
Communicate clearly
50+ SOFTWARE TESTING TERMS THAT ARE MOST OFTEN ABUSED!
THIRUVENGADAM ASHOK

“Communicate Clearly: Language Aspects for Effective Testing”

Master the art of precise communication in software testing with this comprehensive guide. This book offers a structured approach to demystify testing jargon and enhance clarity in communication. By categorising over FIFTY key testing terms into THIRTEEN cohesive groups, it provides clear definitions and a systematic framework to improve understanding and usage. Whether you're a developer, tester, or QA professional, this guide helps to:

- Clarify and standardise software testing terminologies.
- Organise terms into logical categories for improved comprehension.
- Promote effective communication within testing teams and stakeholders.

Elevate your testing practice and foster seamless collaboration within teams. **"Communicate Clearly”** is your go-to guide for effective communication in the world of software testing.

About the author

Thiruvengadam Ashok is the CEO of STAG Software Private Limited & Co-Founder of Pivotrics, based in Bengaluru, India. Ashok has dedicated his career to the pursuit of quality assurance in software, continuously evolving his approaches to match the needs of modern systems. He is the creator of HyBIST, an innovative approach to SmartQA that has revolutionised how teams approach hypothesis-driven testing.

Ashok's professional life is deeply intertwined with his personal philosophy. A passionate ultra-distance runner and long-distance cyclist, he applies the principles of endurance and exploration to his work, constantly seeking out new ways to improve software quality. He is also an avid wordsmith, using his love of language to weave both poetry and technical innovation into his life's work.

He holds an M.S. in Computer Science from the Illinois Institute of Technology, a Bachelor's degree in Electronics and Communication Engineering from the College of Engineering, Guindy, and a Postgraduate Diploma in Environmental Law from the National Law School of India University, Bangalore. His life maxim—"Love what you do & Do only what you love"—is reflected in everything he undertakes, from professional projects to personal passions.

Disclaimer:
The information contained in this book is for educational and informational purposes only. While every effort has been made to ensure the accuracy of the content, the author and publisher make no representations or warranties regarding the completeness, accuracy, or applicability of the information provided. The strategies and methodologies described are for informational purposes and should be adapted to individual circumstances as necessary.

Trademarks:
All product names, logos, and brands mentioned in this book are the property of their respective owners. Use of these names, logos, and brands does not imply endorsement. HyBIST is the intellectual property of STAG Software Private Limited.

Edition: First edition, 2025.

TABLE OF CONTENTS

LANGUAGE ASPECTS FOR EFFECTIVE TESTING

Clarity of thought

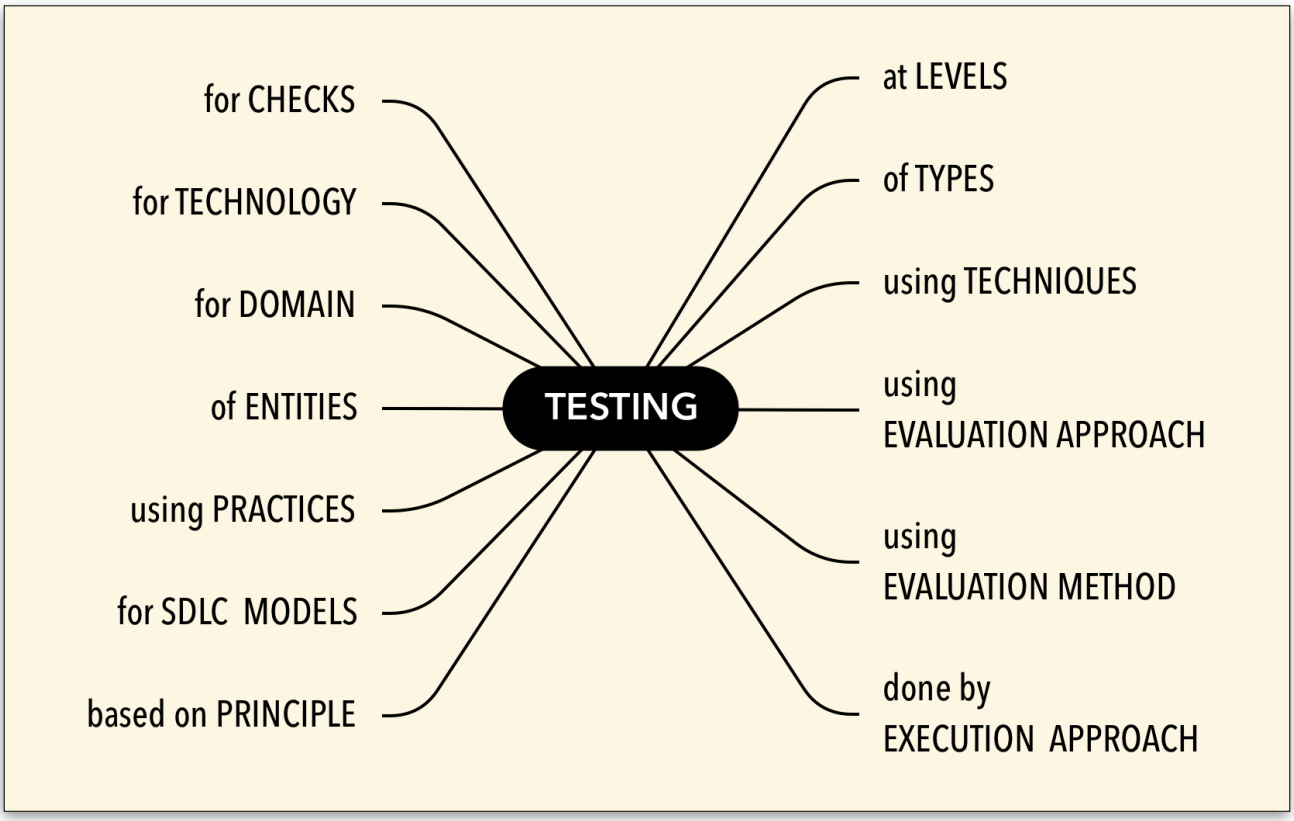
It takes clarity in thinking and clear communication to assure smartly. The language and therefore terms/phrases used have a significant play in this. Given that a lot of terms in software engineering and in particular software testing are takeoffs of typical English terms, they tend to lose its sharpness of intent and most often gets in a general wide sense, in effect abusing the term and therefore resulting in poor thinking and communication.

Abuse via jargons

Jargons abound in software, only to be misused most often! And I think in our discipline of testing, they are abused even more! Most of the terms seem to have an appendage of 'testing' making it far more confusing. Some terms are test types while some are approaches, and some practices and so on. Here I partition 50+ software testing terms into THIRTEEN cohesive groups to sharpen the clarity depicting it as simple mind map and then giving a clear definition to each one.

Categorisation of terms

Here is a mind map that groups the terms based on a phrase to minimise confusion and sharpen clarity.



The mind map above categorises FIFTY common test terms that are typically abused into THIRTEEN categories so that we may communicate clearly. Each category is detailed in the next thirteen sections.

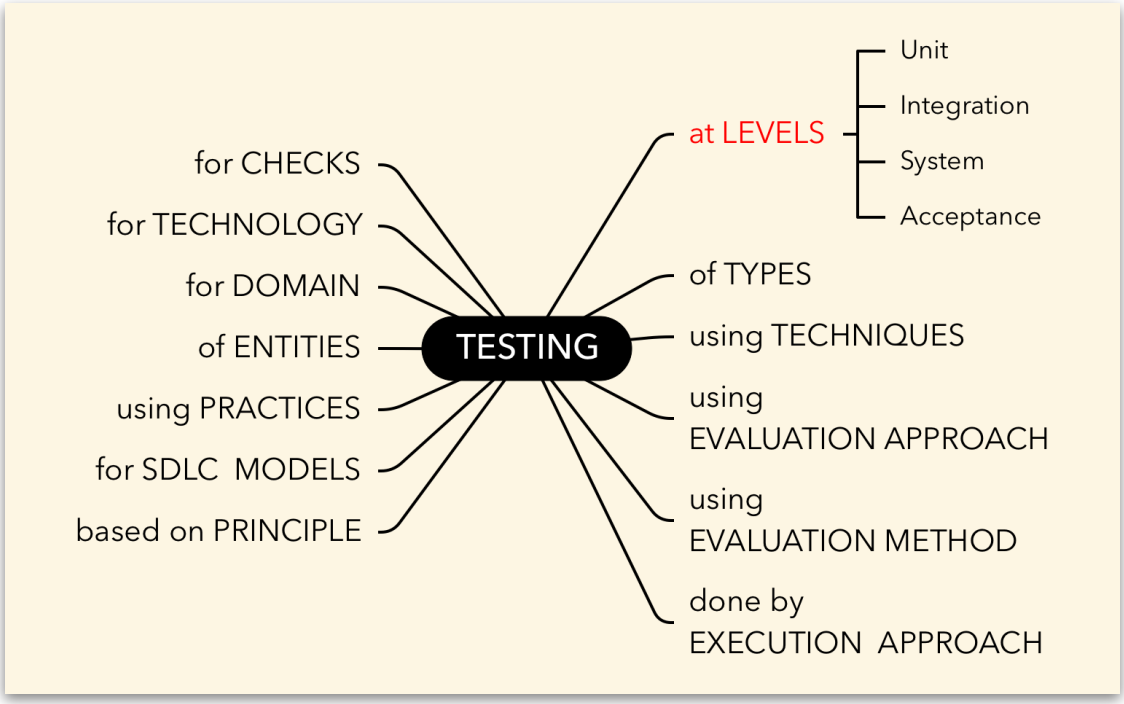
CATEGORY-1: BASED ON TEST LEVELS

The terms Unit, Integration, System, Acceptance testing refer to levels of testing, *implying as to what phase of SDLC it is done*, the earliest being Unit testing.

Levels communicate :

(1) when is it done - early (construction), middle (integration) & late (pre-deployment).

(2) what kind of entity is being validated - a small building block (i.e. Unit) , composition of a few units (i.e. Integration), the whole (System).



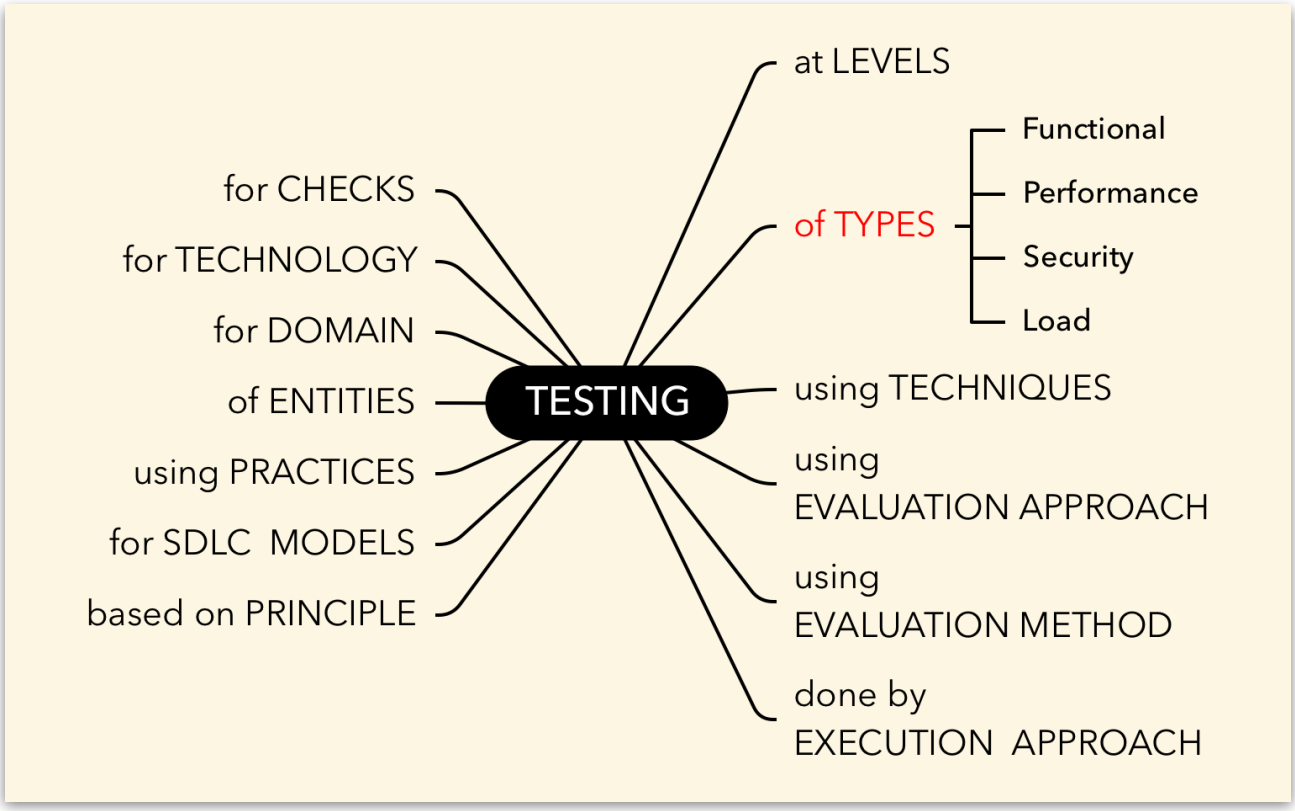
Note each level is focused on uncovering different types of issues:

Test LEVEL	ENTITY validated	Focus on what ISSUE	From POV
Unit	Building block	Basic data validation issues, data interface issues, structural issues & basic functional issues.	Technical/Engineer
Integration	Technical feature	Interface issues & functional issues	Technical/Engineer
System	User Requirement, E2E flow	E2E functional & non-functional issues	End user, Business

CATEGORY-2: BASED ON TEST TYPES

Functional, Performance, Load, Security, Reliability, Stress, Volume and others are really specific types of tests designed to *uncover specific types of issues*, the functional being the behaviour while the others are about the various attributes related to the behaviour.

A mixture of ISSUES to be uncovered represented as **!@#$%^&**							
!	@	#	$	%	^	&	…
Functionality	Usability	Performance	Load	Security	Reliability	Compatibility	…
each TEST TYPE focusses on a specific issue							

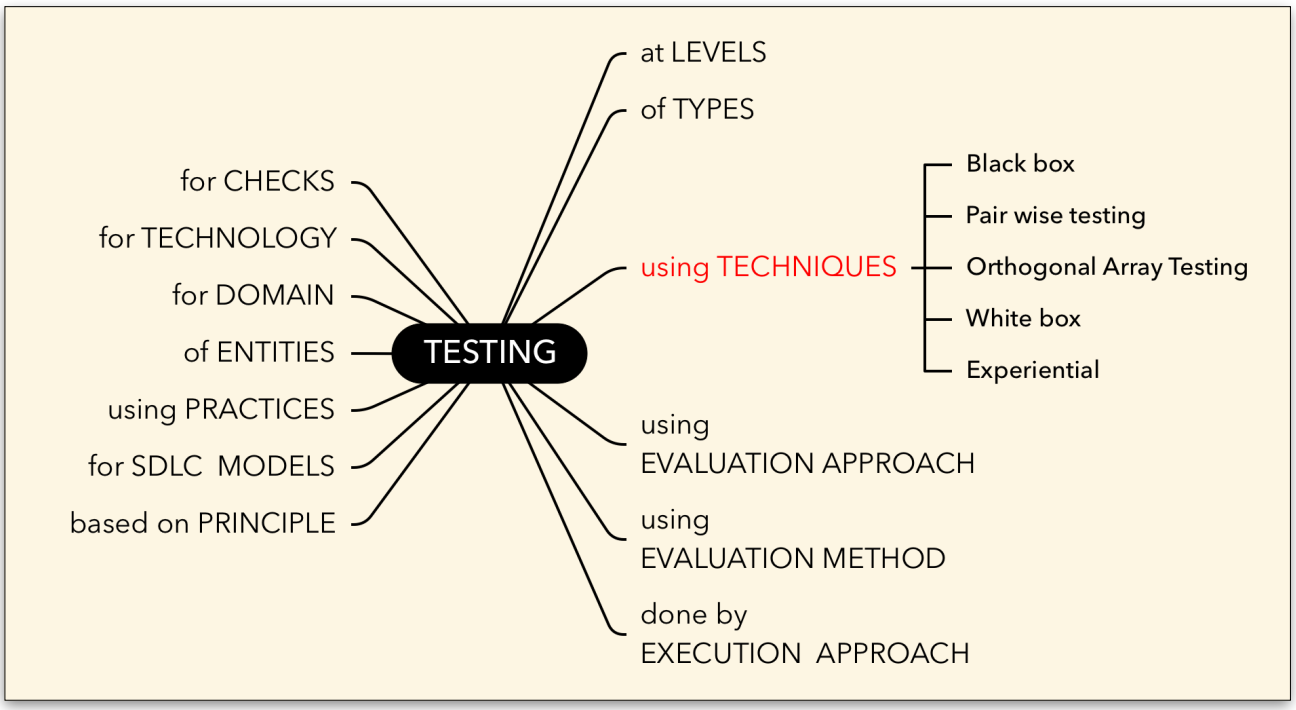


CATEGORY-3: BASED ON TEST TECHNIQUES

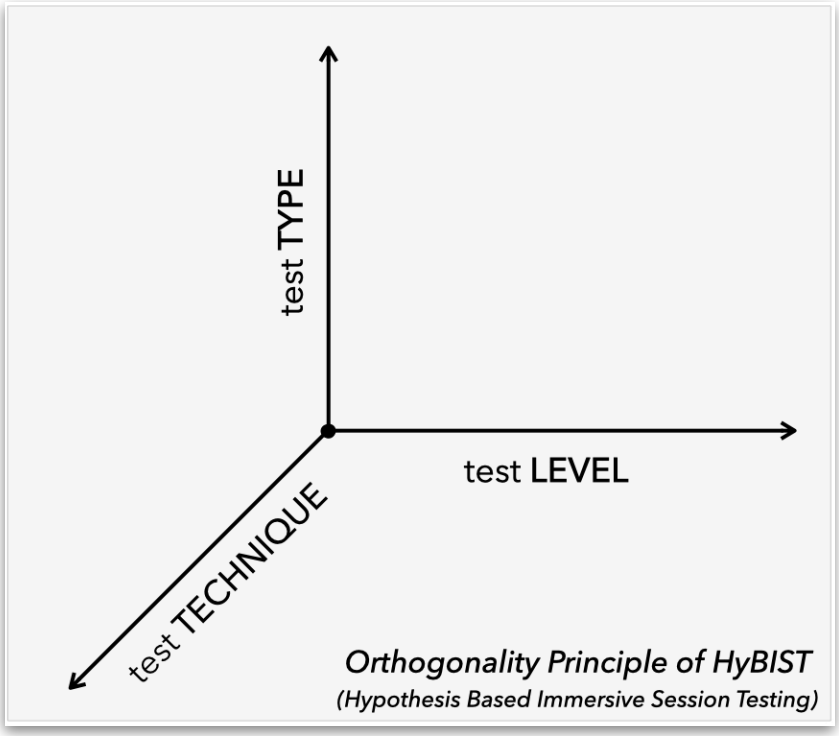
Black box, White box are really test techniques. Black box techniques use external behavioural information to design test cases whilst white box uses internal structural information *to design test cases*. Pair-wise (All-pairs), Orthogonal array is a specific technique to combine test inputs to generate optimal number of test cases.

All pairs testing is a combinatorial method of software testing that, for each pair of input parameters to a system (typically, a software algorithm), tests all possible discrete combinations of those parameters. Using carefully chosen test vectors, this can be done much faster than an exhaustive search of all combinations of all parameters, by 'parallelising' the tests of parameter pairs.
(From Wikipedia) . Click here for more from Wikipedia.

"Orthogonal array testing is a black box testing technique that is a systematic, statistical way of software testing. It is used when the number of inputs to the system is relatively small, but too large to allow for exhaustive testing of every possible input to the systems. It is particularly effective in finding errors associated with faulty logic within computer software systems. Orthogonal arrays can be applied in user interface testing, system testing, regression testing, configuration testing and performance testing. The permutations of factor levels comprising a single treatment are so chosen that their responses are uncorrelated and therefore each treatment gives a unique piece of information. The net effects of organising the experiment in such treatments is that the same piece of information is gathered in the minimum number of experiments."
(From Wikipedia). Click here for more from Wikipedia.



Test LEVELS, TYPES & TECHNIQUES are orthogonal i.e technique(s) is/are used to design test scenarios/cases for a type of test that is performed as part of a test level.



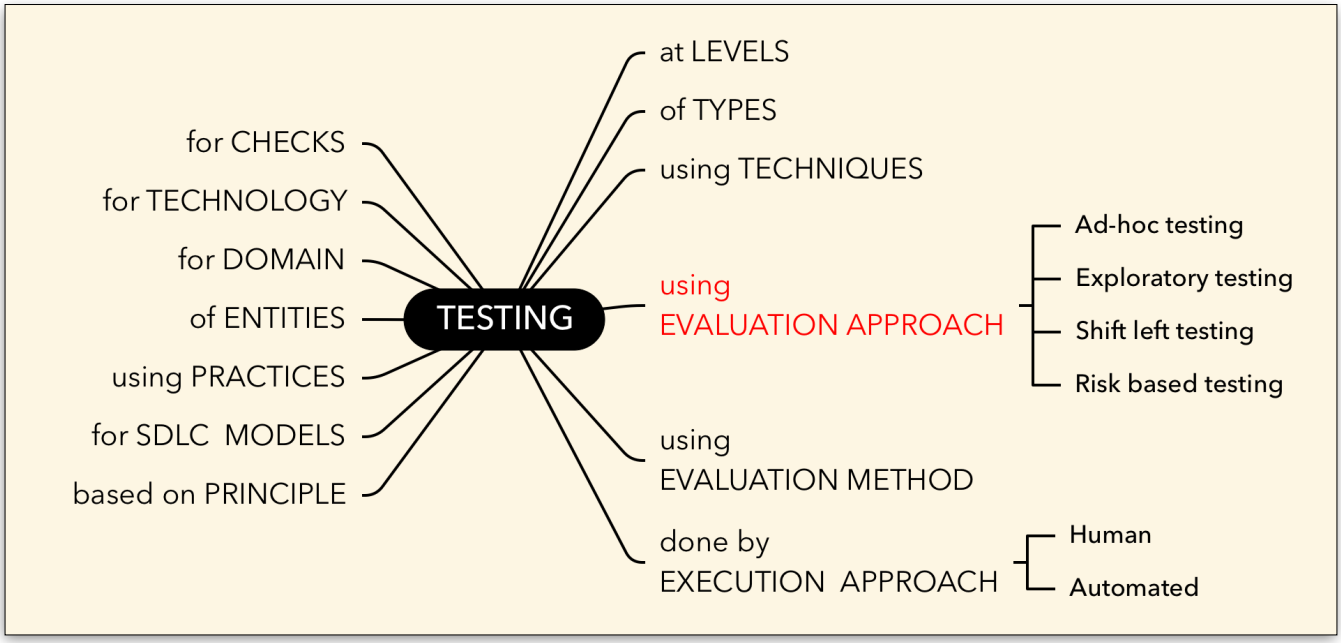
CATEGORY-4: BASED ON EVALUATION APPROACH

Is the approach to validation structured, ad hoc, exploratory, risk-based? This categorisation indicates how what *governs the high level view to validation*.

Ad-hoc testing an approach to testing a software in an unstructured manner with consciously applying any techniques to design test cases, to break the system using unconventional ways.
- Testbytes.

Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, test design and test execution. It is defined as"a style of software testing that emphasises the personal freedom and responsibility of the individual tester to continually optimise the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project."
- Wikipedia.

Risk-based testing (RBT) is a testing approach which considers risks of the software product as the guiding factor to support decisions in all phases of the test process.



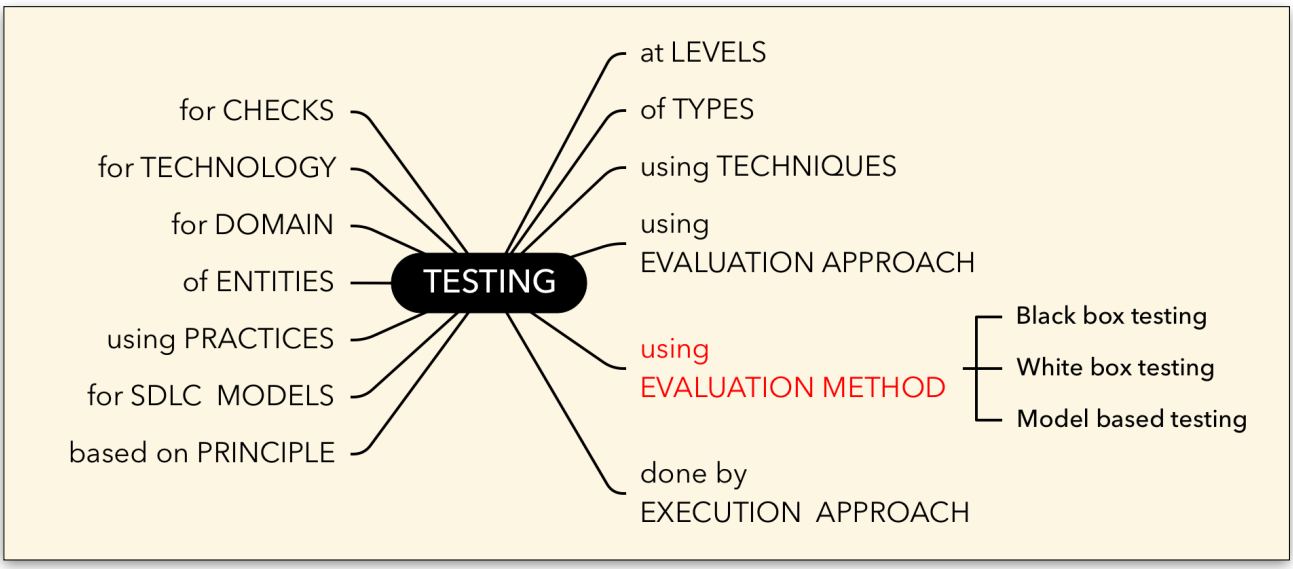
CATEGORY-5: BASED ON EVALUATION METHOD

What are we examining to validate? What information will be used to figure out the strategy and then to design? This category classifies term based on the method of evaluation.

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance. It is sometimes referred to as specification-based testing.
- Wikipedia

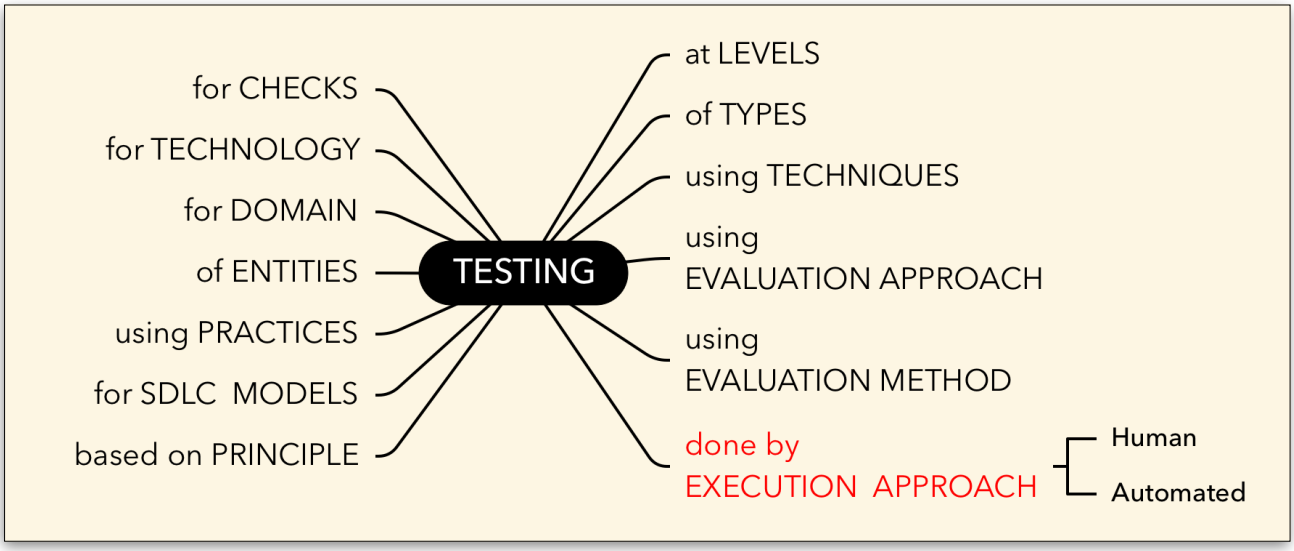
White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.
- Wikipedia

Model-based testing is an application of model-based design for designing and optionally also executing artefacts to perform software testing or system testing. Models can be used to represent the desired behavior of a system under test (SUT), or to represent testing strategies and a test environment.
- Wikipedia.

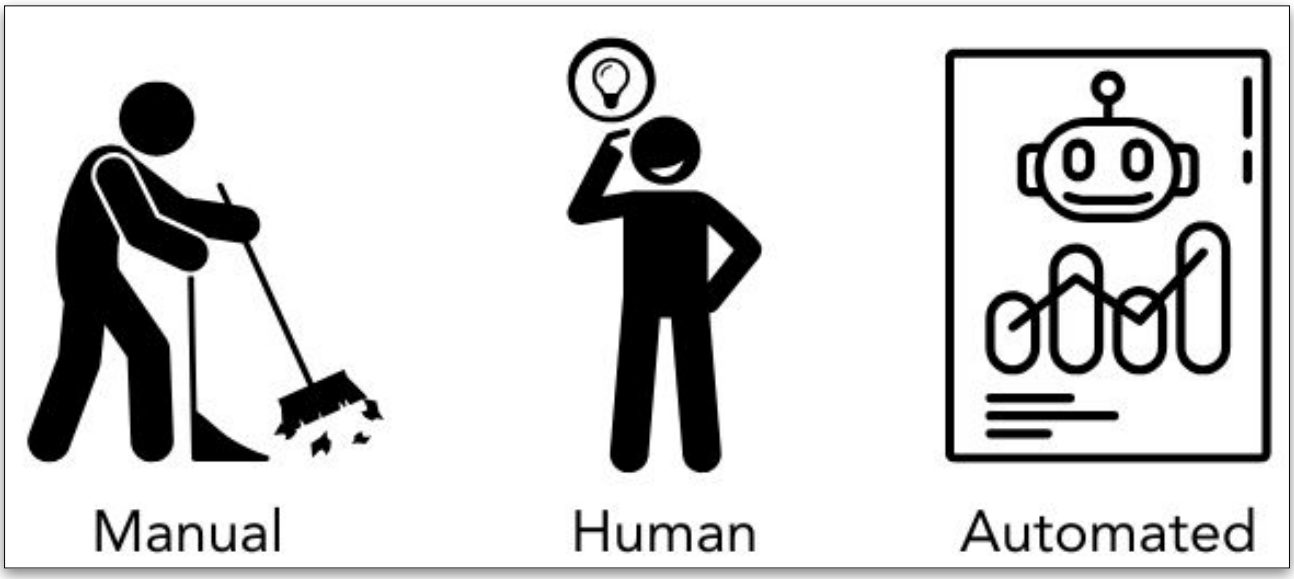


CATEGORY-6: BASED ON EXECUTION APPROACH

The terms "Manual testing" (Human testing really) and "Automated testing" indicate the method of execution of test cases, the former implying human being is executing the test(s) whereas in the latter it is machine that is executing the test cases.



Manual testing seems to connote the physical act of executing tests, observing results and judging correctness, in reality it should be termed Human testing that views the larger act of validating a system that uses intellectual capabilities to analyse, design and then execute physically , learn from context and improvise.



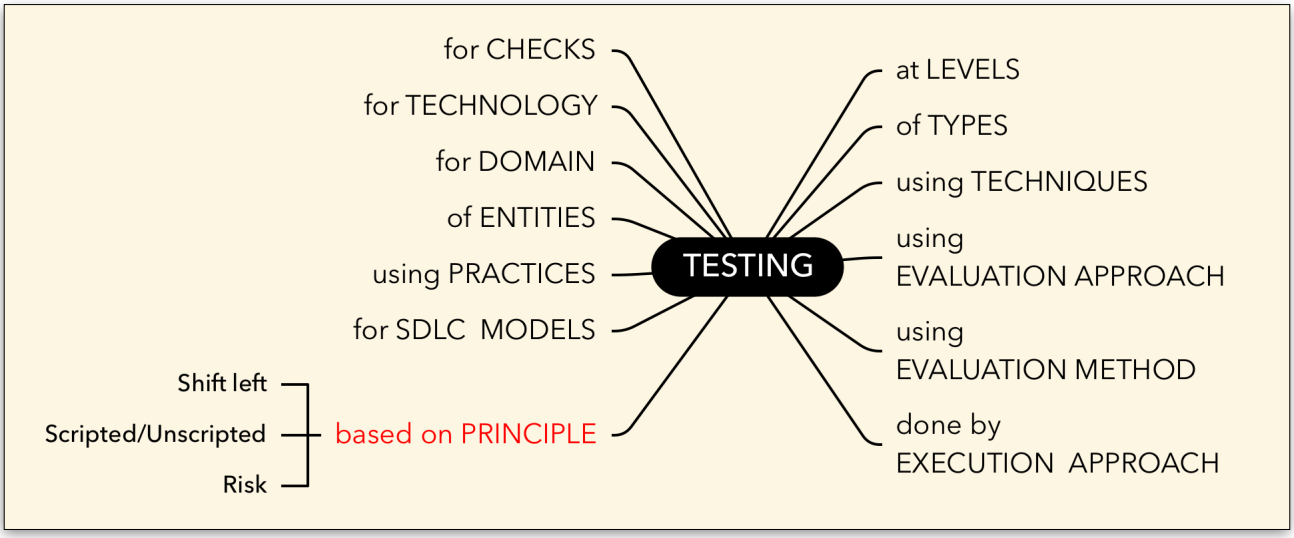
CATEGORY-7: BASED ON PRINCIPLE

Terms like Shift-left testing, scripted/unscripted testing and risk-driven testing indicate principles that govern validation. Some of the principles are:

- Pull forward activities early in the lifecycle to detect issues early, prevent at best
- Perform validation after thinking deeply and coming up with good scenarios/cases
- Exploit content and creativity to uncover new issues while testing
- Given that one can never prove the absence of issues, focus on business risk to focus on key areas/issues and not overdo too.

Shift left testing - While early testing has been highly recommended by software testing and QA experts, there has been a special emphasis on this agile approach of Shift Left testing that recommends reversing the testing approach and involving system/software testing earlier in the lifecycle. Practically, move the testing approach to the left end on the project timeline.

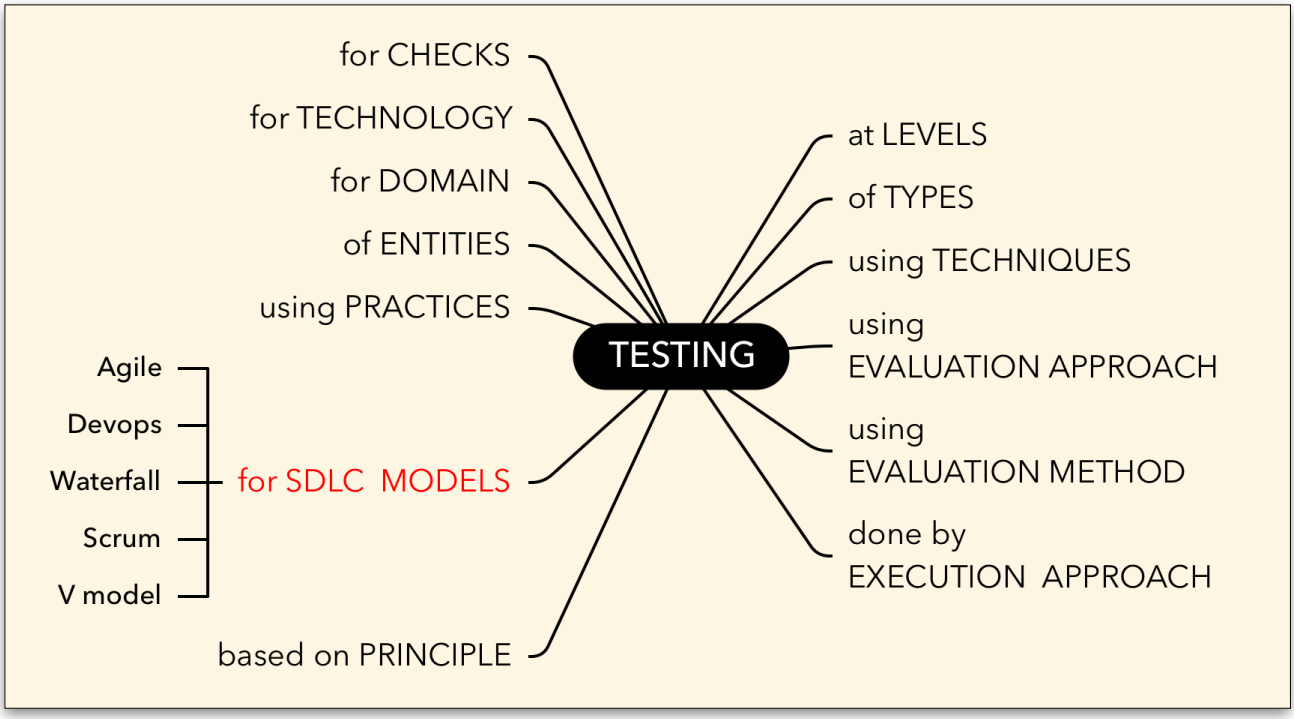
- Cigniti blog



CATEGORY-8: BASED ON SDLC MODELS

Given various SDLC models in vogue, terms like Agile testing, Devops testing etc., indicate when & what test activities are expected in the lifecycle and by whom. After all, different development lifecycle models demand different points of intervention, frequency and roles of validation.

Agile testing is a software testing practice that follows the principles of agile software development. Agile development recognises that testing is not a separate phase, but an integral part of software development, along with coding. Agile teams use a "whole-team" approach to "baking quality in" to the software product.
- Wikipedia



CATEGORY-9: BASED ON PRACTICES

What do CI/CD, TDD, BDD, ATDD etc. indicate? They signify wholesome practices to deliver great working code via sharp focus to finding issue early or preventing as in TDD at early dev stage and BDD, ATDD for complete system with CI/CD focussed on the practice of construction and deployment.

Continuous integration (CI) is the practice of merging all developer working copies to a shared mainline several times a day. Extreme programming (XP) adopted the concept of CI and did advocate integrating more than once per day.
- Wikipedia.

Continuous delivery (CD) is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, doing so manually. It aims at building, testing, and releasing software with greater speed and frequency. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery. CD contrasts with continuous deployment, a similar approach in which software is also produced in short cycles but through automated deployments rather than manual ones.
- Wikipedia

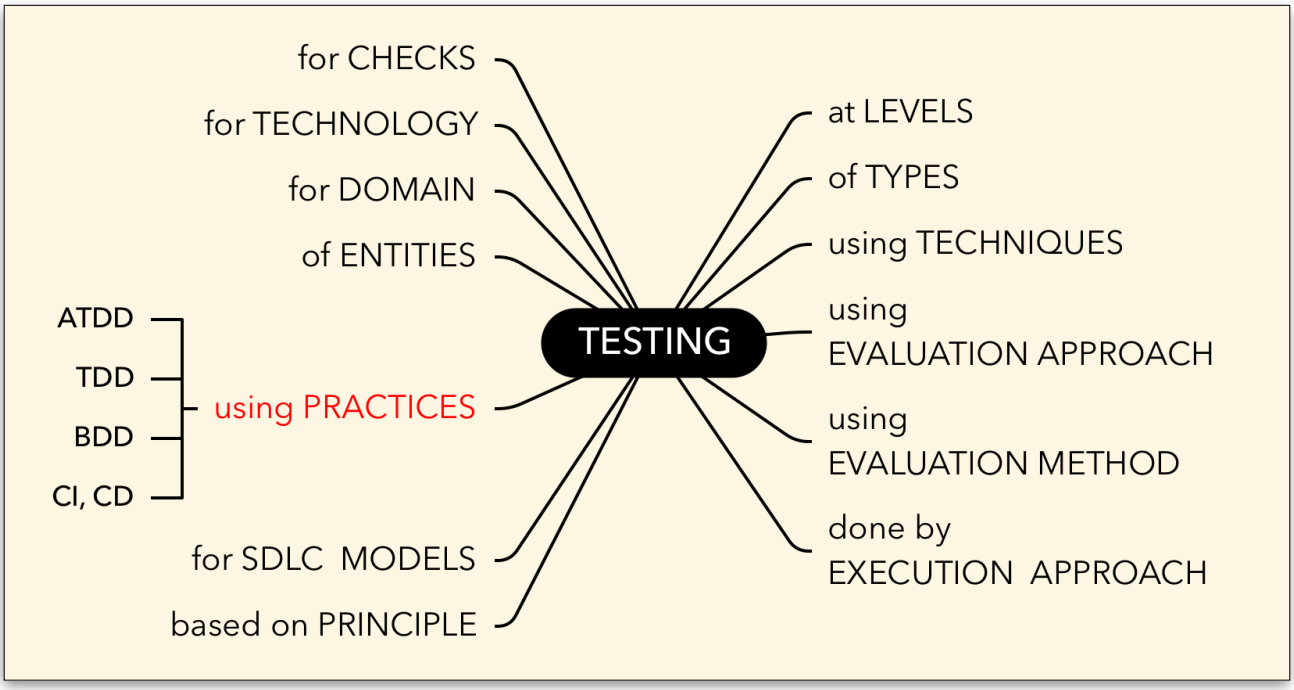
Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only. This is opposed to software development that allows software to be added that is not proven to meet requirements.
- Wikipedia

Acceptance test–driven development (ATDD) is a development methodology based on communication between the business customers, the developers, and the testers. ATDD encompasses many of the same practices as specification by example (SBE), behavior-driven development (BDD),example-driven development (EDD), and support-driven development also called story test–driven development (SDD). All these processes aid developers and testers in understanding the customer's needs prior to implementation and allow customers to be able to converse in their own domain language.

ATDD is closely related to test-driven development (TDD). It differs by the emphasis on developer-tester-business customer collaboration. ATDD encompasses acceptance testing, but highlights writing acceptance tests before developers begin coding.

- Wikipedia

Behavior-driven development (BDD) - This is software development process that emerged from test-driven development (TDD). Behavior-driven development combines general techniques and principles of TDD with ideas from domain-driven design and object-oriented analysis and design to provide software development and management teams with shared tools and a shared process to collaborate on software development.. - Wikipedia.



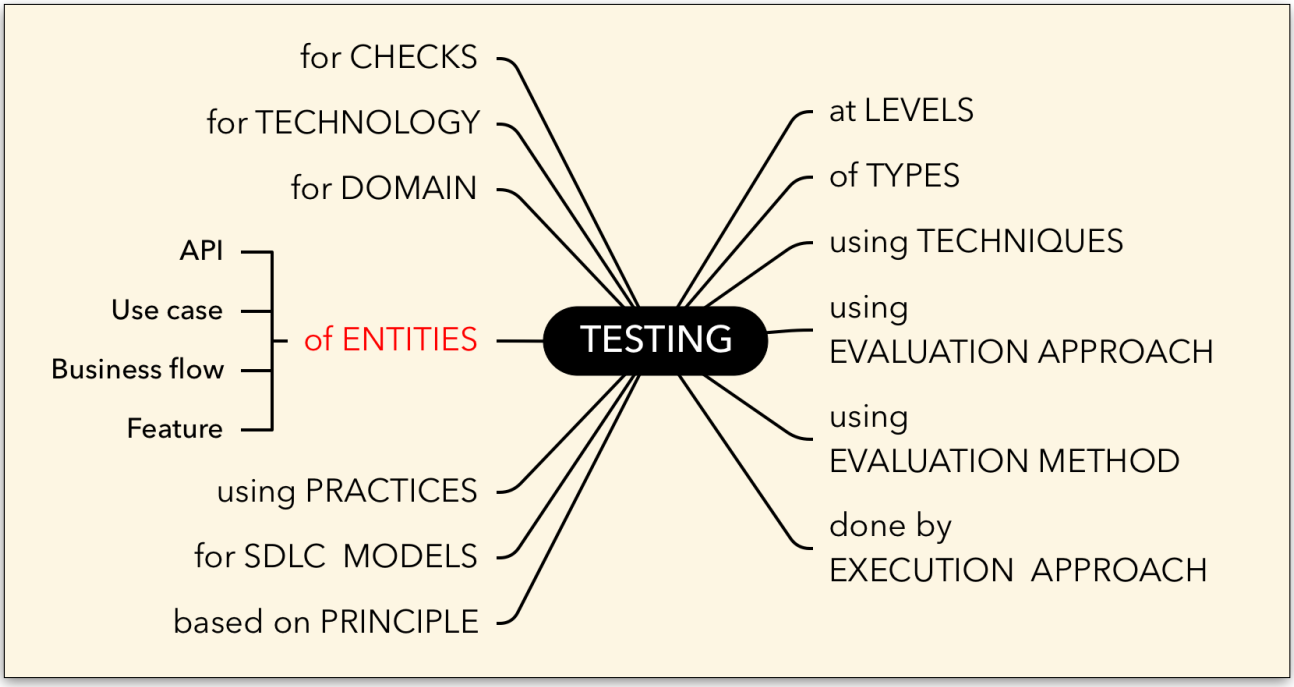
CATEGORY-10: BASED ON ENTITIES

Given that the objective of testing is to uncover key issues in a system, it is important to know where to look for these i.e. in what entities :

1. In the building blocks of a system i.e screens(GUI), API, libraries, components
2. In basic technical features i.e. the behaviour when building blocks are integrated
3. In end user requirements i.e. combination of features in a certain sequence
4. In end to end business flows that may involve different end users

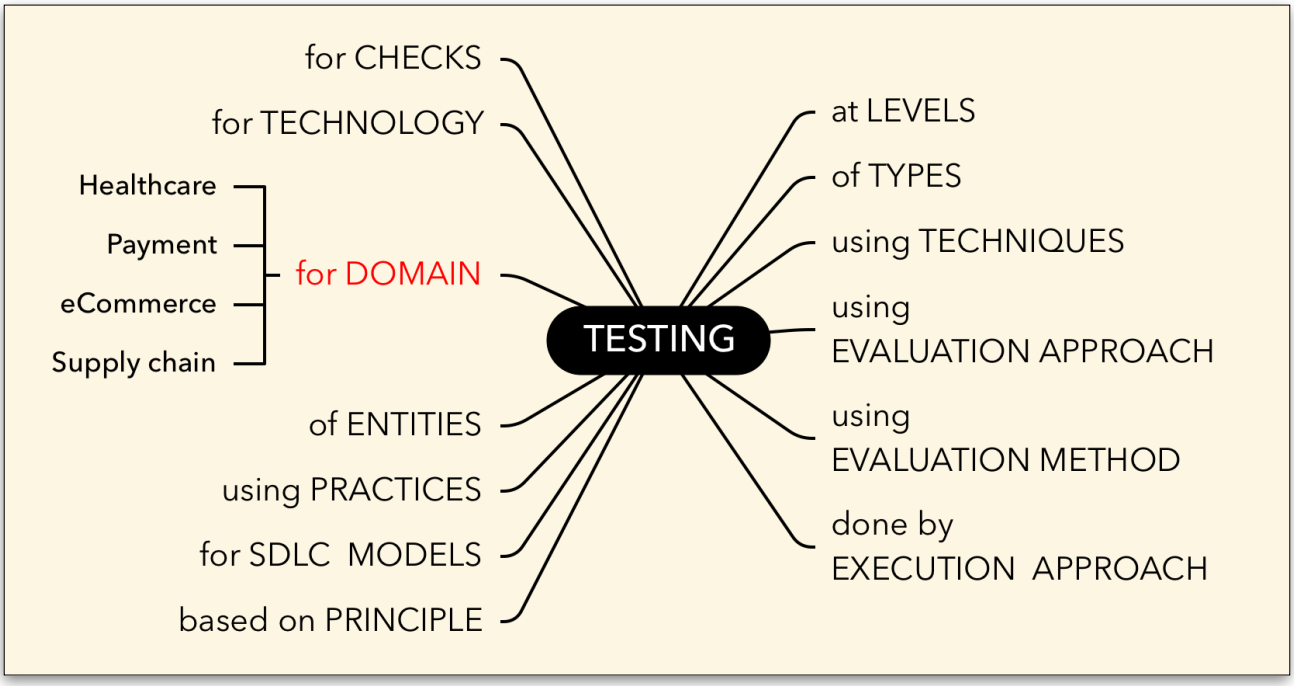
So when we say GUI/API/E2E testing it really means 'what-to-test' i.e. entities.

Note:The key concept "Entity Granularity in HyBIST(Hypothesis Based Immersive Session Testing) highlights this aspect to optimally partition what type of issues to look in different entities when (early/late) and by whom (Dev/QA).



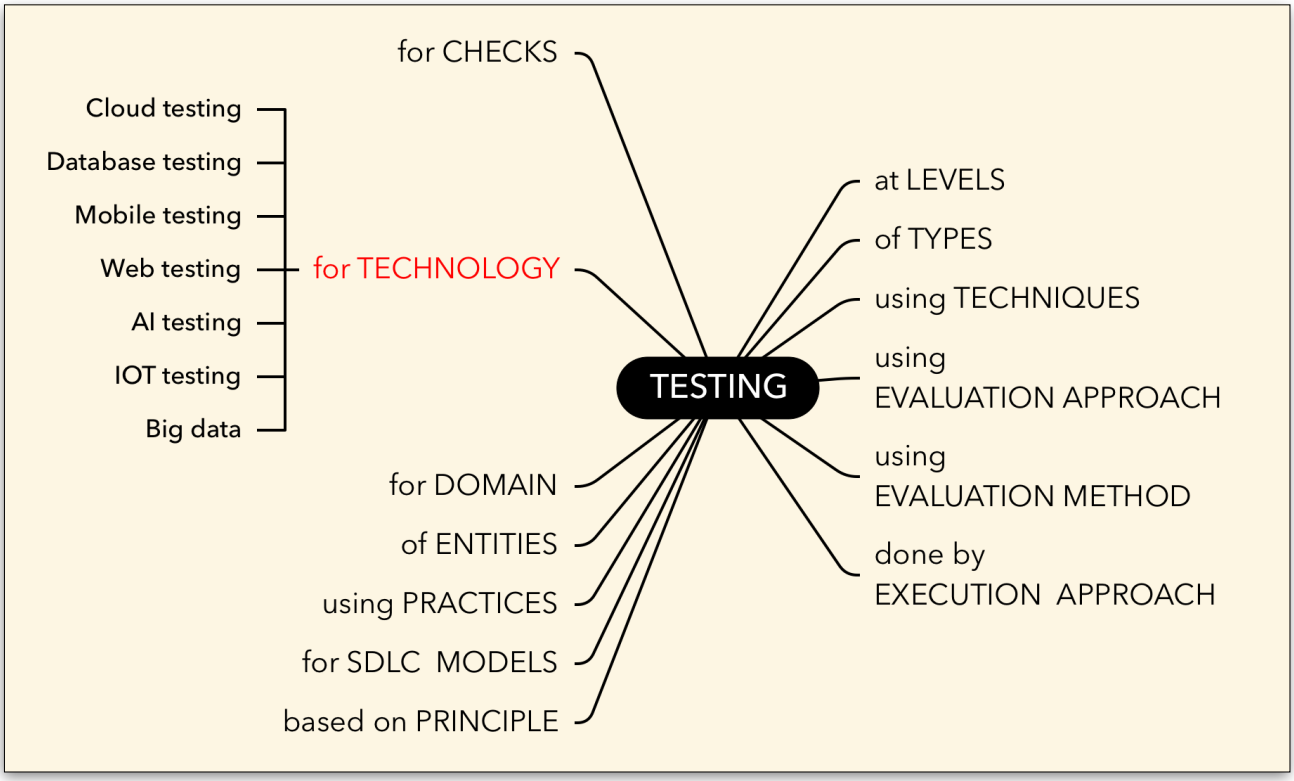
CATEGORY-11: BASED ON DOMAIN

What do terms like eCommerce testing, Supply chain testing etc. mean? Well they signify a business domain context that highlights specific end to end flows end users' expectations to focus on issues unique to that business. Note some domains maybe regulated and may require conformance to specific compliance standards.



CATEGORY-12: BASED ON TECHNOLOGY

When terms like "Database testing" or "Cloud testing" are used, they imply a focus on identifying specific issues unique to that technology. Unlike testing types, levels, techniques, or other categories, these terms highlight the intent to uncover unique issues arising directly from the technology's use.

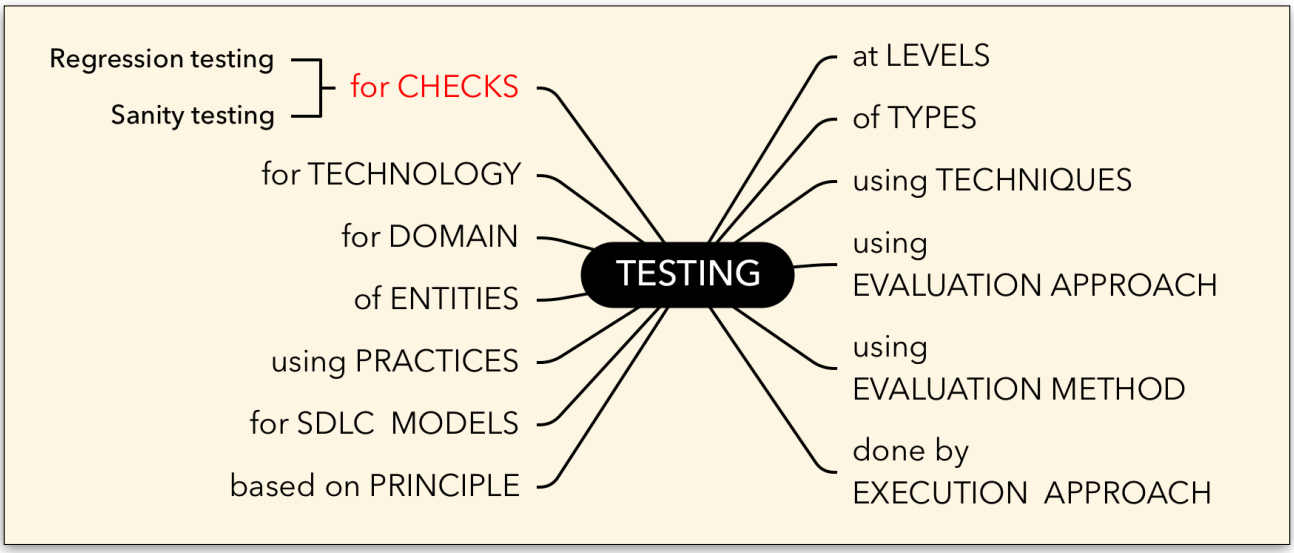


CATEGORY-13: BASED ON CHECKS

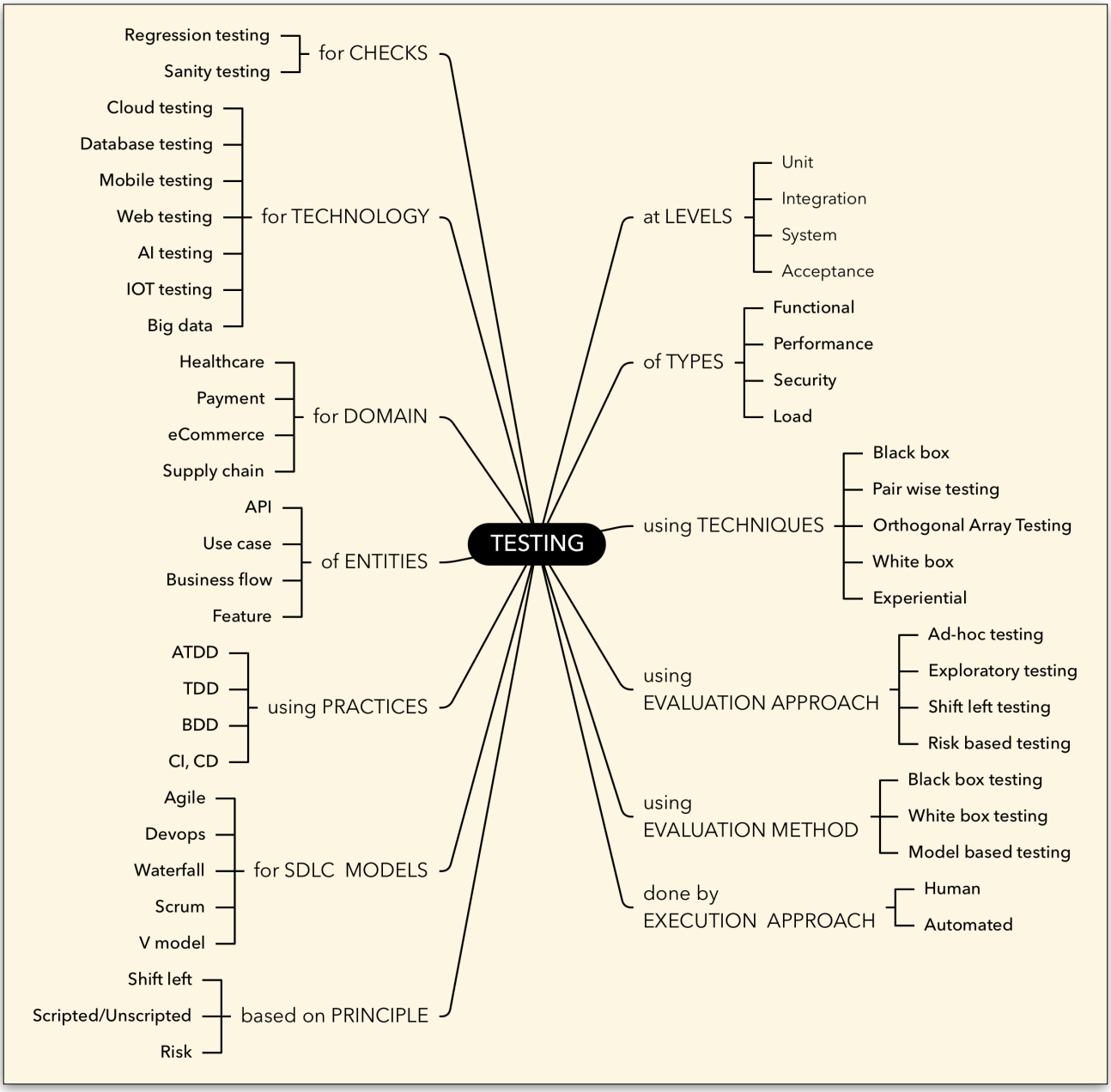
The final categorisation is based on checks for conformance i.e. not looking for new issues but ascertaining if existing system is not compromised and is fit.

Sanity testing is primarily a basic evaluation of system to ascertain if it basically good enough to go ahead so that we may proceed to do thorough testing.

Regression testing on the other hand is about checking if prior health of a system has not been compromised due to changes done. It simply means what was available & working earlier continues to do so.



CATEGORISATION OF TERMS - THE FULL PICTURE



Summarising,

Testing is looking for specific issues using specialised techniques & practices at different levels of various sized entities employing specific evaluation method, approach that may be built following various SDLC for specific business domains and technology executed in a human/automated manner while ensuring conformance all through.

The Symphony of Testing

At varied **levels**, tests take their place,
Each uncovering a unique trace.
Through **types** defined and **techniques** refined,
A path to quality is aligned.

With an **evaluation approach**, we steer,
And a trusted **method** keeps tasks clear.
Execution approaches tailored and wise,
Bring hidden truths to sharpened eyes.

Guided by **principle**, rooted and strong,
Across **SDLC models**, we journey along.
Through practiced hands and diligent care,
Entities thrive under watchful glare.

For every **domain** and **technology's** might,
Testing ensures the systems are right.
With **checks** in place, no flaw too small,
Testing secures the heart of it all.

- ChatGPT on 'Communicate Clearly'

*"**We are SmartQA evangelists.** For over two decades we have transformed how individuals, teams and organisations have practised testing. We espouse methodology to test intelligently. Our mission - Elevate to high performance via SmartQA."*
www.stagsoftware.com

The HyBIST Approach to SmartQA - MASTERCLASS
Testing is deep probing to seek clarity and in the process uncover, preempt issues rapidly. The HyBIST approach enables designing smart probes and probing the system smartly.
https://smartqa.academy/courses/smartqa-using-hybist

doSmartQA - AI based Smart Probing Assistant to interrogate, hypothesise issues, design & evaluate user story or a set of stories in a sprint rapidly. An assistant for smart session-based testing based on HyBIST.
Download personal edition from **here**

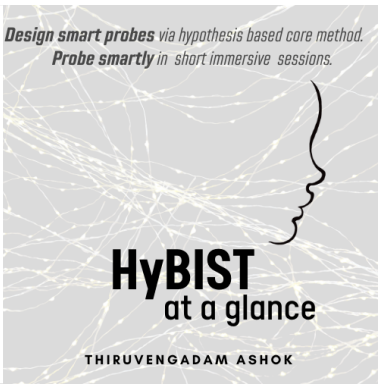
SmartQA Musings - A gentle flurry of interesting thoughts on smart assurance as a weekly webcast. A refreshing view of assurance to broaden & deepen thoughts/actions.
www.stagsoftware.com/subscribe

SmartQA Biweekly - Ignite your curiosity with fresh insights, thought-provoking ideas, and inspiring content delivered straight to your inbox every fortnight.
www.stagsoftware.com/subscribe

A rich collection of original content on smart assurance

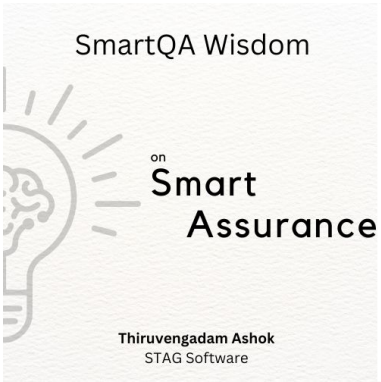
SmartQA eBooks - for Sr Engg/QA managers, Sr Test Practitioner & Young Test Practitioners too.

HyBIST at a glance
do SmartQA -The HyBIST Approach
High Performance QA
50 Tips for SmartQA



Download from www.stagsoftware.com/smartqa-ebooks

SmartQA Wisdom - Profound nuggets of wisdom to think deeply, do rapidly & smartly for Test Practitioners.



on Smart Assurance
on Personal Growth
on Test Design
on Smart Understanding
on Mindset & Habits
on Problem Solving

Download from www.stagsoftware.com/smartqa-wisdom