Chapter 1

Software Testing

ISTQB / ISEB Foundation Exam Practice

Principles of Testing

1 Principles	2 Lifecycle	3 Static testing
4 Test design techniques	5 Management	6 Tools

1	2	3
4	5	6

Principles

Contents

Why testing is necessary

What is testing

Testing principles

Fundamental test process

Psychology of testing

Code of Ethics

Testing terminology

- No generally accepted set of testing definitions used world wide
- New standard BS 7925-1
  - Glossary of testing terms (emphasis on component testing)
  - most recent developed by a working party of the BCS SIGIST
  - adopted by the ISEB / ISTQB

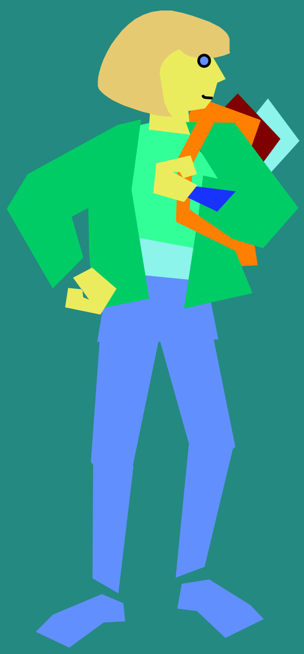
What is a “bug”?

- **Error:** a human action that produces an incorrect result
- **Fault:** a manifestation of an error in software
  - also known as a defect or bug
  - if executed, a fault may cause a failure
- **Failure:** deviation of the software from its expected delivery or service
  - (found defect)

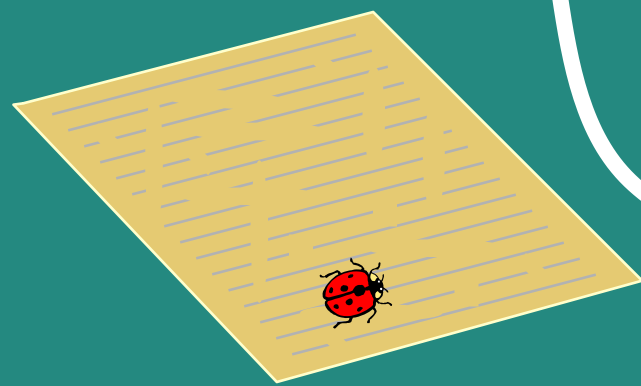
Failure is an event; fault is a state of the software, caused by an error

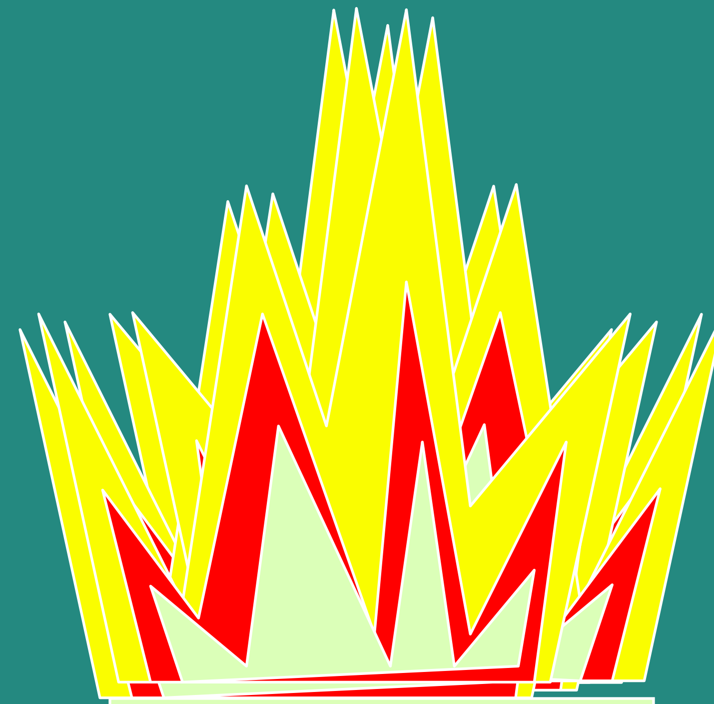
Error - Fault - Failure

A person makes an error ...



… that creates a fault in the software ...





… that can cause a failure in operation

Reliability versus faults

- **Reliability: the probability that software will not cause the failure of the system for a specified time under specified conditions**
  - **Can a system be fault-free? (zero faults, right first time)** ✗
  - **Can a software system be reliable but still have faults?** ✓
  - **Is a "fault-free" software application always reliable?** ✗

Why do faults occur in software?

- software is written by human beings
  - who know something, but not everything
  - who have skills, but aren’t perfect
  - who do make mistakes (errors)
- under increasing pressure to deliver to strict deadlines
  - no time to check but assumptions may be wrong
  - systems may be incomplete
- if you have ever written software ...

What do software faults cost?

- huge sums
  - Ariane 5 ($7billion)
  - Mariner space probe to Venus ($250m)
  - American Airlines ($50m)
- very little or nothing at all
  - minor inconvenience
  - no visible or physical detrimental impact
- software is not “linear”:
  - small input may have very large effect

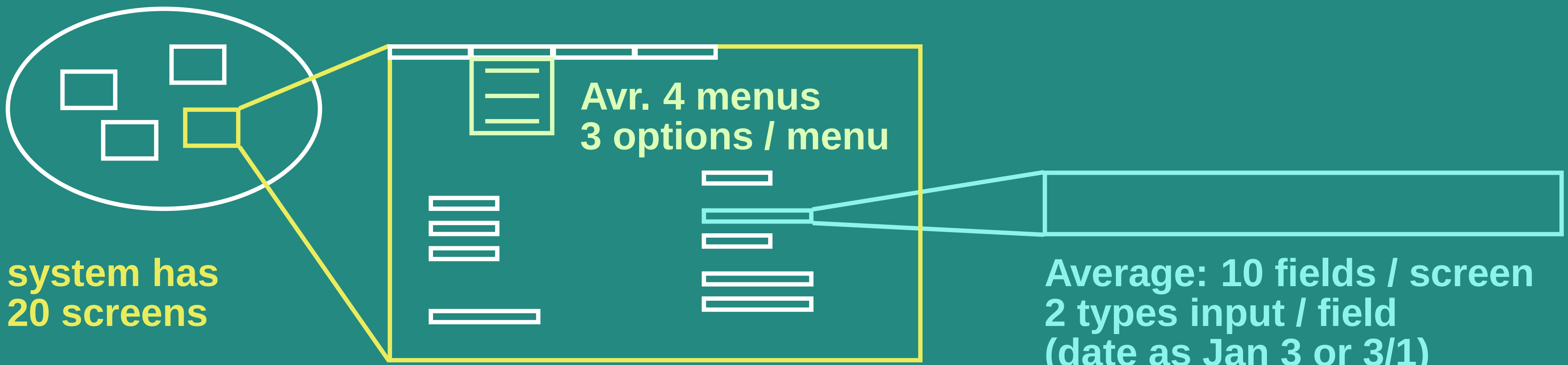
Safety-critical systems

- software faults can cause death or injury
  - radiation treatment kills patients (Therac-25)
  - train driver killed
  - aircraft crashes (Airbus & Korean Airlines)
  - bank system overdraft letters cause suicide

So why is testing necessary?

- because software is likely to have faults ✓
- to learn about the reliability of the software ✓
- to fill the time between delivery of the software and the release date ✗
- to prove that the software has no faults ✗
- because testing is included in the project plan ✗
- because failures can be very expensive ✓
- to avoid being sued by customers ✓
- to stay in business ✓

Why not just "test everything"?



system has
20 screens

Average: 10 fields / screen
2 types input / field
(date as Jan 3 or 3/1)
(number as integer or decimal)
Around 100 possible values

Total for 'exhaustive' testing:

20 x 4 x 3 x 10 x 2 x 100 = 480,000 tests

If <u>1 second</u> per test, 8000 mins, 133 hrs, 17.7 days

(not counting finger trouble, faults or retest)

10 secs = 34 wks, 1 min = 4 yrs, 10 min = 40 yrs

Exhaustive testing?

- What is exhaustive testing?
  - when all the testers are exhausted ✗
  - when all the planned tests have been executed ✗
  - exercising all combinations of inputs and preconditions ✓
- How much time will exhaustive testing take?
  - infinite time ✗
  - not much time ✗
  - impractical amount of time ✓

How much testing is enough?

- it's never enough ✗
- when you have done what you planned ✗
- when your customer/user is happy ✗
- when you have proved that the system works correctly ✗
- when <u>you are confident</u> that the system works correctly ✓
- it <u>depends on the risks</u> for your system ✓

How much testing?

- It depends on **RISK**
  - risk of missing important faults
  - risk of incurring failure costs
  - risk of releasing untested or under-tested software
  - risk of losing credibility and market share
  - risk of missing a market window
  - risk of over-testing, ineffective testing

So little time, so much to test ..

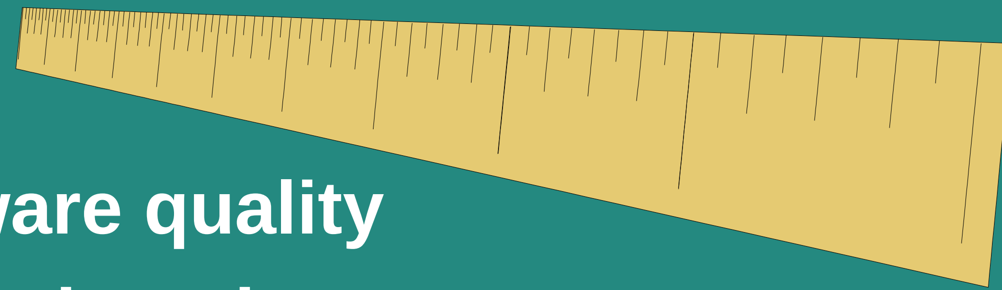
- test time will always be limited
- use **RISK** to determine:
  - what to test first
  - what to test most
  - how thoroughly to test each item
  - what not to test (this time)

  i.e. where to place emphasis
- use **RISK** to
  - allocate the time available for testing by prioritising testing ...

Most important principle

Prioritise tests
so that,
whenever you stop testing,
you have done the best testing
in the time available.

Testing and quality



- testing measures software quality
- testing can find faults; when they are removed, software quality (and possibly reliability) is improved
- what does testing test?
  - system function, correctness of operation
  - non-functional qualities: reliability, usability, maintainability, reusability, testability, etc.

Other factors that influence testing

- contractual requirements
- legal requirements
- industry-specific requirements
  - e.g. pharmaceutical industry (FDA), compiler standard tests, safety-critical or safety-related such as railroad switching, air traffic control

It is difficult to determine how much testing is enough but it is not impossible

1	2	3
4	5	6

Principles

Contents

Why testing is necessary

What is testing

Testing principles

Fundamental test process

Psychology of testing

Code of Ethics

Seven Testing Principles

- Testing shows the presence of defects
- Exhaustive testing is impossible
- Early testing
- Defect clustering
- Pesticide paradox
- Testing is context dependent
- Absence-of-errors fallacy

1	2	3
4	5	6

ISTQB / ISEB Foundation Exam Practice

Contents

Why testing is necessary

What is testing

Testing principles

Fundamental test process

Psychology of testing

Code of Ethics

Test Planning - different levels

Test Policy Test Strategy	Company level
High Level Test Plan	Project level (IEEE 829) (one for each project)
Detailed Test Plan	Test stage level (IEEE 829) (one for each stage within a project, e.g. Component, System, etc.)

The test process

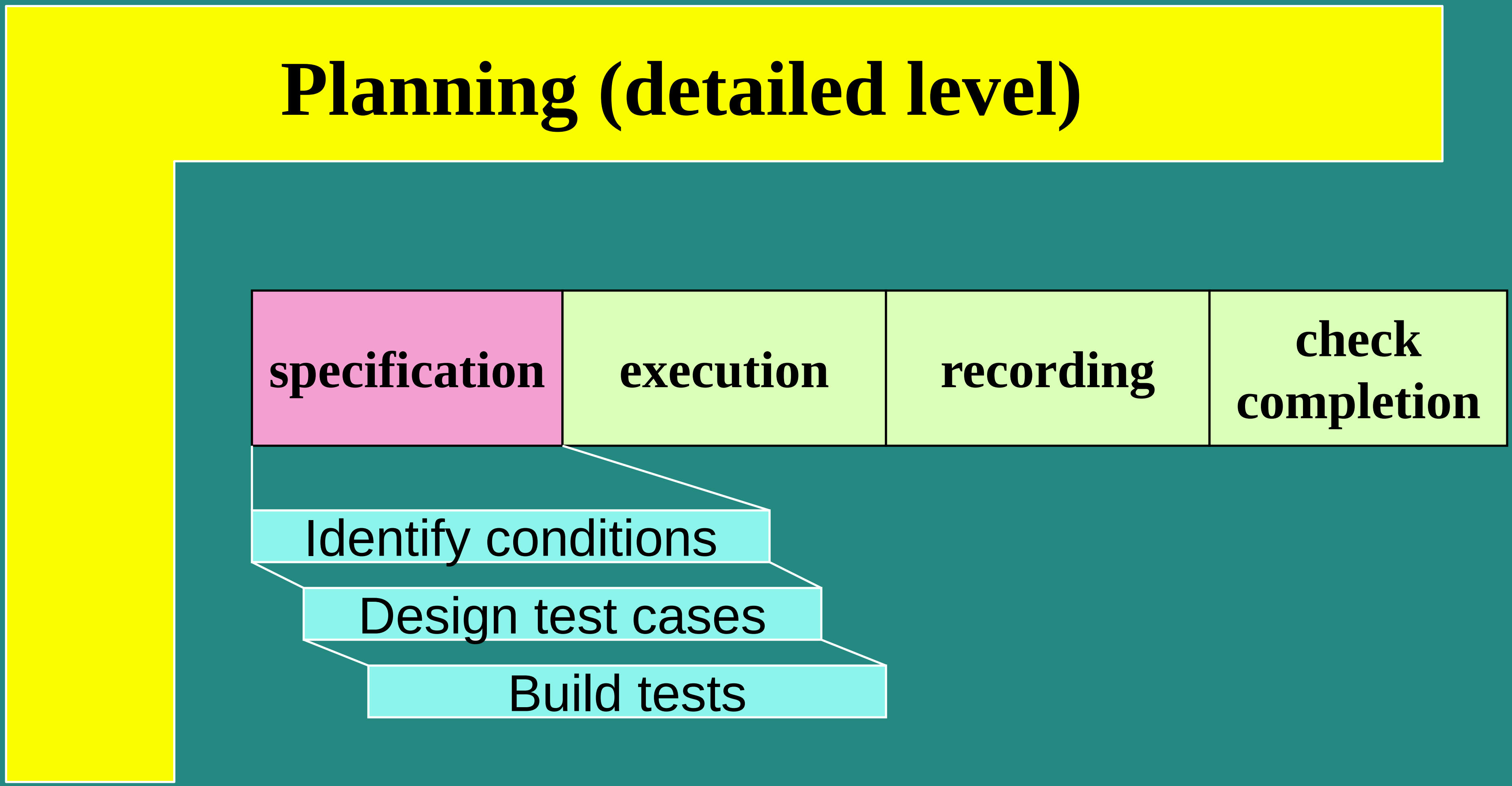
Planning (detailed level)

specification	execution	recording	check completion

Test planning

- how the test strategy and project test plan apply to the software under test
- document any exceptions to the test strategy
  - e.g. only one test case design technique needed for this functional area because it is less critical
- other software needed for the tests, such as <u>stubs and drivers</u>, and environment details
- set test completion criteria

Test specification



A good test case

- effective
- exemplary
- evolvable
- economic

Finds faults

Represents others

Easy to maintain

Cheap to use

Test specification

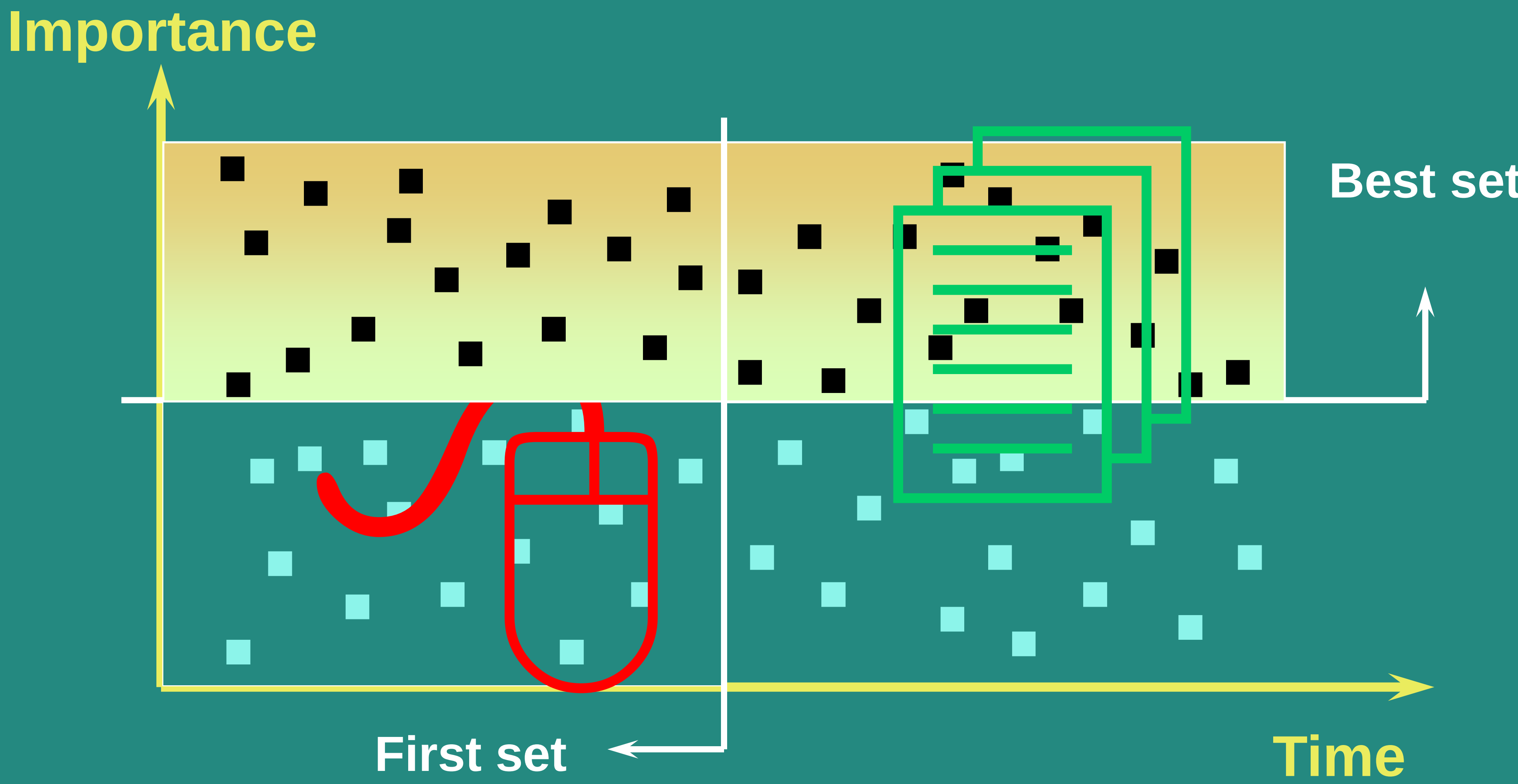
- test specification can be broken down into three distinct tasks:
  1. identify: determine 'what' is to be tested (identify test conditions) and prioritise
  2. design: determine 'how' the 'what' is to be tested (i.e. design test cases)
  3. build: implement the tests (data, scripts, etc.)

Task 1: identify conditions

(determine ‘what’ is to be tested and prioritise)

- **list the conditions that we would like to test:**
  - **use the test design techniques specified in the test plan**
  - **there may be many conditions for each system function or attribute**
  - **e.g.**
    - “life assurance for a winter sportsman”
    - “number items ordered > 99”
    - “date = 29-Feb-2004”
- **prioritise the test conditions**
  - **must ensure most important conditions are covered**

Selecting test conditions

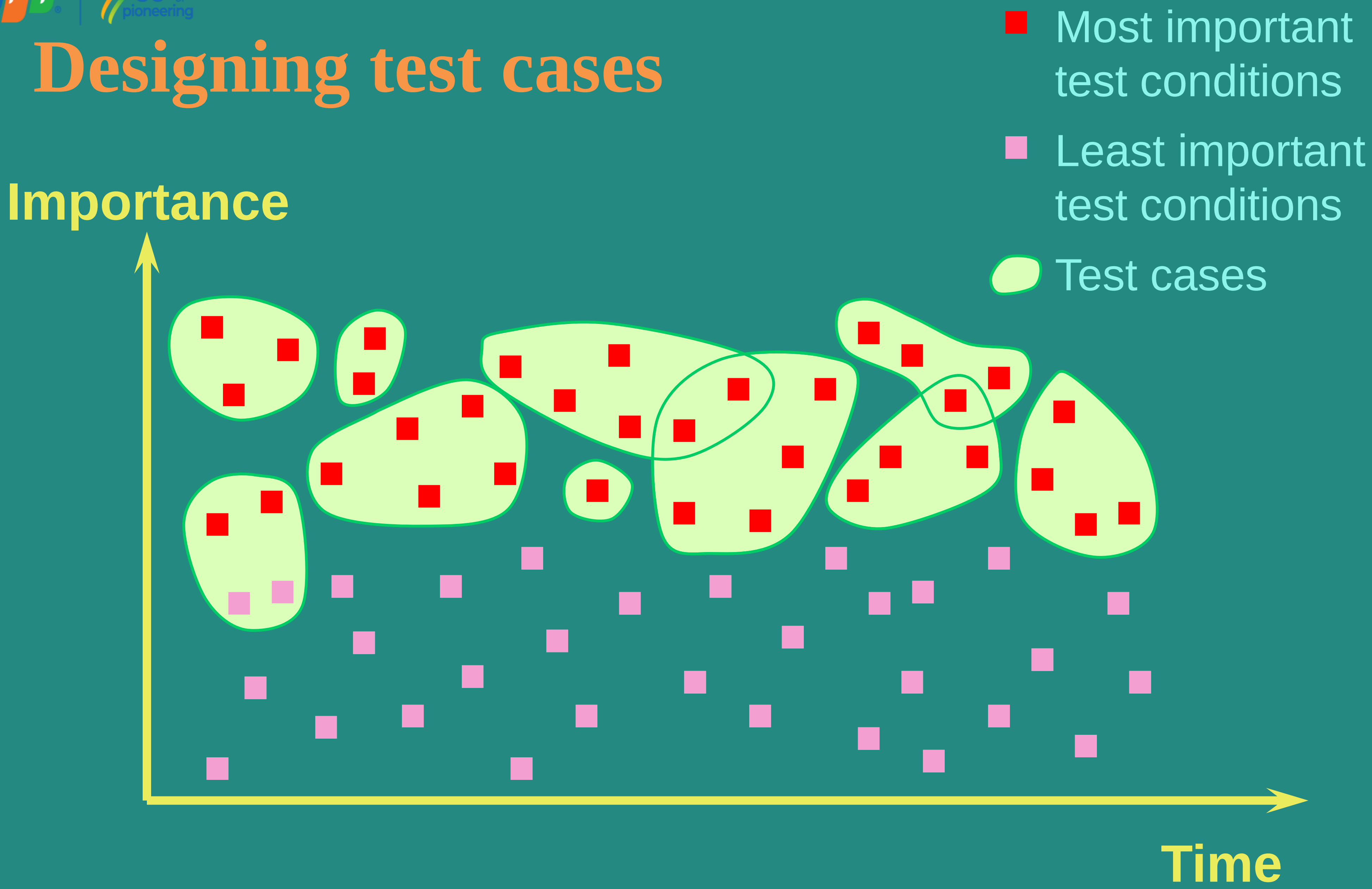


Task 2: design test cases

(determine ‘how’ the ‘what’ is to be tested)

- design test input and test data
  - each test exercises one or more test conditions
- determine expected results
  - predict the outcome of each test case, what is output, what is changed and what is not changed
- design sets of tests
  - different test sets for different objectives such as regression, building confidence, and finding faults

Designing test cases



Task 3: build test cases

(implement the test cases)

- **prepare test scripts**
  - **less system knowledge tester has the more detailed the scripts will have to be**
  - **scripts for tools have to specify every detail**
- **prepare test data**
  - **data that must exist in files and databases at the start of the tests**
- **prepare expected results**
  - **should be defined before the test is executed**

Test execution

Planning (detailed level)

specification	**execution**	**recording**	**check completion**

Execution

- **Execute prescribed test cases**
  - **most important ones first**
  - **would not execute all test cases if**
    - testing only fault fixes
    - too many faults found by early test cases
    - time pressure
  - **can be performed manually or automated**

Test recording

Planning (detailed level)

specification	execution	recording	check completion

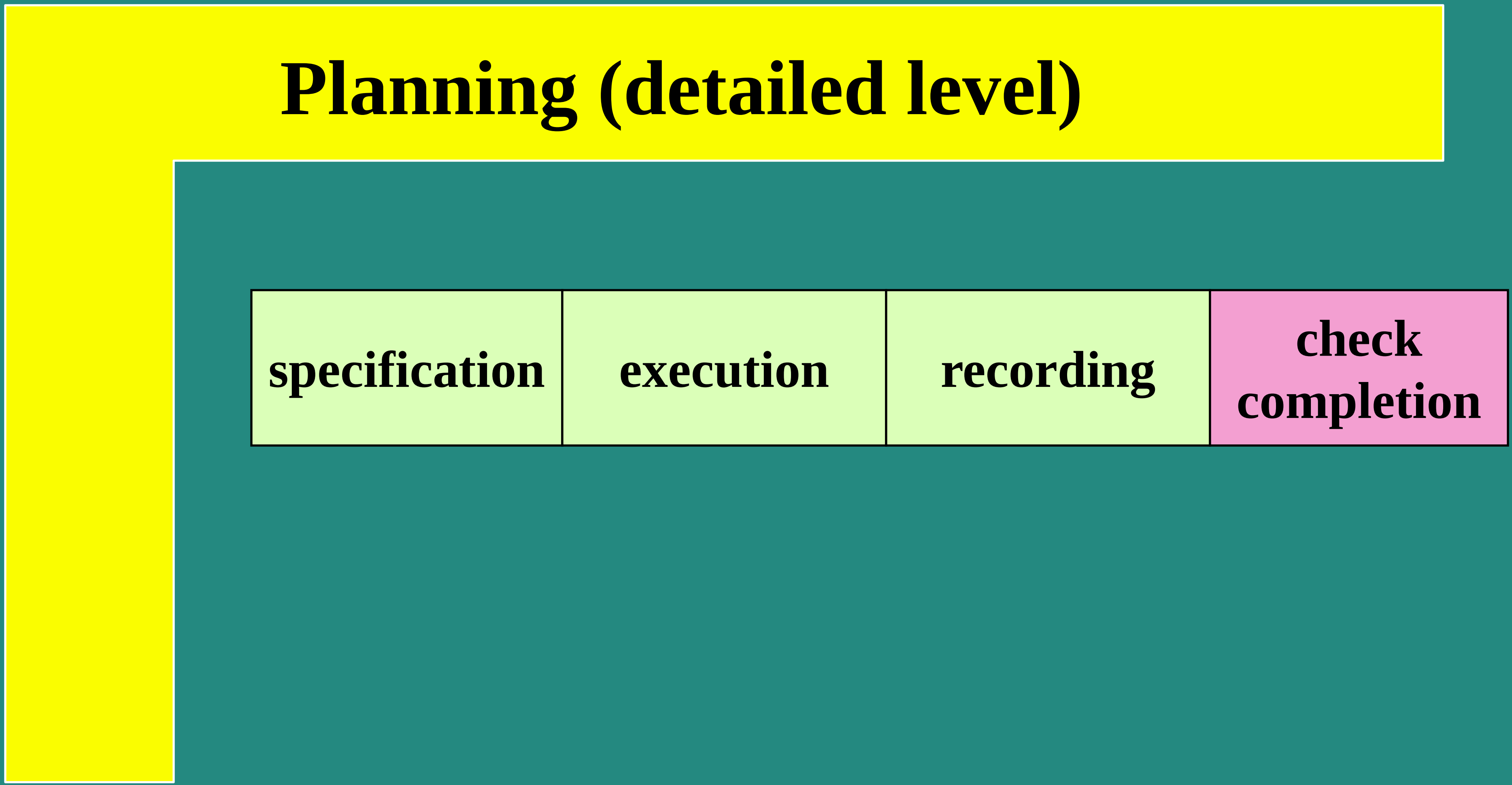
Test recording 1

- **The test record contains:**
  - **identities and versions (unambiguously) of**
    - software under test
    - test specifications
- **Follow the plan**
  - **mark off progress on test script**
  - **document actual outcomes from the test**
  - **capture any other ideas you have for new test cases**
  - **note that these records are used to establish that all test activities have been carried out as specified**

Test recording 2

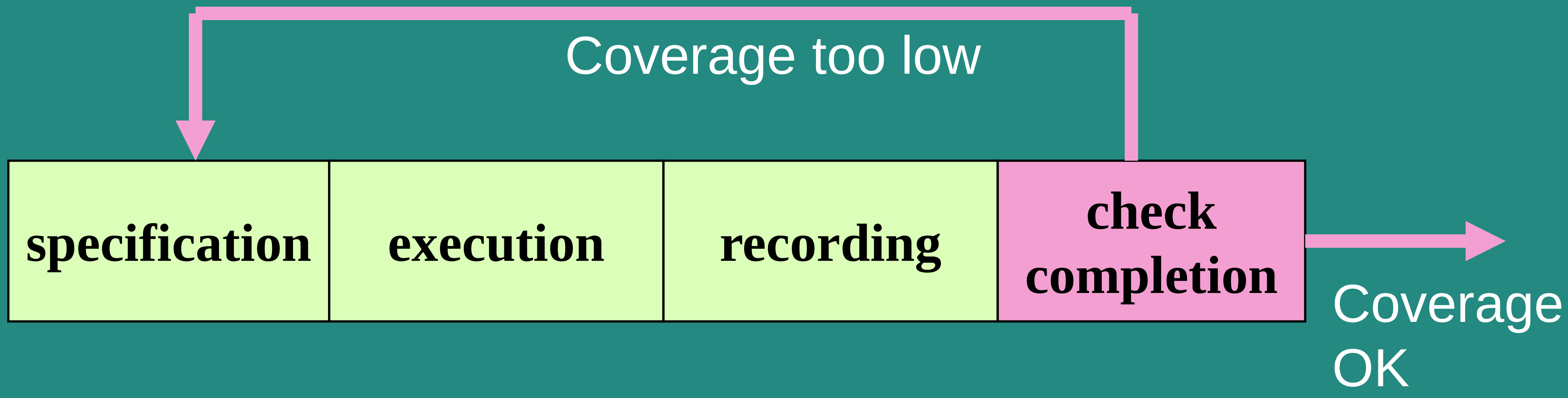
- Compare actual outcome with expected outcome. Log discrepancies accordingly:
  - software fault
  - test fault (e.g. expected results wrong)
  - environment or version fault
  - test run incorrectly
- Log coverage levels achieved (for measures specified as test completion criteria)
- After the fault has been fixed, repeat the required test activities (execute, design, plan)

Check test completion



Check test completion

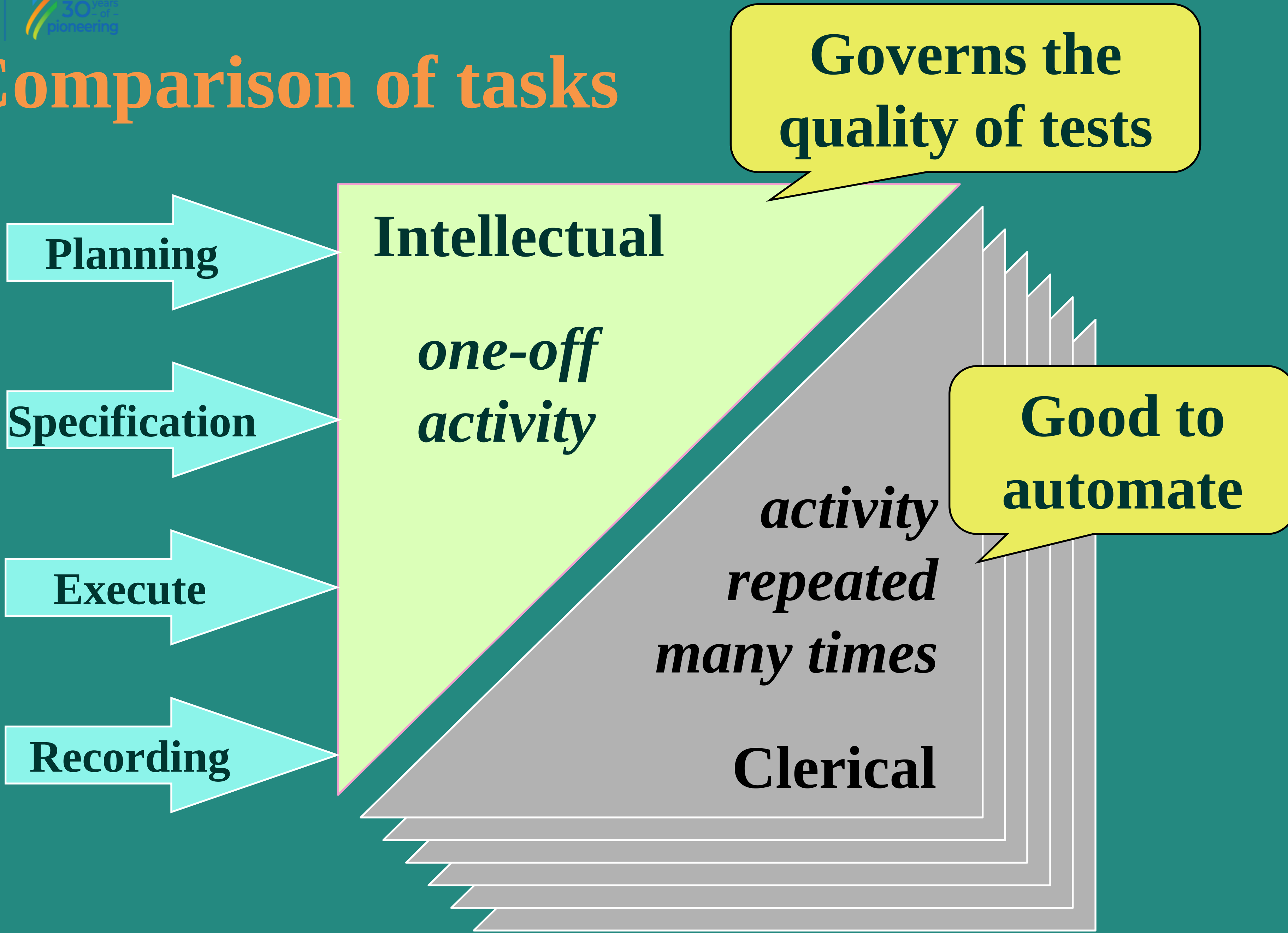
- Test completion criteria were specified in the test plan
- If not met, need to repeat test activities, e.g. test specification to design more tests



Test completion criteria

- **Completion or exit criteria apply to all levels of testing - to determine when to stop**
  - **coverage, using a measurement technique, e.g.**
    - branch coverage for unit testing
    - user requirements
    - most frequently used transactions
  - **faults found (e.g. versus expected)**
  - **cost or time**

Comparison of tasks



1	2	3
4	5	6

ISTQB / ISEB Foundation Exam Practice

Contents

Why testing is necessary

What is testing

Testing principles

Fundamental test process

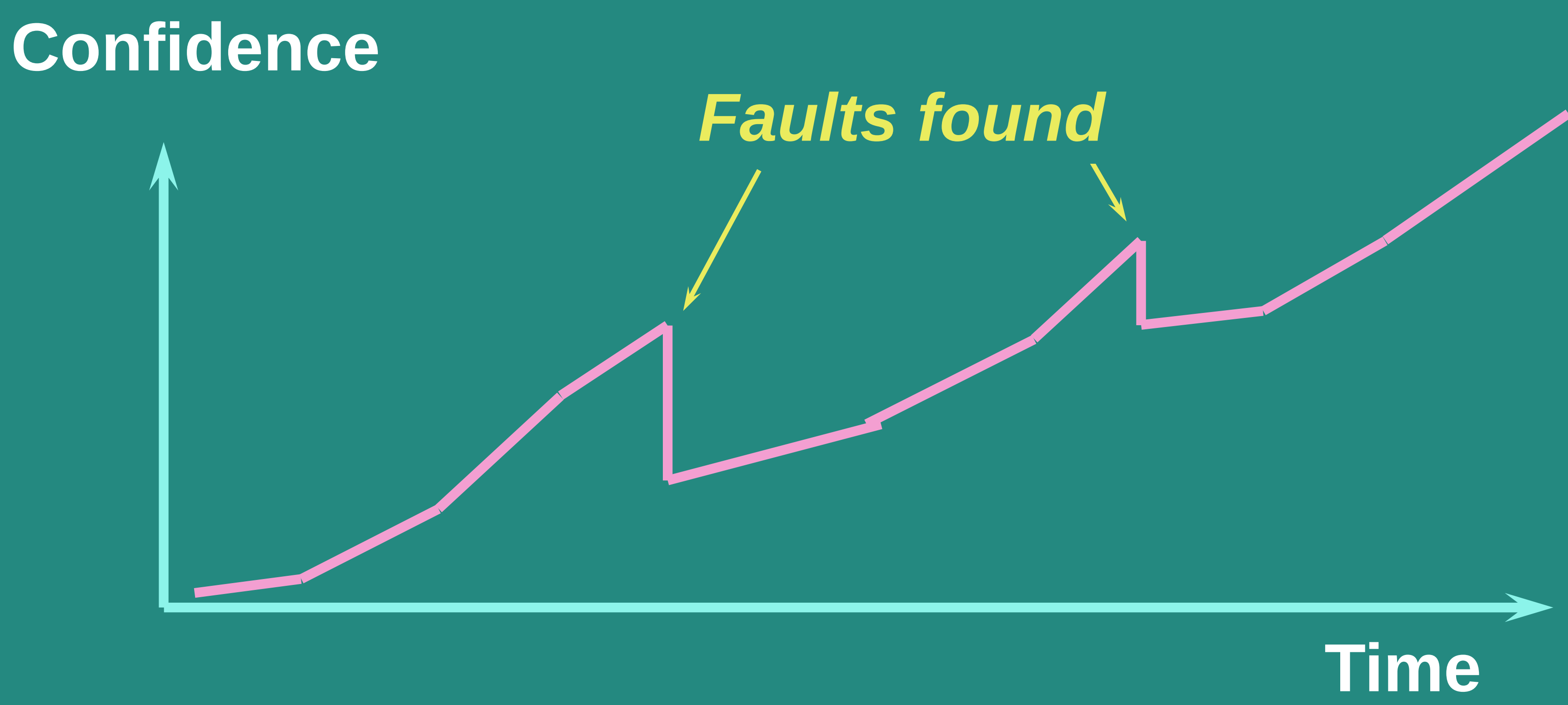
Psychology of testing

Code of Ethics

Why test?

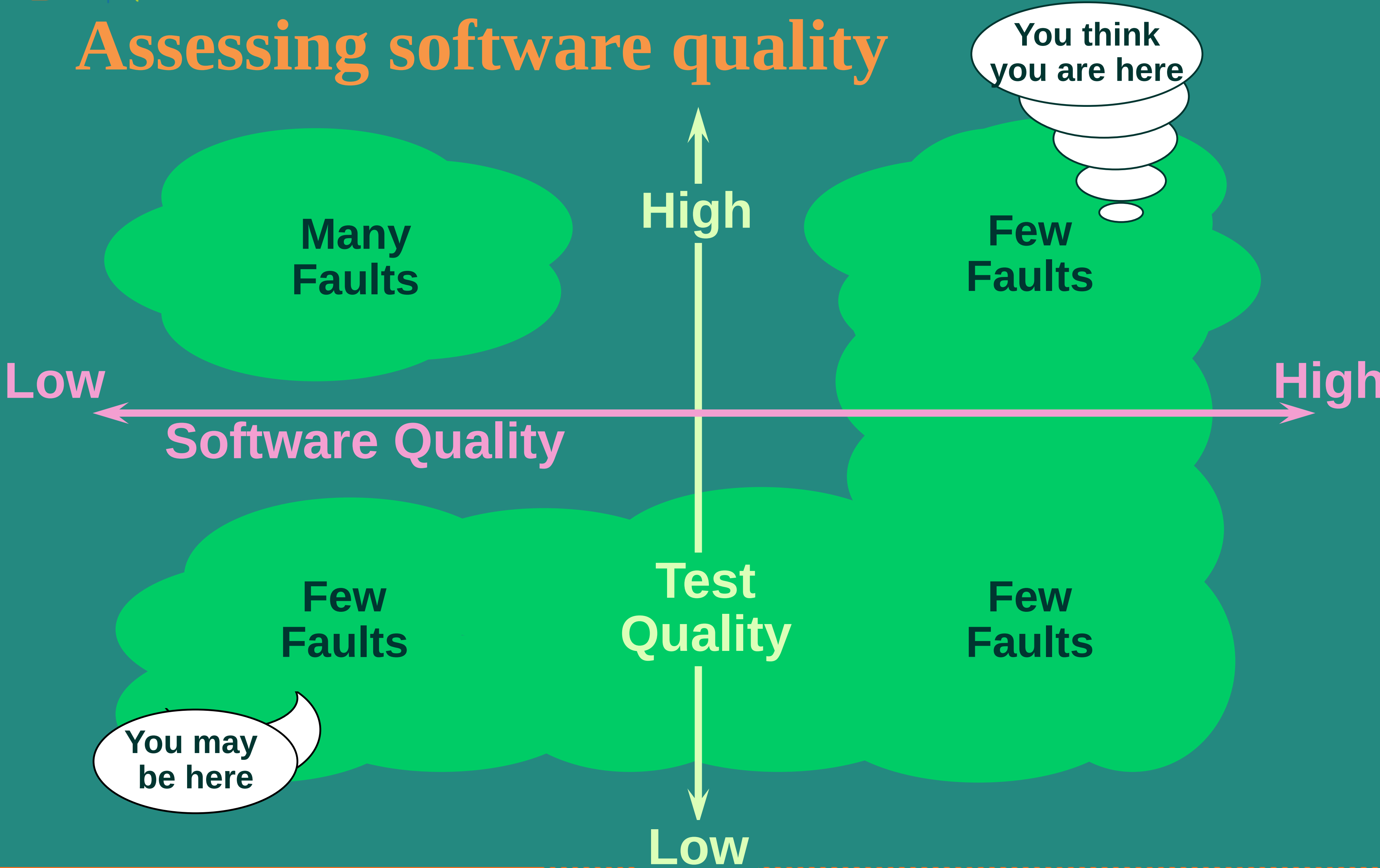
- build confidence ✓
- prove that the software is correct ✗
- demonstrate conformance to requirements ✓
- find faults ✓
- reduce costs ✓
- show system meets user needs ✓
- assess the software quality ✓

Confidence



No faults found = confidence?

Assessing software quality



A traditional testing approach

- Show that the system:
  - does what it should
  - doesn't do what it shouldn't

Goal:	show working
Success:	system works

Fastest achievement: easy test cases

→ Result: faults left in

A better testing approach

- Show that the system:
  - does what it shouldn't
  - doesn't do what it should

Goal:	find faults
Success:	system fails

Fastest achievement: difficult test cases

→ Result: fewer faults left in

The testing paradox

Purpose of testing: to find faults

Finding faults destroys confidence

∴ Purpose of testing: destroy confidence

Purpose of testing: build confidence

The best way to build confidence is to try to destroy it

Who wants to be a tester?

- A destructive process
- Bring bad news ("your baby is ugly")
- Under worst time pressure (at the end)
- Need to take a different view, a different mindset ("What if it isn't?", "What could go wrong?")
- How should fault information be communicated (to authors and managers?)

Tester's have the right to:

- accurate information about progress and changes
- insight from developers about areas of the software
- delivered code tested to an agreed standard
- be regarded as a professional (no abuse!)
- find faults!
- challenge specifications and test plans
- have reported faults taken seriously (non-reproducible)
- make predictions about future fault levels
- improve your own testing process

Testers have responsibility to:

- follow the test plans, scripts etc. as documented
- report faults objectively and factually (no abuse!)
- check tests are correct before reporting s/w faults
- remember it is the software, not the programmer, that you are testing
- assess risk objectively
- prioritise what you report
- communicate the truth

Independence

- **Test your own work?**
  - **find 30% - 50% of your own faults**
  - **same assumptions and thought processes**
  - **see what you meant or want to see, not what is there**
  - **emotional attachment**
    - don’t want to find faults
    - actively want NOT to find faults

Levels of independence

- None: tests designed by the person who wrote the software
- Tests designed by a different person
- Tests designed by someone from a different department or team (e.g. test team)
- Tests designed by someone from a different organisation (e.g. agency)
- Tests generated by a tool (low quality tests?)

1	2	3
4	5	6

ISTQB / ISEB Foundation Exam Practice

Contents

Why testing is necessary
What is testing
Testing principles
Fundamental test process
Psychology of testing
Code of Ethics

Code of Ethics

- ISTQB Code of Ethics:
  - **PUBLIC**
  - **CLIENT AND EMPLOYER**
  - **PRODUCT**
  - **JUDGMENT**
  - **MANAGEMENT**
  - **PROFESSION**
  - **COLLEAGUES**
  - **SELF**

1	2	3
4	5	6

Summary: Key Points

Why testing is necessary

What is testing

Testing principles

Fundamental test process

Psychology of testing

Code of Ethics