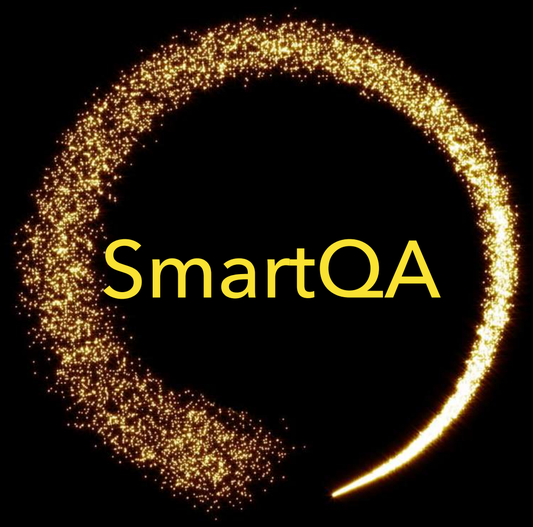


IST Masterclass
Session #5

Exploration phase in detail #2

www.stagsoftware.com

TOPICS

Sprint/Session Plan

Test granularity

Smart checklist

Smart regression

Sprint/Session Plan

Before starting an exploration session, formulate a plan (& revise during session)

Note from a RECON session we would have identified the EUTs.

In case of sprint, we know what user stories are in focus.

Before starting an exploration session, formulate a plan (& revise during session)

Note from a RECON session we would have identified the EUTs.
In case of sprint, we know what user stories are in focus.

- what EUT to focus on
- test or retest(check)
- for what attribute(s)
- where to perform (env)

from EUT perspective
(what-to-test)

Before starting an exploration session, formulate a plan (& revise during session)

Note from a RECON session we would have identified the EUTs.
In case of sprint, we know what user stories are in focus.

- what EUT to focus on
- test or retest(check)
- for what attribute(s)
- where to perform (env)

from EUT perspective
(what-to-test)

EUT- New/Mod/Fix
Interacting EUT
(i.e. impacted)

Before starting an exploration session, formulate a plan (& revise during session)

Note from a RECON session we would have identified the EUTs.
In case of sprint, we know what user stories are in focus.

- what EUT to focus on
- test or retest(check)
- for what attribute(s)
- where to perform (env)

(or)

- what test to conduct
- where to perform (env)

from EUT perspective
(what-to-test)

from TEST TYPE perspective
(test-for-what)

EUT- New/Mod/Fix
Interacting EUT
(i.e. impacted)

Let’s say TF1 is focus of a session, then:

Test TF1 (in case New)

Retest TF1 (in case Mod/Fix)

Let's say TF1 is focus of a session, then:

Test TF1 (in case New)

Retest TF1 (in case Mod/Fix)

technical FEATURE

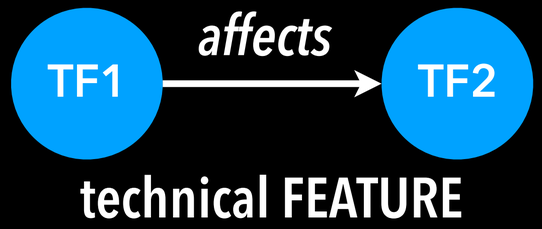
Since TF1 affects TF2

we may need to Retest TF2

Let's say TF1 is focus of a session, then:

Test TF1 (in case New)

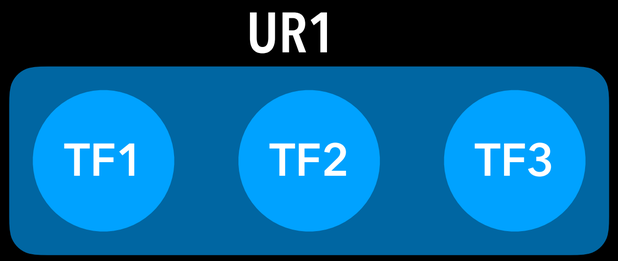
Retest TF1 (in case Mod/Fix)



technical FEATURE

Since TF1 affects TF2

we may need to Retest TF2



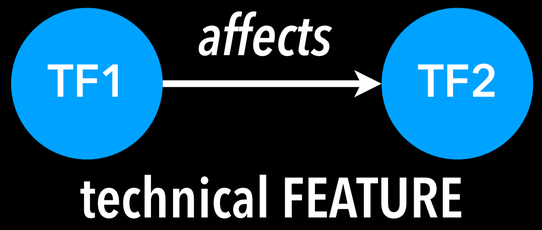
Since TF1 is part of UR1

we may need to Retest UR1

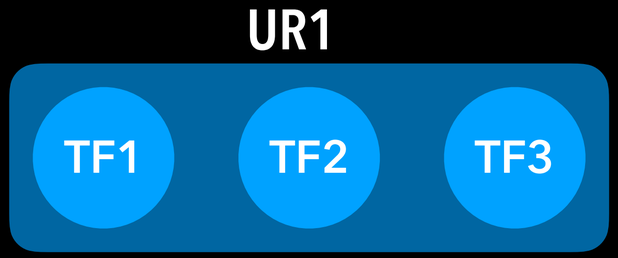
Let's say TF1 is focus of a session, then:

Test TF1 (in case New)

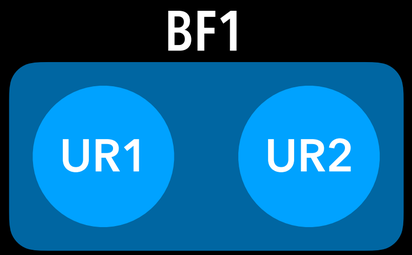
Retest TF1 (in case Mod/Fix)



Since TF1 affects TF2
we may need to Retest TF2

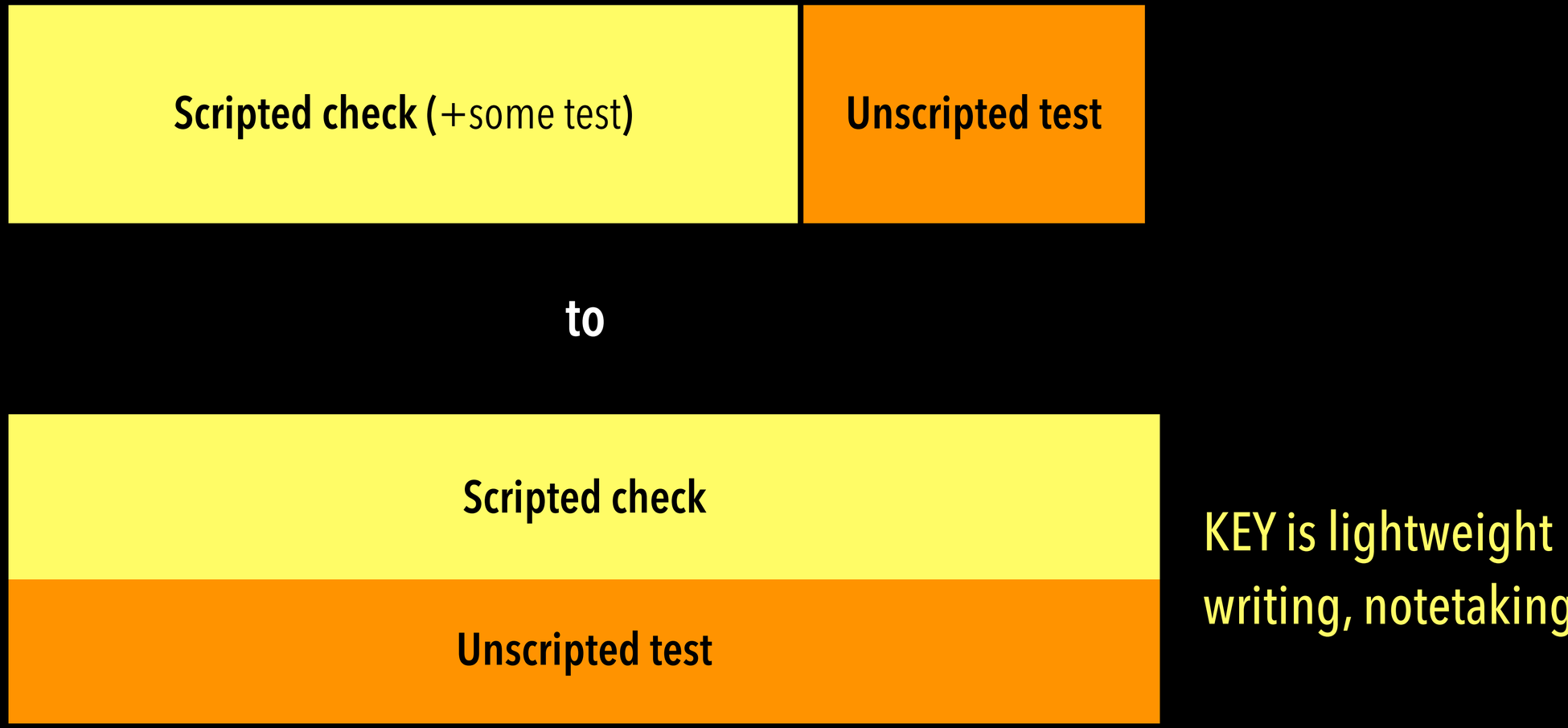


Since TF1 is part of UR1
we may need to Retest UR1



Since TF1 is part of BF1(UR1)
we may need to Retest BF1

REMEMBER - Design & evaluation in rapid tandem



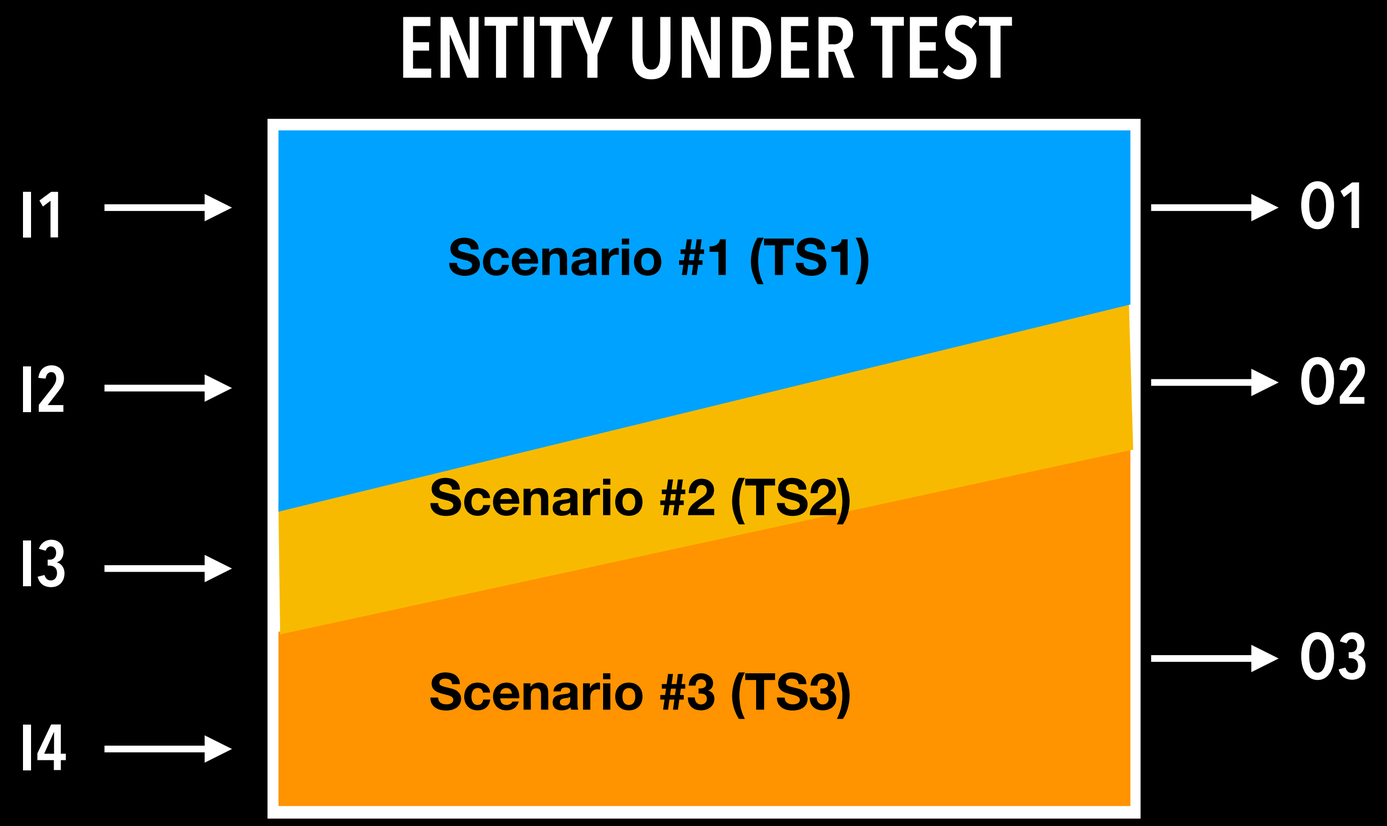
KEY is lightweight writing, notetaking

and then do

RECONNAISANCE | EXPLORATION | RECOUP

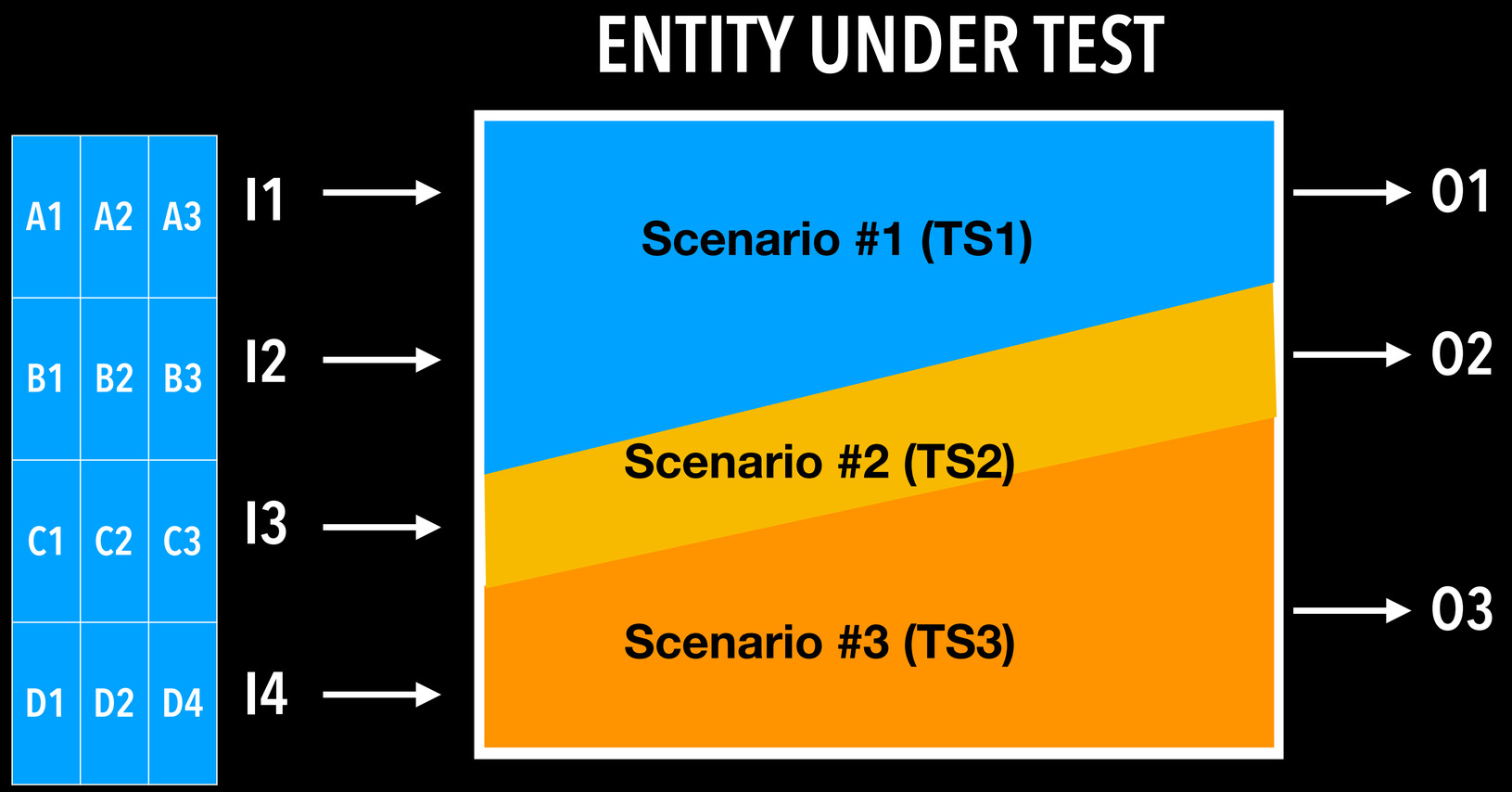
Test granularity

Scenarios & cases



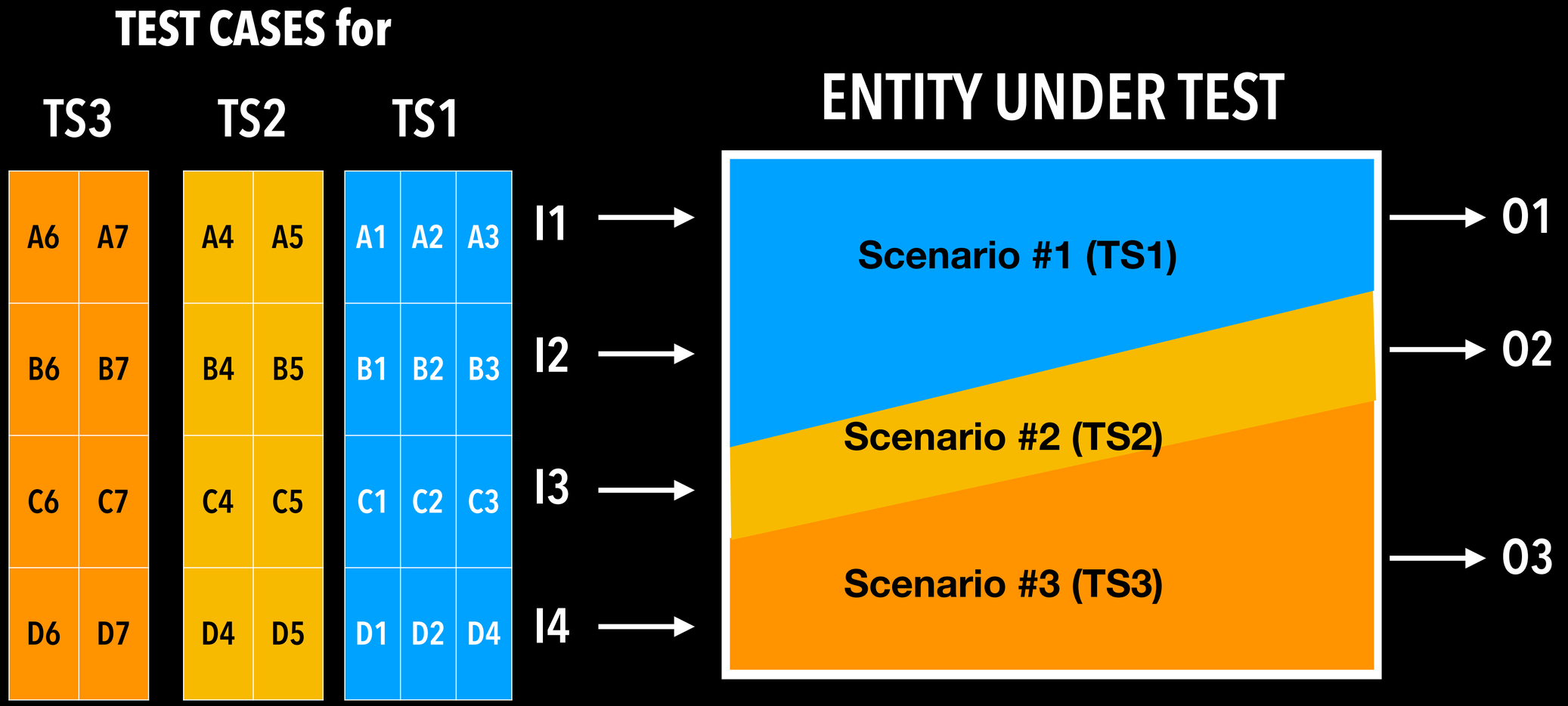
On analysis we see THREE distinct **behaviours**
i.e. **test SCENARIOS**

Scenarios & cases

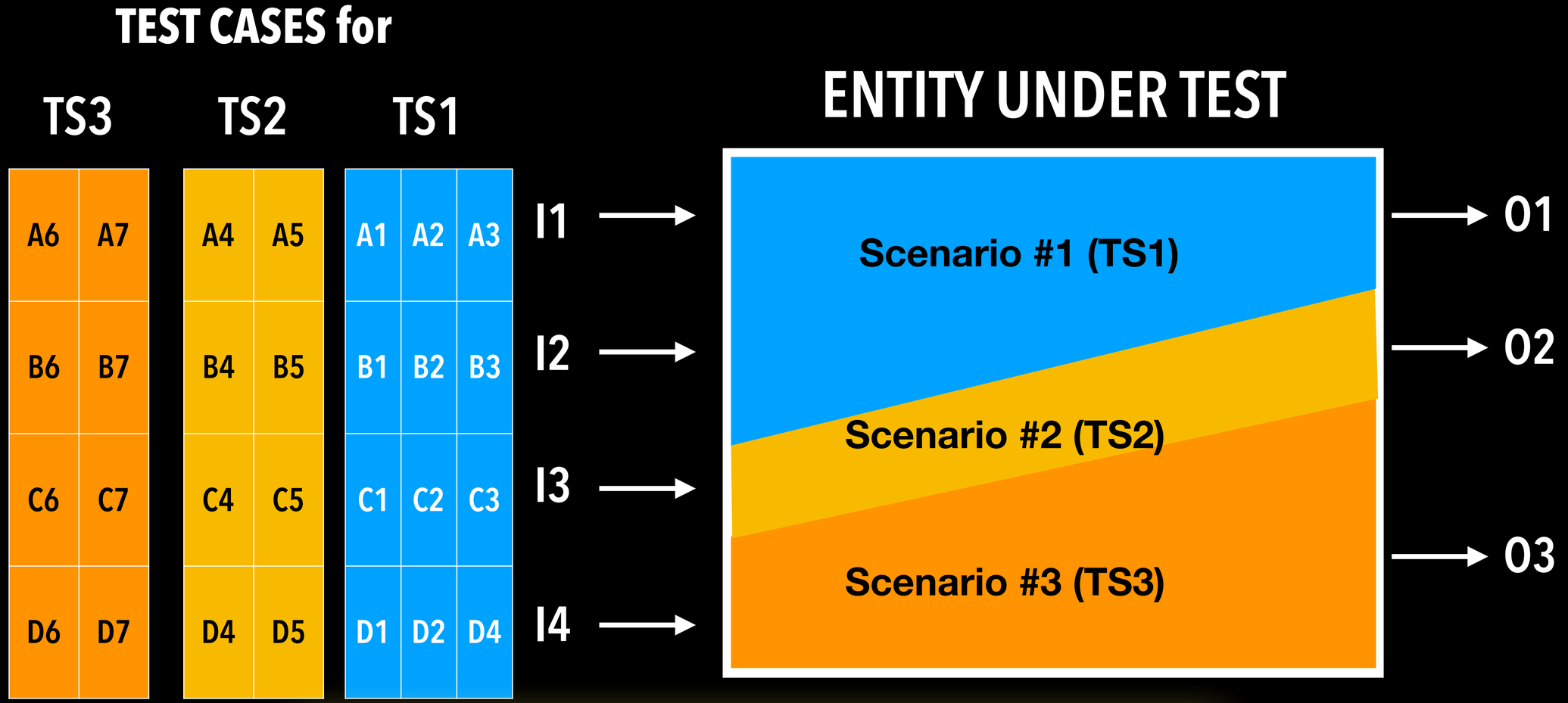


To **stimulate** Scenario #1, it takes THREE sets of distinct combination of inputs i.e. **test CASES**

Scenarios & cases



Scenarios & cases



Test SCENARIOS represent behaviours

Test CASES are stimuli

Scenario can be:

SPECIFIC - Sharp & focussed
precise outcomes for a specific behaviour

focus on DO, to check

Scenario can be:

SPECIFIC - Sharp & focussed
precise outcomes for a specific behaviour

focus on DO, to check

GENERIC - Broad & directional
suggest ideas, possibilities, broad actions on such behaviours/context

suggest, make you think contextually

Scenario can be:

SPECIFIC - Sharp & focussed
precise outcomes for a specific behaviour

focus on DO, to check

GENERIC - Broad & directional
suggest ideas, possibilities, broad actions
on such behaviours/context

SMART Checklist

suggest, make you think
contextually

Smart Checklist

Checklist is seen as a tool to enforce compliance.
Certainly it adds value by ensuring
we do not miss out on important aspects/activities,
but dulls us and tends to make it boring.

But, is that how it supposed to be ?

As smart individuals,
we don’t like checklists.

It somehow feels beneath us to use a checklist,
an embarrassment.

The fear is that checklists enforces
a mindless adherence to protocol.

Well a **Smart Checklist** goes beyond.

It goes beyond checking, to

“Have you thought about these?

Could these be applicable?”

making you see possibilities and think.

It is not just ticking off boxes.
It is helping you think about ideas/possibilities.

It is not just ticking off boxes.
It is helping you think about ideas/possibilities.

It acts as a guide/peer/mentor
that catalyses your thinking to do better,
by leveraging prior wisdom/heuristics.

It is not just ticking off boxes.
It is helping you think about ideas/possibilities.

It acts as a guide/peer/mentor
that catalyses your thinking to do better,
by leveraging prior wisdom/heuristics.

Smart checklist goes beyond mere compliance checking to smart testing.

Checklists are of THREE flavours:

1

Simple activity task checklist

containing simple steps not to be missed/skipped

Checklists are of THREE flavours:

1. **Simple activity task checklist**
   containing simple steps not to be missed/skipped

2. **Coordination activity task checklist**
   activities done by different roles do not cause issues

Checklists are of THREE flavours:

1. **Simple activity task checklist**
   containing simple steps not to be missed/skipped

2. **Coordination activity task checklist**
   activities done by different roles do not cause issues

3. **Complex problems checklist**
   enable ideation, assist in making choices and help you do

On Oct 30, 1985, a massive plane that could carry 5x more bombs roared and lifted off from an airport in Dayton Ohio, and then crashed.

Reason cited was “Pilot error”. A newspaper reported **“this was too much airplane for one man to fly”.**

Boeing the maker of this plane nearly went bankrupt.

On Oct 30, 1985, a massive plane that could carry 5x more bombs roared and lifted off from an airport in Dayton Ohio, and then crashed.

Reason cited was “Pilot error”. A newspaper reported **“this was too much airplane for one man to fly”.**

Boeing the maker of this plane nearly went bankrupt.

So, how did they fix this issue?

By creating a pilot’s checklist, as flying a new plane was too complicated to be left to the memory of any one person, however expert.

On Oct 30, 1985, a massive plane that could carry 5x more bombs roared and lifted off from an airport in Dayton Ohio, and then crashed.

Reason cited was "Pilot error". A newspaper reported **"this was too much airplane for one man to fly".**

Boeing the maker of this plane nearly went bankrupt.

So, how did they fix this issue?

By creating a pilot's checklist, as flying a new plane was too complicated to be left to the memory of any one person, however expert.

Before Takeoff - Run-Up

1. Cabin Doors and Windows - **Closed & Locked**
2. Parking Brake - **Set**
3. Flight Controls - **Free & Correct**
4. Flight Instruments - **Set**
5. Fuel Selector Valve - **ON - Fullest Tank**
6. Elevator Trim - **Takeoff**
7. Mixture - **Rich or As Required**
8. Throttle - **2000 RPM**
   (a) Magnetos - **Check**
   (b) Carburetor Heat - **Check**
   (c) Engine Instruments & Ammeter - **Check**
   (d) Suction Gauge - **Check 5"±0.1**
9. Throttle - **Idle, then 1000 RPM**
10. Radios - **Set**
11. Transponder - **Set, then Altitude**
12. Throttle Friction Lock - **Adjust**
13. Fuel Pump - **ON**
14. Lights - **As Required**
15. Parking Brake - **Release**

On Oct 30, 1985, a massive plane that could carry 5x more bombs roared and lifted off from an airport in Dayton Ohio, and then crashed.

Reason cited was "Pilot error". A newspaper reported **"this was too much airplane for one man to fly".**

Boeing the maker of this plane nearly went bankrupt.

So, how did they fix this issue?

By creating a pilot's checklist, as flying a new plane was too complicated to be left to the memory of any one person, however expert.

RESULT : 1.8 million miles without one accident!

Before Takeoff - Run-Up

1. Cabin Doors and Windows - **Closed & Locked**
2. Parking Brake - **Set**
3. Flight Controls - **Free & Correct**
4. Flight Instruments - **Set**
5. Fuel Selector Valve - **ON - Fullest Tank**
6. Elevator Trim - **Takeoff**
7. Mixture - **Rich or As Required**
8. Throttle - **2000 RPM**
   (a) Magnetos - **Check**
   (b) Carburetor Heat - **Check**
   (c) Engine Instruments & Ammeter - **Check**
   (d) Suction Gauge - **Check 5"±0.1**
9. Throttle - **Idle, then 1000 RPM**
10. Radios - **Set**
11. Transponder - **Set, then Altitude**
12. Throttle Friction Lock - **Adjust**
13. Fuel Pump - **ON**
14. Lights - **As Required**
15. Parking Brake - **Release**

Problem of extreme complexity

The field of medicine

has 13000+ diseases, syndromes, injury types.

(i.e. 13000 ways a body can fail)

and 6000 drugs, 4000 medicines & surgical procedures

each with different requirements, risks & considerations.

Problem of extreme complexity

The field of medicine
has 13000+ diseases, syndromes, injury types.
(i.e. 13000 ways a body can fail)
and 6000 drugs, 4000 medicines & surgical procedures
each with different requirements, risks & considerations.

In an ICU, an average patient requires
178 individual interactions per day!
To save a desperately sick patient it is necessary to:
get the knowledge right & do 178 daily tasks right.

Checklists seem to provide against such failures and instil a kind of discipline of higher performance

Checklists seem to provide against such failures and instil a kind of discipline of higher performance

case study #1

Tackling central line infections in ICU using checklist

prevented 43 infections & 8 deaths and saved USD 2M

(Peter Provonost in 2001)

Checklists seem to provide against such failures and instil a kind of discipline of higher performance

case study #1

Tackling central line infections in ICU using checklist
prevented 43 infections & 8 deaths and saved USD 2M

(Peter Provonost in 2001)

case study #2

In a bigger implementation "Keystone Initiative" (2006)
involving more hospitals of 18 month duration,
USD 17M saved, 1500+ lives saved

Checklists seem to provide against such failures and instil a kind of discipline of higher performance

case study #1

Tackling central line infections in ICU using checklist
prevented 43 infections & 8 deaths and saved USD 2M
(Peter Provonost in 2001)

case study #2

In a bigger implementation "Keystone Initiative" (2006)
involving more hospitals of 18 month duration,
USD 17M saved, 1500+ lives saved

Higher baseline performance is what a smart checklist can do.

Higher baseline performance is what a smart checklist can do.

Surgical Safety Checklist

World Health Organization | Patient Safety
A World Alliance for Safer Health Care

Before induction of anaesthesia
(with at least nurse and anaesthetist)

Has the patient confirmed his/her identity, site, procedure, and consent?
- ☐ Yes

Is the site marked?
- ☐ Yes
- ☐ Not applicable

Is the anaesthesia machine and medication check complete?
- ☐ Yes

Is the pulse oximeter on the patient and functioning?
- ☐ Yes

Does the patient have a:

Known allergy?
- ☐ No
- ☐ Yes

Difficult airway or aspiration risk?
- ☐ No
- ☐ Yes, and equipment/assistance available

Risk of >500ml blood loss (7ml/kg in children)?
- ☐ No
- ☐ Yes, and two IVs/central access and fluids planned

Before skin incision
(with nurse, anaesthetist and surgeon)

- ☐ **Confirm all team members have introduced themselves by name and role.**
- ☐ **Confirm the patient's name, procedure, and where the incision will be made.**

Has antibiotic prophylaxis been given within the last 60 minutes?
- ☐ Yes
- ☐ Not applicable

Anticipated Critical Events

To Surgeon:
- ☐ What are the critical or non-routine steps?
- ☐ How long will the case take?
- ☐ What is the anticipated blood loss?

To Anaesthetist:
- ☐ Are there any patient-specific concerns?

To Nursing Team:
- ☐ Has sterility (including indicator results) been confirmed?
- ☐ Are there equipment issues or any concerns?

Is essential imaging displayed?
- ☐ Yes
- ☐ Not applicable

Before patient leaves operating room
(with nurse, anaesthetist and surgeon)

Nurse Verbally Confirms:
- ☐ The name of the procedure
- ☐ Completion of instrument, sponge and needle counts
- ☐ Specimen labelling (read specimen labels aloud, including patient name)
- ☐ Whether there are any equipment problems to be addressed

To Surgeon, Anaesthetist and Nurse:
- ☐ What are the key concerns for recovery and management of this patient?

This checklist is not intended to be comprehensive. Additions and modifications to fit local practice are encouraged.

Revised 1 / 2009

SMART Checklist

is one

that respects you as smart person

SMART Checklist

is one

that respects you as smart person

gives you hints, not bore you

SMART Checklist

is one

that respects you as smart person

gives you hints, not bore you

is crisp and clear in what to do

SMART Checklist

is one
that respects you as smart person
gives you hints, not bore you
is crisp and clear in what to do
is quick and painless to use

SMART Checklist

is one
that respects you as smart person
gives you hints, not bore you
is crisp and clear in what to do
is quick and painless to use
is not a form to fill & file

SMART Checklist

is one
that respects you as smart person
gives you hints, not bore you
is crisp and clear in what to do
is quick and painless to use
is not a form to fill & file

...helps you build a better habit!

Smart Checklist Intents

Objective oriented - Help you focus

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

Smart Checklist Intents

Objective oriented - Help you focus

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

Idea oriented - Expand thinking

- suggesting ‘have you considered?’
- possibilities to examine
- areas to explore

Smart Checklist Intents

Objective oriented - Help you focus

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

Idea oriented - Expand thinking

- suggesting ‘have you considered?’
- possibilities to examine
- areas to explore

Experience oriented - Leverage experience

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

Smart Checklist design tips

1

SET OBJECTIVE

State objective of smart checklist.

See it as a set of intents to accomplish.

Smart Checklist design tips

1

SET OBJECTIVE

State objective of smart checklist.
See it as a set of intents to accomplish.

2

STATE INTENT

CHECK known for CONFORMANCE (focus)
enable PROBING into KNOWN (question)
enable DISCOVERY of UNKNOWN (ideate)

Smart Checklist design tips

1

SET OBJECTIVE

State objective of smart checklist.
See it as a set of intents to accomplish.

2

STATE INTENT

CHECK known for CONFORMANCE (focus)
enable PROBING into KNOWN (question)
enable DISCOVERY of UNKNOWN (ideate)

3

IDENTIFY ACTION FOR INTENT

as STIMULI to inject
as CRITERIA to check for
as ISSUE to look for

Smart Checklist design tips

1

SET OBJECTIVE

State objective of smart checklist.
See it as a set of intents to accomplish.

2

STATE INTENT

CHECK known for CONFORMANCE (focus)
enable PROBING into KNOWN (question)
enable DISCOVERY of UNKNOWN (ideate)

3

IDENTIFY ACTION FOR INTENT

as STIMULI to inject
as CRITERIA to check for
as ISSUE to look for

4

EXPRESS ACTION

as a TO-DO
as a QUESTION
as a HEURISTIC
as an IDEA/SUGGESTION

Smart Checklist design tips

1

SET OBJECTIVE

State objective of smart checklist.
See it as a set of intents to accomplish.

2

STATE INTENT

CHECK known for CONFORMANCE (focus)
enable PROBING into KNOWN (question)
enable DISCOVERY of UNKNOWN (ideate)

3

IDENTIFY ACTION FOR INTENT

as STIMULI to inject
as CRITERIA to check for
as ISSUE to look for

4

EXPRESS ACTION

as a TO-DO
as a QUESTION
as a HEURISTIC
as an IDEA/SUGGESTION

5

WRITE ACTION

using IMPERATIVE style
using INTERROGATIVE style
using DECLARATIVE style

SMARTDEVTEST CHECKLIST

Am I OK ? — Bad INPUTS rejected? SCREEN/UI well done? — DISPLAYS well? All TEXT fine? — All OUTPUTS checked? Are ACTIONS to do fine? — "I am fine" — 1

Inputs ok?
- ☐ Limits
- ☐ Type
- ☐ Defaults

Screen ok?
- ☐ Alignment
- ☐ Consistency
- ☐ Dependencies
- ☐ Colours fine

Display ok?
- ☐ Responsive
- ☐ Orientation
- ☐ Resolution

Text ok?
- ☐ Spelling
- ☐ Grammar
- ☐ Meaningful
- ☐ Actionable

Outputs ok?
- ☐ Stored fine
- ☐ No duplicates
- ☐ Appropriate msg

Actions ok?
- ☐ Navigation
- ☐ Default
- ☐ Confirmation

Am I ROBUST? — Handled errors/EXCEPTIONS well? Work on different ENVIRONMENTS? — Not affected by DEPENDENCIES? Will system attributes be met? — "I am resilient" — 2

Errors handled?
- ☐ Connection loss
- ☐ Low resources
- ☐ Services down

Env friendly?
- ☐ Diff browsers
- ☐ Diff resolutions
- ☐ Diff devices
- ☐ Diff version OS/SW..

Dependencies ok?
- ☐ In common lib
- ☐ In shared data
  - ☐ memory
  - ☐ files/DB
- ☐ Ext settings/config

Attributes ok?
- ☐ Security
- ☐ Large volume
- ☐ Basic performance

Are you OK? — Used/Released RESOURCES? SETTINGS/CONFIG change side effects? — DATA change side effects? INTERFACE changes side effects? — "I am a good citizen" — 3

I am not messing up the environment

I am not messing up other's code via side effects due to changes that I may have made in my SETTINGS/ CONFIG/DATA/ INTERFACE

Resource usage ok?
- ☐ No leaks
  - ☐ memory
  - ☐ handles
  - ☐ OS resources
- ☐ No tmp files

SETT./CFG side effects?
- ☐ App settings
- ☐ Env. settings
- ☐ Permissions

DATA side effects?
- ☐ Formats
- ☐ Types
- ☐ Defaults
- ☐ Width

I/F side effects?
- ☐ API parameters
- ☐ Msg parameters
- ☐ DB tables
- ☐ File contents

Am I OK ?	Bad INPUTS rejected? SCREEN/UI well done?	DISPLAYS well? All TEXT fine?	All OUTPUTS checked? Are ACTIONS to do fine?	"I am fine"

1

Inputs ok?
- ☐ Limits
- ☐ Type
- ☐ Defaults

Screen ok?
- ☐ Alignment
- ☐ Consistency
- ☐ Dependencies
- ☐ Colours fine

Display ok?
- ☐ Responsive
- ☐ Orientation
- ☐ Resolution

Text ok?
- ☐ Spelling
- ☐ Grammar
- ☐ Meaningful
- ☐ Actionable

Outputs ok?
- ☐ Stored fine
- ☐ No duplicates
- ☐ Appropriate msg

Actions ok?
- ☐ Navigation
- ☐ Default
- ☐ Confirmation

Am I ROBUST? Handled errors/EXCEPTIONS well? Work on different ENVIRONMENTS? Not affected by DEPENDENCIES? Will system attributes be met? "I am resilient"

2

Errors handled?
- ☐ Connection loss
- ☐ Low resources
- ☐ Services down

Env friendly?
- ☐ Diff browsers
- ☐ Diff resolutions
- ☐ Diff devices
- ☐ Diff version OS/SW..

Dependencies ok?
- ☐ In common lib
- ☐ In shared data
  - ☐ memory
  - ☐ files/DB
- ☐ Ext settings/config

Attributes ok?
- ☐ Security
- ☐ Large volume
- ☐ Basic performance

Are you OK? | **Used/Released RESOURCES? SETTINGS/CONFIG change side effects?** | **DATA change side effects? INTERFACE changes side effects?** | **"I am a good citizen"**

3

I am not messing up the environment

I am not messing up other's code via side effects due to changes that I may have made in my SETTINGS/ CONFIG/DATA/ INTERFACE

Resource usage ok?
- ☐ No leaks
  - ☐ memory
  - ☐ handles
  - ☐ OS resources
- ☐ No tmp files

SETT./CFG side effects?
- ☐ App settings
- ☐ Env. settings
- ☐ Permissions

DATA side effects?
- ☐ Formats
- ☐ Types
- ☐ Defaults
- ☐ Width

I/F side effects?
- ☐ API parameters
- ☐ Msg parameters
- ☐ DB tables
- ☐ File contents

SMARTDEVTEST CHECKLIST

Am I OK ? — Bad INPUTS rejected? SCREEN/UI well done? — DISPLAYS well? All TEXT fine? — All OUTPUTS checked? Are ACTIONS to do fine? — "I am fine" — 1

Inputs ok?
- ☐ Limits
- ☐ Type
- ☐ Defaults

Screen ok?
- ☐ Alignment
- ☐ Consistency
- ☐ Dependencies
- ☐ Colours fine

Display ok?
- ☐ Responsive
- ☐ Orientation
- ☐ Resolution

Text ok?
- ☐ Spelling
- ☐ Grammar
- ☐ Meaningful
- ☐ Actionable

Outputs ok?
- ☐ Stored fine
- ☐ No duplicates
- ☐ Appropriate msg

Actions ok?
- ☐ Navigation
- ☐ Default
- ☐ Confirmation

Am I ROBUST? — Handled errors/EXCEPTIONS well? Work on different ENVIRONMENTS? — Not affected by DEPENDENCIES? Will system attributes be met? — "I am resilient" — 2

Errors handled?
- ☐ Connection loss
- ☐ Low resources
- ☐ Services down

Env friendly?
- ☐ Diff browsers
- ☐ Diff resolutions
- ☐ Diff devices
- ☐ Diff version OS/SW..

Dependencies ok?
- ☐ In common lib
- ☐ In shared data
  - ☐ memory
  - ☐ files/DB
- ☐ Ext settings/config

Attributes ok?
- ☐ Security
- ☐ Large volume
- ☐ Basic performance

Are you OK? — Used/Released RESOURCES? SETTINGS/CONFIG change side effects? — DATA change side effects? INTERFACE changes side effects? — "I am a good citizen" — 3

I am not messing up the environment

I am not messing up other's code via side effects due to changes that I may have made in my SETTINGS/ CONFIG/DATA/ INTERFACE

Resource usage ok?
- ☐ No leaks
  - ☐ memory
  - ☐ handles
  - ☐ OS resources
- ☐ No tmp files

SETT./CFG side effects?
- ☐ App settings
- ☐ Env. settings
- ☐ Permissions

DATA side effects?
- ☐ Formats
- ☐ Types
- ☐ Defaults
- ☐ Width

I/F side effects?
- ☐ API parameters
- ☐ Msg parameters
- ☐ DB tables
- ☐ File contents

Design Assessment Checklist

(DRAFT)

DOC LINKS: are all info related to the int/ext links in the document?

- ☐ Internal references ok?
- ☐ External references available?

COVERAGE: quick assessment of requirements/attributes covered

- ☐ All functional requirements covered?
- ☐ Expected attributes designed for?

DESIGN ELEMENTS: are functions of design elements clear?

- ☐ objective of element clear?
- ☐ logic/algorithm clear?
- ☐ logic will meet functionality?
- ☐ input-output data clear
  - ☐ format, values, types

INTERFACE: are the interfaces between elements clear?

- ☐ Interfaces appropriate?
- ☐ Interface spec (req/resp) well defined?
- ☐ Interface data spec non-ambiguous?
- ☐ Interface visibility appropriate?
- ☐ Interfaces secure?

INTERACTIONS: assess interactions between subsystems/components

- ☐ Order of invocation
- ☐ Interface data
  - ☐ data types,
  - ☐ formats
- ☐ Responses clear in send-receive?
- ☐ Blocking/Timeout?
- ☐ Non-blocking/Callback
- ☐ Side effects?

ERROR HANDLING: how robust is the design in handling exigencies

- ☐ What exigencies to handle?
- ☐ Defensive? i.e. bad data rejected
- ☐ Precondition checks?
- ☐ Faults anticipated?
- ☐ Support for
  - ☐ supportability
  - ☐ testability

Smart Regression

the challenge of CHANGE

1. what to retest?

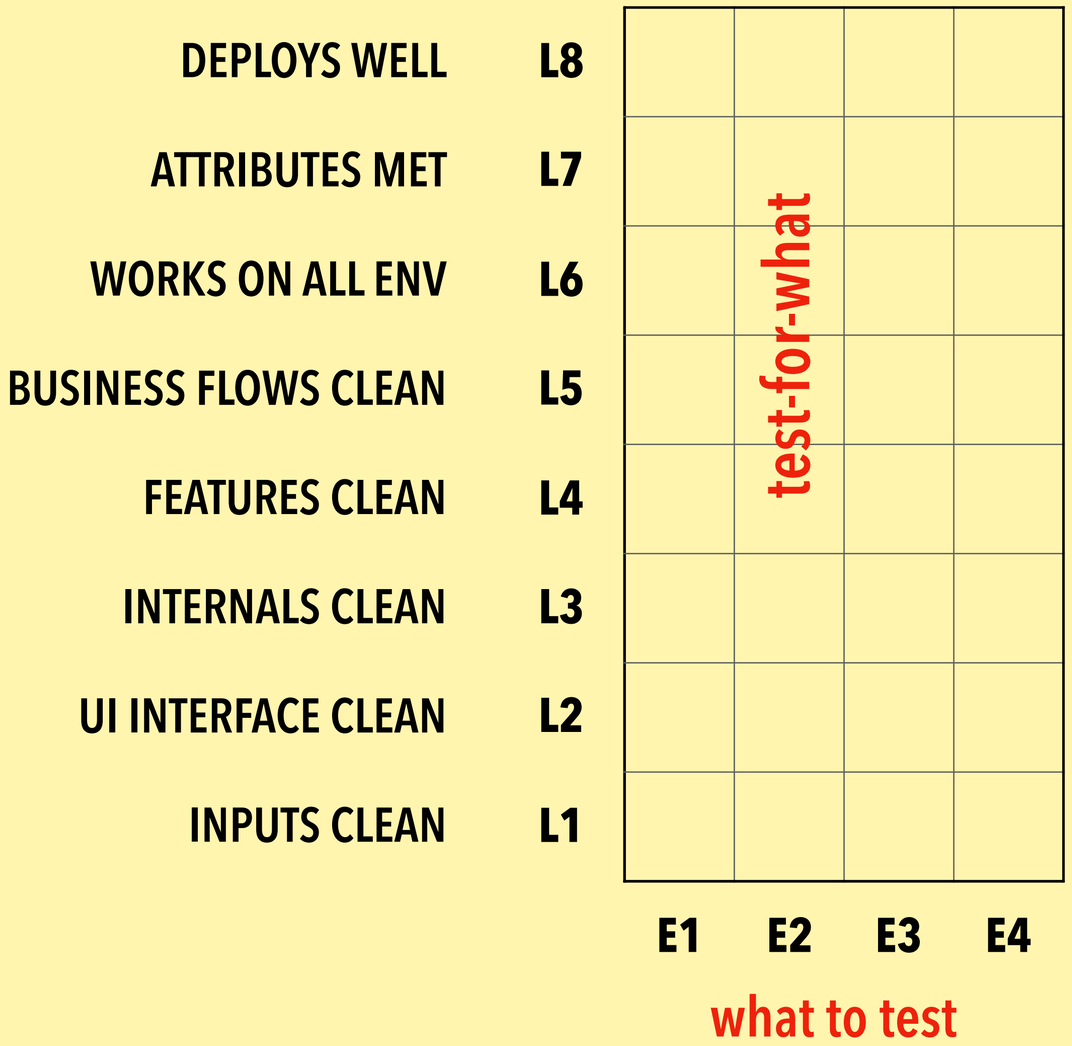
the challenge of CHANGE

1. what to retest?
2. how to retest efficiently?

the challenge of CHANGE

1. what to retest?
2. how to retest efficiently?
3. what not-to retest?

A simple model of “WHAT-is-TESTING” to TEST/(RE)TEST smartly

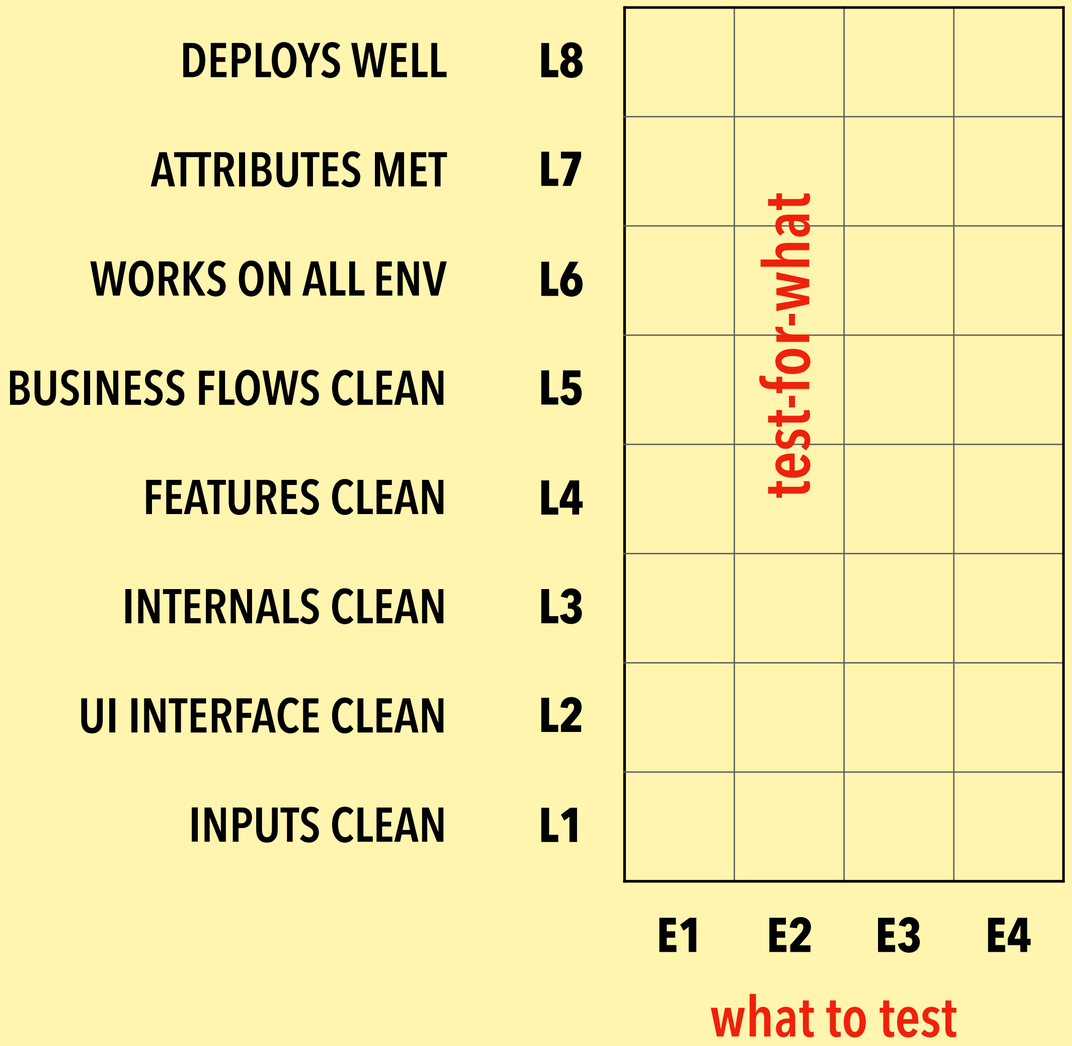


Testing is a cartesian product of

what-to-test

&

test-for-what



Testing is a cartesian product of

what-to-test

&

test-for-what

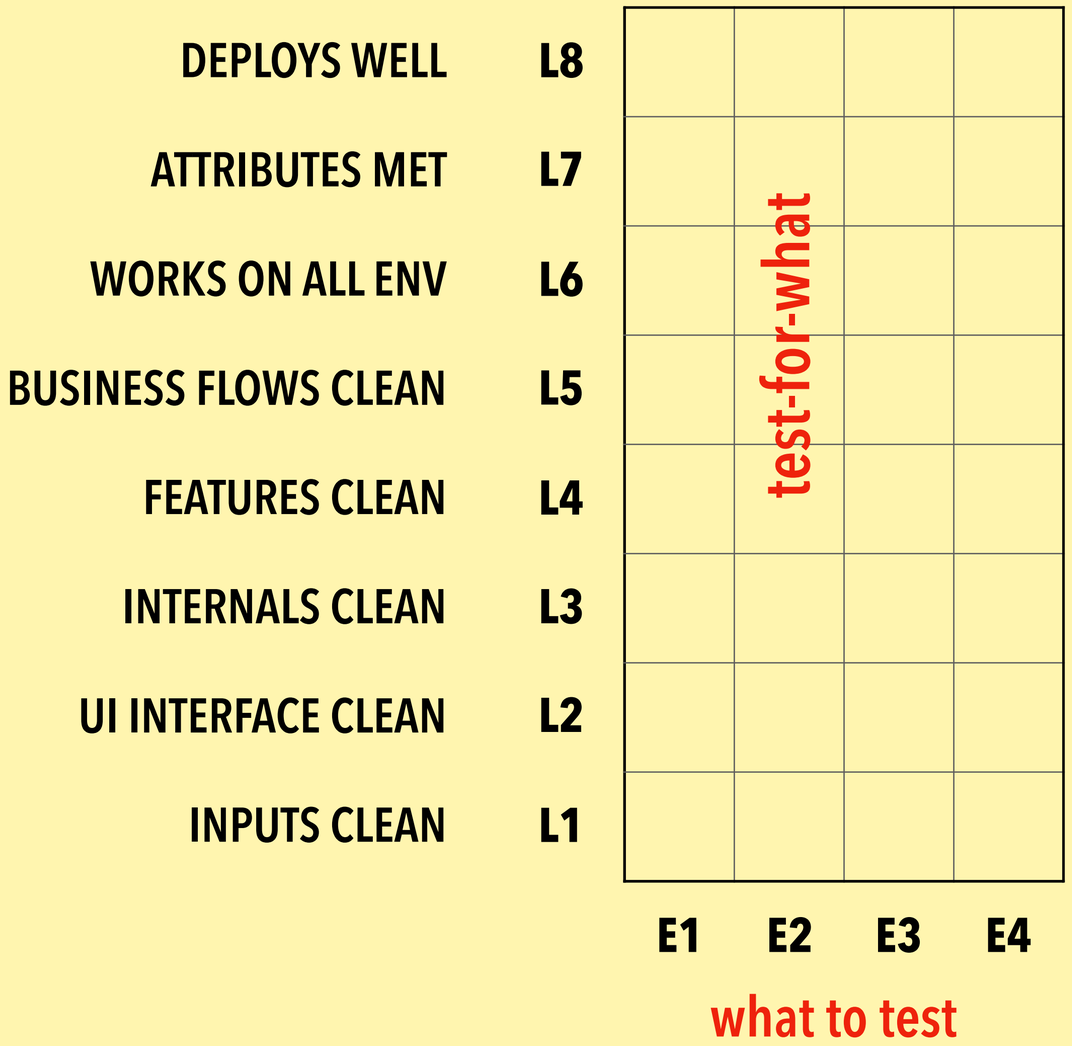
structural
COMPONENT

technical
FEATURE

business
REQUIREMENT

user
FLOW

WHAT IS THE EUT?



Testing is a cartesian product of

what-to-test

&

test-for-what

structural
COMPONENT

technical
FEATURE

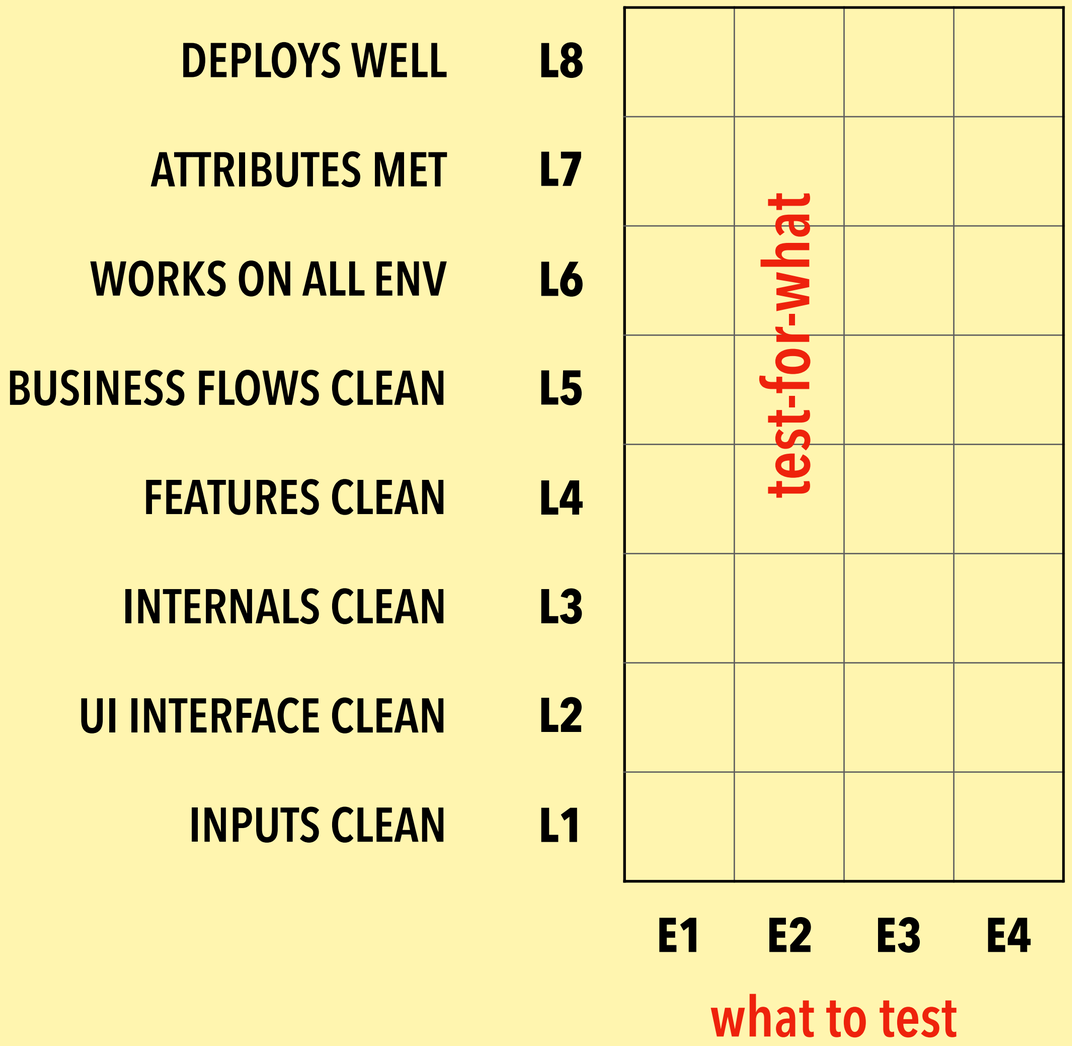
business
REQUIREMENT

user
FLOW

WHAT IS THE EUT?

deploys well?
attributes met?
works on all env?
business flows clean?
features clean?
internals clean?
UI interface clean?
inputs clean?

CLEANLINESS CRITERIA to TEST FOR?



Regression testing is then

what-to-(RE)test

x

(RE)test-for-what

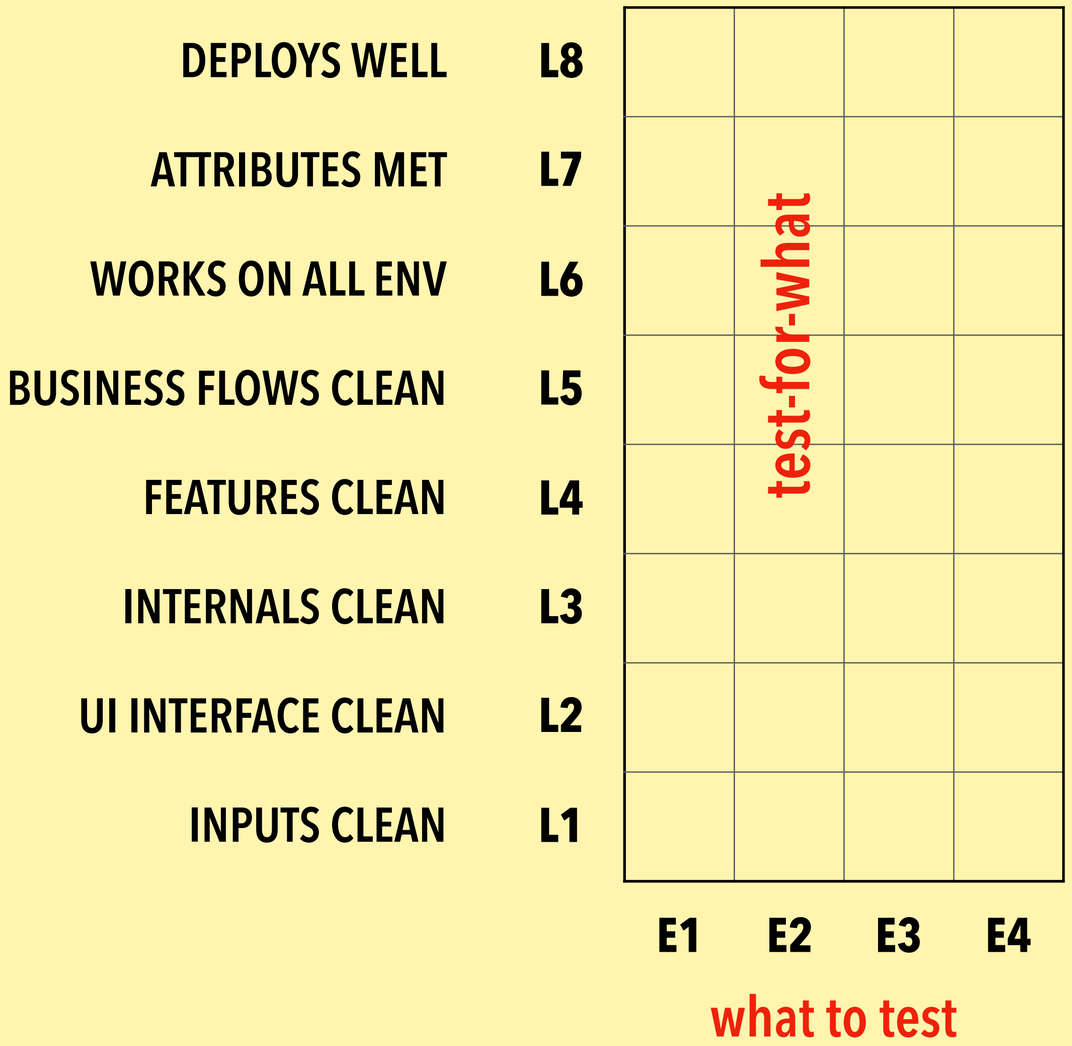
structural
COMPONENT

technical
FEATURE

business
REQUIREMENT

user
FLOW

WHAT IS THE EU(RE)T?



Regression testing is then

what-to-(RE)test

x

(RE)test-for-what

deploys well?
attributes met?
works on all env?
business flows clean?
features clean?
internals clean?
UI interface clean?
inputs clean?

CLEANLINESS CRITERIA to (RE)TEST FOR?

structural
COMPONENT

technical
FEATURE

business
REQUIREMENT

user
FLOW

WHAT IS THE EU(RE)T?

SMART approach to tackling the challenge of CHANGE

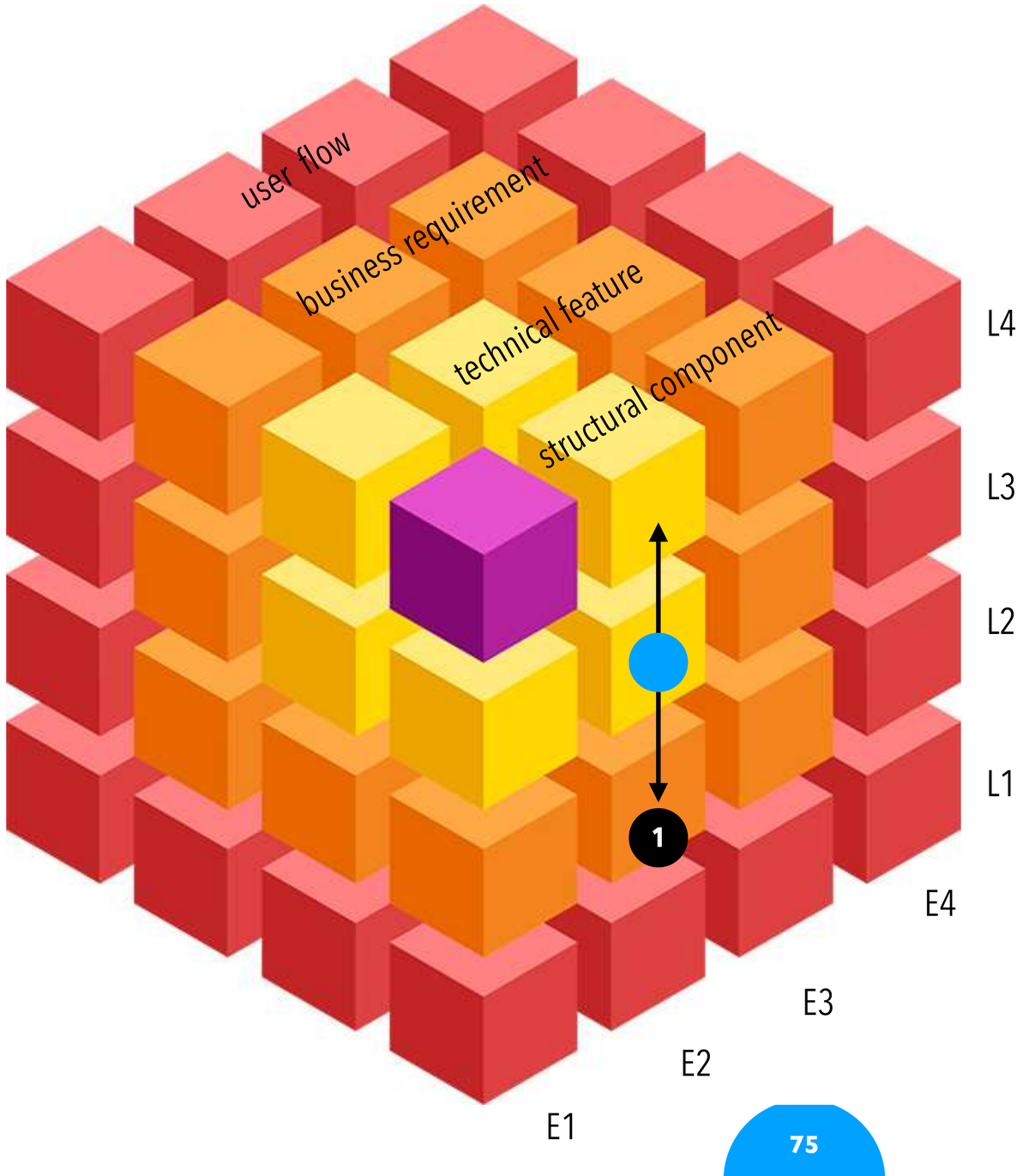
1. what to retest?
   Fault propagation analysis
2. how to retest efficiently?
   Automation analysis
3. what not-to retest?
   Yield analysis

SMART approach to tackling the challenge of CHANGE

1. what to retest?

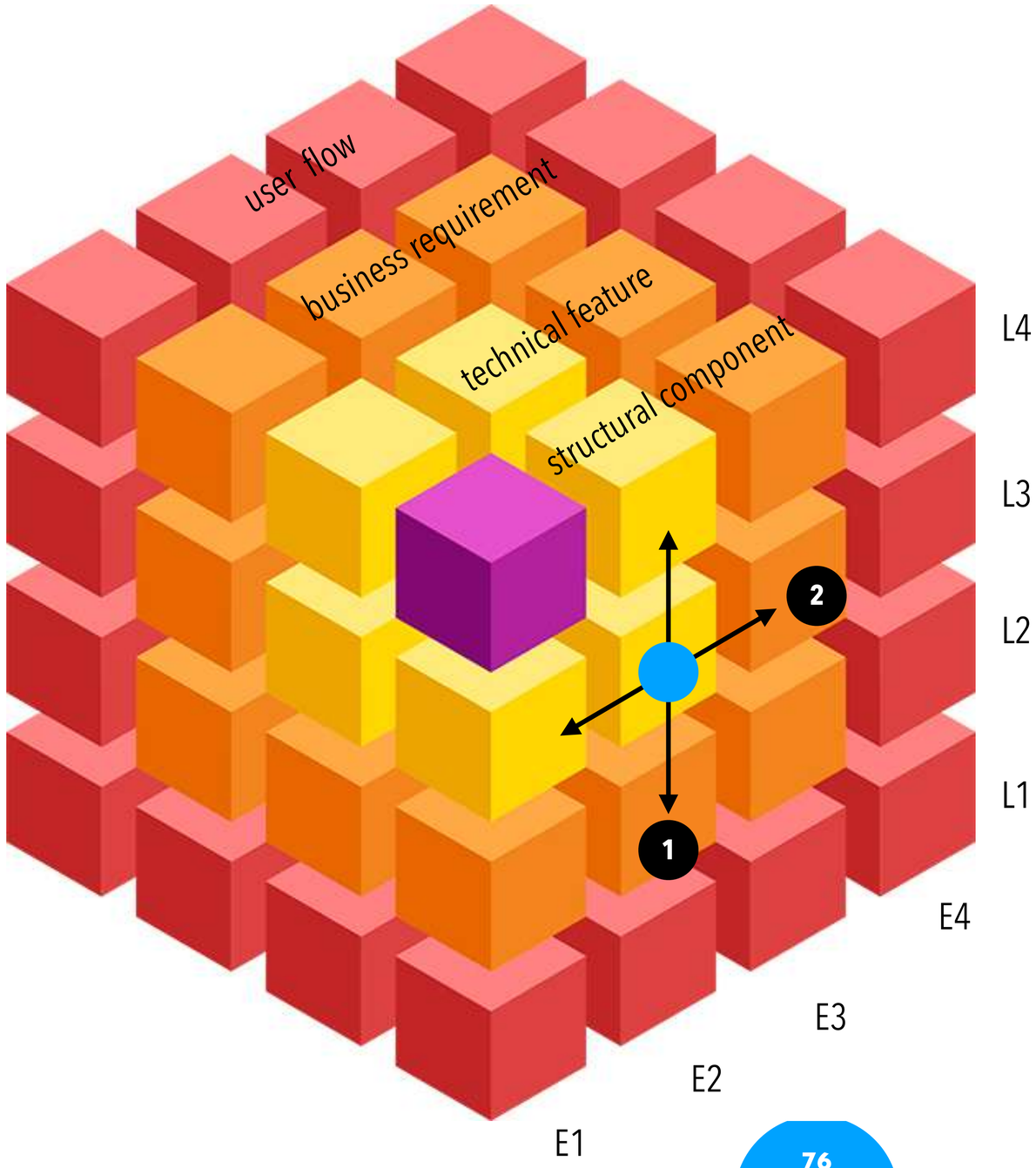
what may be the affected entities that need to be (re)tested?

what criteria of these entities are to be (re)tested for?



Given an entity (say, an component) that has been modified

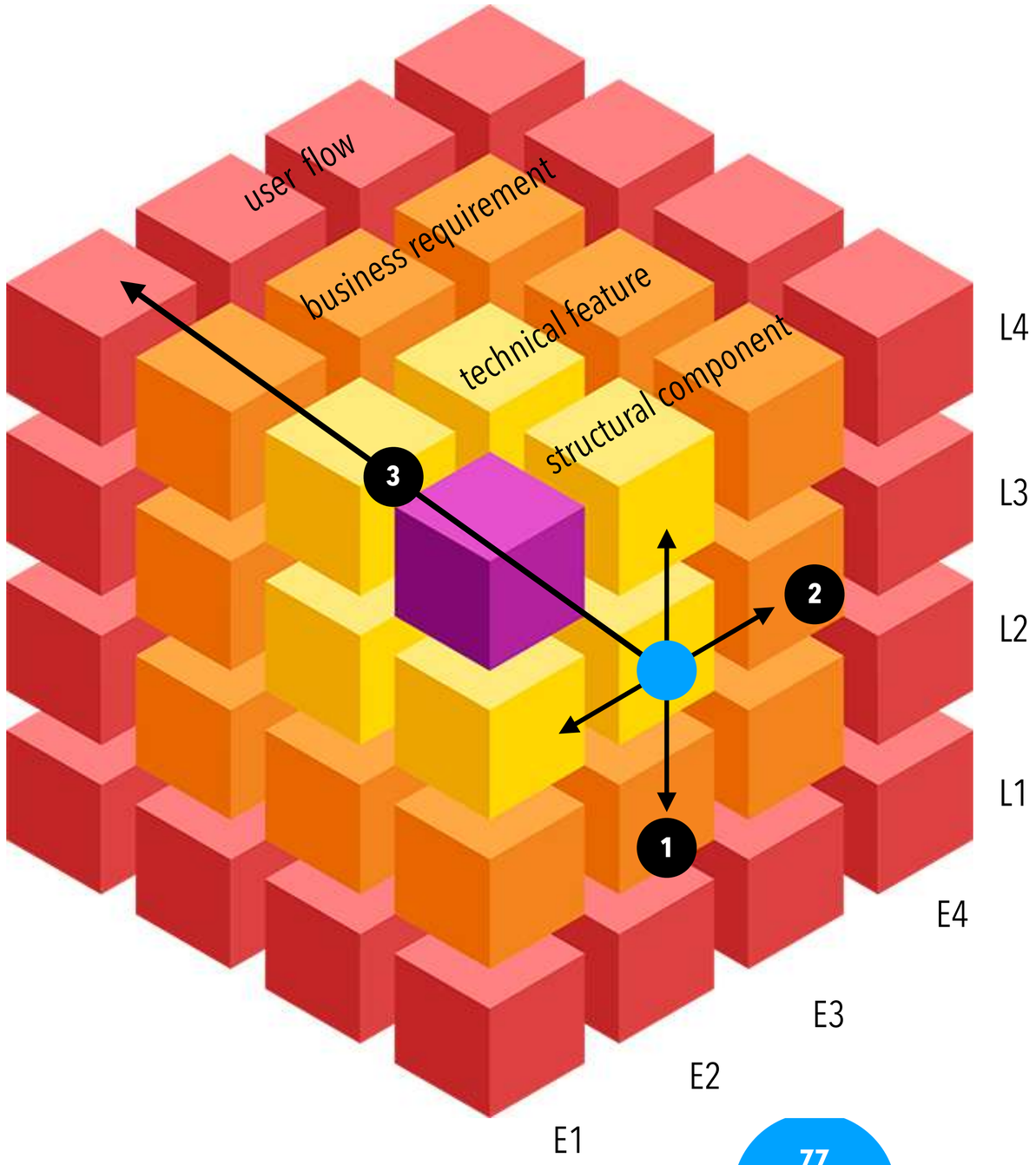
1. Analyze if this could affect any of its other criteria ?
   e.g. performance?



Given an entity (say, an component) that has 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 (say component) and what criteria of that entity



Given an entity (say, an component) that has 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 (say component) and what criteria of that entity
3. Finally analyze which of the larger entity (say feature) that uses this entity could be affected and also the potential affected criteria

GIVEN THE FOLLOWING LEVELS OF QUALITY :

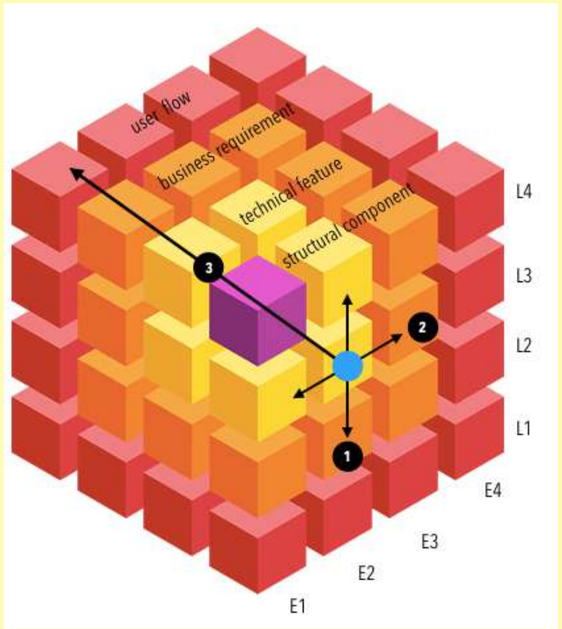
L8	DEPLOYS WELL
L7	ATTRIBUTES MET
L6	WORKS ON ALL ENVIRONMENTS
L5	BUSINESS FLOWS CLEAN
L4	FEATURES CLEAN
L3	INTERNALS CLEAN
L2	UI INTERFACE CLEAN
L1	INPUTS CLEAN

To enable focused purposeful tests that validates correctness of inputs, interfaces, going on to features & flows and then correctness on all environments, finally onto system attributes and deployment.

AID #1:
FAULT PROPAGATION ANALYZER

Who is affected? WHAT-to-RETEST
What is affected?-RE-TES T for WHAT?

Analyse fault propagation level-wise
- within an entity (1)
- across entities (2,3)



Given an entity (say, an component) that has been modified:

1. Analyze if this could any of its other criteria e.g. performance?

2. Next analyze if this could affect any other similar entity (say component) and what criteria of that entity

3. Finally analyze which of the larger entity (say feature) that uses this entity could be affected and also the potential affected criteria

SMART approach to tackling the challenge of CHANGE

1. what to retest?
   Fault propagation analysis

2. how to retest efficiently?

well, automating execution is useful here, challenge is scripts to be in sync with scenarios

ensure that scenarios are segregated by levels so that scripts are shorter and maintainable

L8	DEPLOYS WELL
L7	ATTRIBUTES MET
L6	WORKS ON ALL ENVIRONMENTS
L5	BUSINESS FLOWS CLEAN
L4	FEATURES CLEAN
L3	INTERNALS CLEAN
L2	UI INTERFACE CLEAN
L1	INPUTS CLEAN

Given test scenarios/cases for an entity that does a variety of validations

“LEVELISE” them

Segregate them into quality levels so that that they can be automated easily ensuring that scripts are simple and maintainable.

GIVEN THE FOLLOWING LEVELS OF QUALITY :

L8	DEPLOYS WELL
L7	ATTRIBUTES MET
L6	WORKS ON ALL ENVIRONMENTS
L5	BUSINESS FLOWS CLEAN
L4	FEATURES CLEAN
L3	INTERNALS CLEAN
L2	UI INTERFACE CLEAN
L1	INPUTS CLEAN

To enable focused purposeful tests that validates correctness of inputs, interfaces, going on to features & flows and then correctness on all environments, finally onto system attributes and deployment.

AID #2:
AUTOMATION FITNESS ANALYZER

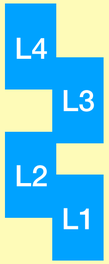
Are your test cases well structured to enable rapid automation/maintenance?

1

TC well structured into levels.
=> FOCUSED TEST CASES

very FIT for automation

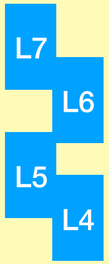
2



TC at levels L1-L4 mixed up.
=>SYSTEM TEST includes DEV TEST
Potentially long scripts, brittle, high maintenance

not FIT for automation

3



TC at levels L4-L7 mixed up.
=> TC validates FEATURES, FLOWS and ATTRIBUTES
Scripts may do too much, fragile high maintenance

not FIT for automation

*TC= Test Case

Segregate them into quality levels so that that they can be automated easily ensuring that scripts are simple and maintainable.

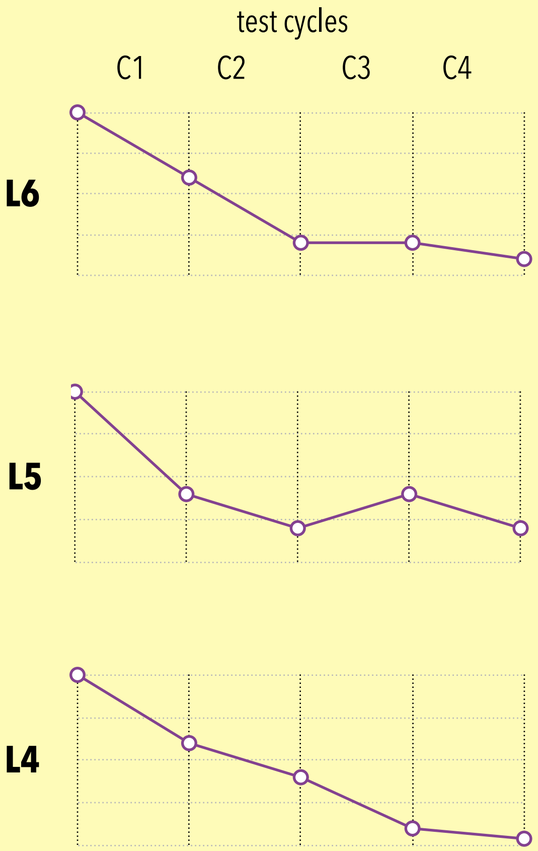
SMART approach to tackling the challenge of CHANGE

1. what to retest?
   Fault propagation analysis
2. how to retest efficiently?
   Automation analysis
3. what not-to retest?

as testing progresses, test cases don't uncover defects

track test cases that do not yield defects , to not execute?

L8	DEPLOYS WELL
L7	ATTRIBUTES MET
L6	WORKS ON ALL ENVIRONMENTS
L5	BUSINESS FLOWS CLEAN
L4	FEATURES CLEAN
L3	INTERNALS CLEAN
L2	UI INTERFACE CLEAN
L1	INPUTS CLEAN



*TC= Test Case

Track test cases that do NOT yield defects each cycle

Normalise defect/TC* ratio for each cycle

Analyse by levels to see yield wrt time

GIVEN THE FOLLOWING LEVELS OF QUALITY :

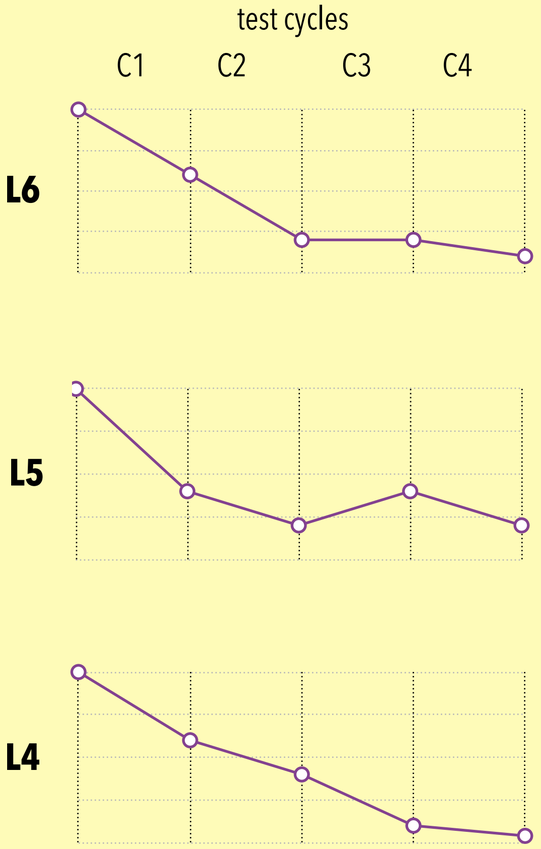
L8	DEPLOYS WELL
L7	ATTRIBUTES MET
L6	WORKS ON ALL ENVIRONMENTS
L5	BUSINESS FLOWS CLEAN
L4	FEATURES CLEAN
L3	INTERNALS CLEAN
L2	UI INTERFACE CLEAN
L1	INPUTS CLEAN

To enable focused purposeful tests that validates correctness of inputs, interfaces, going on to features & flows and then correctness on all environments, finally onto system attributes and deployment.

AID #3:
YIELD ANALYZER

How is the test yield over time?
yield = outcome/effort i.e #defects/#TC_Executed

Normalise defect/TC* ratio for each cycle and analyse by levels to see yield wrt time.



*TC= Test Case

Track test cases that do NOT yield defects each cycle

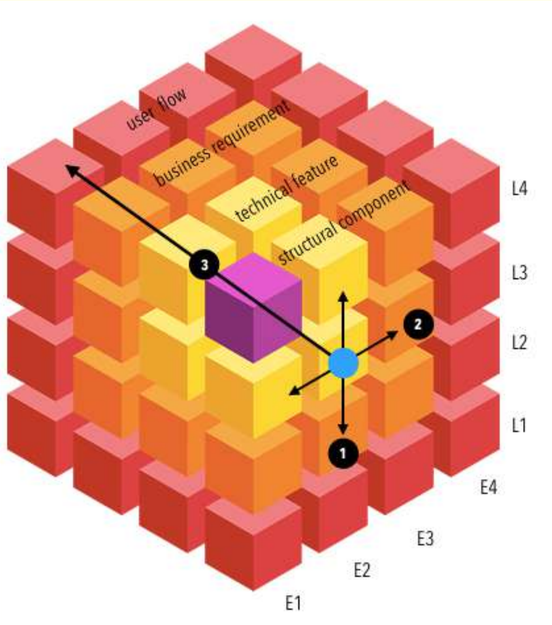
Normalise defect/TC* ratio for each cycle

Analyse by levels to see yield wrt time

AID #1:
FAULT PROPAGATION ANALYZER

Who is affected? WHAT-to-RETEST
What is affected?-RE-TES T for WHAT?

Analyse fault propagation level-wise
- within an entity (1)
- across entities (2,3)



AID #2:
AUTOMATION FITNESS ANALYZER

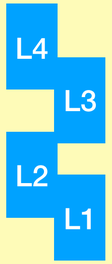
Are your test cases well structured to enable rapid automation/maintenance?

1

TC well structured into levels.
=> FOCUSED TEST CASES

very FIT for automation

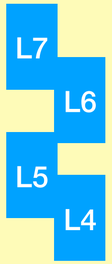
2



TC at levels L1-L4 mixed up.
=>SYSTEM TEST includes DEV TEST
Potentially long scripts, brittle, high maintenance

not FIT for automation

3



TC at levels L4-L7 mixed up.
=> TC validates FEATURES, FLOWS and ATTRIBUTES
Scripts may do too much, fragile high maintenance

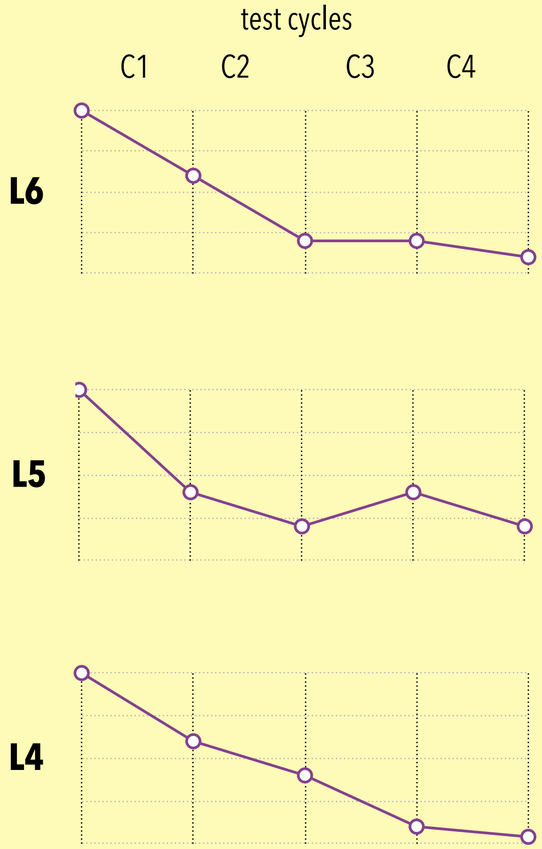
not FIT for automation

*TC= Test Case

AID #3:
YIELD ANALYZER

How is the test yield over time?
yield = outcome/effort i.e #defects/#TC_Executed

Normalise defect/TC* ratio for each cycle and analyse by levels to see yield wrt time.



SMART approach to tackling the challenge of CHANGE

1. what to retest?
   Fault propagation analysis

2. how to retest efficiently?
   Automation analysis

3. what not-to retest?
   Yield analysis

Thank you.

www.stagsoftware.com