U-3 (STQA)
Test Case Design Technique

Black Box Testing	White Box Testing	Grey Box Testing
• internal working ↳ needn't be known	• tester → knowledge of internal working appln	• tester – limited knowledge of internal working appln
• known as closed box testing, data driven testing, or functional testing.	• known as clear box testing, structural testing or code-based testing	• known as translucent testing as tester has limited knowledge of insides of appln
• Performed by endusers & testers & developers	• Normally done by testers & developers.	• endusers, testers & developers
• external expectations - based Internal behaviours - unknown.	• Internal → full known tester → design test data accordingly.	• Basis of high level database diagrams & data flow diagrams
• exhaustive & least time consuming	• most exhaustive & time consuming	• partly time consuming & exhaustive
• Not suited for algo. testing	• suited for algo. testing	• Not suited for algo testing
• Trial & error method.	• data domains & internal boundaries can be better tested.	• data domains & internal boundaries can be tested if known
Adv → • system → works properly performance & security testing • test → user's pov • tester → not know programming language. • test → conducted by body independent • Testcases → specifications.	Adv → • documented → procedure processes • ensures coding standard • if anything wrong → early detection • some parts → verified	Adv → • ensure correctness of s/w → structurally & functionally • combine adv → black & white box testing
Disadv → • coding errors → can't be tested • system design & internal structure → not tested • only small no. of inputs.	Disadv → • No assurance ↳ cust req. meet. • code → not executed fully ↳ no guarantee of working	Disadv → • conducted → using tools • knowledge → tools configurations.

classmate
Date
Page

Agile Testing

- follows principle of agile s/w development.
- continous process → NOT sequential.
- iterative development methodology
- collaboration b/w customer & self organn teams & agile aligns development with customer needs.

Adv →
- saves time & money
- less documentation
- regular feedback → end user
- daily meetings → help determine issues in advance.

Disadv →
- large ones → difficult to assess.
- lack of emphasis → necessary designing & documentations
- get off track → might be diff than customer requirement.
- senior programmers → can take decisions → development process.
- no place for newbie → with experience (exceptional)

Adhoc Testing.

- informal unstructured testing.
- main aim → break the system
- doesn't follow any test design techniques
- conducted by tester with lot of knowledge
- randomly creating test cases altogether.

Adv →
- tester → no. of defects.
- can be applied anywhere in SDLC
- coupled with any other testing
- doesn't mandate any documentation.
- valuable → test coverage & quality.

Disadv →
- testing → not documented.
  → so difficult for tester
- execution of invalid test case → invalid errors.
- tester → no prior knowledge
  → uncover many errors will not be able
- no assurance of error to be found
- uncertainty → document, time & efforts

Clip Board

100% ECOFRIENDLY BOARD

MACGREEN educare series

White Box Testing Test Case Design Techniques.

Static Testing	Dynamic Testing
• performed → early stage of develop^n	• later stage of softw/ develop^n
• whole code → not executed	• whole code → executed.
• prevents defects	• finds & fixes defects.
• before code deployment	• after code deployment
• Less expensive	• More expensive.
• more statement → less time	• fewer statement → small area of code.
• completed prior to code deployment	• after code deployment
• Verification stage	• Validation Stage
• Doesn't run code.	• executes code
• Code & Documentation	• identifies software system bottlenecks.
• checklist → testing procedure.	• test case → dynamic testing
• Walk through & code reviews	• functional & non fuctional aut types.

a) Informal Review b) walk through
c) Technical Review d) Inspection

Types of Code Coverage Testing

a) statement coverage
b) Branch coverage
c) Path Coverage
d) conditional coverage
e) Loop Coverage

} Testing.

* Structural testing -

- Testing → structure of system or component.
- tester → knowledge of internal impln of code
- How s/w works?
- run against → predesigned test cases.
- system & acceptance testg.

Types

Code Functional Testing

- developer → checks code
- code → coverage or complexity testing
- early phase of testing code
- before submitting code.
- debugging sort of activities
- activities → code knowledge
- developer → error → inputs.
- module → logic & cond^n (complex).
- diff IDE → run module
- focus on debugging than testg
- test → input & expected output
- debug → program → complex logic & conditions

Code Coverage Testing.

- form of white box testing
- generates report on test suite
- part of feedback loop in dev process.
- loop → till specified target achieved
- execute all line of code.
- follow logic of decision path.
- discovering code within program
  ↳ not exercised → testcases
- developers to quickly & easily improve quality of unit tests.
- determine how much → tested
- code coverage = $\frac{\text{No. of line of code}}{\text{Total no. of lines of}} \times 100$ %
- lower chance of code containing undetected software

Types of Code Coverage Testing.

1) Statement coverage testing –
   - coverage metric → whether flow of control reached every executable statement of source code at least once.
   - caculate coverage levels.
   - structured program → decision conditions & control loops → program pl

$$\text{statement coverage} = \frac{(\text{No. of executed statement})}{(\text{Total no. of statement})} \times 100.$$

- Adv →
  - conforms quality code → executing each program atleast once.
  - checks flow of program statement
- Disadv →
  - necessary → not sufficient way of testing
  - Missing statement → source code not covered.
  - no of test cases required → type of statement
  - decision making statement → testers → confirm correct part.
  - costly due to time & money.

2) Path Coverage Testing →
   - considers all paths present in block code.
   - execute paths flow from start to end.
   - loops → infinite no. of paths.
   - report → possible paths in each fun^{n}

$$\text{Path coverage} = \frac{\text{Total path exercised}}{\text{Total no. of Paths}} \times 100.$$

- Adv →
  - reduce redundant test.
  - focus on programic logic
  - test cases → execute every statement → program at least once.
- Disadv →
  - no. of paths → exponential to no. of branches
  - path testing → expert & skillful testers
  - difficult to test → all paths → product more complex.

3) Conditional coverage Testing -

- known as predicate coverage.
- covers → True or false → Boolean expression.
- a cond^n → operand of a logical operator → don't contain logical operators.
- metric → better sensitivity → control flow → not evaluate all Boolean exp^n

Conditional Coverage = $\frac{\text{total decision exercised}}{\text{total no. of decision of program}} \times 100$.

- Adv →
  - validate all cond^n in code
  - ensure no condition led to any abnormality of program's operation
  - remove issues → cond^n coverage method.
  - allows to find areas which → not tested by other methods.
  - finds a quantitative measure of code coverage.

- Disadv →
  - tedious to determine minimum set of test cases → very complex boolean expr
  - no. of test cases → cond^n → similar complexity.

4) Functional Coverage Testing -

- mapped into function blocks & logical units of program.
- metric reports → invoked each function of given program.
- determined by tracing how many fun^n/modules of program.

Functional coverage = $\frac{\text{total functions exercised}}{\text{total no. of functions of program}} \times 100$.

- Adv →
  - easily identified → program so test case → writing easy
  - easy to measure → function → in program.
  - improves performance & quality of a product.

✱ Branch Coverage Testing -

- white box testing method.
- every outcome → from code module
- ensure that each decision condition from → each branch executed at least once.
- measure fractions of independant code.
- Branch Coverage = $\frac{\text{Number of Executed Branches}}{\text{Total no. of Branches.}}$

Adv →
- approve all branches → code are reached.
- to guarantee → no branches prompt.
- removes issues because of statement coverage testing
- find areas not tested
- find quantitative measure of code
- ignores branches inside Boolean expressions.

Disadv →
- costly
- only applicable to operations other than Boolean operations.
- Not a competent method →
- metric → control structure completely exercised.
-

✱ Loop Coverage Testing -

- validity of loop constructs
- white box testing • test loops in programs.

Adv →
- no. of loop iterations → loop testing
- s/w doesn't enter an infinite loop.
- initialization of all variables used within loop
- aids in detection of various issues → inside loop
- aids in capacity assessment

Disadv →
- loop issues → low level appl^n
- Flaws → loop testing are not significant
- Numerous defects → operating system, pointer, files etc.

Black Box Approaches

Black Box Approaches To Test Case Design -

Adv
- Test → user pov
- tester → need not know programming language
- Body independent → developers
- Test cases → specific → complete.

Disadv -
- small no. of input → tested.
- unclear specification → difficult design
- tests → redundant

1) Boundary Value Analysis (BVL) -
- boundary condition data testing.
- testing boundaries b/w two partitions.
- both valid & invalid boundaries
- test edge of each equivalence class.
- edge of both input & output classes.

2) Equivalence Class Partioning
- equivalence class partioning → ECP
- input data → two equivalent partitions.
- large range of data input
- reduce no. of test cases to save time
- type of black box testing
- Reduces time → required for testing.

3) State Transition Testing -
- Blackbox dynamic testing
- depedant → values/condition
- when → appln → real time system → diff states
- appln → tested against finite set of input values

4) Cause effect Graphing -
- represent graph which maps specification writing in natural language
- tester → boolean logic & logic operators
- testers to use combination of input.
- graph → covered → decision table.

5) Decision Table -
- excellent tool → testing & requirements management.
- diff input combinations.
- capture & effects for better test coverage.
- black box test design technique → test scenarios for complex business logic

adv →
- easily → decision table
- simple → easily interpreted

disadv →
- no. → input increase → table complex
- each rule → test case → decision table.

* Use Case Testing -

- functional black box testing
- identify gaps in s/w applⁿ → not be found → testing individual s/w components

*Characteristics
- → Use Case → b/w → actors & systems
- → Actors → users & interactions each user takes part into.
- → Test case → use cases
- → identify gaps in system
- → efficient in defining scope

* Diff types of Experience based Testing Techniques

1) Error guessing

- takes adv of testers skill, intuition.
- depends on tester experience
- creation of defect & failure lists.
- used to design tests & user experience

2) Exploratory testing

- doesn't use any specific test case
- testers explore & identify → errors to improve quality of s/w
- test cases are not created in advance but testers check system on fly.
- used in agile models.
- used when specifications are either missing or inadequate.

adv →
- bug early.
- less preparation
- bug → missed in testcases
- test → new features.

disadv → time consuming
test engineer → misunderstand → tester as bug
bugs → misunderstand as feature.

Types of exploratory testing.

1) Freestyle
- don't follow rules.
- no max. coverage
- Adhoc testing
- freestyle exploratory

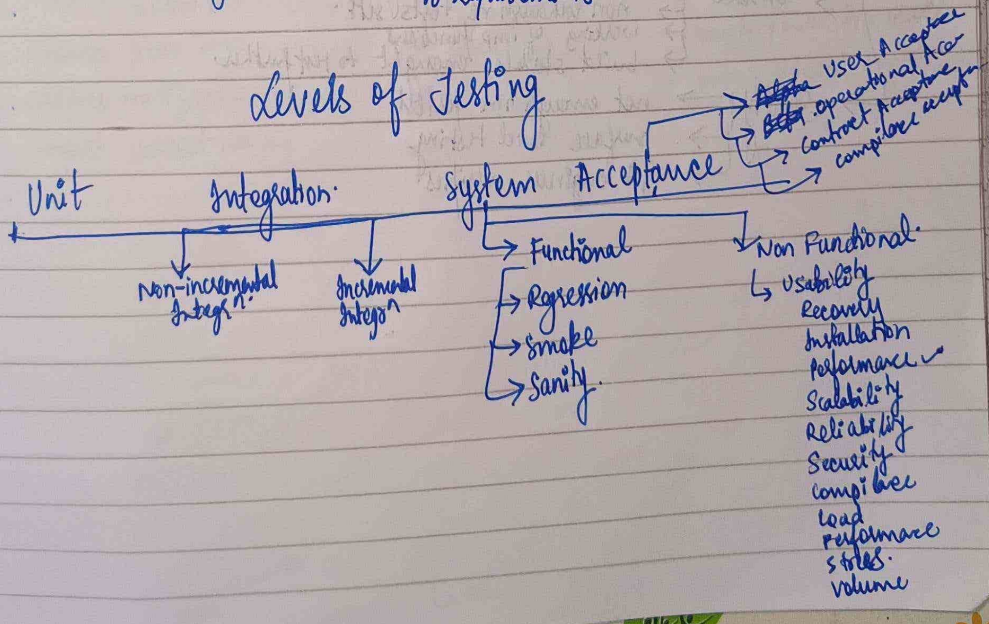
2) strategybased
- multiple testing technique.
- experienced tester
- knows appln
- appln → longest time.

3) Scenario based.
- multiple scenerio
- end to end, test scenerio
- test engineer → find defects
- set of possibilities.

* Levels of Testing

Unit Testing	Testing individual	By developers.
Integration Testing (Non-incremental integration testing / Incremental integration testing)	Testing components group	By developers & testers.
System Testing	Testing entire system	By testers.
Acceptance Testing	Testing final system w.r.t. to requirements	By end user/ customer.

Levels of Testing



Unit Testing

- test each part of program. individual parts → correct
- kind of white Box Testing
- performed → anytime
- doesn't verify → code.
- executed → developer
- maintenance → cheap
- can't be subdivided
- specifications of modules
- not detect → any system wide error

Integration Testing.

- combine modules in appln. test → as group → if they work
- kind of Black Box testing.
- carried out after unit testing & before system Testing.
- verifies → code.
- executed → testing team.
- maintenance → expensive
- can be subdivided → top down / bottom up / hybrid
- Interface specification of modules
- detect all type of errors.

✱ Functional Testing

- Regression → previously developed → tested → ensure same kind of working
  - changes may include software enhancement
- Smoke
  - Build verification testing
  - non exhaustive tests sets.
  - working of imp functions
  - build stable enough to test further
- Sanity
  - not enough time for test
  - surface level testing
  - QA engineer verifies.

Smoke Testing

- confirm → critical functional of program working fine.
- verify stability of system in order to proceed more rigorous testing
- performed by developer or tester
- tests entire system from end to end.
- usually documented or scripted.
- general health checkup of software.
- performed first on initial builds

Sanity Testing.

- check new functionality / bugs have been fixed.
- verify rationality of system in order to proceed with rigorous testing
- performed by testers.
- only particular component of entire system
- not documented or is unscripted.
- specialized health checkup of software.
- performed after smoke testing.

* Regression Testing –

- verify changes made in codebase don't impact current functionality.
- confirm recent program or code change → affect existing features.

Adv
- lot of efforts & time.
- run multiple tests
- ensures → no new bug
- easily automated → tools
- ensures fixed bugs.
- verifies modification do not impact correct work

Disadv →
- time & resource → consuming
- required even after small changes.
- each & every time
- achieve max test coverage with fewer test case
- release & build bug fixes.

Functional Testing.

- What the product does.
- checks → opn & actions of appln.
- based on business requirement.
- actual result → working → acc to expected
- carried out manually
- as per customers requirements.
- reducing risk of product
- testing func" of s/w
- types → unit, Integration, system, acceptance

Eg-
Login page must show textboxes' enter username & password.

Non-Functional

- checks behaviour of an appln
- customer & performance requirement
- checks response, time & speed
- more feasible to test.
- tests → customer requirement
- customer feed → more valuable
- testing performance functionality
- includes → performance, load testing, stress testing, security, Installation, Recovery.
- eg→ Testif a login page
  ↳ loaded in 5seconds.

Performance Test classification:

- Load Testing
- stress Testing
- Data Volume Testing
- Storage testing

* I18n Testing - (Internalization)

- Process of designing & developing product.
- non-functional testing technique
- ensuring adaptability of software to diff culture & language
- 18 → no. of characters b/w I & N in internationalization.

Adv →
- easier adaption
- reduced time & cost
- simpler maintenance
- improved quality & code architecture
- adherence to international standard.
- good quality & architecture
- reduced ownership cost for versions.

* L10N - (Localization)

- process of localization testing to check appropriate linguistic & cultural aspects
- testing globalized appln
- 10 → no. of letters b/w l & n → specific language.

Adv
- cost reduces (testing)
- cost reduced (support)
- reducing time for testing
- more flexible & scalability.

Disadv
- requires domain expert
- Hiring local translator → expensive.
- tester → schedule challenges.

U-4 (STQA)
Software Quality Assurance & Quality control.

- Software Product Quality - (Measure) -
  - define goal of software product
  - determine how to measure success of your software
  - Identify which s/w quality metric are imp.
  - choose test metric → easy to implement & analyze.
  - Setup a system for collecting data on this test metric evaluate

* Requirement of Product.
  - stated & Implied requirements
  - general & specific requirement
  - Present & Future Requirement.

* Requirement Categories Based on Priority
  - Primary requirement (must / must not)
  - secondary " (should / should not be)
  - Tertiary Requ " (could be / could not be)

* Agile Model -

adv →
- customer satisfaction by rapid
- people & interaction
- face to face conversation.
- regular adaption to circumstances.
- late changes in requirements all welcomed.

Disadv →
- lack of emphasis
- easily get taken off track
- only senior programmers are
- customer representative not

Software Development Model.

Waterfall Model.

adv → • simple & easy
• processed one at a time
• smaller projects
• defined stages
• easy to assign tasks
• well documented.

disadv → • late processed
• high risk & uncertainty
• not for complex
• past model → long project
• difficult → measure progress
• adjusting model.

Iterative Model.

adv → • working s/w during SDLC
• very flexible
• cheap
• feedback taken
• errors & bugs → early detect
• smaller dev teams

disadv → • not good for small project
• more resource intensive
• difficult to manage.
• risk analysis → specialists
• all requirement → gathered upfront.

Incremental

adv → • develop quickly
• result → early & periodic
• parallel development
• progress can be measured
• risk analysis ↓ → better
• supports changing requirement
• risk → easily managed

Disadv → • More resource → required
• more management required.
• not suitable for smaller project
• management complexity → more
• dependent on risk analysis phase.

Spiral

adv → • high amt risk analysis
• good for large projects
• approval & documentation control
• s/w → early in SDLC
• high risk projects

disadv → • costly.
• Risk analysis
• not for smaller projects
• hard to define objects w.r.t milestones

Proto typing

adv → • users → actively involved in dvlp
• errors → detected earlier
• Mission functionality → identified easily
• diff functions → identified.

disadv → • leads to implement & repeating system
• increase complexity
• Incomplete appln → full system designs.

RAD.

adv → • reduced development time
• modularized built
• ~~highly skilled developer~~
• ~~high dependency → modelling skills~~
• quick initial reviews
• encourages customer feedback.
• Integ^n → solves integration issues.

disadv → • depends on strong team.
• only system → modularized.
• highly skilled developers.
• High dependency on modelling skill.
• Inapplicable → code generator very high.

* Types of Software Product

1) Products affecting life
2) Products affecting investment
3) simulation based products
4) other products

* Schemes of Criticality Definitions:

1) classification talks about dependency of busness on system.
2) classification talks about products operating environment.
3) " talks about complexity of a system on basis of development capabilities required.

* Software Quality Management.

Activities of software Quality Management:

1) Quality Assurance
2) Quality Planning
3) Quality Control ⇒ correction, corrective actions, prevention actions.
4)

* Factors → defect in software →
- Technology → source of defects
- communication gap
- requirement → dynamically changes
- customers → unaware about stated requirements
- software developers → confident about stated goals
- programming errors
- time pressures
- egotistical or overconfident of team
- poorly documented code
- lack of skilled testers
- peer reviews & self reviews

* Process related to software Quality

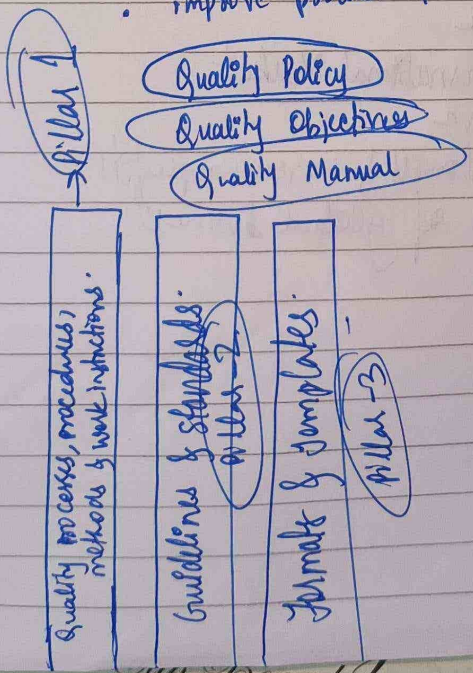
→ 1) Vision
2) Mission
3) Policy
4) objectives
5) strategy
6) Goal
7) Values.

* Quality Management System's Structure -

↓

- process of overseeing all activites & tasks needed to maintain desired level of excellence
- collection of business processes focused on consistely meeting ustomer requirements

Benefits → • Managing product & process quality
• Achieve greater consistency
• Reduce expensive mistakes
• Increase efficiency → improve use of time & resources
• customer satisfaction
• Market business → efficiently
• new market sector & territories.
• improve product process & system



* Important Aspects of Quality Management.

1) Quality planning at Organization level
2) Quality planning at Project level
3) Resource Management.
4) Work Environment
5) Customer related processes
6) QMS document & Data Control.
7) Verification & Validation
8) software project management.
9) Software configuration management
10) Software Metrics & measurement.
11) software quality audits.
12) Subcontract management
13) Information security management
14) Management Review.

* Need for Quality

- A competitive issue
- Survival issue
- Entry in International Market
- Cost effective
- Retaining customers & increasing profits
- Hallmark of global business.

Software Quality Models.

McCall's Quality Model

- General Electrics Model.
- Vol. of 55 quality characteristics.
- quality factors → 11 main factors.
- 11 factors → describe view of s/w
- 23 quality criteria → internal view of s/w.

3 major perspectives of model -

i) Product Revision
   a) Maintainability
   b) Flexibility
   c) Testability

ii) Product Transition -
   a) Portability
   b) Re-usability
   c) Interoperability

iii) Product Operations
   a) Correctness
   b) Reliability
   c) efficiency
   d) Integrity
   e) Usability

Boehm's Quality Model.

- quality of s/w on basis of set of credential & measurement.
- Hierarchial manner
- aim to address shortcomings of models that automatically & quantitatively evaluate.

High level of characteristics.
- As in utility
- Maintainability
- Portability -
  - Reliability
  - efficiency
  - Usability
  - Testability
  - Understandability
  - Flexibility

Quality Assurance (QA)

- procedure focuses on.
  ↳ QA → quality requested achieved.
- Prevent the defect
- method to manage quality → verification.
- Not involve executing program
- preventive technique
- proactive measure.
- procedure to create deliverables.
- full software development life cycle.
- defines standards & methodologies.
- performed before Quality Control
- Low level activity → identify error &.
  mistakes which QC cannot.
- prevent defects in system.
- Less time consuming
- ensures → everything is executed in right way
- whole team involved
- statistical technique on QA
  ↳ statistical Process Control (SPC).

Quality Control (QC)

- procedure focuses on
  ↳ fulfilling quality requested
- identify & fix defects
- validation
- involves executing program
- corrective technique
- Reactive measure.
- procedure to verify deliverables
- full s/w testing life cycle
- confirms standards → followed
- after QA is done
- high level acting:
  identify error that QA cannot
- identify defects or bugs in system
- more time
- ensures → as per requirement
- Testing teams involved
- statistical technique on QC
  ↳ statistical Quality Control (SQC)

Benefits of Quality Management Systems

1) Defining, improving & controlling processes
2) Reducing waste
3) Preventing mistakes
4) Lowering costs
5) Facilitating & identifying training opportunities.
6) Engaging staff
7) Communicating a readiness to produce results.

Activities → Quality Assurance
Quality Planning
Quality Control.

Objective → Improving internal processes
Lower cost
Reusability
Optimum utilization of resources
Helps in achieving organization goal.
Data Management
Continously improved customer satisfaction.

Principles →
- Customer Focus
- Engagement of ppl
- Process Approach
- Evidence based decision making
- Relationship management.

Complexity Metrics →
- Lines of Code
- Cyclomatic Complexity

* Customer Satisfaction -

How s/w testing improving

1) Understanding Customers.
2) Defining validation requirements to customer satisfaction
3) Add customer experience of testing Quality Assurance efforts.
4) Conduct in depth performance testing.
5) Should have enough knowledge about crowdsourcing
6) Pay attention to customer feedback.

* CASE tools -

→ Computer Aided Software Engineering.
→ computer facilitated tools
→ high quality & defect free s/w
→ warehouse for documents.

- Improve quality of system.
- Increase speed of design & develop.
- Ease & improve testing
- Integration of devlp activities
- promote reusability
- program maintenance.

Objectives
→ improve productivity
→ improve Info System Quality
→ improve effectiveness.

CASE Support.
→ Prototyping Support
→ Structured Analysis & Design
→ Code Generation
→ Test Case Generator.

ISO 900

- International Standard Organization
- any type of industry
- operate business processes
- min req. product → identified
- No levels specified
- Sequence of steps
  - ↳ quality sys not specified
- Pass or fail criteria provided
- What is required?
- focuses on hardware, s/w, processed materials & services
- aims at Level 3 of SEI-CMM Model

CMM.

- Capacity Maturity Model
- software industry
- software Engineering activities.
- technical aspects of s/w engineering
- Five levels → Initial, Repeatable, Defined, Managed, optimization
- step by step progress among its maturity levels.
- Grade for process maturity provided.
- How to fulfil requirement
- achieving total Quality Management
  - beyond quality assurance

U-5 (STQA)

Automation Testing Tools / Performance Testing Tools

Automation Testing

- repetition of same operation each time.
- automation helps if code changed
- useful when set of test cases needs to execute frequently
- automation test suites ready
- heterogeneous environment
- Testers → complex appln test
- runs test case faster than human
- not helpful → testing UI
- Build Verification Testing → Automating
- Initial cost → more than manual
  Always useful

Manual Testing

- not reliable since result of test execution is not accurate all
- difficult to catch defects after regression using manual
- needs to run once or twice
- test case needs to run once after
- impossible to carry on diff machine
- not involve any programming task
- slower than automation
- very much helpful → testing UI
- Build Verification Testing → execution
- less cost than automation testing

1) Requirement understanding
2) Defining scope of automation
3) Selecting right tool
4) Framework creation
5) Scripting test cases
6) CI CD

Benefits –

- enhanced results
- swifter feedback system
- Brand enhancement
- cost-effective
- efficiency testing
- increase in coverage area
- Detailed testing
- Reusability
- Earlier detection of defects
- Time to market

Automated testing framework.

Linear Framework (Automation)
- record & playback framework.
- Tester → record each step
- Adv → • no need of custom code.
  - fastest way → test scripts
  - workflow → easier
  - most automated.
- Disadv → • Not reusable script
  - maintenance → hard
  -

Modular Based Testing Fram
- break → appln → individual modules
- test script → client's requirements.
- abstract changes.
- Adv → • changes → module & appln
  - easy maintenance
  - scalability
  - less effort for test case.
- Disadv → • data → hard coded.
  - programming knowledge needed.
  - additional time.
  - scripting knowledge.

Library Architecture
- modular + add" benefits.
- scripts → identified.
- Adv → • high level modularization
  - test maintenance.
  - cost efficient
- Disadv → • Test data → coded script
  - Technical expertise.
  - Test scripts → more time

Data Driven Framework.
- same features → diff data → test
- Adv → • test → multiple data sets.
  - multiple scenario → reduce no. of scripts
  - save time
  - run in short test period.
  - changes → test data
    ↳ no alter to test script code
  - code → Reusable.
- Disadv → • High experienced tester
  - amt of time
  - programming knowledge.
  - quite complex

Keyword Driven
- each func" → appln → test
  laid out → instructions.
- Adv → • minimal scripting knowledge
  - single keyword → multiple test scripts → code → reusable
  - Test script → indep ndent.
- Disadv →
  - cost high
  - employee → good test automation
  - keywords → hassle.

Hybrid Testing
- comb" of any previously mentioned framework.
- teams → agile model
- easily adapt → for better results.

Criteria for Selecting Test Tool.

- Data driven capabilities
- Debugging & logging capabilities
- Platform Independance
- Extensibility & customizability
- Email Notifications
- Version control friendly
- Ease of use
- Environment support
- Object Identification.
- Testing of Database
- Scripting Language Used.
  - easy to debug
  - support for multiple testing frameworks
  - extensive test reports & results
  - Minimize training cost of selected tools.

Automation Tool selection criteria Suggested by Experts.

1) Meeting Requirements →
2) Technology Expectations.
3) Training & Skills
4) Management Aspects.

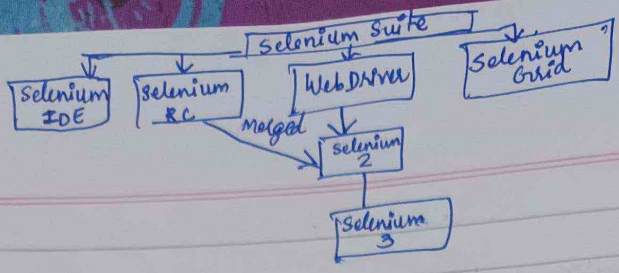
Selenium –

- open source
- free web Automation tools or suit-
- components → a) Selenium Integrated Development Environment (IDE)
  Selenium Suite
  b) Selenium Remote Control (RC)
  c) WebDriver
  d) Selenium Grid.

Features →
- functional automation tool for Web application-
- open source tool
- HTML, Java, PHP, Perl, Python, etc-
- browsers like IE, Moxilla, Safari, chrome & Opera
- functional regression testing.
- supports OS like windows, Linux & Mac
- flexible → functional tools
- executing one's tests on multiple browser platforms.

Advantages →
- free & no license cost.
- supports various OS.
- support programming languages
- supports browsers
- uses less Hardware resources.
- supports parallel test case execution.

Disadvantages →
- supports web appli" only.
- no centeralized maintenance of objects & elements.
- new features may not work properly
- no reliable support from anybody as open source tool.
- Difficult use, take more time to create test case.
- difficult to set up test environment.
- Limited support for image testing



* Selenium IDE -

- Integrated Development environment → selenium tests.
- Selenium Record
- automatically add assertions to all pages.
- export WebDriver or Remote control scripts.
- allows you options to select a language

Features →
- record & playback
- Intelligent field selection will use IDs, names or XPATH.
- auto complete for all common Selenium commands.
- walkthrough → test cases & test suites
- debug & set breakpoints.
- save tests as HTML, Ruby scripts
- support for selenium user-extension.js file
- option to automatically assert title of every page
- Roll up common commands.

Categorization of logs → Debug
Info
Warn
Error

* Selenium RC -

- Selenium Remote Control
- server, written in Java that accepts to browser via HTTP.
- possible to write automated tests for web application.
- API & library for supported languages.
- high-level programming language.
- automated web appln UI.

Features -
- test scripts → Programmy language.
- user run → test against diff browsers.
- use any Javascript enabled browsers.
- work with any HTTP website
- automatically launches server & kills browsers
- test automation developer → developing test logic
- configures the browser.

Components of Selenium RC
- → Selenium Server
- → Client libraries.

* Selenium Web Driver.

- API helps in communication b/w languages & browsers.
- supports many languages.
- Architecture →
  - selenium client Library
  - JSON Wire Protocol over HTTP
  - ~~ISEN~~ Browser Drivers
  - Browser.

Selenium Grid 1

- Requires Apache Ant to be installed
- Has its own remote control (diff from RC server)
- Supports only Selenium RC commands.
- Automate only one browser per remote control.

Selenium Grid 2.

- Apache Ant installation is not req
- Bundled with Selenium Server
- supports both Selenium RC & WebDriver scripts
- You can automate up to 5 browsers per remote control.

* Automation Tools -

Help teams & organization to automate their software testing needs

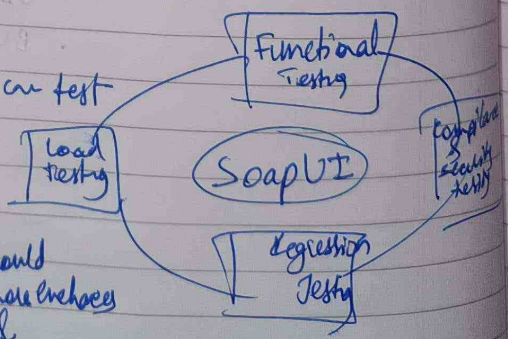
* Soap UI -

- World's leading open-source testing platform.
- open source cross platform API testing tool.

Features → . simple & easy UI
. supports all standard protocols
. provide security or vulnerability testing of system
. own building plugin for diff open source environment

adv → . desk top based appln
. costs less
. supports drag & drop.
. creates mock where testers can test
. fast & well organised.

disadv → . require enhancement
. mock reponse module should be more enhanced
. longer to request big data & files



RPA.

- Robotic Process Automation
- form of businessprocess automation that allows anyone to define a set of instructions for a robot or bot to perform

RPA features →
- Rich analytical suite
- Simple creation of bots
- Scriptless automation
- Security
- Hosting & deployment
- Debugging.

Benefits →
- Reduces cost
- Reduce operational risks
- Quality & accuracy
- Scalability
- Reduced workload
- Improved customer satisfaction.
- Improved business results.

Types →
1) Unattended RPA / Autonomous
2) Attended RPA.
3) Hybrid RPA.

Working of RPA –
1) Planning
2) Design & Development.
3) Deployment & testing
4) Support & maintenance

* Tosca
- Topology & Orchestration Specification for cloud appl'n.
- set of rules given by industry group OASIS.

Features –
- Automated & Manual Testing
- Browser Based Performance Testing
- Unicode Compliance
- Hierarchial View
- Parameterized Testing
- supports parallel execution
- Requirements Based Testing
- Security

Adv –
- one stop soln → automated tests
- no script to function
- drag & drop features
- creation, maintenance of traditional → increased potential
- supports both GUI & non GUI.
- regressing time → reduced.
- explicit framework → not needed
- Low maintenance
- less time consuming.

Disadv –
- very expensive
- Heavy tool → to maintain
- less performance → while scanning the application.

Components →
- TOSCA Commander
- TOSCA Executor
- TOSCA Wizard or Xscan
- TOSCA Repository

* Appium –

• open source Mobile Automation Tool → platform like Android, IOS, etc

Features – • open source s/w
• cross platform
• all types of Mobile s/w
• supports visual testing
• Multiple Language support
• no source code of library
• strong & active community
• multiple platform support
• small change → no need to install again

Adv → • open source tool
• supports desktop app
• cross platform

Disadv → • lack of detailed reports
• tests → slow
• only supports SDK, 16 & higher
• one instance → one mobile OS device.

* Performance Test Advantages

↳ • Validate Features
• Measure speed, accuracy & stability
• Keep users happy
• Identify discrepancies
• Improve optimization & load capacity.

Tools for Performance Testing.

JMeter

- performance of both static & dynamic resources
- designed on JAVA appln.

Features →
- open source appln
- User friendly GUI
- Support various testing approach
- platform independent
- Support various server types
- support multi protocol
- Simulation
- Framework
- Remote distributed testing
- Test Result Visualizer

Load Runner –

- supports XML → views & handle data
- large range of appln supported
- detailed performance test reports.
- reduce cost of distributed load testing
- provide operational tool for deployment tracking
- reduce cost of s/w & h/w.

WebLoad –

- → flexible test scenario creation
- → detects bottleneck automatically
- → easily approachable
- → evaluate performance test results → browser or mobile
- → generate load from cloud.

Load complete

- provide load modeling for performance testing
- record & playback.
- various OS environment
- test various types of app
- generate load test reports

U-6 (STQA)

Testing Framework.

* Software Quality -

- Quality - a developed product meets it's specifications.
- degree to which a system, component, or process meets specified requirements

* Targeted Quality Factors

- Intuitiveness
- Efficiency
- Robustness
- Richness

* Software Quality Dilemma.

1) Good Enough Software
2) The Cost of Quality ⟷ Prevention cost. / Appraisal cost / Failure cost
3) Risks.
4) Negligence & Liability
5) Quality & Security
6) Impact of Management Actions

* SQA

- Software Quality Assurance
- set of activities that ensure quality of s/w engineering processes

objectives →
- Quality Management Approach
- Measurement & reporting mechanisms
- effective software - engineering technology
- procedure to assure compliance with s/w development standards
- Multitesting strategy is drawn
- formal technical reviews that applied throughout software process

elements of SQA
1) Standards
2) Reviews & Audits
3) Testing
4) Error/defect collection & analysis
5) Change Management
6) Education
7) Vendor Management
8) Security Management
9) Safety
10) Risk Management

SQA tasks
1) SQA plan (prepare)
2) Participate in development
3) review s/w activities to verify compliance
4) Audit designated work products
5) ensure → any deviation in software work products are documented
6) Recording evidence of noncompliance & reports to management

Goals →
- Requirements quality
- Design "
- Code "
- Quality Control effectiveness

Six Sigma

- system of statistical tools & technique focused on eliminating defects & reduce process variability.
- strategy for statistical quality assurance.

Features -

- eliminates waste & inefficiency
- follows structured methodology
- data driven methodology
- putting results on Financial statements.
- business driven, multidimensional structured approach.

Key Concept

- Critical to Quality
- Defect
- Process Capability
- Variation
- Stable operations
- Design for Six Sigma.

Core Steps -

1) Define customer requirements
2) Measure the existing process along with output.
3) Analyze defect metrics & determine imp problem
4) Improve process by eliminating root cause of defects
5) Control process to ensure future work ⇒ no reintroduce causes of defects.

* ISO 9000 -3 & MS principles.

1) Customer Focus
2) Leadership
3) Involvement of ppl
4) Process approach
5) System approach to management
6) Continual improvement
7) Factual approach to decision making
8) Mutually supportive supplier relationships

* Adv of ISO standard.

- Proper quality management can improve & increase business as it puts measures in place
- positive effect on investment, shares etc
- expectation → easily determine
- highlight business objective & new opportunities
- organizations to identify & address risk associated
- Internal costs down
- meet necessary statutory & regulatory
- organizations can expand → new market/sectors

* Limitations

- misleads companies → certification → lethal quality
- high amt of money, time & paperwork required.
- no idea about required resources.
- risks & uncertainty.
- failure → get certified → risk of poor company image
- doesn't guarantee a succesful quality system.
- ISO → doesn't validate technical solution.

* SQA Plan -
- Management Section
- Documentation "
- Standards, practices & conventions "
- Reviews & audit "
- Test section
- Problem reporting & corrective action section

* TQM → Total Quality Management.
→ Total Quality Control (TQC)
- way of organization to acheive excellence in managing
- Art of managing whole to acheive excellence.

*Key elements → • customer focus (satisfaction)
Basic concept • Leadership
• Employee involvement
• Supplier partnership
• Continous process improvement
• Performance measure
Teamwork, Communication & Recognition

* TQM Frameworks
- Plan Do Check Act
- The Quality Improvement Paradigm/ Experience Factory Organization
- SEI capability Maturity Model
- Lean enterprise Management.

* Product Quality Metric.
  - 1) Mean time to failure
  - 2) Defect density
  - 3) Customer problems
  - 4) Customer Satisfaction.

The defect density Matric
- Lines of Code (LOC)
- Customer Perspective.
- Fuctional points

* Inprocess Quality Metrics
  - defect density Machine Testing
  - defect arrival Pattern during Machine Testing
  - Phase based Defect Removal Pattern
  - Defect Removal effectiveness (DRE).

$$DRE = \frac{\text{defect removed during development phase}}{\text{defects latent in product}} \times 100$$

* ~~Ishi~~ Ishikawa's 7 basic tools →
  - Checklists
  - Pareto Diagrams
  - Histogram
  - Run charts
  - Scatter diagram
  - Control chart
  - Cause effect diagram

* Checklists :-
  - most basic tool for quality
  - easy to apply & understand
  - collect data in systematic way.
  - To determine source of problem.
  - efficient & powerful tools.

Disadv → not have effective ability to analyze quality problem

* Histogram -
  - understand glance variation → exists in process
  - shape → process behavior
  - dig deeper → unseen causes of variation.
  - shape & size of dispersion → help identify hidden sources of variation
  - determine capability of process
  - starting points for improvement

* Pareto -
  - find nonconformity from data figures
  - helps graphically displays figures result.
  - tells what to work on first,
  - X → defect cause, Y - defect count

* Run charts.
  - separates gathered data from varities of sources.
  - TO check pattern.
  - run sequence plot.
  - Trends output of manufacturing process
  - process has any cycle or any shift
  - non-random pattern in behavior over period of time.

* Scatter Diagram.
  - displays rel^n b/w two interval variables.
  - identify correlations → exist b/w → quality characteristic & factor desired
  - Two variable in process
  - dot dust together in line → across diagram.

* Control Chart.
  - consists of central line represent average or mean of two parallel lines
  - predict process out of control & limits
  - statistical process control
  - Diff b/w statistical / causes of variation.

Cause Effect diagram.

- Fish bone chats.
- identifies factors of causing an undesired effect.
- break problem → bit size pieces → find root cause
- foster team work
- common understanding of factors causing problem
- Road map to verify picture of process
- relations brainstorming relationship.

Defect Removal effectiveness of Process.

↳ DRE → process metric

$$DRE = \frac{\text{Defects removed during development phase}}{\text{Defects latent in product at phase}} \times 100\%$$

1) error detection efficiency $= \frac{\text{errors found during inspection}}{\text{total errors in product before inspection}} \times 100\%$

2) Removal efficiency $= \frac{\text{defects found by removal opn}}{\text{defect present at " "}} \times 100\%$

3) Early detection ~~feedback~~ $= \frac{\text{no. of major inspection errors}}{\text{Total no. of errors}} \times 100\%$

4) effectiveness measure $= E = \frac{N}{N+S} \times 100\%$

E = effectiveness of activity
N = No of faults by activity phase
S → no. of faults found by subsequent activities.