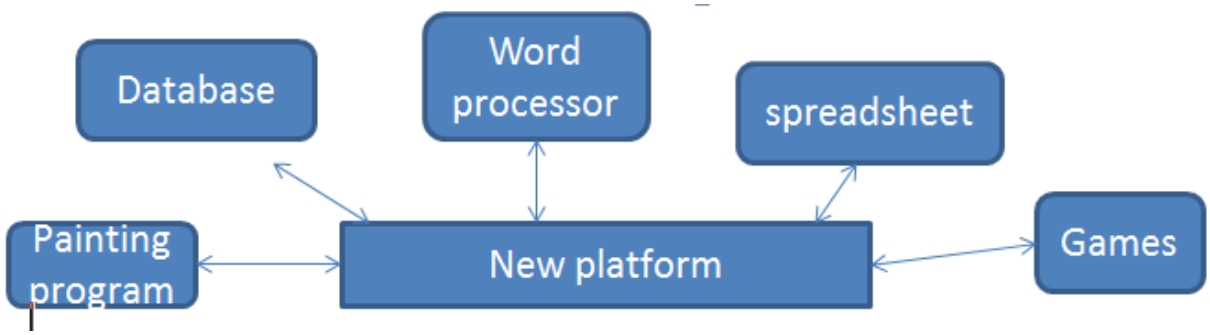
❖ Compatibility testing

- Check whether s/w interact with and shares information correctly with other s/w.
- Team specify what level of compatibility are required by the system on which your system will run on.
- Eg : stand alone medical application that run on its own OS , store its data on its own memory and doesn't connect to any other devices would have no compatibility consideration
- Eg : fifth version of word processor , that reads and writes files from other version of word processor , allow multi user editing over internet has compatibility consideration.
- When performing this testing keep following points :
  1. what other platform and other applications s/w is your s/w designed to be compatible with ?
  2. what compatibility standards and guidelines should be followed ??
- Example :
- 1. cutting text from web page and pasting it into word document
- 2. saving accounting data from spreadsheet and then loading it into completely
  different spreadsheet program.
- 3. upgrading to a new database program and having all your existing databases load in.

❖ Back ward compatibility testing

- Backward : whether s/w work with previous versions of the s/w.
- Forward : whether s/w work with future versions of the s/w.





- There could be hundreds of existing programs for the current versions of the OS
- Equivalence partition can be applied to reduce the amount of work.
- Criteria :
- 1. popularity 2. age
- 3. Type 4. manufacturer
- Forward compatibility testing is difficult to test

❖ Acceptance testing

- Formal testing conducted to determine whether s/w satisfies its acceptance criteria
- Types of acceptance testing :
  1. user acceptance test : validate by user
  2. operational acceptance test: system meet rqmts for operation
  3. contract acceptance testing: validate against contract acceptance criteria
  4. compliance acceptance testing: validate against regulation
- Work at each stage of s/w development.
- Starting from proposal stage till the point where the system is formally accepted by user
- May be used as basis on which exit criteria for each phase and entry criteria of next phase may be defined.
- Designs must fulfill acceptance criteria so coding phase can start.
- Consider following attributes :
- -- data integrity --usability --performance
- --availability --documentation

❖ Alpha testing

- Consider as internal acceptance testing in which End users test the s/w at developer's site.
- Assess the performance of s/w in the environment in which it is developed.
- Users report the errors to s/w developers.
- Advantage :
- 1. simulate real time user behavior and environment ?
- Early detection of errors
- Disadvantage :
- 1. sometimes developers and tester are dissatisfied with result of alpha testing

❖ Beta testing

- Consider as external acceptance testing in which s/w sent out to a selected group of customers who use it in a real world environment.
- 'live testing' and conducted in an environment , which is not controlled by the developers.
- Testing is performed without any interference from the developers.
- Beta tests can be a good way to find compatibility and configuration bugs ??
- Advantage :
  1. allow to test post lunch infrastructure
  2. reduces s/w failure risk via validation.
  3. Improveproduct quality via customer validation
  4. Increase customer satisfaction
- Disadvantage :
  1. test management is an issue
  2. finding right beta users could be a challenge.

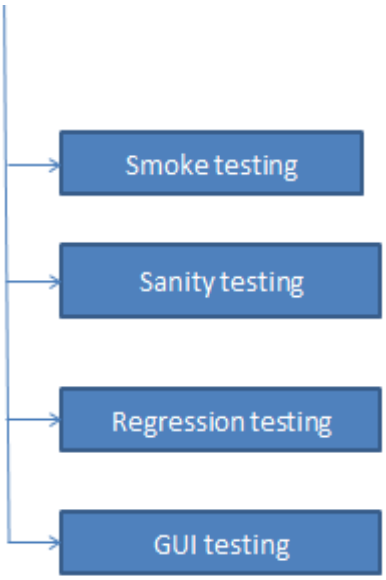
alpha testing

- Performed at developers site
- It involves white box black box testing
- Requires lab environment
- Ensure quality of product
- Test in controlled environment

Beta testing

- Performed at customer site
- It Involves black box testing
- Doesn't require lab environment
- Ensure product is ready for the real time users
- No controlled environment

❖ Special tests



❖ Smoke testing

- s/w build : developers makes executable files by combining all files. These executable files are known as s/w build.
- s/w build includes all data files , libraries , reusable files that are required to implement function.

- Smoke testing :Check normal health of the build and make sure it is possible to continue testing.
- Smoke testing performed after s/w build to ascertain that the critical functionalities working fine.
- Activities :
  1. components translated into code are integrated into a build.
  2. series of test is designed on build
  3. build is integrated with other builds. And entire build is smoke tested daily.

- Objectives of smoke testing :
- 1. uncover errors that have highest impact on project schedule.
- 2. not perform exhaustive testing , but to verify that the critical functionalities of the system are working fine.
- Example :check application launches successfully , GUI is responsive.
- Example : let 4 modules
- --New employee --existing employee
- --admin employee --user module
- --Smoke test execute all major functions of modules.
- Smoke test done by development team before release.
- When the build release to the testing team where testing team perform smoke test.
- Advantage:
- 1. integration risk minimized
- 2. quality of product improved
- 3. progress is easier to assess.

❖ Sanity testing

- Sanity testing performed after build has clear the smoke test . It check the major functionality with finer details.
- Tester team check the extreme functionality of the modules , do not go in deep.
- This testing is done after thorough regression testing is over , it is done to make sure that any defect fixes or changes after regression testing doesn't break the core functionality.
- Sanity tests are mostly non scripted.
- Objective : not to verify thoroughly the new functionality, but to determine that the developers has applied some rationality while producing s/w.
- When we performed sanity testing??

Smoke testing

- Check whether critical functionalities are working fine or not.
- Objective : verify stability of the system
- Performed by the developers or testers
- Usually documented or scripted
- Subset of regression testing
- Exercise entire system from end to end
- General health check up

Sanity testing

- Check whether bugs have been fixed or not
- Verify rationality of the system
- Performed by testers.
- Usually not documented or scripted.
- Subset of acceptance testing
- Exercise only particular component.
- Specialized health check up

❖ Regression testing

- Re testing of a s/w to verify that system modification doesn't affect system core requirements.
- Regression testing is required whenever new features added or introduce modification.
- Different classes of test cases :
  1.test cases that will exercise all s/w functions.
  2.tests that focus on functions that are likely to be affected by change.
  3. tests that focus on modules that have been changed.

❖ Regression testing techniques

- Retest all
- Retest part of product
- Retest based on prioritized test cases.
- --selecting test cases based on:
- --frequent defect
- --verify core feature
- --function that has recent change
- --boundary value
- --failure test case(test to fail)

❖ GUI testing

- How application handles keyboard , mouse events , and how menu bars , tool bars reacts to user input.
- 7 important traits common to good UI

1. follow standards and guidelines.

- when to use check box instead of option
- when to use warning , critical messages.

2. intuitive :

- functions be there when you expect them.
- easily get from one function to other.
- move back
- is any excessive functionality that make s/w complicate.

3. consistent:

- Users have a habit that if they do something a certain way in one program , another will do the same operation in same way.
- Eg : in Notepad search find or F3 in word pad edit find
- Shortcut keys and menu selection. Eg : F1 for Help
- Terminology and naming . Eg : find , search
- Placement of buttons : position of OK Cancel button

4. Flexible :

- users like to select what they want to do , and how they want to do.
- windows calc two views : 1. standard 2. scientific
- state jumping : more ways to complete same task.
- data input / output : eg : to put test into a word pad type , paste it , load it.

5. Comfortable :

- appropriateness: look and feel. Eg: financial business application should not have loud color , sound.
- error handling : eg :warning before critical message
- performance : eg: show progress through status bar

6. correct :

- Test whether GUI does what its supposed to do..
- WYUIWYG :Eg: click on save button should saved document on disk.
- Language &spelling

7. Useful:

- features actually valuable to s/w.

❖ Object oriented application testing

- Traditional prog: operate on data
- OO : operate on objects that are instance of classes.
- Identify &specify object & services provided by object (data members , methods)

1.Sate based testing :

- verify whether methods of class are interacting properly with each other.
- Finite state machine represents possible states of objects.

2. Fault based testing:

- detect the presence of faults.
- start by examining analysis and design models of OO s/w

3. scenario based testing:

- detect errors that are caused due to incorrect specification .
- It describe how a user accomplish a task.

❖ Client server testing

1. components testing.

- test client and server individually.
- when server is tested , we may need a client simulator
- while testing client , need a server simulator.
- test network by using client server simulator.

2. integration testing:

- after successful testing of server , clients , n/w , they are put together to form system and system test cases are executed.

- **Performance testing :** when number of client are communicating with a server at a time , system performance is tested for maximum throughput.

- **Concurrency testing :** multiple users may be accessing same records at a time , so concurrency test is required to understand response of a system.

- **Disaster recovery testing:** test for disaster recovery and business continuity.

❖ Web based testing

- **Web page features:**

  --text of different sizes , fonts , colors

  --Graphics , photos

  --hyperlinked text and graphics

  --rotating advertisement

  --fields in which the users can enter data

- **Dynamically changing text**

  1. text : check contact information - Check ALT text
  2. hyperlinks : tied to text or graphics.
     - --make sure text link is underline , mouse pointer changes
     - --orphan pages
  3. graphics: if graphics cant load than error must be displayed
  4. forms: text box , list box – for entering or selecting data
     - -- accept correct and reject wrong data.
     - --mandatory fields.

Chap -04 Test Management

Test planning **Test management**

Test process **Test reporting**

❖ Test Plan

- Test plan act as anchor for the execution , tracking and reporting of the entire testing project
  1. preparing test plan
  2. scope management
  3. Deciding test approach
  4. Setting up criteria for testing
  5. identifying responsibility
  6. Staffing and training needs
  7. Resource requirements
  8.Test deliverables
  9. Testing task

❖ 1.Preparing test plan

- It includes :
  1.Methodology.
  2. Requirements: h/w , s/w tools ,training , criteria for pass / fail of test cases , schedule.
- What need to be tested
- How testing is going to be performed
- What resources are needed for testing
- When each test occur.
- Time lines

- Test plan types :
  1. master test plan
  2. testing level specific:
  -- unit test plan -- integration –acceptance
  3. testing type specific :
  -- performance -- security

❖ 2.Scope management

- It include what items , features , procedures , functions will be tested.
- For testing scope management needs:
  1. understanding what constitutes a release of a product.
  2.Prioritizing the features for testing
  3.Deciding which features will be tested and which will not be
  4.Estimation of resources for testing
- Factors for prioritization of features:
  1. features that is new and critical for release.
  2. features whose failures can be disastrous
  3. features that are expected to be complex to test
  4. features which are extension of earlier features.

❖ 3.Deciding test approach

- Deciding right type of test for each feature.
- After prioritizing features identify :
  1. testing tools
  2. special training
  3. configuration / compatibility test

❖ 4.Setting up criteria for testing

- Pass / fail criteria : specify the criteria to determine passed or failed criteria
- Suspend criteria : specify criteria to be used to suspend the testing activity
- Resume criteria : specify criteria to be used to redone the testing activity

❖ 5.Identifying responsibility :

- List responsibility of individual
- Responsibilities define who is responsible for what activity at each stage of development

❖ 6.Staffing and training needs

- Staffing needs estimated based on efforts involved and availability of time.
- Provide number of individuals required for each role.
- Training needs to enhance skills , learn new technology , tools.
- Training needs for use of tools for test cases execution and for reporting.

❖ 7.Resource requirements

- Estimation of h/w , s/w ,human resource , environment needs (support staff , office space)
- Resources required are :
  1. RAM,Processor ,HD
  2. s/w , OS
  3. supporting tools
  4. special testing tools : for load , performance test

❖ 8.Test deliverables

- Documents that are provided before testing phase , or during testing phase or after testing phase
- Different test deliverables are :
  1. test plan
  2. testing strategy
  3. test scripts
  4. test cases
  5. tools
  6. Test summary report
  7. defect report.

❖ 9.Testing task

- Task required to perform testing
- Identifying inter dependency among different testing related task.
- Estimation happens in 3 phases :
  1. size estimation :quantify actual amount of testing
  -- size can be estimated based on
  --LOC : line of code
  --FP : application feature
  --number of screens , report
  2. effort estimation: quantify person
  3. schedule estimation: translate effort into time frame.

❖ Test Management

- Organizing test assets such as test requirements , test cases and test results to enable accessibility and reuse.
- Tools : pen paper , word processor , spread sheet.
- Choices of standards
- Test infrastructure management
- Test people management
- Integrating with product release

❖ 1.Choice of standard

- Standard Defined to provide exact meaning to any subject.
- Testing standard are mainly used to control activities .
- Two types :

  1. **external std :**

  --made by entity other than organisation.

  -- 3 types :

  a. customer std:

  --defined by customers based on business requirements.

  forced by customer to supplier (developer)

  b. national std :

  defined by regulatory authorities.

  applicable to both producer customer.

  violating stds lead to legal actions.

  c. International std:

  globally defined

  applicable to all producer customer.

  eg : ISO , IEEE

2. Internal Std :
--defined by s/w development organization
-- internal std are :
--naming and storage conventions : eg test cases , test results have to be named meaningfully.
-- document std: header level comment at the beginning of file , sufficient in line comment , up to date change history information
--test coding std : consistency during scripts writing
--test reporting std : timely view of the prgress of test

❖ 2. Test infrastructure management

- Testing requires a robust infrastructure.
- Infrastructure made up of 3 elements
  1. test code db
  2. defect repository
  3. configuration management repository

- 1. test case db :
  --Captures all relevant information about test cases.
  Content of test case db:
- 2. defect repository:
  -- capture all relevant details of defects reported for product
- 3. configuration management repository:
  -- labeling , tracking , controlling changes in s/w element.
  --keep track of change control and version control
  --change control ensure : change made by one test engineer , each change produces a distinct version , have access to only the most recent version .
  --version control ensure : provide history of each s/w change , who did what , why ,when

❖ 3.Test people management

- Mainly defects found based on effective testing techniques, but it also depend on skills and knowledge of tester and team dedication.
- Test Team members with different expertise levels have to be integrated to maximize quality.
- Test lead :
  --lead team of testers.
  -- do test plan and approved by management.
  -- Check skills required.
  -- If skill gap than plan training.
  -- create healthy environment
  -- encourage team: bridge gap between team and management
  -- monitor test progress, delay in schedule.
- Consideration of test team management:
  --understand testers : with experience , testers learn to break code and find defect
  -- tester work environment:
  --tester work in pressure due to deadline
  --frustration , since mgmt might not response +vely
  -- role of test team :
  -- 100% testing is practically not achieved.
  -- question mark on tester role

❖ 4.Integrating with product release

- Success of product depend on effective integration of activities from development and testing phases.
- Product release related with schedule of testing phase.
- So , testing must work in integration with product release.

- Points related to planning:

  1. sync point:

  -- between development and testing

  --provide schedule for each type of testing.

  --eg : start , commence date for integration , system testing.

  2. service level agreements :

  --between developer and tester team

  --total period required by testing team to complete testing job.

  3. consistent definition:

  -- consistent defn of defect among team members.

  -- also define severity and priority of defects.

  4. communication channel :

  -- communication between testing and documentation team to keep documentation in sync with the current growth of product

❖ Test process

- Testing is not a single activity instead it's a set of number of processes.
- Test process is divided into following steps :
  1. test plan and control
  2. test analysis and design
  3. test implementation and execution
  4. evaluation of exit criteria and test report
  5. test closure activities

❖ 1.Base lining a test plan

- Base line: form the base for performing further activities
- Validation of documents and specification using which test cases would be planned and designed.
- Test plan is also a base line that must be followed by test team.
- Test plan developed by competent people and approved by higher authority.
- If any change occur than it first reflected in test plan

❖ 2.Test case specification

- Test plan specify :
- unit to be tested , approaches , tools
- Test case specification done separately for each unit.
- Deals with details of testing a unit.
- Testing team designs test case specification which then becomes the basis for preparing individual test cases.
- It state :
  -- item being tested
  --environment need to run test case
  --input data
  --steps to be followed to execute the test
  --Expected results
  --relation with other tests

❖ 3.Update of traceability of matrix

- It is a tool to validate that every requirements is tested.
- It associate requirements to its work product and test cases.
- It is created during rqmt gathering phase (unique identifier) is assigned to each.
- Than in design and code phase , unique identifier for design and code is entered in traceability matrix.
- When test case specification complete , corresponding rqmts which is being tested is updated with test specification identifier.
- This ensure two way mapping between rqmts and test cases.

Template of rqmts traceability matrix

Unique number	Requirements	s/w rqmt specification	Program module	Test specification	Test cases	Modification of rqmts	remarks

❖ Test report

- Providing information about testing result.
- It include:
  --defect
  --impact , severity
  --risk of releasing of product with existing defects.
- types of reports:
  1. test incident report
  2. test cycle report
  3. test summary report

  1. test incident report :
  -- testing report when defect encountered.
  -- entry made in the defect repository.
  2. test cycle report :
  -- test project take place in units of test cycles.
  --running certain tests in cycle.
  --each cycle using a different build of the product.
- It include :
  -- a summary of the activities carried out during that cycle.
  -- defect summary
  -- progress from previous cycle in terms of defect fixed.
  --outstanding defects that yet to be fixed.
  --any variations observed in effort or schedule.
  3. test summary report:
  --final stepto recommend suitability of a product release.
  two types of test summary report:
  1. phase wise test summary.
  2. release test report.

- Test summary report include:

  1. summary of test activities.
  2. variance between planned and carried out activities.

  -- test that could not be run

  --modification to tests from what was in original tests.

  -- additional tests that were run

  --Difference in effort and time.

  3. summary of results:

  --tests that failed –with root cause descriptions.

  --severity of impact of defects.

  4.assessment and recommendation for release:

  --"fit for release" assessment

  --recommendation of release

❖ Test report

1.recommending product release

2. matrix executing test cases

3. collecting and analyzing metrics

4 preparing test summary report

❖ 1.Recommending product release

- Based on test summary report , organization decide whether to release the product or not.
- Senior manager consult customer support team , development team , testing team for decision.
- Product release recommended based on :

  -- market pressures. -- low priority/impact defect

- Test report summarize following , in order to decide product release:

  -- what is impact / severity of defect?

  --risk of releasing product with existing defects?

❖ 2.Matrix executing test cases

- Test cases has to be executed at appropriate times during project
- Eg : test cases corresponding to smoke tests may be run on daily basis.
- Eg : system testing test cases will be run during system testing
- As test cases executed , defect repository updated : it contain all information about defects un covered by testing
- Test case run only when entry criteria satisfied and should be complete when exit criteria satisfied
- Trace ability matrix updated.

❖ 3.collecting and analyzing metrics:

- Information about test execution gets collected in test logs
- And converted into meaning full metrics by formulas.

❖ 4. preparing test summary report :

- Handover to senior management about the fitness of product release.
- It is prepared at the end of testing project.

Chap 5 : Defect Management

❖ Defect

- Inconsistency in behavior of s/w
- s/w doesn't meet requirement
- Errors in coding or logic
- Expected result don't match with actual results.
- Root causes of defects :
  --requirement are not well defined
- Defect management process focuses on:

  --preventing defects.

  --catching defects as early

  --minimizing impact of defects

- Defect classification:

  -- severity wise --status wise

  --work product wise -- error wise

❖ Defect classification

- Severity wise:

  -- major -- minor --fatal

- Work product wise:

  --SSD --FSD --ADD

- Source code
- User documentation

- Error wise:

  --comment : incorrect , inadequate

  --computational error

  --logic error

  --interface error

  --Boundary condition neglected

- Status wise:

  --open --deferred

  --close --cancel

❖ Defect Management Process

- Objectives :

  --finding defects

  --reducing them at low cost

  1. Defect prevention
  2. deliverable base line
  3. defect discovery
  4. defect resolution
  5. process improvement

❖ 1. Defect prevention:

a. identify critical risks:

-- missing key requirements

-- H/w malfunctioning

-- performance is poor

b. estimate expected impact:

-- E= P * I I→ impact

P→ probability of risk to become real

c. minimizing expected impact:

--eliminate risk

-- reduce probability of risk

-- reduce the impact :

eg : disaster recovery for reducing security bug

❖ 2. deliverable base line:

- Deliverables is a base lined when it reaches a predefined mile stone.
- Milestone involve transferring the product from one stage to next
- establish milestones where deliverable will be considered complete and ready for further activities.
- a deliverables should be base lined when changes to the deliverable or defect In the deliverable , can have impact.

❖ 3. defect discovery:

- defect is said to be discovered when documented and acknowledged as valid defect from developer side
- find defect
- report defect
- acknowledge defect

❖ 4. defect resolution:

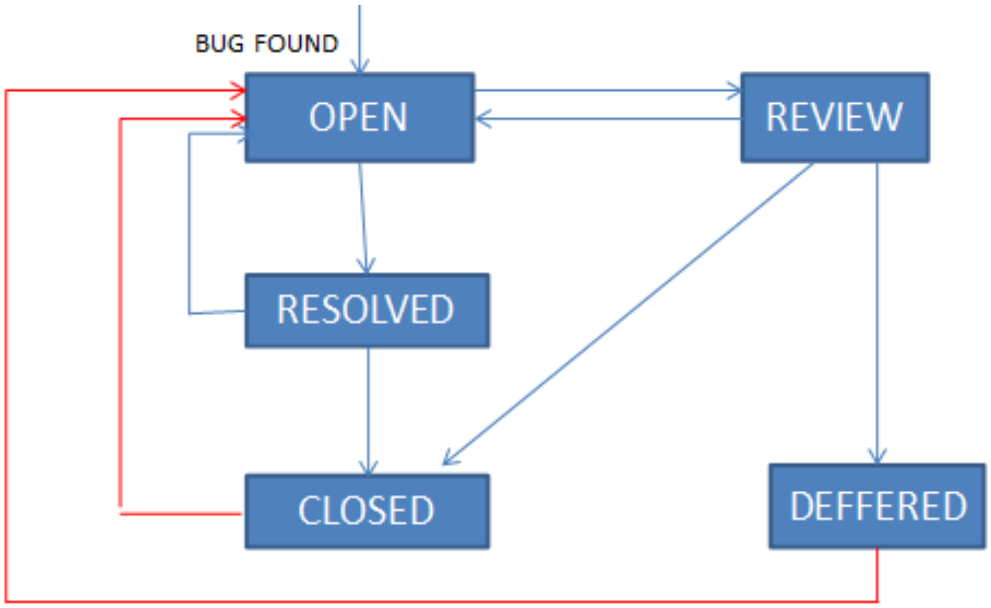
- done by developer
- prioritize risk : critical , major , minor
- schedule fix: based on priority, defect fix should be scheduled
- fix defect: correcting deliverables .
- report resolution: developer response back to tester??

❖ 5. process improvement:

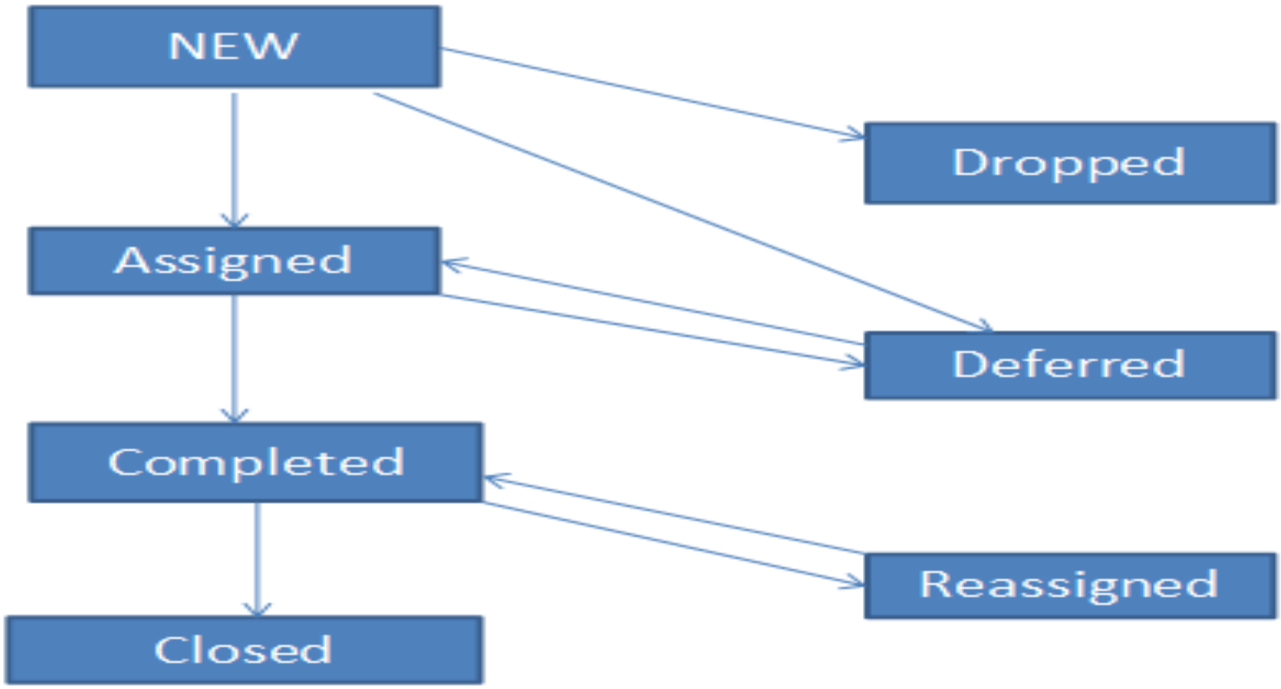
- identify and analysis of process in which a defect originated
- improve process to prevent future occurrence of similar defects.

❖ Defect bug life cycle

- It start when defect is found and end when a defect is closed.
- Bug has different states in the life cycle.



- OPEN: when bug found by tester
- OPEN-RESOLVED : once programmer fixes the code
- RESOLVED-CLOSED : tester close the bug after verifying
- OPEN-REVIEW : CCB decide whether the bug should be fixed or not
- REVIEW- CLOSED : CCB decide that bug is not a real problem
- REVIEW-DEFERRED: CCB decide that bug should be fix in future , but not for this release of the s/w
- REIEW-OPEN: CCB decide that bug should be fix.
- RESOLVED – OPEN : tester find bug is not fix
- CLOSED – OPEN : bug may occur in future
- DEFERRED – OPEN: bug may occur in future.



❖ Defect template

- Capture defect data
- Purpose: state problem as clearly as possible so that developers can replicate the defect easily and fix it.
- Severity :
  1. system crash
  2. data loss
  3. minor problem : misspelling , UI layout
  4. suggestion
- Priority:
  1. immediate fix
  2. must fix before the product is released
  3. should fix when time permits
  4. would like to fix but the product can be released as it is.

❖ Techniques for finding defects

- Static techniques:
  --testing is done without physically executing a program
  --code review , walkthrough , inspection
- Dynamic techniques :
  --components are physically executed.
- Operational techniques :
  --defect is found as a result of failure.

❖ Defect reporting

- Discovered defect must be brought to the developers attention.
- Guidelines while reporting :
- 1.be specific:
  --don’t say anything which add confusion.
  --in case of multiple paths , mention exact path
  --be detailed : provide more information
- Be objective: stick to fact and avoid emotions
- Reproduce the defect:
  -- don’t be impatient .
  --replicate it at least once more
  -- state exact test condition
- Review the report

Chap -06 Testing tool and measurement

❖ Manual testing

- Process of manually testing s/w for defects
- Testers takes over the role of an end user and test the s/w to identify any unexpected behavior or bugs.
- Advantages :
  1. it is covered in limited cost
  2. easy to reduce and add test cases according to project movement
  3. less time required to begin manual testing
- Limitation :
  1. time consuming
  2. limited support for regression
  3. Not consistent or repeatable
  4. error prone testing
  5. impractical performance testing
  6. limited scope
  7. no batch testing

❖ Automated testing

- Automating the manual testing process.
- Reduce manual testing work by using tools.
- Advantages:
  1. speed
  2. efficiency
  3.accuracy and precision
  4. resource reduction
  5. simulation
  6. relentlessness

❖ Need for automated testing tools

- 1. Effective testing:
  --automation perform test repeatedly, so save human time.
  --Eliminate required think time.
  --Perform test at machine speed.
- 2. reducing testing costs
  --test s/w faster with fewer errors than individual.
  --testing tools can replicate the activity of large number of users.
  --require only fraction of h/w to perform load/stress testing
- 3. replicating testing across different platform
- 4.greater application coverage
  --test tool cover all modules
- 5.result reporting
  --produce convenient report

❖ Testing tool

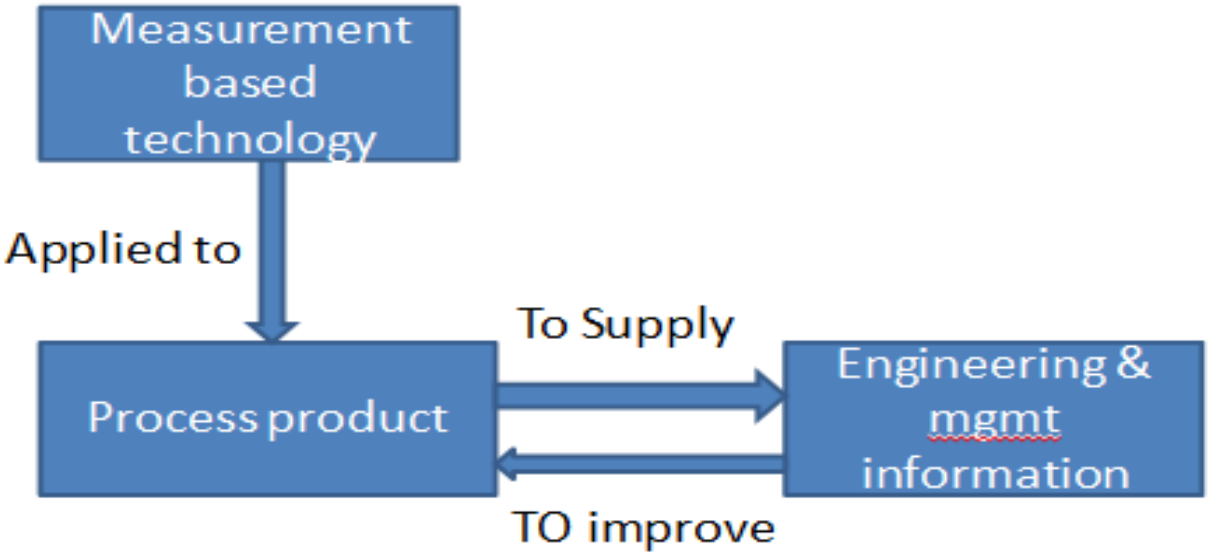
- Two types:
  1. static testing tool 2.dynamic testing tool
- 1. static testing tool
  --code is not executed.
  --it is send as input data to the tool
  --it is extension of compiler technology.
- 2. dynamic testing tool:
  --code are in running state
  --to identify arithmetic errors

- 1. static testing tool include :
  --flow analyzer : ensure consistency in data flow
  -- path test : find unused code
  --coverage analyzer: ensure all logic path
  --interface analyzer : examine effect of passing variables between modules.
- 2.Dynamic test tool include:
  --test driver : input data into under tested modules.
  --test bed : display source code along program under execution
  --mutation analyzer : errors are purposely fed in order to test fault tolerance
- Advantages of test tool
  --Reduction of repetitive work
  --Consistency
  --Ease of access of information about tests(chart & graph)
- Disadvantages of test tool:
  --Unrealistic expectation from tool
  --People make mistake by ignoring time , cost , effort
  --People doesn't maintain tests assets
  --Depend on tool a lot

- When to use automated test tool:
  -- regression testing for stable system
  --fast data creation
- automated test tool not suitable when:
  -- testing new functionality
  -- when user interfaces changes

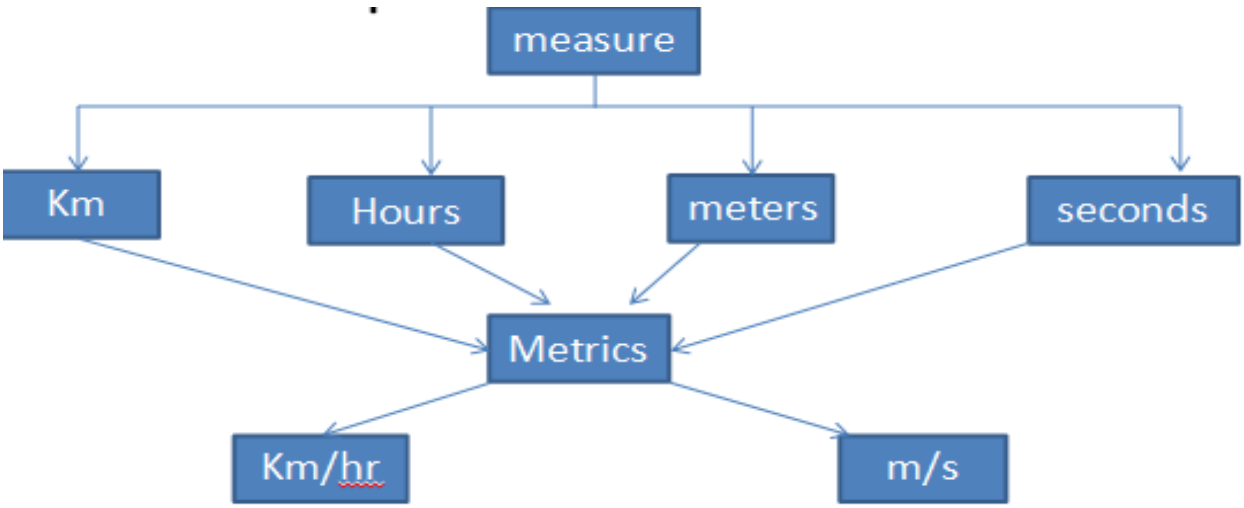
❖ Metrics and measurement of s/w testing

- Metrics : quantifies results
- Measuring progress of testing
- Measure how much testing completed and how much more time is needed.



- Measurement:

  --is the quantitative indication of extent , amount , dimension , capacity of some attribute of product



❖ Metrics Life Cycle

- 1. Analysis :

  --identify metrics to use

  --Define metrics identified

  --Define parameter for evaluating the metrics identified

- 2. Communication:

  --explain the need of metrics

  --educate testing team about the data points need to be captured for processing the metrics

- 3. evaluating :

  -- capture the data

  --Verify the data

  --calculating the metrics value using the data captured

- 4. reporting:

  --report with effective conclusion

  --distribute report to stakeholder

  --take feedback from the stakeholder

❖ Metrics category

- 1. Base metrics : derived from test cases development & execution --keep track of : --total number of test cases developed -- executed -- pass fail

Sr.no	Testing metric	Data retrieved during test case development and execution
1	Total no.of test cases executed	100
2	No.of test cases pass	65
3	No.of test cases failed	30
4.	Total no.of test cases unexecuted	35
5.	Total no.of defects	30

- 2. calculated metrics：

  --derived from data gathered in base metrics

  -- track by leader / mgr for test reporting

  Formulas for calculating metrics:

- --%age test cases executed :

  (no.of test cases executed)/total test cases *100

- %age test case not executed:

  no.of test cases unexecuted/total test cases *100

❖ Project metrics

- Assess status of project
- Track risk
- Evaluate project team ability
- Production rates are measured
- Errors uncovered during each phase are measure
- Minimize the development schedule by making adjustments.
- How different activities are progressing
- Track planned vs actual over time
- Man hours/test case executed
- Planned hours/actual hours
- Test cases executed/planned
- Test cases executed / defect found

❖ Productivity metrics

- Calculate how many “simple task” can be delivered by day
- “simple task” :amount of time required by a resource to deliver something
- Eg : 5 hours planned , 8 hours actual
- Productivity=

  ((planned effort/5)/actual effort)*8