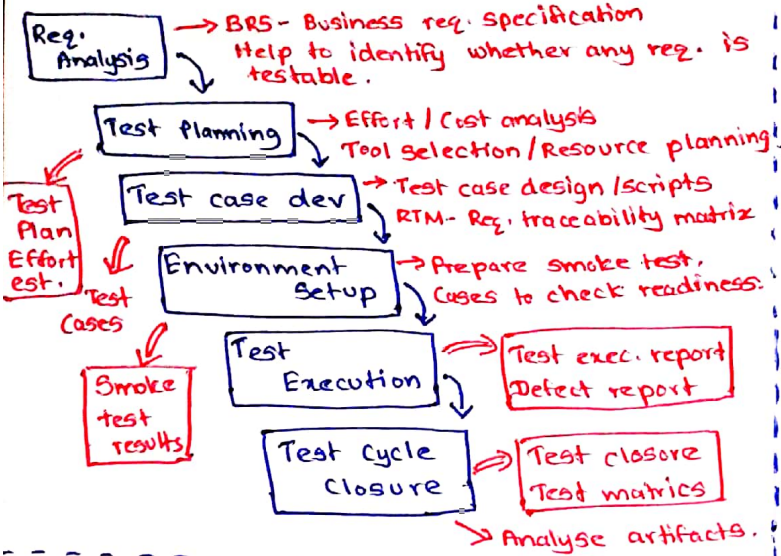
SOFTWARE TESTING LIFE CYCLE



WHAT ?

Process which assures all s/w engineering processes methods, activities and work items are monitored and comply against defined standards.

S/W QUALITY

Ensures customer satisfaction by offering all deliverables on

⇒ PREVENTION - Performance standards and ease of operation.

⇒ DEFECTION - s/w testing.

S/W TESTING

An activity to check whether the actual result match the expected result and to ensure the s/w system is error free.

Quality Assurance	Testing
Controls the testing process and verifies the s/w is able to work under given set of conditions.	concentrate on case stodying / implementation and evaluation.
Aim : Assure the quality	Aim : Control the quality
* Process oriented	* Product oriented
Main Focus : Quality control & meeting req.	Main Focus : System Inspect & bug finding.

WHY BUGS?

- Miscommunication / No communication
- Exponential growth in s/w complexity
- Programming errors.
- Dependencies among code modules / services.
- Changing requirements
- Time pressure
- Poorly designed / documented code.
- s/w development tools.

COMMON PROBLEMS IN S/W DEV.

01

- Poor requirements
- Unrealistic schedule.
- Inadequate testing.
- Misunderstanding about dependencies.
- Miscommunication.

* S/W Products are

(01) High complexity

(02) Invisible

(03) Opportunity to detect bugs are limited.

* SQA METHODS ARE DEVELOPED FOR

[01] CONTRACTUAL CONDITION

- commitments and conditions defined in the contract.
- defined list of functional requirements / budget / time schedule.

[02] CUSTOMER - SUPPLIER RELATIONSHIP

- continous co-operation and mutual support, review to enhance product quality with customer.

[03] REQUIRED TEAMWORK

- three factors to motivate
  - time
  - need for variety of specializations
  - the wish to benefit from professional mutual support.

[04] CO-OPERATION / CO-ORDINATION WITH OTHERS

[05] INTERFACES WITH OTHER SYSTEMS

[06] CONTINOUS WORKFLOW OF PROJECT DESPITE OF TEAM MEMBER CHANGES.

[07] CONTINOUS MAINTENANCE FOR EXTENDED PERIOD

- to ensure project's quality and success.

SOFTWARE QUALITY

(IEEE)

Computer programs, procedures, documentation & data pertaining to the operation of a computer system.

(ISO) ⇒ Four components of a s/w

- Computer programs
- Procedures
- Documentation
- Data necessary for operation.

SOFTWARE DEFECTS

s/w Errors	s/w Faults	s/w Failures
grammatical / logical (syntax) errors.	Incorrect functioning	Incorrect behaviour
• Made by dev.	• Not all faults are detected	• Observable.
• May or not be detected in coding / testing		

* CASES FOR S/W ERRORS

- Faulty requirement definition.
  - root cause of s/w errors.
  - incorrect/incomplete definitions.
  - un needed requirements.
- Client - developer miscommunication.
  - misunderstanding - instructions/graphics
  - lack of attension.
- Deliberate deviation from S/W requirements.
  - reuse previous code / last minute changes
  - unapproved enhancements.
- Logical design errors.
  - boundary conditions
  - erroneous algorithms / system states.
- Coding errors.
  - Syntax / Logic / Run-time
  - Tools / data selection.
- Non compliance with documentation and coding standards.
  - hard to understand during collaborati-ve development / replacement.
- Shortcomings of testing process.
  - Incomplete test plans
  - failure to detect / report errors.
  - failure to prompt / fix "
- UI and procedure errors.
  - unsatisfactory UI.
- Documentation errors.
  - meaningless error msgs. etc.

* S/W QUALITY DEFINITION.

- Degree to which a system / component or process meets specified customer expectations.

QUALITY ASSURANCE:

A planned, systematic pattern of all actions necessary to provide adequate confidence that an item / product conforms to established technical requirements.

QUALITY CONTROL:

A set of activities designed to evaluate the quality of a developed or manufactured product
* Reviews & Testing

QUALITY ASSURANCE	QUALITY CONTROL
- Start at req. gathering	- Start at dev. & testing
- Make sure that we do the right thing right way	- Make sure the results are what we expected.
- Process	- Product
- Applicable for entire LC	- Applicable for testing.
Focus: Building quality products	Focus: Detecting defects and testing for quality products.
- Preventive oriented	- Detection oriented.
- Relates to all products created by a process	- Relates to a specific product.

OBJECTIVES OF SQA ACTIVITIES

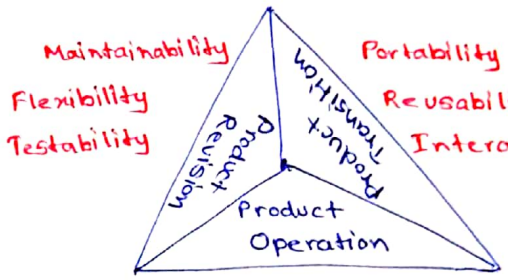
s/w development (Process oriented)	s/w maintenance (Product oriented)
- ensure that the s/w meet functional & technical req.	- same as above
- schedule / budget req.	
- greater efficiency	

SOFTWARE QUALITY FACTORS

NEED: To assure the full satisfaction of the users which include.
- all attributes of s/w
- aspects of the use (usability etc.)

CLASSIFICATION OF REQUIREMENTS INTO S/W QUALITY FACTORS

(01) McCALL'S CLASSIC FACTOR MODEL (11)



Usability Correctness Reliability
Efficiency Intergrity

* PRODUCT OPERATION

Requirements that directly affect the daily operation of s/w.

(01) Correctness.
- Functionality should match the specificat^n
- Output/Time between event & response.
- Completeness / Standards in coding & doc.
- Uptodateness / Availability of information.

(02) Reliability
- The extent to which the system fail.
- Service failure

(03) Efficiency
- Enhance the usages of system resources

(04) Intergrity
- Ability to withheld access to secured inf for unauthorized persons. (Security)

(05) Usability
- How easy to use
- UI / Workflow

* Product Revision

Requirements that affect the range of s/w maintenance

- CORRECTIVE MAINTENANCE
  Correction of faults / failures
- ADAPTIVE MAINTENANCE
  Adapting the current s/w to new features or customers without changing.
- PERFECTIVE MAINTENANCE
  Improvement / Enhancement of existing s/w with respect to issues.
- PREVENTIVE MAINTENANCE

(06) Maintainability

How easy / inexpensively maintenance can be done
Modular structure / Program documentation.

(07) Testability

Ability to verify requirements.

(08) Flexibility

Ability to customize the product according to demands.

* Product Transition

Adaptation of s/w to other environments and its interaction with other systems.

(09) Portability

How easily it can be adapted to run in a different execution environment.

(10) Re-usability

Ability to reuse significant component in another development.

(11) Interoperability

Ability to work with other s/w systems.

02 EVAN'S & MARCINAK FACTOR MODEL (12)

03 categories

- Design
- Performance
- Adaptation.

11 + verifiability
+ Expandability
- Testability

03 THE DEUTSCH & WILLIS FACTOR MODEL (15)

04 Categories

- Functional
- Performance
- Change
- Mgt.

11 + Verifiability
+ Expandability
+ Safety
+ Survivability
- Testability

THE COMPONENTS OF THE S/W QA SYSTEM

- Can be classified into 06 classes.

01 PRE-PROJECT COMPONENTS

To assure
- project commitments have been adequately defined. (resources / budget / development / quality schedule).
- plans have been correctly determined

⇒ MAJOR COMPONENTS
- CONTRACT REVIEW
- DEVELOPMENT & QUALITY PLANS

* CONTRACT REVIEW
- Clarification of customer's requirements.
- Review of project's schedule / resource req.
- Evaluation of professional staff capacity.
- Evaluation of customer capacity.
- Evaluation of development risks.

Proposal Draft Review + Contract Draft Review ⇒ CONTRACT REVIEW

* DEVELOPMENT & QUALITY PLANS

↓ Project ↓ QA activities

- Plans are revised to reflect the changes that occurs.

⇒ Main issues treated in development plan.
- schedule
- manpower / H/W resources
- risk evaluations
- organizational issues
- project methodology / development tools
- s/w reuse plans.

⇒ Main issues treated in quality plan
- quality goals
- criteria for starting / ending each stage
- lists of reviews / tests and other scheduled verifications & validations.

---

02 S/W PROJECT LIFE CYCLE COMPONENTS

Composed of two stages

- Development Life Cycle Stage
- Operational / Maintenance Stage.

⇒ MODELS OF DEVELOPMENT
- SDLC Model
- Prototyping Model
- Spiral Model - iterative / user involved
- OO Model - reuse of s/w.

* **VERIFICATION** – Are you building the Product Right?

The process of evaluating a system/component to determine whether the products of a given dev. phase satisfy the conditions imposed at the start of that phase.

Concerned : System is error free
Well engineered.

* **VALIDATION** – Are you building the Right product?

The process of evaluating a system/component during or at the end of development to determine whether it satisfies specified req.

Concerned : Customer's actual need.

* **QUALIFICATION** – Operational aspects

The process used to determine whether a system/component is suitable for operational use.

⇒ **MAJOR COMPONENTS**

- REVIEWS
- EXPERT OPINIONS
- S/W TESTING
- S/W MAINTENANCE
- Assurance of the quality of subcontractor's work and supplied parts.

* **REVIEWS**

A process/meeting during which a work product or set of work products is presented to project personnel, managers, users, customers or other interested parties for comment or approval.

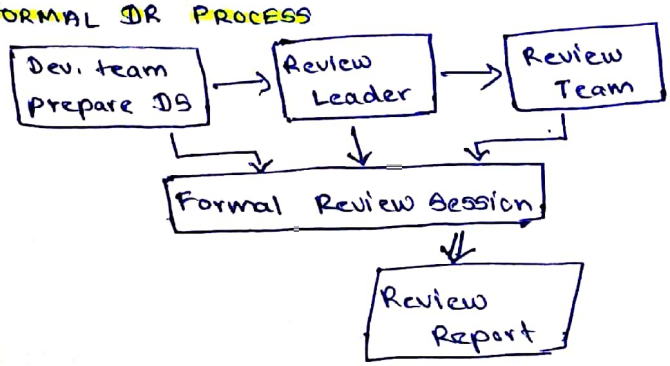
FORMAL DESIGN REVIEWS (DR)

- only reviews that are necessary for approval of the design product.
- may be conducted at any milestone.

PEER REVIEWS

- list of detected faults
- defects summary
- statistics.

FORMAL DR PROCESS



PEER REVIEWS

- INSPECTIONS
- WALKTHROUGHS

INSPECTION

Used to verify the agreement of the product with defined standards and requirements.

⇒ Participants : Moderator (Scribe)
Author
Coder/Implementer
Designer
Tester

WALKTHROUGH

Informal process

⇒ Participants : Coordinator (Scribe)
Standards enforcer
User representative
Author (Presenter)
Maintenance expert.

Inspection	VS	Walkthrough
- Formal		- Informal
- Corrective action		- Comments on the document reviewed
- Findings are inco-operated into efforts to improve dev. methods. ∴ more significant		

* **EXPERT OPINIONS**

- Support quality assesment efforts by introducing additional external capabilities
- Getting outside expert opinion may be useful.
  - S/W testing
  - S/W maintenance
  - Assurance of quality of sub contractor's work.

---

03 **INFRASTRUCTURE COMPONENTS FOR ERROR PREVENTION.**

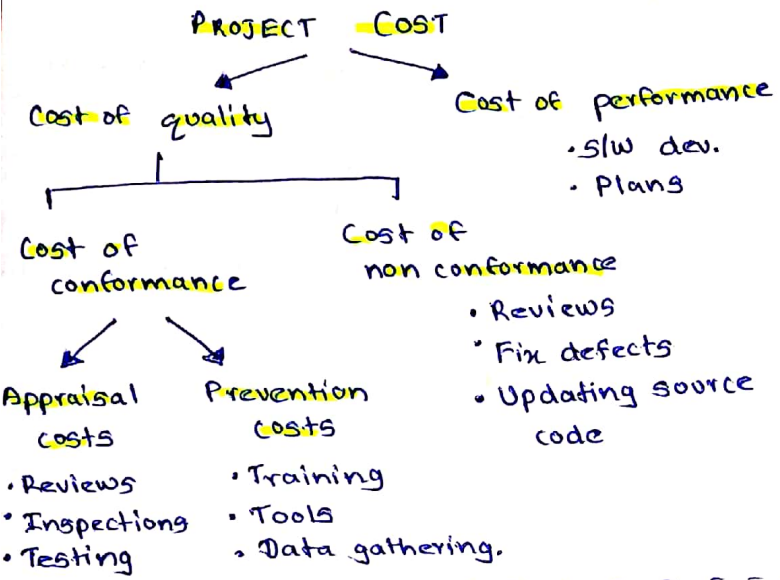
Main objective of these components are to eliminate/reduce the rate of errors.

⇒ **MAJOR COMPONENTS**

- Procedures and work instructions
- Supporting quality devices
- Staff training/retraining/certification
- Preventive & corrective actions.
- Configuration management.
- Control of documentation/quality records

04 MANAGEMENT SQA COMPONENTS

- Support managerial control of s/w projects and maintenance services.
- Prevent schedule / budget failures.
- Includes
  - Project progress control
  - s/w quality matrics / costs.



05 SQA STANDARDS / SYSTEM CERTIFICATION AND ASSESMENT

⇒ MAIN OBJECTIVES

- Utilization of internal professional knowledge
- Improvement of coordination of organizational quality system with other org.
- Assesment of achievements of quality system according to a common scale.

⇒ STANDARDS

- Quality management standards
- Project process standards

06 THE HUMAN COMPONENTS

⇒ MAIN OBJECTIVES

- Initiate and support the implemcntation of SQA components.
- Detect deviations from SQA procedures.
- Suggest improvements.

⇒ INCLUDES

Managers, testers, developers etc.

SOFTWARE TESTING

TYPES OF TESTING

- MANUAL TESTING
  - manually execute test cases.
  - helps to find bugs.
  - done by an experienced tester.
  - expensive / requires more effort.
  - does not require knowledge of a tool.
  - prone to human error.
  - use test plans / cases / scenarios.
- AUTOMATED TESTING
  - Uses a s/w to test the product.
  - write scripts and use.
  - more reliable / robust.
  - quality depends on how well the scripts are written
  - saves time / improve accuracy
  - less cost.

Any where a large number of users can access the s/w simultaneously should be automated.

Eg: Login / Registration
DB connections

* When to automate?
  - High risk business cases
  - Repeatedly executed test cases.
  - Time consuming test cases.

* FUNCTIONAL TESTING

- testing done against the business req. or the application.
- verifies each function with the SRS.
- each functionality is tested by providing appropriate input and verifying the output and comparing the actual result with the expected result.
- UI / API / DB / Security etc.
- can be done manually / automated.

Eg: Unit, System, Sanity, Integration, Smoke, Regression

* NON FUNCTIONAL TESTING

- Check for non-functional aspects performance / usability / reliability
- conducted using automation tools.
- expressed in a testable way.
- has a greater influence on customer

Eg: Performance, Security, Recovery, Volume, etc.

Functional	vs	Non-functional
• Can be done manually		• Hard to perform manually.
• Based on requirements		• Based on expectations
• Describe what the product does		• Describe how the product works.

Re-test all modules	Re-test selected modules	Re-test prioritized cases.
• very expensive	• re-usable test cases	• depending on impact/ freq. of use etc.
• takes a lot of time & effort.	• obsolete test cases.	• saves time and cost.
	• can be selected	

UNIT TESTING

- Individual unit or a component of a S/W are tested. (function/method/procedure/module)
- Purpose is to validate that each unit performs as expected.
- Cheap to automate.
- A white box testing technique.
- Saves time and money.
- Identifies bugs during development stage Hence reduce cost.

Eg: JUnit TestNG
PHPUnit nUnit

⇒ **MOCK OBJECTS**

When a part of the code is not completed mock objects fill the missing parts.

INTEGRATION TESTING

- Modules are integrated logically and tested as a group.
- Ensures/verifies different modules or services can work together.
- Expensive
- Focus mainly on interfaces and flow of data.

Big Bang Approach

- All integrated together at once and then tested.

Incremental Approach

- Done by joining two or more logically related modules & others are added incrementally.
  - Top down - LL → HL
  - Bottom up.

REGRESSION TESTING

- Done to verify that a code change in the software doesnot impact the existing functionality of the product.
- Ensures that everything works fine still after doing changes.

Eg: Selenium testsigma
Katalan

SMOKE TESTING - Build verification confidence

- Determine whether the deployed build stable or not..
- Purpose: To confirm whether the QA team can proceed with further testing
- Designed to expose errors in the build
- usually done manually.

SANITY TESTING - Tester acceptance testing

- Performed after recieving a software build with minor changes in code or functionality to ascertain the bugs have been fixed & no further issues are introduced due to these changes.

Features:
- Follows narrow and deep approach
- Not scripted.
- Subset of regression testing
- Used to verify requirements of end users are met.

* Both smoke and sanity testing avoid wasting time and effort.
* Both can be executed manually or automated.

Smoke testing	vs	Sanity
Performed to ascertain that critical functions of the S/W are working fine.		Used to check the new functionality / bugs have been fixed.
Goal: Verify stability		Goal: Verify rationality
Done by developers and testers		Done by testers
Documented / Scripted		Doesn't document or scripted.
Verifies the entire system.		Verifies only a particular component.

ACCEPTANCE TESTING

- Performed to verify/accept s/w before moving to the production environment.
- Test whether it is in compliance with the business requirements and verifies if it meets the required criteria for delivery to end users.
- Done in final phase.
- Kind of black box testing.

⇒ REASONS TO CONDUCT

- Requirement changes may not be effectively commonicated to the developer
- Difference in actual client req. to that is developed by developer according to their understanding.

- User acceptance
- Business acceptance
- Alpha testing → dev. env. → internal employee
- Beta testing → released to limited set of users.

SOFTWARE TESTING STRATEGIES

CODE INSPECTION

- Main purpose isto find defects and spot process improvements if any.
- Led by a trained moderator who's not the author.
- Conducts a formal meeting.
- Peer examination
- Follow up process to makesure that the corrective action was taken.

CODE WALKTHROUGHS

- Form of peer review
- Led by a programmer
- Ask questions and spot possible errors dgainst development standards and other issues.
- Informal.

STRATEGIES

"Big-Bang" testing

- Test the s/w as a whole unit once ~~and~~ the entire thing is finished
- Small and simple s/w projects.

Incremental testing

- Test the s/w piecemeal

Unit test
↓
Integration
↓
System

- Good for very big/complex projects.
- Yeilds higher % of

* STUBS & DRIVERS FOR INCREMENTAL TESTING

- s/w replacement simulators required for modules not available when performing tests.

In top-down ⇒ STUBS (Dummy Modules)
- replaces unavailable lower level module to the module tested.

In bottom-up ⇒ DRIVER
- substitute module that activates the module tested.
- driver passes and accepts test data on the tested module.

SOFTWARE TEST CLASSIFICATION

(CONCEPT)

Black Box Testing
- Focus only on the output
- less expensive (functionality)

White Box Testing.
- examine the internal paths of calculations in order to identify bugs (structural)

(REQUIREMENT)

- McCall's classification - done earlier

⇒ TEST SCENARIOS

A high level description of the functionality we need to test.

⇒ TEST CASES.

The condition which we test for any particular scenario.

⇒ BUGS / DEFECTS

- Output of testing effort
- Any behavior of the system which is not according to the requirements and any behavior which impacts negatively to the system.

⇒ BUG / DEFECT LIFECYCLE

- Cyclic process which a defect follows through during its life time.
- Begins when a tester logs a bug and ends when he decides to close it after thorough verification.

WHITE BOX TESTING.

* Data processing and calculation correctness tests.

- large number of possible processing paths and multitudes of lines of code possible.

Two alternative approaches

PATH COVERAGE

- Try to cover all possible paths
- Percentage of possible paths of S/W processing activated by the test cases.
- Different paths → Conditional Statements
- For high risk S/W modules

LINE COVERAGE

- Try to cover all lines in the source code.
- Every line of the code should be executed atleast once during the process of testing.
- Cover each block with a minimum number of test cases as possible

$$LC = \left(\frac{\#\text{ of statements executed}}{\text{total \# of statements}} \right) * 100$$

McCABE'S CYCLOMATIC COMPLEXITY MATRICS

- Used to measure the complexity of a program
- A quantitative measure.
- Determines the maximum number of independent paths to achieve full line coverage.

⇒ The cyclomatic matrix [V(G)]

$$V(G) = R$$
$$V(G) = E - N + 2$$
$$V(G) = P + 1$$

R - # of regions
E - # of edges
N - # of nodes
P - # of decisions (nodes with more than one edge leaving)

V(G) < 5 → Simple Program

SOFTWARE QUALIFICATION TEST

Determine whether S/W development responded positively to questions reflecting a specific set of criteria.

- Code structure instructions / procedures
- Coding style
- internal program documentation

MAINTAINABILITY TESTS

- Tests how easy it is to maintain the system.
- Special features such as those installed for
  - detection of cause of failure
  - module structure (adaptat^n / improvements)
- Done on already developed S/W
- Testing done on enhancement, change and migration cycle of a S/W after deployment is known as maintainability testing.

MAINTENANCE STRATEGY TYPES

01 CORRECTIVE MAINTENANCE

- implemented after a defect has been detected.
- measured in terms of time taken to diagnose and fix problems identified within that system.

02 PERFECTIVE MAINTENANCE

- Enhancements
- Measured in terms of effort taken to make required enhancement to the system
- Changing existing functionality by refining, adding / deleting features.

03 ADAPTIVE MAINTENANCE

- Infrastructure of the S/W
- Adapting to changes in the environment. (O/S, H/W, platform)
- Measured in terms of effort required to make required adaptations to that system.

04 PREVENTIVE MAINTENANCE

- Carried out before any breakdown/failure at predetermined intervals or according to prescribed criteria.
- Aimed to reduce failure to risk.
- Refers to actions to reduce future maintenance costs.

REUSABILITY TESTS

- Determines whether packaging and documentat^n of program/modules that is listed to reuse conf-orms to the procedures demanded in the reusable S/W library.

BLACK BOX TESTING

- Identifies bugs only according to the mal-func. of software as revealed from its outputs. (functionality).
- Equivalance classes for output correctness tests.
  - Select test cases intelligently from a pool of test cases.

Equivalance class partitioning

- divide all possible test cases into groups where the system behav-ior will be the same.
- pick only one from each partition, make the assumption that if the selected value passes, all the values of that partition passes

Boundary value analysis

- Test the boundaries between the partitions.

* REVISION FACTOR TESTING CLASSES

- Assures the s/w package's success, long service and its use for longer populat^ns.
  - Maintainability tests
  - Flexibility tests
  - Testability tests

* TRANSITION FACTOR TESTING CLASSES

- Transition features such as operative, with minor adaptations in different environments, permits interfacing with other s/w are required.
  - Portability tests
  - Reusability tests
  - Interoperability tests

SOFTWARE TESTING IMPLEMENTATION

THE TESTING PROCESS

Main issues with the test implementat^n

Test effectiveness

- Reduction of percentage of undetected errors remaining in the s/w

Test efficiency

- Performance of tests with fewer resources

TEST EFFECTIVENESS

- How effectively testing is done / goal is achieved that meets customer requirements.
- Starts at the begening of development.
- Defects can be valid or invalid.
  - valid ← system doesn't function according to the req. spec. → Calculated as the % of # of valid defects divided by the sum of defects injected & defects escaped
  - invalid ← Misunderstanding requirements → wrong test cases
    - * change in req. not updated in test cases
    - * deviations due to external factors, env, network issues. → Can be ignored

---

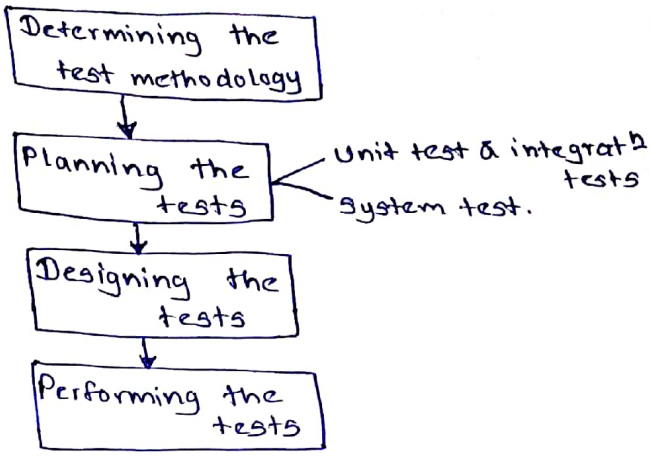
TEST EFFICIENCY

- The most efficient way to utilize your resources to achieve the organizational goals.
- An internal process for an organization that evaluates the utilization of resources to develop a s/w product.

↓

Calculated as % of number of alpha(α) testing defects divided by sum of α testing defects + β testing defects.

TESTING PROCESS



* DETERMINING THE TEST METHODOLOGY

- Selecting the appropriate, required s/w quality standards
- Determining the test strategy.
- Level of quality standard selected mainly depend on the characteristics of the s/w.

* TEST PLANNING

Includes unit tests, integration tests and system tests

⇒ MAIN ISSUES

* What to test?
  - Feasibility to test everything is very limited
  - Which modules to be unit tested?
  - " " " " integrated?
  - For system: ignore low priority modules?
  - Prioritize based on Damage Severity & s/w Risk level
    - Risk level ↑ Probability of failure.
    - Severity Level ↑ Damage if fails.
* Which types of sources are to provide test cases?
  - Real life or synthetic test cases.
* Who performs the test?
  - dev. team / independent test teams outsource etc.
* Where to perform the test?
  - Unit / Integration → dev's site
  - System → client site.
* When to terminate the test? (System)
  - (01) The completed implementation route } Perfection Approach
  - (02) The mathematical models application route : % of undetected errors.
  - (03) The error seeding route.
    - errors are seeded (hidden) in the tested s/w prior to testing.
  - (04) Dual independent test teams route.
  - (05) Terminat^n after resource peter out.

⇒ DOCUMENTATION

The planning stage of s/w system test. is commonly documented in a (STP) "Software Test Plan"

Main Sections of STP

- Scope of the test
- Testing environment
- Test details (for each test)
- Test schedule including time estim-ates

TEST DESIGNING

- Designing is carried out based on STP.
- The products of this stage.

s/w test procedure document

- detailed design & procedure for each test case

Test case file document

- test case / db file

→ **STD**

(Software Test Description)

Main Sections of STD

- Scope of the tests
- Testing environment
- Testing process
- Test cases
- Actions to be taken in case of failure.
- Procedures to be applied according to the test results summary.

TEST IMPLEMENTATION

Consists of

- a series of tests (run according to
- correction of detected STD procedures) errors.
- re-tests (regression tests)

↑ Conducted to verify that detected errors have been properly corrected.

⇒ DOCUMENTATION

STR

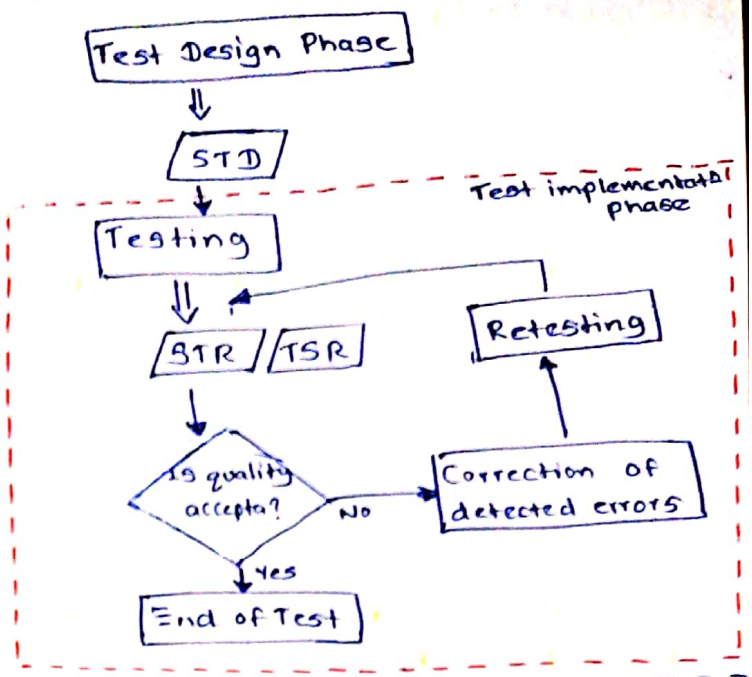
Software Test Report

- The results of individual tests and retests

TSR

Test Summary Report

- The summary of set of planned tests.



TEST CASE: DESIGN

A test case is a

- documented set of data inputs
- operating conditions required to run a test item
- expected results of the run.

Tester runs the program for the test item accordingly. Compare the actual results vs the expected results.

Obtained results agree with expected results

- no errors present
- error not identified

Obtained results do not agree

- Error identified

TEST CASE: DATA COMPONENTS

Numerical
Alphabetical
Error message

} Types of expected results.

TEST CASE : SOURCES

- Real Life cases
- Synthetic / simulated cases.
  - cover all possible operating conditions & parameters.

Real Life test cases	Synthetic Test cases
• Low effort for preparat°	• High effort for preparat°
• High size of test case file	• Relatively small test case file
• High effort to perform test cases (low efficiency)	• Low effort to perform test cases (High efficiency)
• Relatively low probability of error detection	• High probability of error detection

STRATIFIED SAMPLING

Sample is divided into groups and random samples taken as required.

↓ # of repetitions → ↑ coverage of less frequent and rare conditions

AUTOMATED TESTING PROCESS

- Test Planning
- Test Design
- Test case preparation
- Test performance
- Test log and report preparation
- Retesting after correction of detected errors. (regression)
- Final test log / report preparation.

TYPES OF AUTOMATED TESTING

[*] CODE AUDITING

- Automated qualification testing
- code auditor checks compliance of code to specified standards and procedures.
- Auditor's report ⇒ deviations from standards and summary of findings.

[*] COVERAGE MONITORING

- Produce reports about the line coverage, achieved by implementing a given test case file.
- A measurement of how many lines, statements / code blocks are tested.
- A good tool for whitebox testing
- Measured as a percentage.

[*] FUNCTIONAL TESTS

- Tests for functionality of components.
- Test cases are recorded in the test case database and executed

[*] LOAD TESTS

- Based on simulated scenarios of maximal load situations the s/w will confront.

⇒ TEST MANAGEMENT

The main objective of these automated tools are to provide comprehensive follow up and reporting of testing and correction of detected errors.

- Features ⇒
  - Test plans / results / correction and follow up.
  - Test execution
  - Maintenance and follow up.

ALPHA TESTING (α)

- A kind of acceptance testing where customer tryout the s/w at developer site.
- The errors identified are expected to include the errors only the real user can reveal.
- More difficult to organize
- Helps to simulate real-time user environment before it is sent to beta testing.
- Helps to identify whether it is eligible for beta testing.

BETA TESTING (β)

- Performed by real users in a real environment.
- Helps to provide authentic feedback from real users.
- A form of external user acceptance testing.
- A group of selected customers are given to use the s/w and asked to report the encounted errors they find.
- Direct feedback from customers.

INDUSTRY SESSIONS

WHAT IS A TEST TOOL ?

- A s/w product that supports one or more test activities such as
  - Planning and control of test
  - Test specification
  - Building files / data used for test.
  - Test execution
  - Test result analysis.
- Anything that makes testing faster / easier / more accurate.

OUTCOMES OF A TESTING TOOL ⇒

- Time transformation
- Xtreme automation
- Smart platforms & in-life intelligence

WHAT IS TEST AUTOMATION ?

Involves automating a manual process which already in place and that uses a formalized testing process.

WHY AND WHEN TO AUTOMATE ?

- Frequent regression testing
- Repeated test case execution
- Increasing testing scope coverage
- Faster feedback to the developers
- Reduces human effort.
- Test same application on multiple env.

WHAT IS A TEST FRAMEWORK ?

A set of guidelines which will be followed during automation scripting.

⇒ Benefits : increase code re-usage
higher portability
reduced script maintenance

- Data-driven / Keyword-driven / Table-driven
- Hybrid test automation.

WHAT IS A WEB DRIVER?

A web automation framework that allows you to execute your tests against different browsers.

Eg: TestNG
- → A testing framework for JAVA
- → Main features
  - Annotation support
  - Support for parameterized / data-driven testing.
  - Support for multiple instances of same test cases.
  - parallel execution.

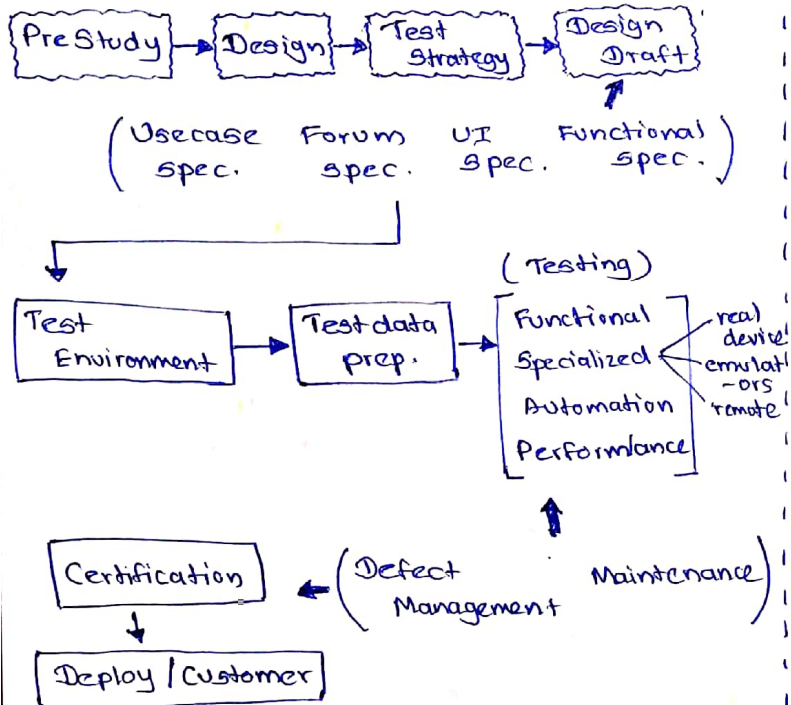
* Coding part in videos ⇒ check

INDUSTRY SESSIONS

MOBILE APPLICATION TESTING

- Mobile testing types (Manual / Automated)
  - Functional - validate the functionality of the app.
  - Usability - user experience / interactions / themes.
  - Compatability - multiple devices / OS / browser screens etc.
  - Integration & System testing
  - Performance testing
  - Regression testing - new feature.

MOBILE TESTING METHODOLOGY



TYPES OF MOBILE APPLICATIONS

- NATIVE - Platform specific
  Rich UI
- HYBRID - Multiple platforms
  Leverages device capabilities
- WEB - Cross platform
  Instant updates

NATIVE APPS

Benefits	Cons
• Best performance • Recieve complete support • Interactive / Smooth to use. • Allow developer to access the full features of the OS. • UX is superior.	• Difficult programming languages for developer • Initial expenses are high. • Not the best for simple applications.

WEB APPS

Benefits	Cons
• Relatively easy to use • Can be built for all platforms. • Less expensive upfront • Don't adhere to standard OS protocols.	• Smaller scope to device features and H/W • A browser is required • Poor discoverability • Slower / Less responsive • Less interactive.

HYBRID APPS

Benefits	Cons
• Doesn't require a web browser • Can access to device's internal API / H/W • One codebase.	• Much slower than native • Dependent on a third party platform. • Cost is high due to customization for the apps.

MOBILE TESTING STRATEGY

01 User Experience
Test cases should be designed from the end user's perspective.

02 OS
Test cases should be designed to support all available versions of OS.

03 Screen sizes.
Simulators / Emulators can be used.

04 Network Testing
The same set of test cases are to be executed for different networks as per requirement (2G/3G/4G...)

05 Different types of Mobile Apps.
There should be different sets of test cases / test efforts for each type.

INDUSTRY SESSIONS

INTRODUCTION TO WEB SERVICES

WHAT IS A WEB SERVICE?

- A method of communication between two electronic devices over the internet.
- A software function provided at a network address over the web or 'cloud'.

WEB SERVICES

REST (Representational State Transfer)	SOAP (Simple Object Access Protocol)
• Data available as resources.	• Data available as services
• Can return data in multiple formats (JSON/XML)	• Fully described using WSDL
• Only works on top of HTTP	• Consumes a lot of bandwith
• Social Media	• Uses only XML
• Social Networks	• Financial Services
• Web chat	• Payment gateways
	• Telecommunication

WEB SERVICE ARCHITECTURE

Individual roles of each web service actor

- SERVICE PROVIDER
  - implements the service and make it available on the internet.
- SERVICE REQUESTOR
  - consumer
- SERVICE REGISTRY
  - Logically centralized directory of services

Emerging web service protocol stack

- SERVICE TRANSPORT
  - transporting msgs between applications
- XML MESSAGING
  - encoding msgs in common XML format
- SERVICE DESCRIPTION
  - Describing the public interface to a specific web service (WSDL)
- SERVICE DISCOVERY
  - Centralizing the services into a common registry (UDDI)

INTRODUCTION TO MICROSERVICES.

- An architectural style that arranges the application as a set of loosely coupled, self contained, scalable set of services.
  - A variant of SOA.
  - Each service is self contained and implements a single business functionality.
  - Multiple services work together as a system.

Advantages ⇒

- Independent development
  - can be easily implemented to accomplish individual functionality
- Independent deployment
- Fault isolation
  - failures in one may not affect other
- Mixed technology stack
- Granular scaling
  - can scale only particular service based on the demand

TESTING WEB SERVICES

- Approach is the same
- Based on API documentation', design test cases.
  - Identify requirements
  - Prepare high level test cases
  - document detailed test cases
  - start testing using suitable tools
  - defect tracking

TOOLS : SOAPUI - Both SOAP & REST
Postman - Only REST
VREST - Support REST APIs.

HTTP METHODS

GET : Used to extract information from a given server using URI.

POST : Send data to server.

DELETE : Remove all current representations of the target resources.

PUT : Replaces all current representation of the target resources.

PERFORMANCE TESTING

- To demostrate that the system functions to the specifications with acceptable response time.
- A non functional testing technique performed to determine the system quantitative parameters in terms of responsiveness under various workload.
- Measures the quality attributes of the system such as scalability, reliability and resource usage.
- Mainly intended to :
  - Assess risks
  - Make informed decisions
  - Plan for future
  - Find out the bottlenecks
- WHY?
  To demostrate that the system meets req. for transaction throughput & response time simultaneously.

- Speed ⇒ Respond quickly enough?
- Scalability ⇒ Handle expected user load & beyond?
- Stability ⇒ Stable under expected & unexpected user loads.
- Confidence ⇒ Users will have a positive experience?

* Common Performance Problems

- Long load time
- Poor response time
- Poor scalability
- Bottlenecking
  - CPU utilization
  - Memory "
  - Network
  - OS limitations
  - Disk usages.

Performance Testing Approach

- Review applicatn architecture and design
- Production configuration & Interfaces
- Transaction volume & performance goals.
- Develop performance strategy and plan

→ Strategy and Planning

- Determine optimal performance tools
- Architect performance lab environment.
- Conduct tool(s) POC for suitability

→ Tool selectn and Lab design

- Performance testing objectives, design and scripts.
- Types of performance test & test results.
- Current performance and scalability baseline

→ Performance and Baseline

- Design and conduct performance tests.
- Analyze results & determine root cause of issues.
- Present findings, reporting and solutions recommended

→ Performance testing & Diagnostics

Types of Performance Tests

01 Load Test

- Makesure the slw product can handle the expected user loads.
- Simulates real world traffic and activity for the app. under test.
- Throughput / Stability / Responsiveness
- Identify bottlenecking under expected

02 Stress Test

- Understand the breaking points of the system with huge load.
- Used to determine stability of the application under intense conditions.
- With increased load for an extended time, the system is subjected to stress
- The behavior of the application is assesed under stress.

03 Scalability Test.

- To makesure the application can scale up to a level without deviating from application goals.
- By loading the system beyond the full load in definite increment till the system response becomes unacceptable, the scalability can be calculated.
- This will reveal how long the existing system can sustain based on projected load growth.

04 Reliability Testing / Endurance

- To makesure there's no harm to the system while running for longer periods.
- Determines how long the system can sustain optimum performance levels under expected loads.
- This places on steady / consistent workload on the application for a considerable period of time.

05 Volume Test

- To makesure no other performance deviations due to the volume of the database.
- Testing a slw application with a large amount of data to be processed to check efficiency.
- Main goal is to monitor the performance of the application under varying database volumes.

Pre-Requisites For Performance Testing

- Quantitative, Measurable, Relavent, Realistic and Achievable requirements.
- Stable system - freedom from functional defects.
- Realistic test env - similar to producth env.
- Controlled " "
- Performance testing tools.

TOOL IDENTIFICATION & EVALUATION

Objective is to ensure that the tools identified supports and helps in measuring the performance test goals.

⇒ Assesment against selection criteria.

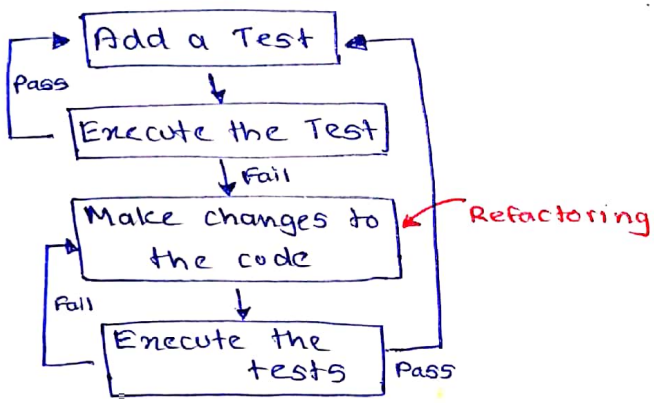
- Customer preference
- Test requirements
- S/W and H/W used
- Known tool limitations
- License cost.
- Protocol support
- Vendor support
- Efficiency of tool.

INDUSTRY SESSIONS

TEST DRIVEN DEVELOPMENT (TDD)

- A S/W development process that relies on the repetition of a very short dev. cycle.
- Test cases are written before code.
- Each iteration starts with a set of tests written for a new piece of function.
- Relates to test first programming concepts.

TDD PROCESS



BENEFITS ⇒

- Less debug time
- Refactoring allows to improve the design of the code.
- Low level regression test suit.
- Reduce the cost of the bugs.

DRAWBACKS ⇒

- Developer consider as a waste of time.
- Test become a part of maintenance overhead.
- Rewrite the tests when requirements are changed.

BEHAVIOR DRIVEN DEVELOPMENT (BDD)

- An extension of TDD
- Write the test first and add the application code.
- The major difference to TDD are
  - tests are written in plain descriptive English.
  - tests are explained as a behavior of application and are more user focused.
  - Using examples to clarify req.

⇒ FEATURES

- Shifts from think in "tests" to think in "behavior".
- Collaboration between stk. holders, BA, QA team and developers.
- Easy to describe / simple language.
- Driven by business value.
- Extends TDD by utilizing natural language that non technical stk. holders can understand.
- BDD frameworks such as Cucumber acts as the bridge between business and technical language.

- Can be utilized for unit level test cases and for UI level test cases.

STANDARDS / CERTIFICATION / ASSESMENT

- provide a way to benchmark to see the level of quality in org / product.
- designed to ensure the companies meet the minimum requirements to become a fundamental part of the industry.

BENEFITS

- The ability to make use of the most sophisticated and comprehensive professional methodologies & procedures.
- Better understanding and co-operation between users of same standards.
  - team members and teams
  - developer & external parties.

STANDARDS

Constantly updated SQ standards, used internationally by professionals & SQ managers.

Eg: IEEE / ISO

CERTIFICATIONS

Certification through independent auditors, valid till next evaluation.

ASSESMENT

- Provide professional support, tools for self assesment.
- Organization-created assesment programs have detailed documentation which act as instruction manuals
  Eg: CMM
  ISO/IEC Std. 15504

⇒ ISO/IEC 9126

- Defines a quality model for S/W product evaluation.
- Objective: Address some of the human biases that can adversely affect the delivery & perception of S/W
- Tries to develop a common understanding of project's objectives and goals.
- An extension of defining a set of S/W quality characteristics.

↓ replaces

⇒ ISO/IEC 25010:2011

- Security and compatability were added as the main characteristics.

CLASSIFICATION OF SQA STANDARDS

Quality Management Standards	Project process Standards
⇒ What to achieve	⇒ How to perform
• Mgt. of S/W development	• S/W development
• Assuring supplier's S/W quality and assesing it's S/W process capability.	• Assuring the quality of the software product.

QUALITY MANAGEMENT STANDARDS.

⇒ Includes certification and assesment methods.

⇒ Focus on organization's SQA system/infrastructure/req.

⇒ Organizations can continously ensure that their S/W products achieve an acceptable level of quality.

* CERTIFICATION STANDARDS

ISO/IEC 90003:2014 - buy/sell/operate and maintain S/W & related services.

↓

ISO 9001

- provide guidelines

* not a certification

ISO 9000

- Certification serves as a reference for contract between independent parties.
- Specifies guidelines for maintaining a quality system.
  ⇒ guidelines for repeatable and high quality product development.
  ⇒ addresses organizational aspects.
  ⇒ responsibilities, reporting, procedures, processes and resources for implementing quality mgt.
- A set of guidelines for production process
  - not directly concerned about the product

- ISO 9001 → design/development/production servicing of goods.
  * only certification.
- ISO 9002 → production (existing)
  Eg: manufacturing industries.
- ISO 9003 → installation & testing (finished product)

WHY GET ISO 9000?

- Increase the confidence of customers
- Make the dev. process focused and cost effective
- Increase marketability
- Reduce operational expenses
- Better mgt. control.
- Improve internal communication
- Attractive to investors.

HOW TO GET?

01 Application Stage
↓
02 Pre-Assesment
↓
03 Document Review & Adequecy audit
↓
04 Compliance audit
↓
05 Registration
↓
06 Continued Surveillance

ISO 9001/2008

Addresses all products and services in all industry segments.
Requires 08 principles to be implemented
Replaced by ISO 9000/2015

- customer focus
- leadership
- engagement of people
- process approach
- improvement
- relationship mgt.
- evidence based decision making.

FEATURES OF ISO 9001

- All documents concerned should be properly managed / authorized and controlled.
- Proper plans should be prepared and progress against this should be monitored.
- Important documents independently checked and reviewed for effectiveness and correctness
- product should be tested against specification.

SHORTCOMINGS

- Not fault proof
- Individual skills / experience is significant
- Doesn't guarantee results.

* ASSESMENT STANDARDS

CAPACITY MATURITY MODEL (CMM)

Capability.

- developed by SEI
- establishes a framework for continous process improvement.
- 05 level evolutionary path.
- helps organization to
  - Improve the quality
  - realize adaptation of CMM for significant benefits.
- measures the maturity of the org.
- can be used in two ways

Capability Evaluation

- s/w process capability of an org.
- helps in selecting a contractor.
- indicates the likely contractor performance

s/w process assesment

- asses current process.
- suggest ways to improve the process capability
- For purely internal use.

CMM MATURITY LEVELS

Optimizing (5)
↑
Managed (4)
↑
Defined (3)
↑
Repeatable (2)
↑
Initial (1)

CMM DRAWBACKS

- Activity based approach
  - considers only the completion of a specific activity.
- Importance on paper-work
  - mgt. of time and effort.

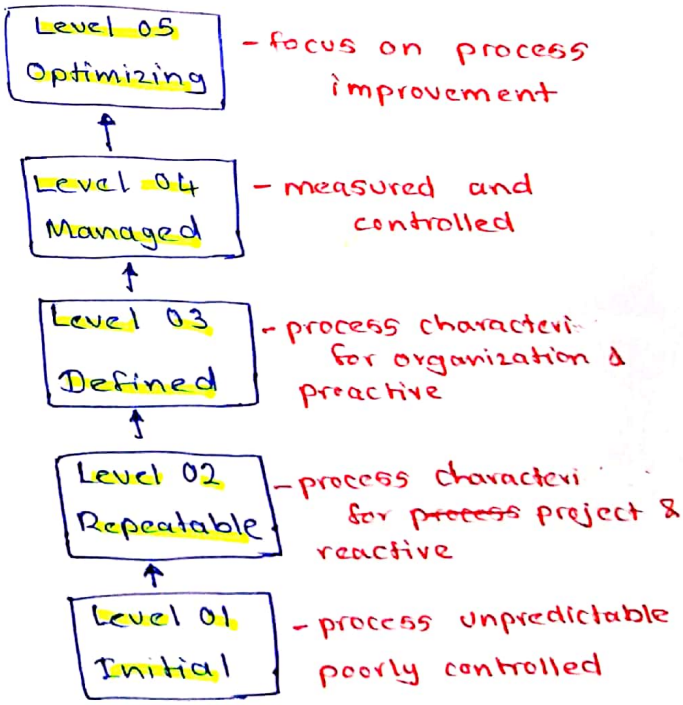
CAPABILITY MATURITY MODEL INTEGRATION (CMMI)

- less specific
- applicable for H/W, S/W and service
- enables org. to measure, build and improve capabilities and overall performance.

PRIMARY OBJECTIVE / GOAL

Creation of reliable environments where products, services and departments are pro-active, efficient and productive.

- Quality services / products
- Increase customer satisfaction
- Increase value of stk. holders.
- Achieve industry wise recognition.
- Building a larger market share.



COMPARISON OF CMMI & ISO

CMMI	ISO
• S/W Engineering practices	- Customer satisfaction
• Program / project mgt. practices.	- Process discipline across entire organization
• increasing maturity levels.	- continous improvement
• appraisals when going to next level.	- principled mgt.

S/W PROJECT PROCESS STANDARDS

- Focus on the methodologies for carrying out S/W dev, maintenance projects.
- Define the steps to be taken
  - design doc. requirements
  - the content of design doc.
  - design reviews & review issues.
  - S/W testing to be performed.
  - testing topics.

IEEE S/W ENG. STANDARDS

- CONCEPTUAL STD.
  guiding principles and overall approach.
- PRESCRIPTIVE STD.
  address the req. to which the S/W dev. must conform
- GUIDANCE STD.
  Implementation of std.
  conformance requirements.