

Design smart probes via hypothesis based core method.

Probe smartly in short immersive sessions.



HyBIST at a glance

THIRUVENGADAM ASHOK

Unlock the Future of Smart Quality Assurance with HyBIST.

Explore "*HyBIST at a glance*" guide to revolutionise software testing with **Hypothesis-Based Immersive Session Testing (HyBIST)** methodology. This eBook introduces an intellectual approach that transcends traditional QA practices by emphasising **smart probing, intellectual testing, and rapid defect discovery**.

Key highlights of HyBIST

- **SmartQA redefined:** Shift from repetitive test execution to intelligent probing, ensuring deeper coverage with minimal effort.
- **Hypothesis-driven testing:** Learn to formulate smart hypotheses for risk-based testing and early defect detection.
- **Three-phase testing approach:** Master the phases of Reconnaissance, Exploration, and Recoup for focused, efficient testing sessions.
- **Key concepts & practices:** Gain insights into powerful concepts like 360 POV, Quality Levels, and Smart Probes while adopting effective practices such as Mindful Sessions and Focus & Meander.
- **doSmartQA Assistant:** Leverage the AI-powered HyBIST assistant tool for smarter, faster, and more insightful testing.

Why choose HyBIST?

- Enhance testing **efficiency** while reducing redundant efforts.
- Align **QA strategies** with modern Agile and DevOps environments.
- Achieve **faster releases** with higher software quality.
- Empower both **managers and testers** with practical tools for better decision-making and smarter testing.

Who should read this?

- QA Professionals seeking **modern testing strategies**.
- Agile & DevOps Teams aiming for **faster, smarter QA** practices.
- Test Managers wanting **data-driven insights** and efficient defect detection.

Transform your QA practice today! Dive into "*HyBIST at a glance*" and start building a smarter, faster, and more effective testing practice.

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.

Copyright © 2025, Thiruvengadam Ashok

All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations used in reviews and certain other noncommercial uses permitted by copyright law.

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

Chapter 1 : SmartQA - Setting the context	1
State of testing practice- The challenges	1
Challenges in today's software development	1
Need of the hour- Smart Assurance	1
What is SmartQA?	1
Key principles of SmartQA	2
Transition from traditional QA to SmartQA	2
SmartQA & HyBIST	2
HyBIST approach to SmartQA	2
The superior outcomes of HyBIST/SmartQA	3
Chapter 2 - Introduction to HyBIST	4
HyBIST Overview	4
The central concept	4
Why it matters	4
Inspirations for HyBIST	4
Role of hypotheses in testing	4
Comparison with traditional testing	5
Predefined test cases vs. Adaptive testing	5
Human vs. Automated testing	5
Probing vs. Checking	5
Role of HyBIST in solving the challenges	6
Why managers & testers should care about HyBIST	6
For managers: Assuring well & faster	6
For testers: Doing smarter	6
"The SHIFT" in HyBIST	7
From mere evaluation to PROBING	7
Emphasis on PRACTICE along with process	7
Leverage INTELLECT, not just machine	7
Beyond mere compliance to DISCOVERY	8
Experience based to METHOD oriented	8

Preempt, not just detect	8
To doing less, not more	9
DIVE into HyBIST	9
The key idea is PROBING	9
SMART Probes & Probing	9
The three phases of HyBIST	10
Key concepts of HyBIST	10
Key practices of HyBIST	10
Chapter 3 - KEY CONCEPTS in HyBIST	12
360 POV	12
Entity granularity	13
Quality level	14
Orthogonality principle	15
Landscape	16
Product Map	17
Test Scope	18
Design Method	19
Perturbation Analysis	20
Potency & Immunity	21
Smart Checks	22
Defect Map	23
Chapter 4 - KEY PRACTICES of HyBIST	24
Note taking	24
Language style	24
Survey, Swoop & Iterate	24
Focus & Meander	24
Chapter 5 - HyBIST in PRACTICE	25
Validate user story	25
Validate an entity	25
Validate a sprint validate a user story	25
Chapter 6 - HyBIST Assistant - doSmartQA	26

What is doSmartQA?	26
Key functions of doSmartQA	26
WHO benefits and HOW	27
Chapter 7: Conclusion - The Future of Smart QA with HyBIST	28
A quick recap	28
SmartQA philosophy:	28
HyBIST methodology:	28
Key concepts	28
Key practices	28
doSmartQA Assistant	28
The transformative impact of HyBIST	28
Why HyBIST matters for modern QA	29
The path forward	29

CHAPTER 1 : SMARTQA - SETTING THE CONTEXT

State of testing practice- The challenges

Software development has significantly evolved and matured, but practice of QA is still archaic. It is caught between do-more/test-continuously and preempt/do-less on the other, without a clue on how to marry them in simple terms.

There is extreme reliance on experience and rote continuous testing via automation, rather than smart assurance by harnessing human intelligence. The problem is capacity throttling due to delayed and excessive tasks, struggling to match the speed of rapid development, causing frustration for the tester(s).

Challenges in today's software development

Increased Complexity: Modern software systems are far more complex than in the past, with numerous interdependent components and integrations (e.g., microservices, third-party APIs, cloud infrastructure).

Faster Development Cycles: Agile and DevOps practices have shortened development cycles, demanding faster releases. This requires QA to be agile and adaptive, ensuring quality without becoming a bottleneck.

More Demanding Users: User expectations for reliability, performance, and security have never been higher. Even a minor issue can lead to a significant loss in user trust or revenue.

Limited Time and Resources: Testing teams are often pressed for time, resources, and budget. Prioritising the most critical tests using a hypothesis-driven approach is essential to ensuring coverage within these constraints.

Need of the hour- Smart Assurance

It is shifting focus from evaluating to probing, following a process to setting up an intelligent practice, from rote automation(machine) to exploiting intellect , checking (for compliance) to discovering(issues), reliance on experience to applying logical method(s), detection to preempting and striving to doing less testing rather than more.

What is SmartQA?

SmartQA is about probing deeper to seek clarity and in the process uncover/preempt issues rapidly, not limited to mere validation of code.

SmartQA is a quality assurance philosophy that goes beyond traditional testing. It focuses on being efficient, adaptive, and smart in the way tests are designed, executed, and analysed. SmartQA encourages testing strategies that evolve with the software development process, combining automation, intelligence, and human insight.

Key principles of SmartQA

Do Less, Accomplish More: Focus on the most critical parts of the system rather than testing everything. SmartQA aims to reduce redundant tests and unnecessary cycles, promoting efficiency in testing.

Human-Powered, Machine-Aided: SmartQA values human intellect in testing. While automation is important, real insight often comes from humans probing the system, thinking critically, and forming hypotheses. Automation should support human testers, not replace them.

Prevention over Detection: SmartQA advocates for testing early in the development process (Shift Left Testing) to prevent issues from making it to production. The goal is to preemptively catch problems before they become defects.

Constant Adaptation: Testing is not static. SmartQA encourages continuous refinement of test cases, scenarios, and approaches based on real-time feedback and system behavior.

Transition from traditional QA to SmartQA

Traditional QA Mindset: Testing is often treated as an isolated step in the development process, executed after development is complete. This can lead to inefficiencies and missed defects.

SmartQA Mindset: Testing becomes an integral, continuous part of the development cycle. SmartQA shifts testing left, meaning testing starts as early as possible (even during requirement gathering or design phases). HyBIST, with its emphasis on hypotheses and immersive sessions, is the tactical implementation of this philosophy.

SmartQA & HyBIST

HyBIST is a natural evolution of SmartQA principles. SmartQA encourages intellectual testing, shifting focus from repetitive manual checks to insightful explorations of potential system weaknesses. This aligns perfectly with HyBIST's hypothesis-driven approach.

HyBIST approach to SmartQA

Hypothesis-Based Immersive Session Testing (HyBIST) approach to SmartQA is about designing smart probes and probing the system smartly. What is probing? It is about being logical, curious & observant, to understand what is in there, what may be expected/intended and in the process of exploration spot anomalies, issues, observations, come up with suggestions, a significantly elevated approach.

A hypothesis based method consisting of 'key concepts' enables logical design of smart probes whilst 'key practices' ensure rapid expansive probing of the system in immersive sessions.

The superior outcomes of HyBIST/SmartQA

The outcomes of typical TESTING are: Test plan, test cases, defects, scripts (driven by templates), larger documentation, extreme focus on execution with lot of rote activities largely driven by experience.

On the other hand the outcomes of SMART probing are:

Questions, observations, suggestions, ideas in addition to test plan, scenarios, defects, scripts. These are NOT template driven and are about : write less, to do well now, focus on what-if & what, driven by logic & outcomes not relying on experience only.

CHAPTER 2 - INTRODUCTION TO HYBIST

HyBIST Overview

Hypothesis-Based Immersive Session Testing (HyBIST) is an intellectual practice of probing via hypothesis based core method to analyse & design, and do immersively in sessions to test deeply & rapidly.

Validation is done in THREE phases of Reconnaissance, Exploration & Recoup using 'key concepts' to design smart probes and 'key practices' to probe system expansively and rapidly in short immersive sessions.

The central concept

Unlike traditional test strategies that rely on predefined test cases, HyBIST encourages testers to create hypotheses about potential risks, failures, or edge cases based on a deep understanding of the system. These hypotheses guide immersive testing sessions aimed at validating or disproving them.

Why it matters

As systems grow more complex and users demand faster releases, traditional testing approaches often fail to keep up. HyBIST bridges the gap by enabling more targeted and efficient testing that focuses on areas most likely to break or cause issues, thus improving the overall quality of software.

Inspirations for HyBIST

The key inspirations to science and engineering of testing were:

- Properties of matter to state purity in terms of criteria & potential issues
- Sherlock Holmes-ian deductive thinking to hypothesis & proof
- Fractional distillation of separating elements of a mixture,
- The concept of 'flow' to be mindful, immersed
- The fishnet as a metaphor to understand test effectiveness

Role of hypotheses in testing

Instead of randomly testing parts of the application or relying solely on regression tests, HyBIST involves forming intelligent hypotheses about system vulnerabilities. These hypotheses may involve security risks, performance bottlenecks, or functional edge cases.

Examples of Hypotheses:

"The system may fail if the payment gateway is slow to respond."

HYBIST AT A GLANCE

"The search functionality might not return results when complex filters are applied."

"An error might occur if the user cancels the booking process after seat selection."

The benefit - Testing based on hypotheses ensures that testers focus on areas with the highest risk of failure, thus reducing time spent on unnecessary tests and increasing the likelihood of catching critical issues

Comparison with traditional testing

Predefined test cases vs. Adaptive testing

Traditional testing strategies involve creating predefined test cases based on the system's requirements. While effective in many cases, this approach is rigid and may not adapt to unexpected issues. HyBIST, on the other hand, is dynamic and evolves during the testing process based on real-time insights.

Human vs. Automated testing

HyBIST is not limited to manual testing. While it emphasises intellectual testing (manual investigation of potential issues), it can work in conjunction with automation. Hypotheses can be used to guide automation scripts, ensuring that automated tests focus on the most critical areas.

Probing vs. Checking

HyBIST encourages probing (exploring new potential issues) rather than simply checking if something works as expected. This results in a deeper understanding of the system and more comprehensive test coverage.

ASPECT	TRADITIONAL TESTING	HYBIST
Focus	Predefined test cases based on requirements	Hypotheses about potential issues
Test design	Static, planned in advance	Dynamic, evolves during testing
Execution	Scripted, often repetitive	Immersive, focused & creative
Defect detection	Post-code defect detection	Preempt & Prevent
Test adaptability	Rigid, may not adapt well to changes	Adaptive, responds to real-time findings
Automation role	Primarily regression checks	Complements human, hypothesis-driven sessions
Test coverage	Broad, often shallow, experiential	Deep, driven by hypothesis & logic

HYBIST AT A GLANCE

ASPECT	TRADITIONAL TESTING	HYBIST
Time efficiency	High time/resource expenditure	Higher through smarter testing

Role of HyBIST in solving the challenges

Efficient testing through hypotheses: HyBIST helps testers prioritise what to test by focusing on areas of the system that are most likely to cause problems. This ensures that time and resources are spent wisely.

Immersive sessions for deeper exploration: Rather than spending time running scripted tests, HyBIST allows testers to dive deep into specific areas of the application, exploring real-world behaviours and potential issues through immersive sessions.

Early defect detection (Shift-Left): HyBIST aligns with the Shift Left Testing approach, encouraging early-stage validation to catch issues before they become expensive to fix. Testers engage with the system from the beginning, forming hypotheses as soon as requirements are available.

Balancing automation & human testing: HyBIST works alongside automation by guiding it with hypotheses. Automation handles repetitive checks, while immersive sessions explore the system deeply and intelligently.

Why managers & testers should care about HyBIST

For managers: Assuring well & faster

Focus on business value: Managers need to ensure that QA efforts contribute directly to business goals. HyBIST helps align testing efforts with business value by focusing on high-impact areas and risks.

Increased efficiency: By reducing redundant tests and focusing on what matters most, HyBIST helps teams save time and deliver high-quality software faster.

This leads to reduced time-to-market and better resource allocation.

Data-driven decision making: HyBIST provides actionable insights. By constantly refining hypotheses and tracking defect escape rates, managers can make informed decisions about the readiness of a product for release.

For testers: Doing smarter

Develop critical thinking: HyBIST encourages testers to think critically about what might go wrong and how to test for it. This makes testing more engaging and intellectually stimulating.

Better test coverage: By focusing on potential issues, testers can ensure they are covering the most critical aspects of the system without wasting time on unnecessary tests.

Adapt to rapid development cycles: HyBIST fits perfectly in Agile and DevOps environments, allowing testers to adapt to continuous changes in the system and requirements.

“The SHIFT” in HyBIST

From mere evaluation to PROBING

Probing is deep questioning to gain clarity and preempt/uncover potential issues. Unlike traditional code validation, it goes beyond mere confirmation, focusing on proactive discovery and prevention of risks. Rapid generation of questions, suggestions, observations, and ideas is integral to illuminating uncertainties and uncovering latent problems.

At its core, probing fosters curiosity to understand well and unravel issues. It moves away from conventional test practices such as being limited by plans, scenarios that focus on “how-to”. It emphasises a flexible and adaptive mindset not limited by templates to a broader & deeper spirit of inquiry.

Emphasis on PRACTICE along with process

Process is a set of activities we do, normally agreed upon by stakeholders in an organisation. Process is great in enabling overall discipline but the “individual habits” of good practice are vital to being effective and rapid.

Practice on the other hand is how we do these activities, largely driven by individual brilliance. It is about how we breakdown problems, ideate solutions and rapidly implement them. It is about techniques, principles, guidelines(heuristics) that one uses to solve problems.

The shift has been to make processes light so that they are nimble and responsive. Process is like route map whilst Practice are driver skills to get to the destination quickly and safely. Equipping an individual with good method(s) that when practiced continually enables one to deliver great work, the outcome of ‘practice’. Mere processes don’t cut it; however when good individual practices become common place, they get converted to “Best/Good practice” becoming part of the process.

Leverage INTELLECT, not just machine

Testing is more than assessing whether something is correct or incorrect. It is an exploration into the unknown, of discovering interesting stuff that demands an intelligent human. Testing is not always postcode-based; it is about doing earlier to detect, prevent, or to enhance clarity.

Figuring out test approach, understanding behaviour and identifying the conditions, understanding the human psyche, the environment in which they use it, and the constraints, expectations, and potential mistakes that one might make are things that require human intelligence.

If we can do faster using a machine, then do so. If some things can be observed, perturbed, or exercised by using a tool or technology, then it is 'smart' to exploit it to do so. But understanding, figuring out questions, connecting various parts to see the big picture and using that to evaluate requires human intelligence.

It is necessary to exploit human intelligence, as testing is not merely a notion of evaluation at the end. It demands a systems-thinking approach to "look at individual trees but also understand the larger context of the forest". Human intelligence plays a definitive role, and today it is aided by the intelligent tools that are more supplemental than replacement.

Repetition of tests and probing that is hard to be done by a human are those that demand tools/technology i.e., machine intelligence, that we state as automated testing. A harmonious mixture of machine intelligence and human intelligence helps deliver clean code.

Beyond mere compliance to DISCOVERY

Testing is not a mere compliance-driven activity to validate if meets specification/needs, rather it is probing to understand well and see out potential interesting issues.

It is about questioning, embracing uncertainties, and discovering scenarios in not explicitly stated in specifications. It's a human-powered exploration, not always scripted, which leads to intriguing observations or potential issues. These observations might not always yield a binary outcome but prompt further consideration based on their significance to the project.

Testing delves beyond specifications, akin to evaluating system wellness rather than just compliance. It necessitates human intelligence due to its probing nature, although some observations might be automated for subsequent checking.

Experience based to METHOD oriented

Domain experience seems to be play a pivotal role to be able to detect issues. Is that good enough? What if we do not have a very deep domain experience? That is where the science & engineering of testing becomes necessary, the METHOD.

Experience is indeed very useful and necessary but not sufficient. The act of bounding the system user test, the ability to go beyond the stated boundary to seeing clarity, hypothesising probable issues and devising means to preempt/detect them is where the METHOD becomes paramount.

Preempt, not just detect

Testing is more often seen as an intense activity of evaluation. Is it just about executing or running? Not really. It requires good preparatory work before the act of evaluation, to understand the big picture, ask interesting questions about who, where, what, why, what-if and so on and in the process find missing information, seek clarifications,

HYBIST AT A GLANCE

discover potential issues, jot down observations, and come up with suggestions too. Not just viewing testing as an act of evaluation but pulling it forward. This process helps clarify things and preempt issues.

Validation is not a physical act of execution; it is intellectual, an invisible process that goes on in one's brain to figure out possibilities and probabilities of what may go wrong, what may happen that is not yet spec'd out, and come up with interesting questions to answer to preempt, to assure. Just like for good health, activity or exercise is good for the body, but rest and sleep are equally important.

Testing is not always an intense activity, it is silent thinking, quietly exploring in the mind's eye too, to visualise correct flows and see issues to preempt.

To doing less, not more

In today's world, there is an extreme focus on speed, the need to test quickly, to test frequently so that build/release is not compromised. The focus is on automation to accomplish this. Is this good enough?

It is not about doing more, and more quickly it is about doing smart QA to do less to accomplish more. It is about:

- don't do at all (prevent) - sensitive to probable & preempt.
- do early - detect issues early so that they do not fester & become a pain later.
- do optimally - regress smartly, test just enough.
- do less - delete via smart automation.

DIVE into HyBIST

The key idea is PROBING

Probing is deep questioning to gain clarity and preempt/uncover potential issues. Unlike traditional code validation, it goes beyond mere confirmation, focusing on proactive discovery and prevention of risks. Rapid generation of questions, suggestions, observations, and ideas is integral to illuminating uncertainties and uncovering latent problems.

At its core, probing fosters curiosity to understand well and unravel issues. It moves away from conventional test practices such as being limited by plans, scenarios that focus on "how-to". It emphasises a flexible and adaptive mindset not limited by templates to a broader & deeper spirit of inquiry.

SMART Probes & Probing

The HyBIST approach to SmartQA is smart probing i.e., designing smart probes and probing a system smartly.

Smart PROBES are those that help uncover issues. There may be viewed as : (1) Intellectual probes that really are observations, questions, suggestions, criteria, potential

issues & conditions, ideas integral to illuminating uncertainties and uncovering latent problems to help in *preempting* issues and (2) Executable probes which really SmartChecks and test cases to *detect* issues.

Smart PROBING relates to the act of probing and is about (1) being immersive i.e mindful, take notes in free form , focussing to understand depth and also meandering to understand the broader picture and (2) doing in short sessions of Reconnaissance, Exploration & Recoup.

The three phases of HyBIST

The act of probing is performed in three phases - Reconnaissance, Exploration & Recoup, in short focussed sessions.

- Reconnaissance: Get a big picture of system, and create maps to explore.
- Exploration: Dive deep to understand entities and then evaluate them.
- Recoup: Analyse what has been done, learn and course correct.

Key concepts of HyBIST

The TWELVE key concepts of HyBIST is either a technique/principle/guideline. These assist you in logically dissect, analyse and understand information to sharpen clarity, preempt issues, figure out test approach, design & review scenarios/cases, analyse impact to regress smartly and assess practice.

The TWELVE key concepts are :

- 360 POV
- Entity granularity
- Quality level
- Orthogonality principle
- Landscape
- Product map
- Test scope
- Design method
- Perturbation analysis
- Potency & Immunity
- Smart checks
- Defect map

Key practices of HyBIST

The FIVE practices of HyBIST enable one to immerse deeply, respond rapidly and stay connected to the context.

The FIVE key practices are :

HYBIST AT A GLANCE

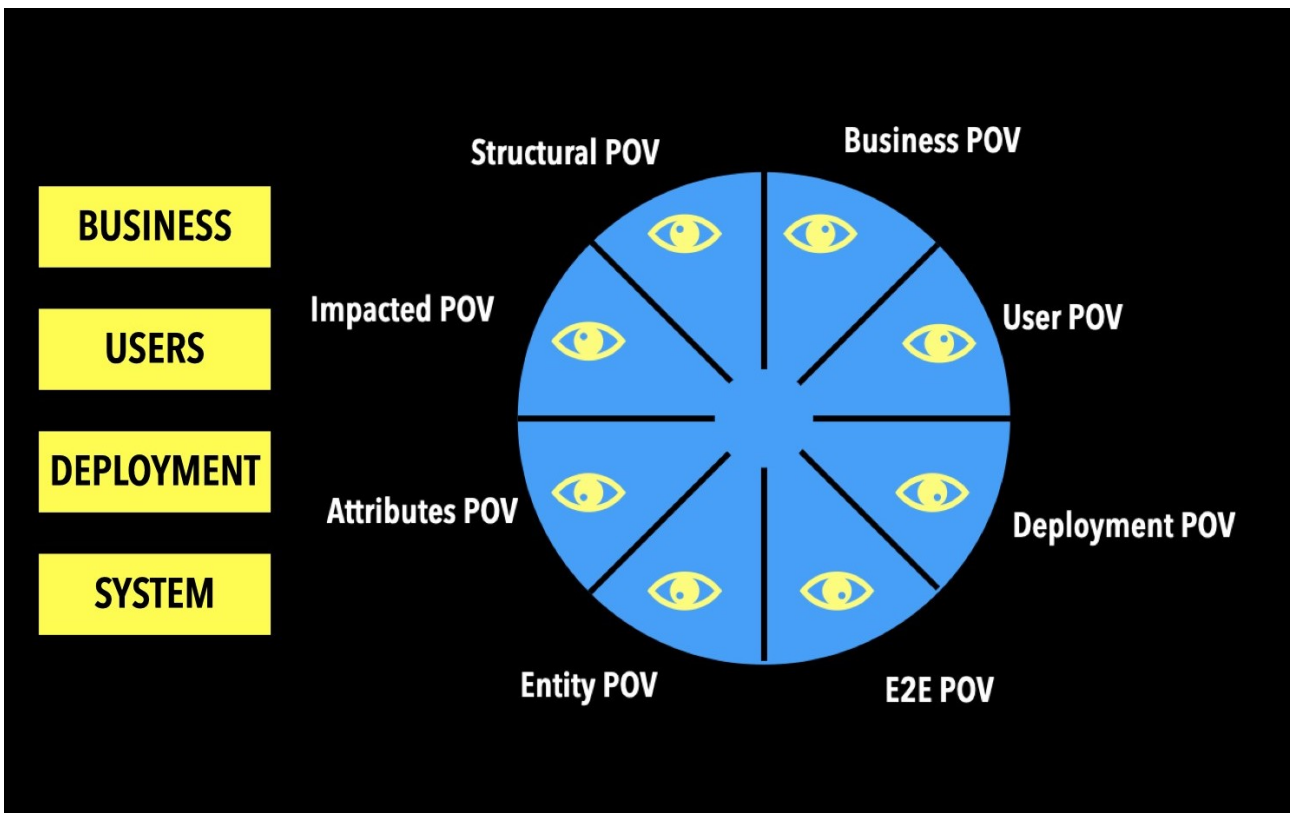
- Note taking
- Mindful sessions
- Language style
- Survey, Swoop & Iterate
- Focus & Meander

CHAPTER 3 - KEY CONCEPTS IN HYBIST

360 POV

See system/entity from 360 view to get a holistic picture to review/test/retest smartly.

This is about seeing system or entity in a comprehensive manner, examining it from multiple perspectives: business objectives, end-user needs, deployment strategies, and internal structural aspects. This holistic approach helps understand the entire system's

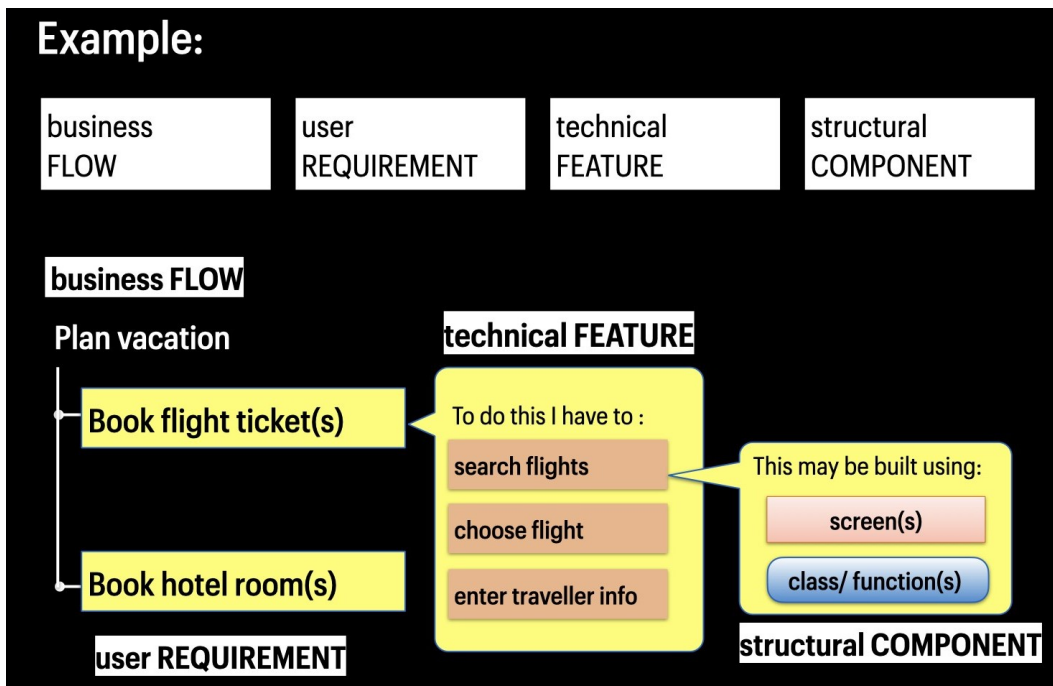
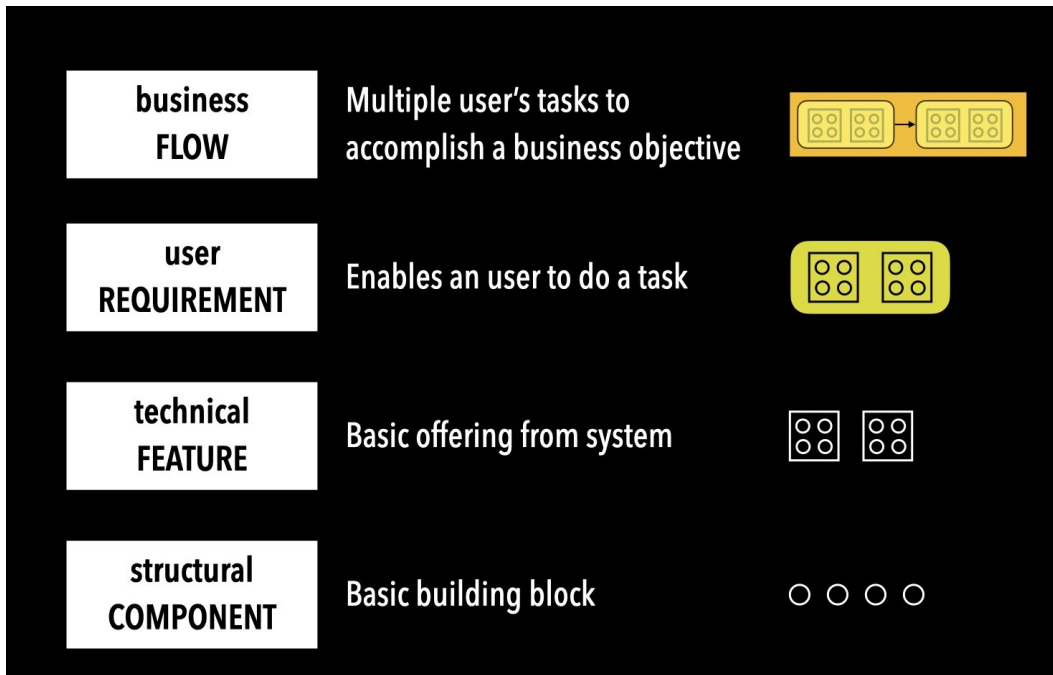


functioning.

Entity granularity

See the system as composed of different sized entities to characterise system well and test smartly.

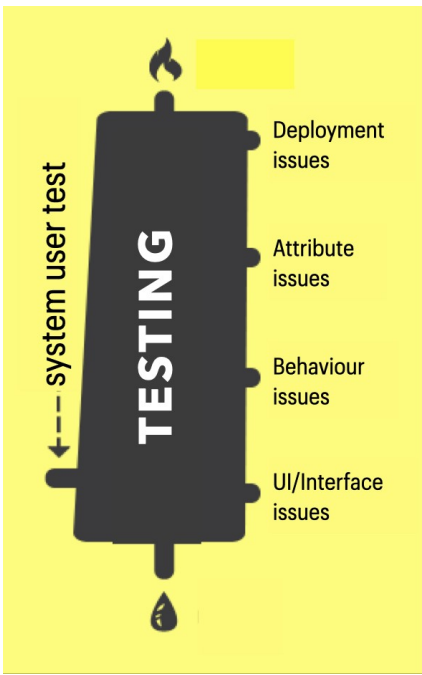
This is about seeing a system as a collection of different-sized entities, from small structural components to complete business flows, aiding in understanding system composition for effective testing. In HyBIST the various entities are - structural COMPONENT, technical FEATURE, user REQUIREMENT & business FLOW.



Quality level

See validation as multi staged filtration.

This is about treating validation as a multi-staged filtration process, akin to fractional distillation, with the aim to separate and uncover different issues optimally at different levels that consists of specialised focussed tests.



apply this to TESTING SOFTWARE

Think of SUT as a mixture of issues.

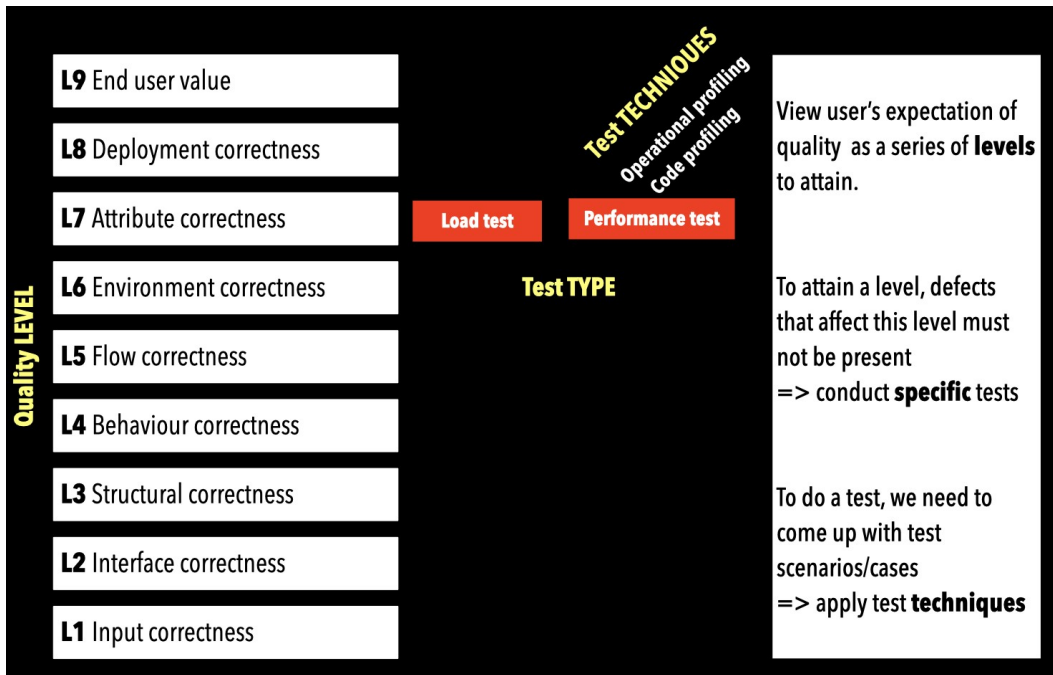
Different tests uncover different issues at various levels. Sharpens focus on what to find, how to find and when. Enables scenarios to be minimal, crisp and complete.

L9 End user value	That end user expectations are met
L8 Deployment correctness	That it deploys well in the real environment
L7 Attribute correctness	That expectations (i.e. attributes) are met
L6 Environment correctness	That it works well on intended environments
L5 Flow correctness	That end-to-end flows is correct
L4 Behaviour correctness	That functional behaviour is correct
L3 Structural correctness	That the internal structure is robust
L2 Interface correctness	That interfaces are clean
L1 Input correctness	That bad inputs are rejected

Orthogonality principle

See validation as multi staged filtration. Understand Levels, Types & Techniques are orthogonal.

This establishes a relationship between quality levels, test types, and design techniques. By applying specific tests and techniques at corresponding levels, one can uncover various potential issues optimally.



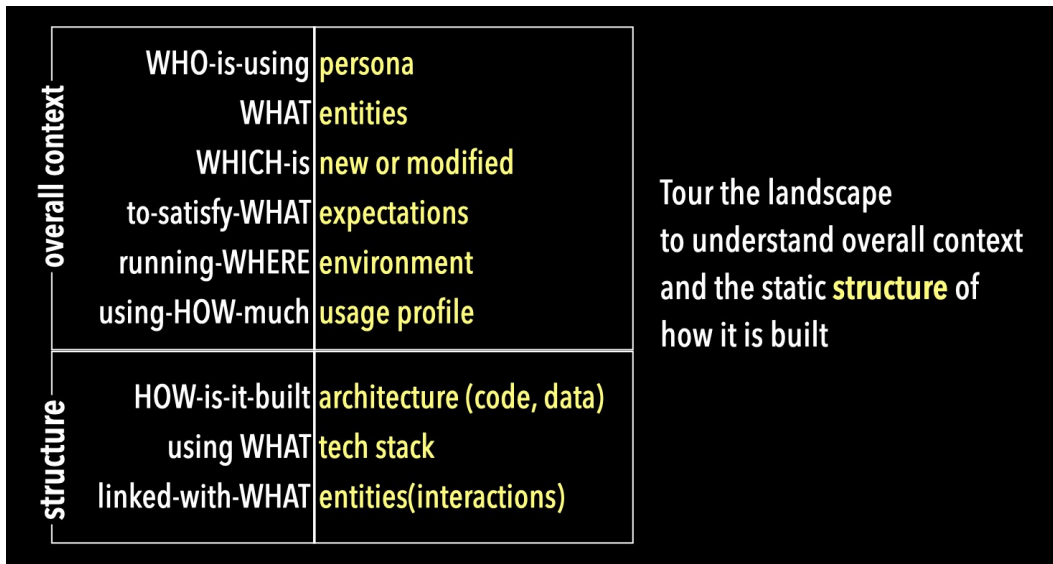
i.e. Different issues at each level by specific tests using appropriate techniques

satisfy QUALITY level	by TESTing for ISSUES
L9 End user value	User acceptance test
L8 Deployment correctness	Deployment test Data migration test Installation & CFG test
L7 Attribute correctness	Load test Performance test Security test Volume test
L6 Environment correctness	Configuration test- compatibility issues Attribute issues
L5 Flow correctness	Use case test- Higher order behaviour End-to-end test - Business flow issues
L4 Behaviour correctness	Functional test- Behaviour correctness Access control test- Roles & access issues
L3 Structural correctness	Structural test - Resources, Exceptions, Timeouts, Synchronisation, Side effects, Coverage
L2 Interface correctness	Interface correctness test (Data/UI) - UI interface, Data, Message, File format
L1 Input correctness	Input validation test - Limits, duplicates, data type, non-unique, data dependency

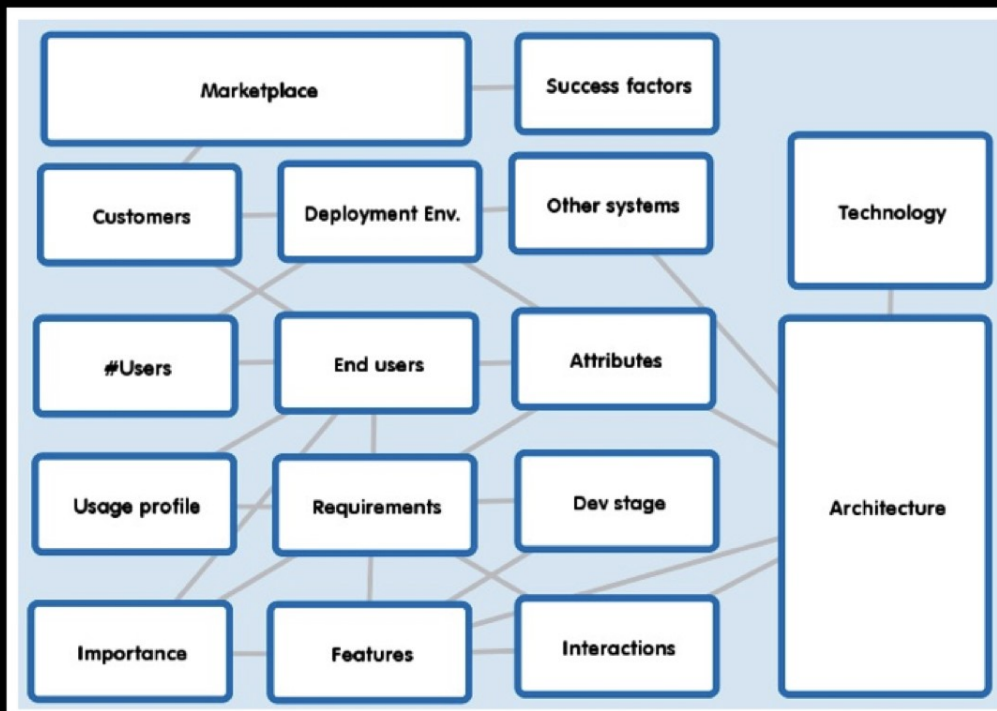
Landscape

See the BIG picture.

This helps in understanding the overall system context, usage profile, and the static structure of the system, to aid in asking pertinent questions and clarifying test objectives. This focuses on gaining a holistic and detailed overview of the system's context and structure.



LANDSCAPING is about connecting the different pieces

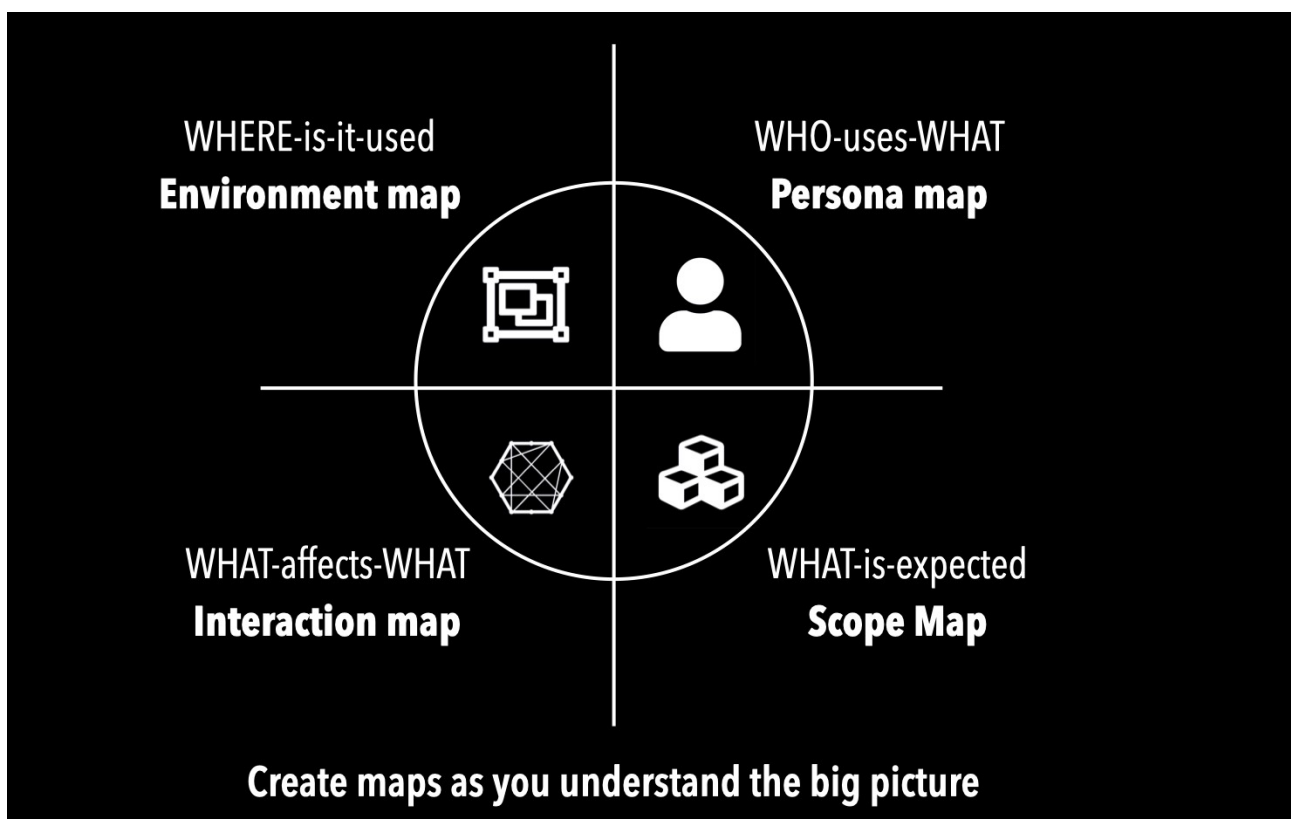


Product Map

Map what is used by who, where & expected to satisfy what.

The various maps help in understanding who uses what, what may be the expectations, how different system entities interact, and the various environments in which the system operates to help you understand well and test effectively & efficiently.

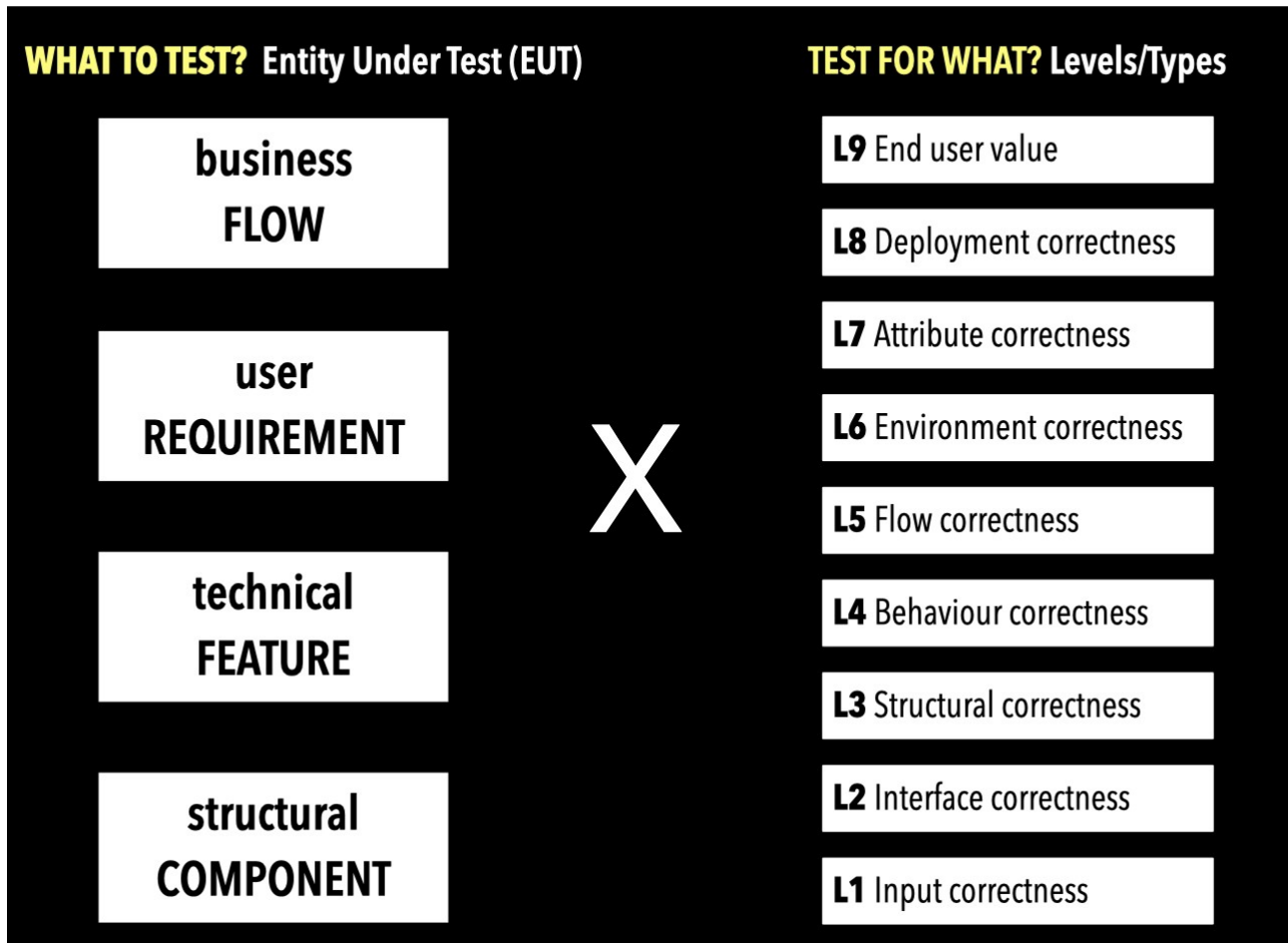
- Persona map to understand who and what they need & expect
- Scope map to understand attributes expected from various entities
- Interaction map to establish relationships between various entities to regress smartly
- Environment map to understand the various environments to validate on.



Test Scope

See scope of validation as a cartesian product.

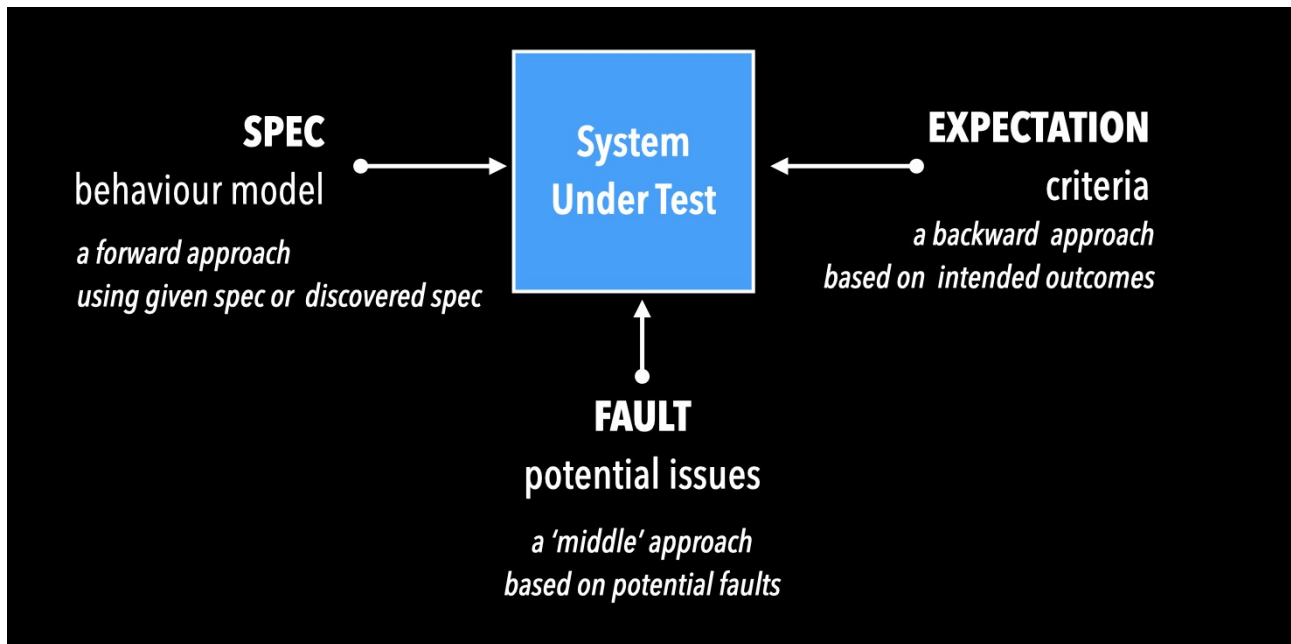
This guides in formulating a plan or strategy based on the validation scope, seen as a cartesian product what to test and test for what. The 'what' here signifies the various entities under test while 'test for what' are the various types of tests of interest. Aligning entities, levels, and types establishes the scope of validation, laying the foundation for a strategy that evolves into a plan.



Design Method

SmartDesign- coming up with different scenarios from different views.

This outlines the approach to designing probes to uncover potential issues in the system under test. This is a three-fold approach: using a spec-driven forward approach, an expectation-based backward approach, and fault-oriented middle approach to come top with probes(scenarios). This diversification in viewpoints aids in crafting smart probes from various perspectives.



SmartDesign : scenarios from different views

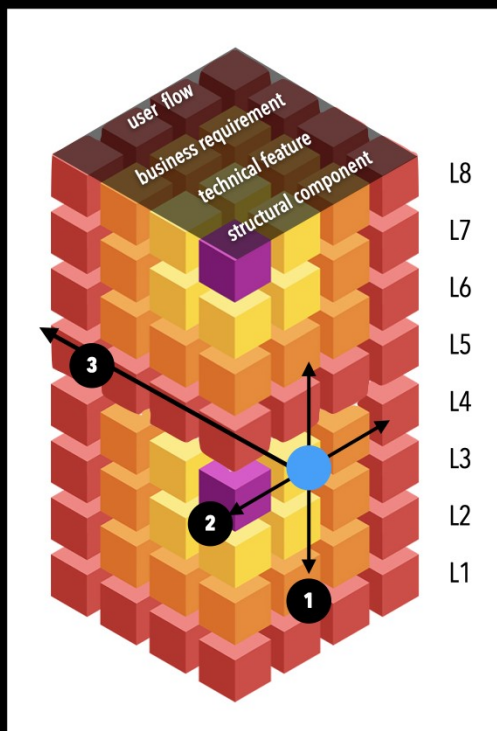
END USER view	user-spec based	user-observation based	user-experience based	I want expect would-like
ANALYTICAL view	techniques based	model based	standards based	behaviours to satisfy needs
CONSTRUCTION view	coverage based	technology based	architecture based	that are implemented well and comprehensively covered
TEST /QUALITY view	fault oriented	robustness oriented	failure mode based	exploration based questioning driven
EXPERIENCE view	support issue(s) based	fault patterns based		repeating no prior mistakes
OPERATIONAL view	usage profile based	deployment oriented		to help me do well on my environments
EVOLUTION view	code-change propagation	env-change impact		with no side effects

"I want | expect | would-like behaviours to satisfy needs that are implemented well and comprehensively covered to help me do well on my environments with no side effects"

Perturbation Analysis

Analyse impact of change via level wise criteria.

The concept of analysing the impact of change, known as perturbation analysis uses quality levels and associated criteria to understand potential impacts due to a change. It helps discern potential effects of modifications on entities by analysing impact across levels, similar entities, and different granularities, to gauge the extent of perturbation into the broader system.



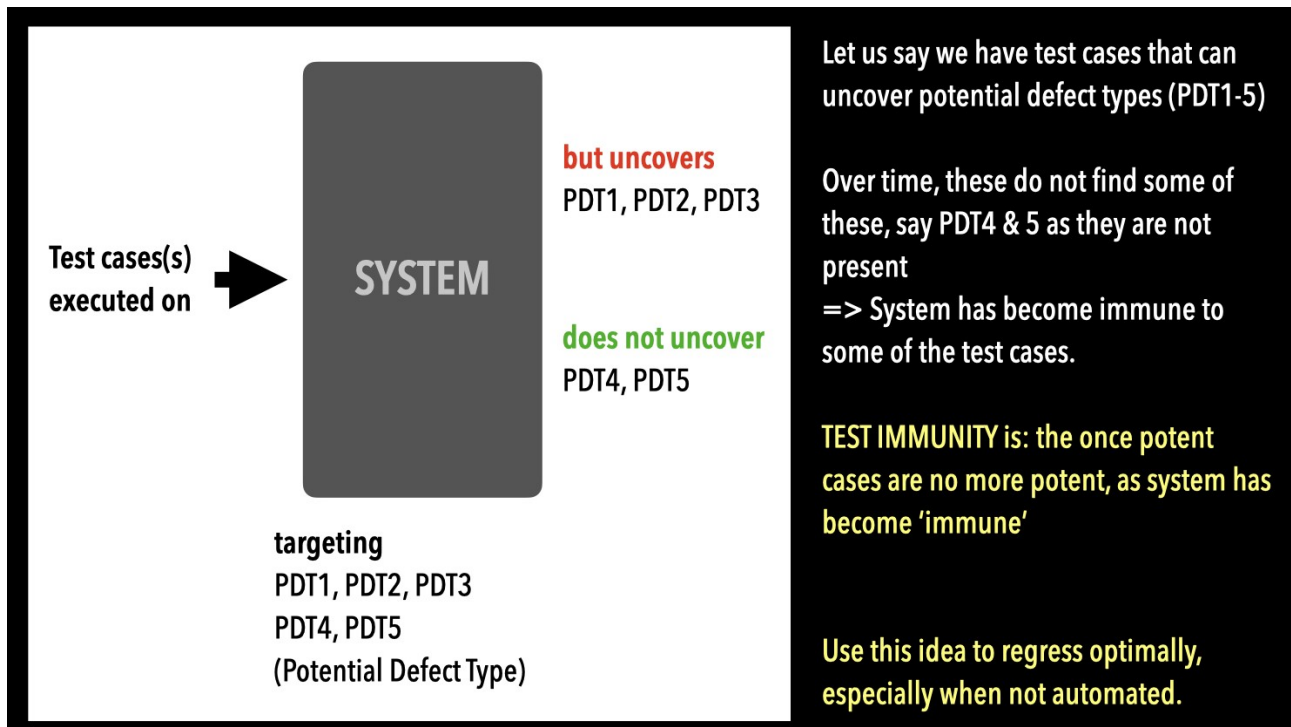
Given an entity been modified:

1. Analyze if this could affect any of its other criteria ? e.g. performance?
2. Next analyze if this could affect any other similar entity & what criteria of that entity
3. Finally analyze which **larger entity** that uses this entity could be affected and also the potential affected criteria

Potency & Immunity

Effective design & efficient regression (How-to).

Potency and immunity, as concepts, centre around assessing the effectiveness of test cases. Potency evaluates the ability of test cases to uncover specific defects within entities, aiming for minimal yet effective test coverage. Conversely, immunity recognises instances where the system becomes robust against specific test cases due to their continuous passing, prompting a reconsideration of their necessity.



Smart Checks

Assistance to probe well - focus, expand, leverage.

Transitioning to a smart check from a checklist aims to refine the process. Smart checks serve as guides rather than strict instructions, encouraging focus, expanding thinking, and leveraging experience. They suggest areas for consideration without imposing rigid mandates, enhancing the testing process.

**Smart Checks acts as
a guide/peer/mentor
catalysing thinking to do better,
leveraging wisdom/heuristics.**

Help you focus - Objective oriented

- what do we want to satisfy - criteria
- what issues do you want to uncover

Expand thinking - Idea oriented

- suggesting 'have you considered?'
- possibilities to examine
- areas to explore

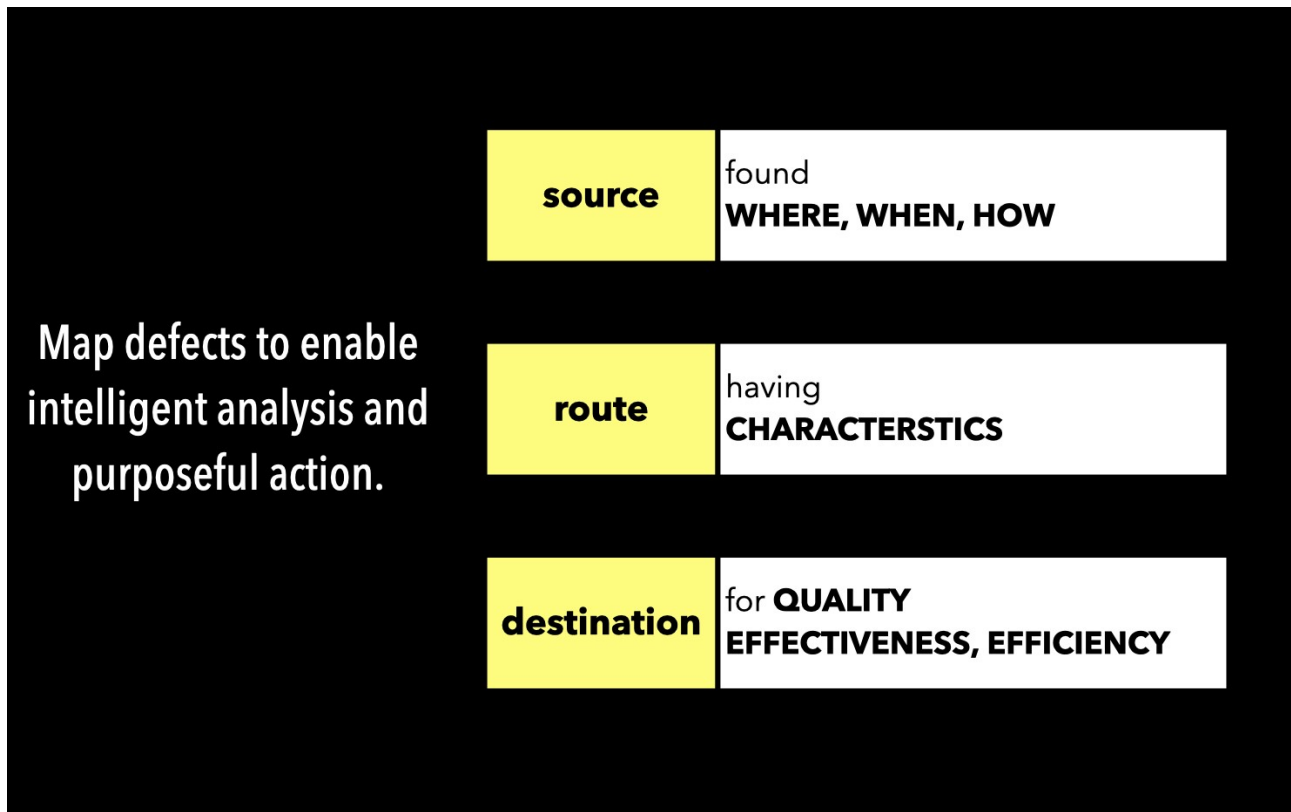
Leverage experience- Experience oriented

- interesting situations to consider
- sensitising to user's expectations
- implementation nuances/gotchas

Defect Map

Map defects well to assess & act.

The concept of a defect map aids in intelligent analysis and purposeful action. It maps defects based on their source, characteristics, and impact on quality, test effectiveness, and process efficiency, enabling informed decisions and actions.



CHAPTER 4 - KEY PRACTICES OF HYBIST

Note taking

Jot down interesting bits of information.

The first key practice involves note-taking, a critical aspect of observing and assimilating information when analysing systems, reading specifications, formulating ideas, or exploring the product under test. It aids in tracking interesting observations, issues, suggestions, and questions. Note-taking is fundamental to smart queuing strategies.

Mindful sessions

Be fully immersed, connected, in a state of flow.

This practice aims to maintain complete immersion, connection, and a state of flow during short, rapid testing sessions. This emphasises being fully engaged and connected during the testing process.

Language style

Communicate well.

This practice focuses on effective communication. It explores the styles and modes of communication—be it textual, pictorial, or otherwise—to facilitate thorough and quick communication during testing activities.

Survey, Swoop & Iterate

Understand well.

This is about 'how-to understand complex systems'. It encompasses various approaches: surveying from a top-down perspective, smoothing processes from a bottom-up angle, and iterating through multiple cycles to gain comprehensive understanding.

Focus & Meander

Focus on intent, but explore.

This practice advocates for a balance between staying focused on the task at hand and exploring beyond the typical routes. It emphasises the necessity to not only remain focused but also explore alternate paths for better contextual understanding.

CHAPTER 5 - HYBIST IN PRACTICE

Validate user story

This is about probing a requirement at early stage to preempt issues rapidly & effectively.

Here probing is about :

- clear about big picture? : who uses it, what stage of dev it is, is it new/enhanced, where may it be deployed and what are expectations of this
- what existing entities could this affect
- clear about intended criteria & behaviours?
- is enough info for first cut design available i.e. conditions, i/o clear & complete? conflicts, attributes ?
- is it testable?

Validate an entity

This is about probing a feature/requirement/flow (new or enhanced) to detect issues rapidly & effectively i.e. test a feature rapidly.

Here probing is about :

- understanding the big picture
- figuring out what all to retest
- figuring out criteria & behaviours
- designing first cut scenarios, develop Smart Checks and
- while evaluating, observe & improvise

Validate a sprint validate a user story

This is about probing a given build to detect issues rapidly & effectively. i.e. test entities delivered in a sprint.

Here probing is about :

- connecting to the big picture
- figuring out flows to retest
- scoping out what to test/retest
- figuring out criteria & behaviours
- designing first cut scenarios, develop Smart Checks and
- while evaluating, observe & improvise

CHAPTER 6 - HYBIST ASSISTANT - DO SMARTQA

What is doSmartQA?

doSmartQA is an assistant that aids testers in applying the HyBIST methodology, to test software & systems smartly. It assists one to interrogate, hypothesise issues, design and evaluate user story or a set of stories in a sprint deeply and rapidly.

It is based on HyBIST (Hypothesis Based Immersive Session Testing), a smart probing approach to validate deeply and rapidly in Agile environment. It is a SmartQA toolbox to design probes (analyse, design) & probe smartly (validate, assess) and available in two editions- as Personal (Free) as Chrome Extension & Enterprise (Premium) web application.

It acts as an aid to apply HyBIST, to help in the practice of probing. It helps you streamline your thinking, sharpens your practice of session so that you may do rapidly and be in a flow.

It is not a typical test automation, documentation or management tool; it is an assistant for smart session-based testing. It streamlines hypothesis formation, session management, and test execution, providing a structured framework for both manual and automated testing efforts.

Key functions of doSmartQA

Hypothesis management: Allows testers to document and track hypotheses about potential system failures.

Persona and Entity Mapping: Helps visualise how different user personas interact with system entities and identify areas that require focused testing.

Immersive Session Management: Provides the ability to organise immersive testing sessions, where testers can deep-dive into system behaviours and potential failure points.

Insights and Reporting: Gathers insights from testing sessions, producing actionable reports on system quality and test coverage.

The act of probing is performed in three phases - Reconnaissance, Exploration & Recoup, in short focussed sessions.

The key elements of doSmartQA to assist in this are:

Landscaper: Assist in understanding the big picture- persona, expectations, interactions, environment. Help you take notes- observations, questions, suggestions for multidimensional view.

Mind mapper: Help see the big picture of an entity(user story)

Mapper: Assist in charting, persona, scope & and interaction map to explore well.

Deep diver: Dig in and discover conditions, criteria, potential issues to design well.

HYBIST AT A GLANCE

Designer: Assist in development of smart checklist, test ideas, free-form & level based scenarios for superior coverage.

Executor: Assist you in track, improvise and enhance.

Session planner: Help you micro-plan, assist in setting up a 'probing session' and conducting it.

Intelligent Assistant: Based on Generative AI, it assists in discovering user expectations, potential issues, criteria, behaviour conditions to accelerate smart probing.

WHO benefits and HOW

It benefits QA Practitioners in tech companies with Agile development practices, by boosting their intellectual QA practice.

Note: The enterprise edition of doSmartQA facilitates team collaboration and richer insights.

CHAPTER 7: CONCLUSION - THE FUTURE OF SMART QA WITH HYBIST

The journey through "HyBIST at a glance" has offered a comprehensive exploration into the innovative Hypothesis-Based Immersive Session Testing (HyBIST) methodology. Throughout this book, we have examined how HyBIST redefines quality assurance by moving away from traditional, checklist-driven practices towards a more intellectual, hypothesis-driven approach.

A quick recap

SmartQA philosophy:

- Shifts from repetitive testing to intelligent probing.
- Encourages deeper coverage with minimal effort.
- Focuses on intellectual testing over mere automation.

HyBIST methodology:

- Built around the principles of forming smart hypotheses and performing immersive testing sessions.
- Involves three key phases: **Reconnaissance, Exploration, and Recoup** for efficient, rapid defect discovery.

Key concepts

- 360 POV, Quality Levels, Entity Granularity, Smart Probes, and more.
- These concepts guide systematic probing and ensure clarity in test design and execution.

Key practices

- Effective note-taking, language style, mindful sessions, and techniques like **Focus & Meander** for deep probing.

doSmartQA Assistant

- A powerful AI-powered tool supporting HyBIST principles.
- Enhances test strategy with hypothesis management, test mapping, and immersive session management.

The transformative impact of HyBIST

HyBIST represents a paradigm shift in quality assurance, emphasising:

- **Efficiency:** Reducing redundant testing while enhancing coverage.
- **Intellect-Driven Testing:** Leveraging human curiosity and logical thinking.

HYBIST AT A GLANCE

- **Adaptability:** Aligning perfectly with modern Agile and DevOps environments.
- **Early Defect Detection:** Promoting a proactive approach to defect discovery.

Why HyBIST matters for modern QA

- **For Testers:** HyBIST empowers testers with tools to think critically and test smarter, not harder.
- **For Managers:** It offers data-driven insights for faster, more effective decision-making.
- **For Organizations:** HyBIST ensures faster releases with improved software quality, directly aligning with business goals.

The path forward

HyBIST is not just a methodology but a mindset shift towards **smarter, faster, and more effective quality assurance**. By combining intellectual testing with minimal yet impactful test execution, it promises to reshape the future of software quality.

Embrace HyBIST and step into the future of Smart Quality Assurance. Test smarter, probe deeper, and deliver exceptional software with confidence.

This concludes your journey through "HyBIST at a Glance." Implement these principles and tools to transform your QA practice today.



"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

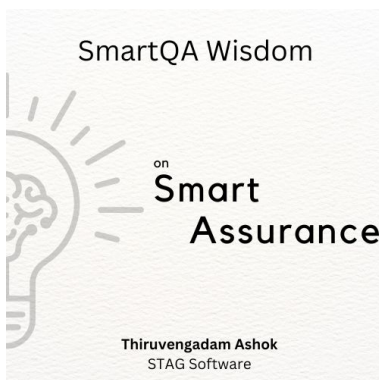
SmartQA eBooks - for Sr Engg/QA managers, Sr Test Practitioner & Young Test Practitioners too.

Communicate Clearly
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