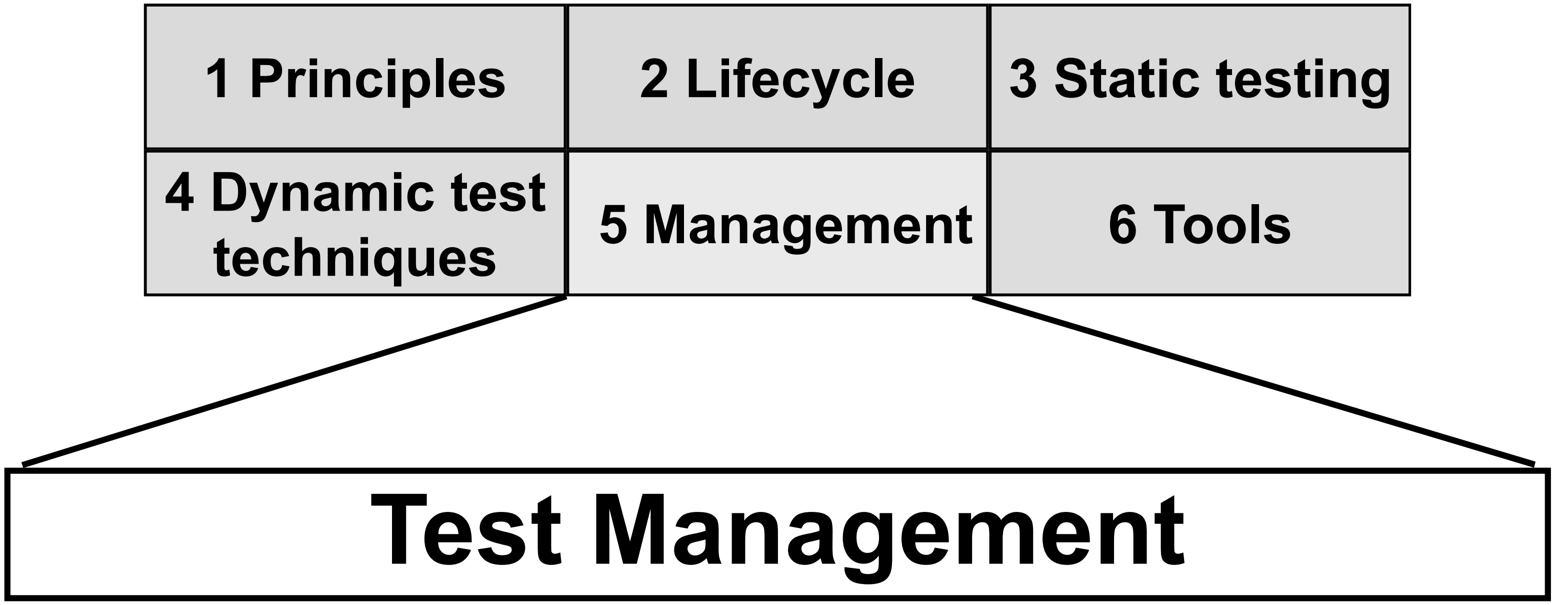
Chapter 5

Software Testing



1 Principles	2 Lifecycle	3 Static testing
4 Dynamic test techniques	5 Management	6 Tools

Test Management

Contents

Organisation

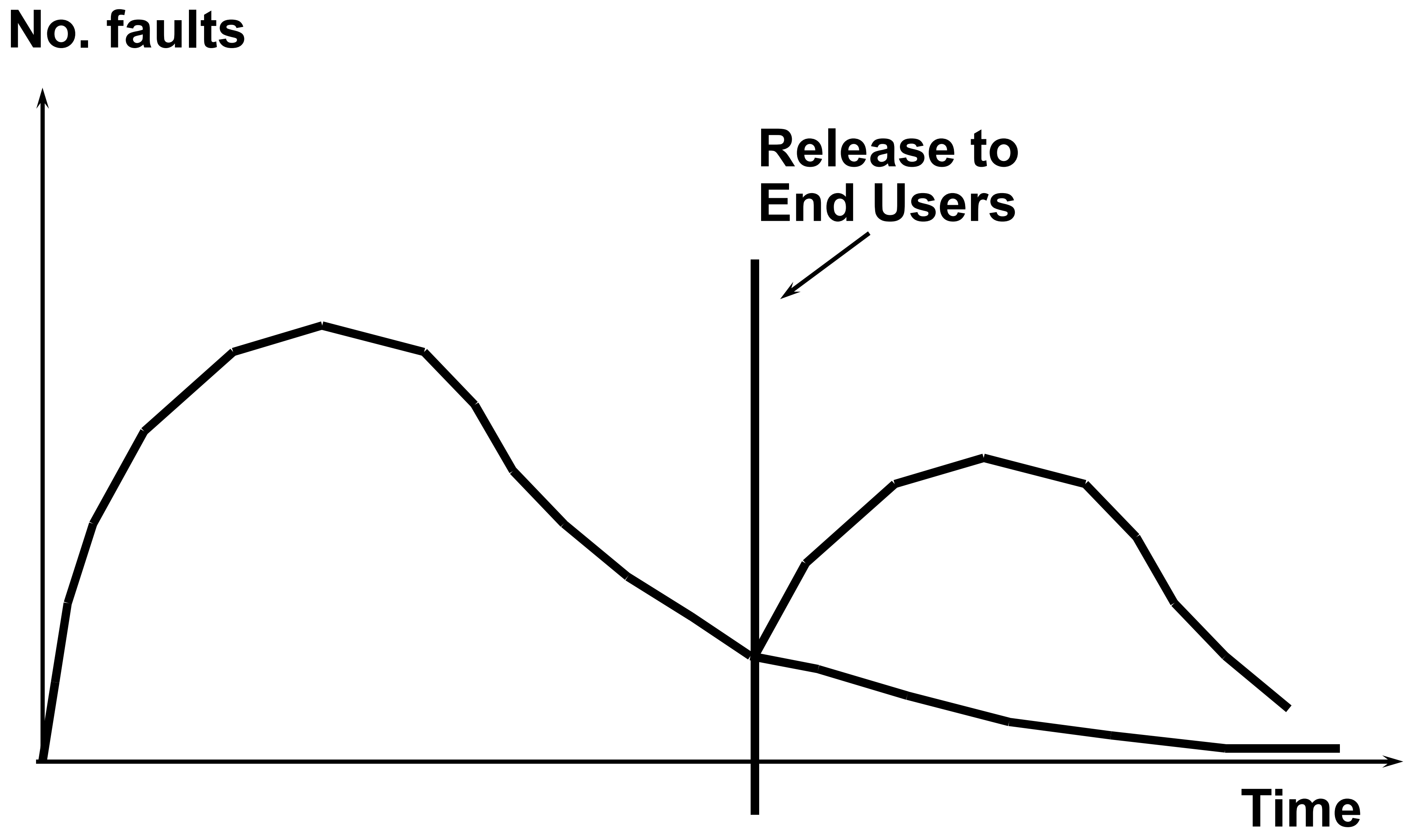
Configuration Management

Test estimation, monitoring and control

Risk and Testing

Incident management

Importance of Independence



Organisational structures for testing

- Developer responsibility (only)
- Development team responsibility (buddy system)
- Tester(s) on the development team
- Dedicated team of testers (not developers)
- Internal test consultants (advice, review, support, not perform the testing)
- Outside organisation (3rd party testers)

Testing by developers

- **Pro's:**
  - know the code best
  - will find problems that the testers will miss
  - they can find and fix faults cheaply
- **Con's**
  - difficult to destroy own work
  - tendency to 'see' expected results, not actual results
  - subjective assessment

Testing by development team

- **Pro's:**
  - some independence
  - technical depth
  - on friendly terms with "buddy" - less threatening
- **Con's**
  - pressure of own development work
  - technical view, not business view
  - lack of testing skill

Tester on development team

- **Pro's:**
  - independent view of the software
  - dedicated to testing, no development responsibility
  - part of the team, working to same goal: quality
- **Con's**
  - lack of respect
  - lonely, thankless task
  - corruptible (peer pressure)
  - a single view / opinion

Independent test team

- **Pro's:**
  - dedicated team just to do testing
  - specialist testing expertise
  - testing is more objective & more consistent
- **Con's**
  - "over the wall" syndrome
  - may be antagonistic / confrontational
  - over-reliance on testers, insufficient testing by developers

Internal test consultants

- **Pro's:**
  - **highly specialist testing expertise, providing support and help to improve testing done by all**
  - **better planning, estimation & control from a broad view of testing in the organisation**
- **Con's**
  - **someone still has to do the testing**
  - **level of expertise enough?**
  - **needs good "people" skills - communication**
  - **influence, not authority**

Outside organisation (3rd party)

- **Pro's:**
  - **highly specialist testing expertise (if out-sourced to a good organisation)**
  - **independent of internal politics**
- **Con's**
  - **lack of company and product knowledge**
  - **expertise gained goes outside the company**
  - **expensive?**

Usual choices

- **Component testing:**
  - **done by programmers (or buddy)**
- **Integration testing in the small:**
  - **poorly defined activity**
- **System testing:**
  - **often done by independent test team**
- **Acceptance testing:**
  - **done by users (with technical help)**
  - **demonstration for confidence**

So what we have seen thus far..

- independence is important
  - not a replacement for familiarity
- different levels of independence
  - pro's and con's at all levels
- test techniques offer another dimension to independence (independence of thought)
- test strategy should use a good mix
  - "declaration of independence”
- balance of skills needed

Skills needed in testing

- Technique specialists
- Automators
- Database experts
- Business skills & understanding
- Usability expert
- Test environment expert
- Test managers

1	2	3
4	5	6

Test Management

Contents

Organisation

Configuration Management

Test estimation, monitoring and control

Risk and Testing

Incident management

Problems resulting from poor configuration management

- can't <u>reproduce</u> a fault reported by a customer
- can't roll back to previous subsystem
- one change overwrites another
- emergency fault fix needs testing but tests have been updated to new software version
- which code changes belong to which version?
- faults which were fixed re-appear
- tests worked perfectly - on old version
- "Shouldn't that feature be in this version?"

A definition of Configuration Management

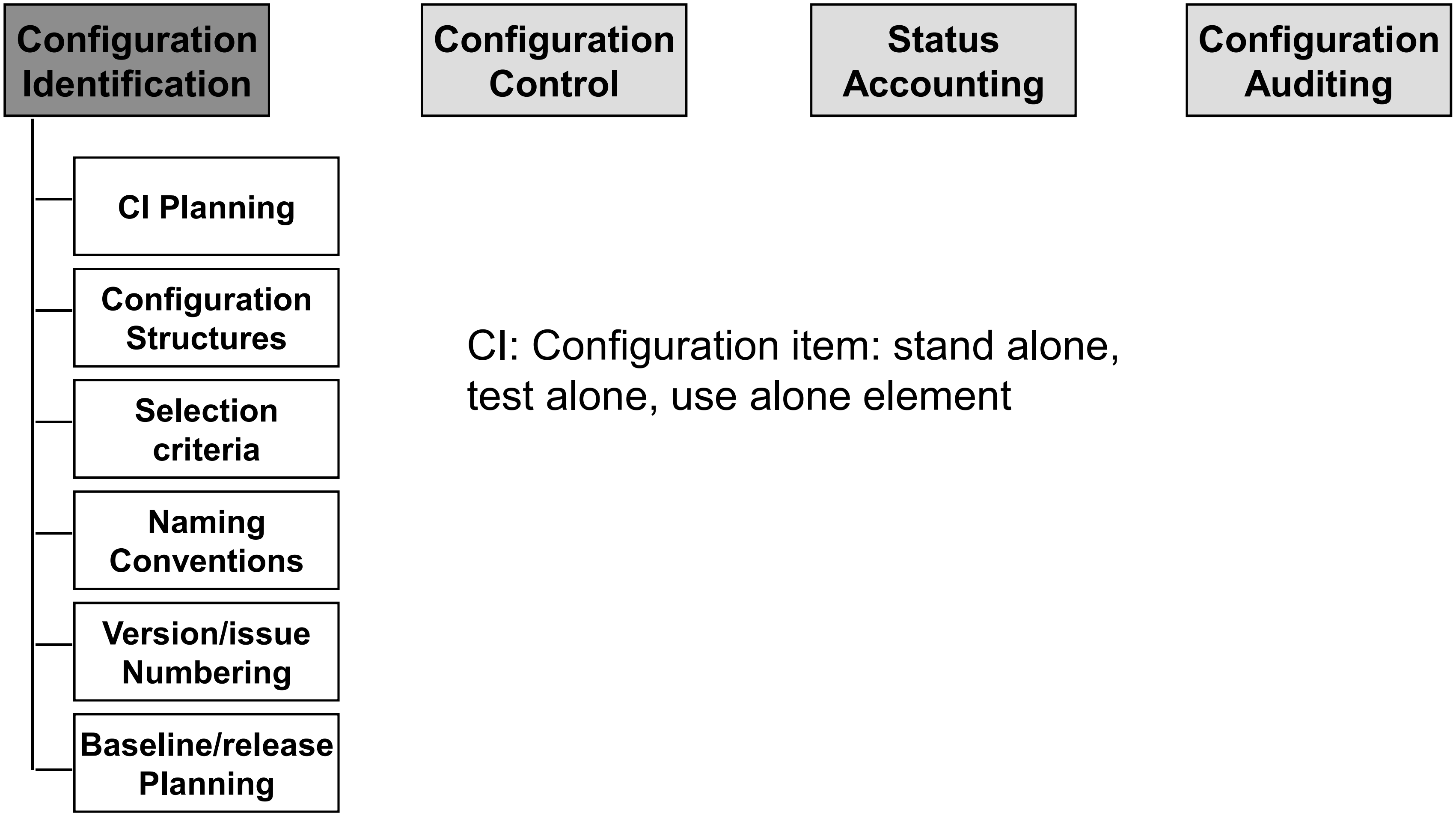
- “The process of identifying and defining the configuration items in a system,
- controlling the release and change of these items throughout the system life cycle,
- recording and reporting the status of configuration items and change requests,
- and verifying the completeness and correctness of configuration items.”
  - ANSI/IEEE Std 729-1983, Software Engineering Terminology

Configuration Management

- **An engineering management procedure that includes**
  - **configuration identification**
  - **configuration control**
  - **configuration status accounting**
  - **configuration audit**

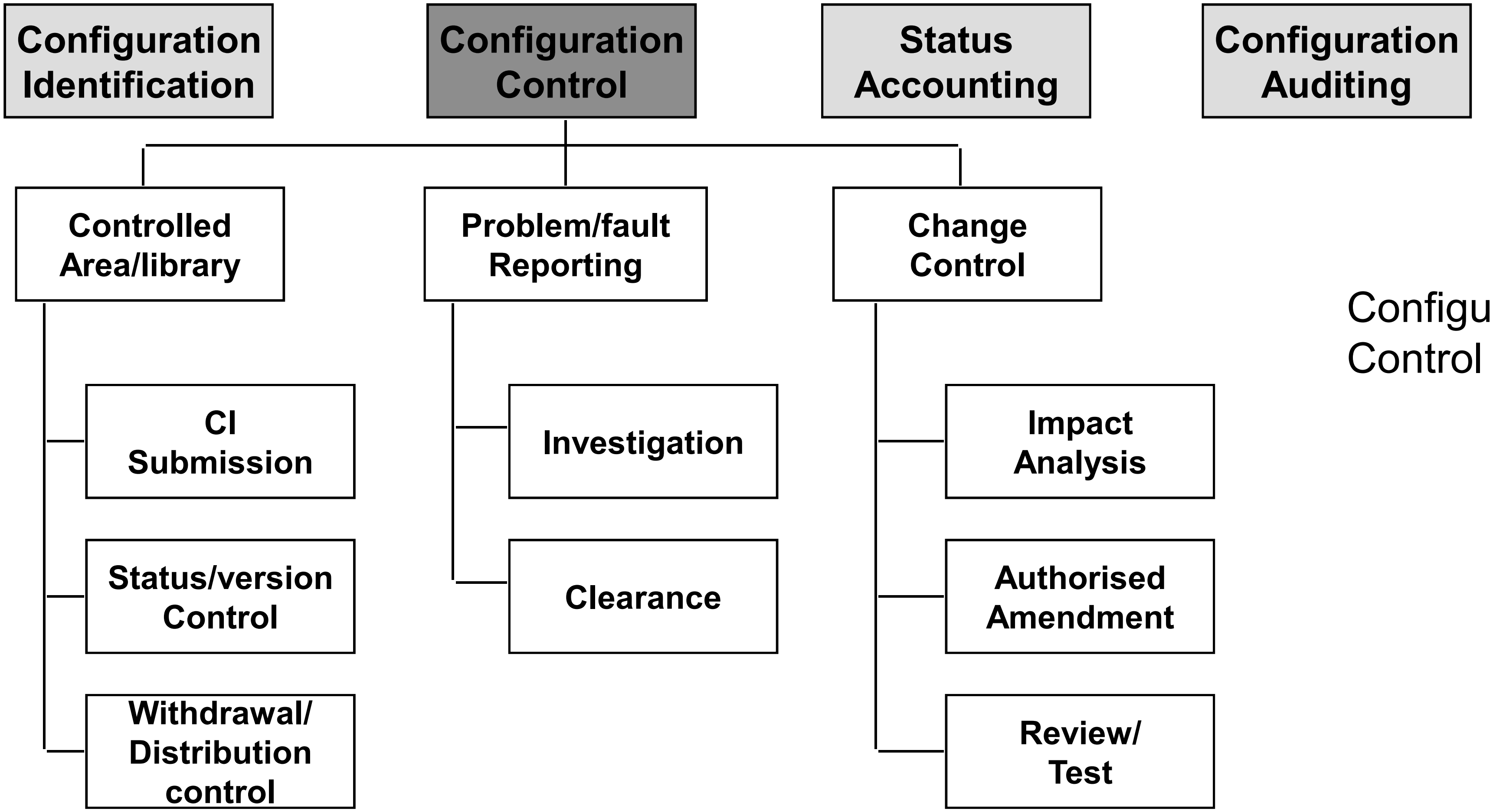
    - Encyclopedia of Software Engineering, 1994

Configuration identification



CI: Configuration item: stand alone, test alone, use alone element

Configuration control


Configuration Identification

Configuration Control

Status Accounting

Configuration Auditing

Controlled Area/library

Problem/fault Reporting

Change Control

CI Submission

Status/version Control

Withdrawal/ Distribution control

Investigation

Clearance

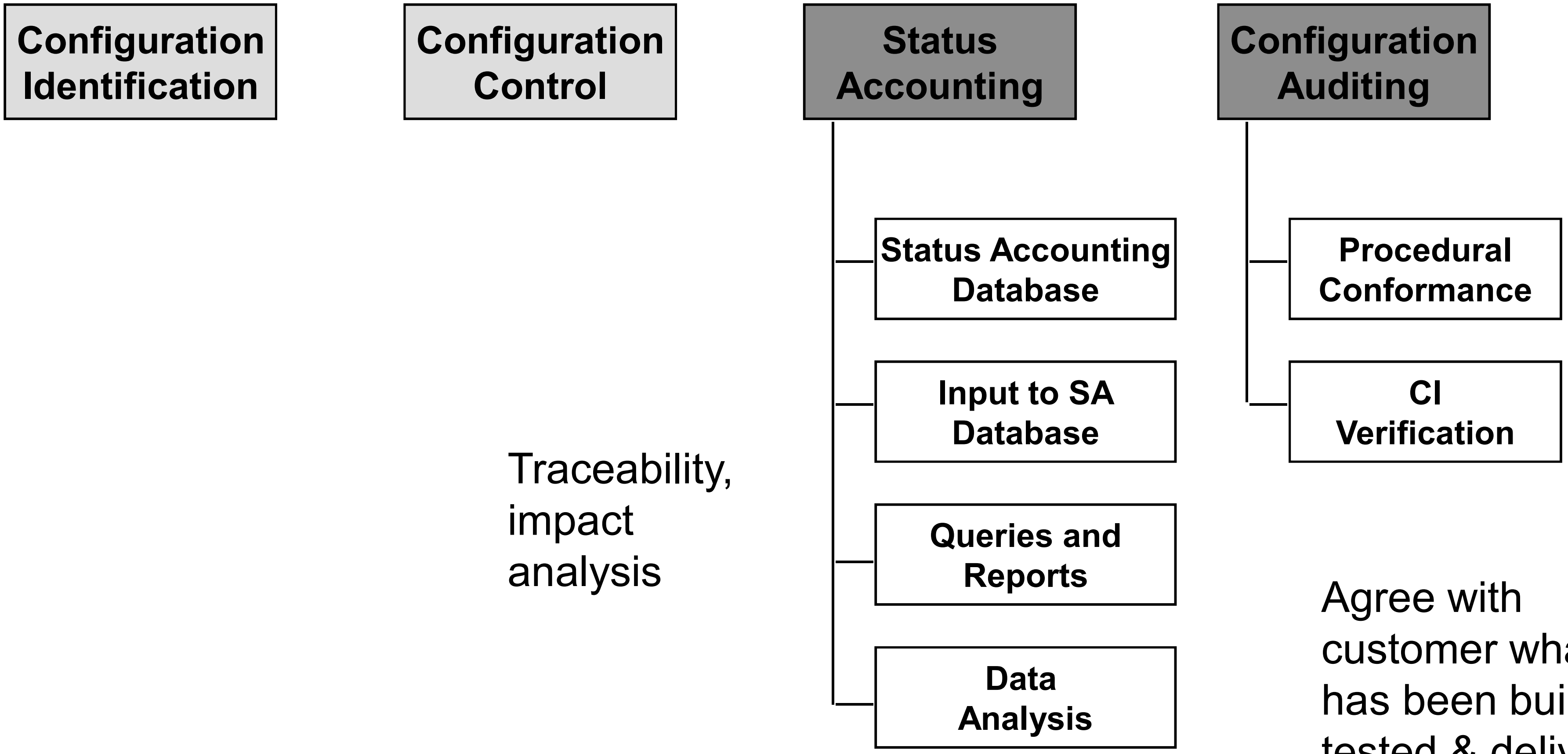
Impact Analysis

Authorised Amendment

Review/ Test

Configuration Control Board

Status accounting & Configuration Auditing



Traceability, impact analysis

Agree with customer what has been built, tested & delivered

Products for CM in testing

- test plans
- test designs
- test cases:
  - test input
  - test data
  - test scripts
  - expected results
- actual results
- test tools

CM is critical for controlled testing

What would not be under configuration management?

1	2	3
4	5	6

Test Management

Contents

Organisation

Configuration Management

Test estimation, monitoring and control

Risk and Testing

Incident management

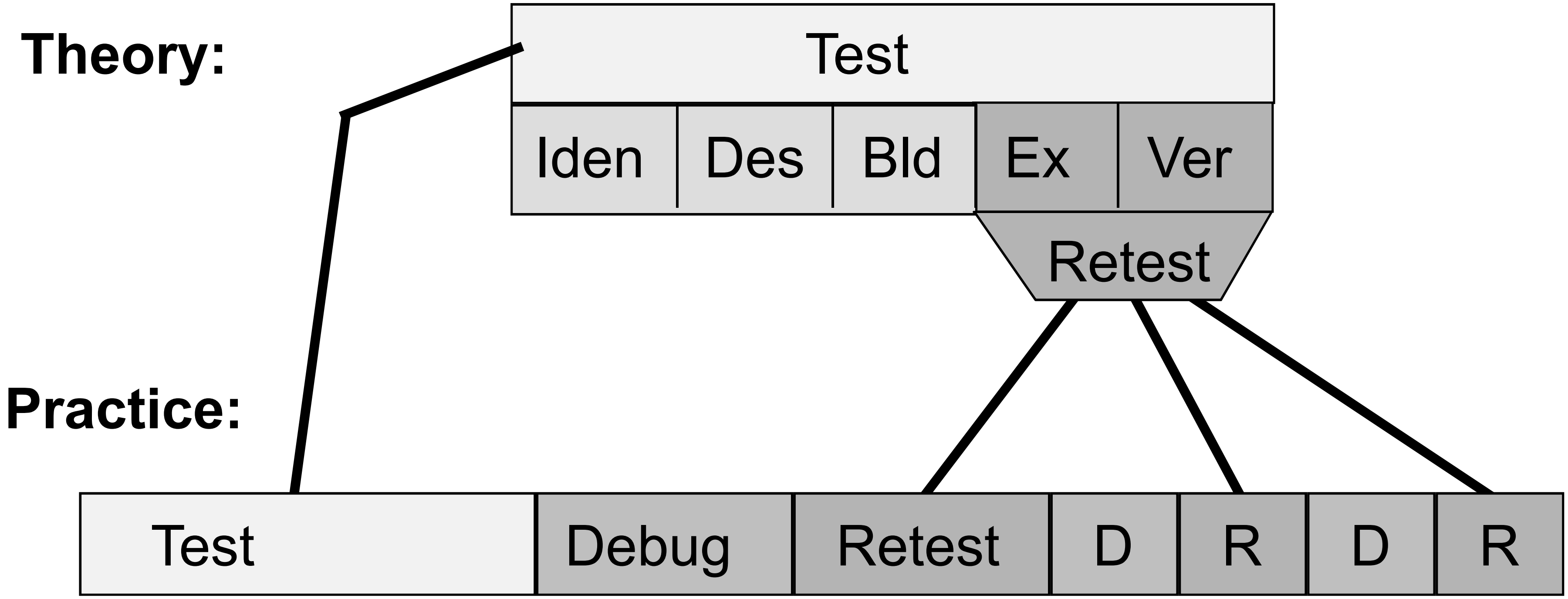
Estimating testing is no different

- **Estimating any job involves the following**
  - **identify tasks**
  - **how long for each task**
  - **who should perform the task**
  - **when should the task start and finish**
  - **what resources, what skills**
  - **predictable dependencies**
    - task precedence (build test before running it)
    - technical precedence (add & display before edit)

Estimating testing is different

- **Additional destabilising dependencies**
  - testing is not an independent activity
  - delivery schedules for testable items missed
  - test environments are critical
- **Test Iterations (Cycles)**
  - testing should find faults
  - faults need to be fixed
  - after fixed, need to retest
  - how many times does this happen?

Test cycles / iterations



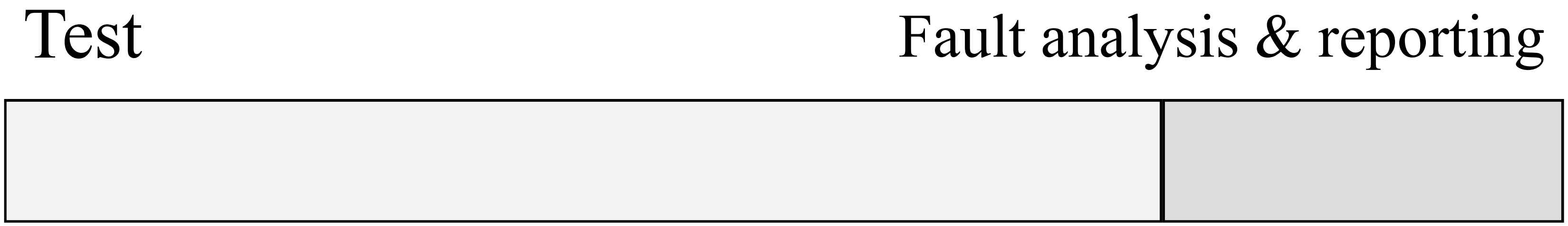
3-4 iterations is typical

Estimating iterations

- past history
- number of faults expected
  - can predict from previous test effectiveness and previous faults found (in test, review, Inspection)
  - % faults found in each iteration (nested faults)
  - % fixed [in]correctly
- time to report faults
- time waiting for fixes
- how much in each iteration?

Time to report faults

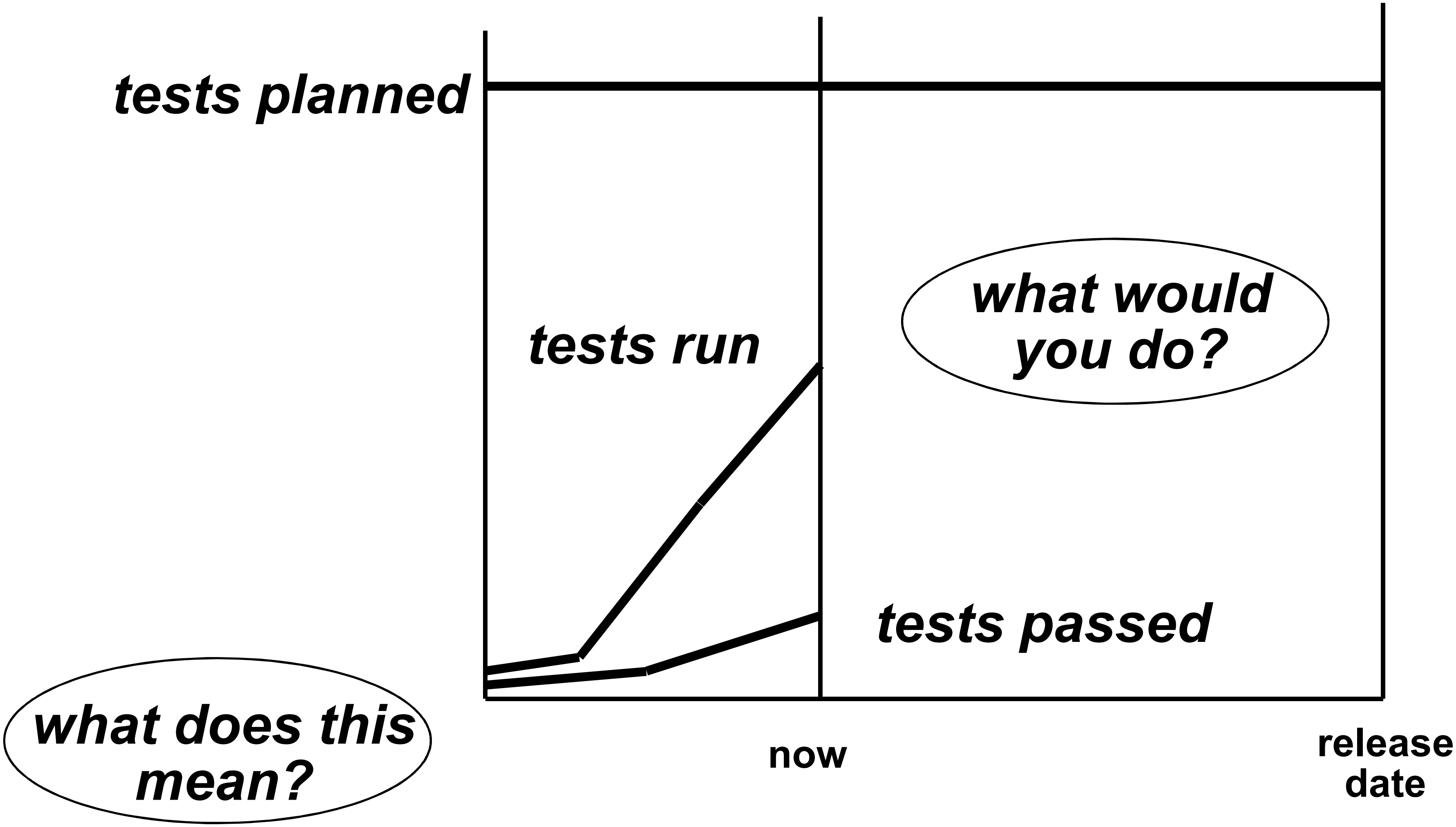
- **If it takes 10 mins to write a fault report, how many can be written in one day?**
- **The more fault reports you write, the less testing you will be able to do.**


Test

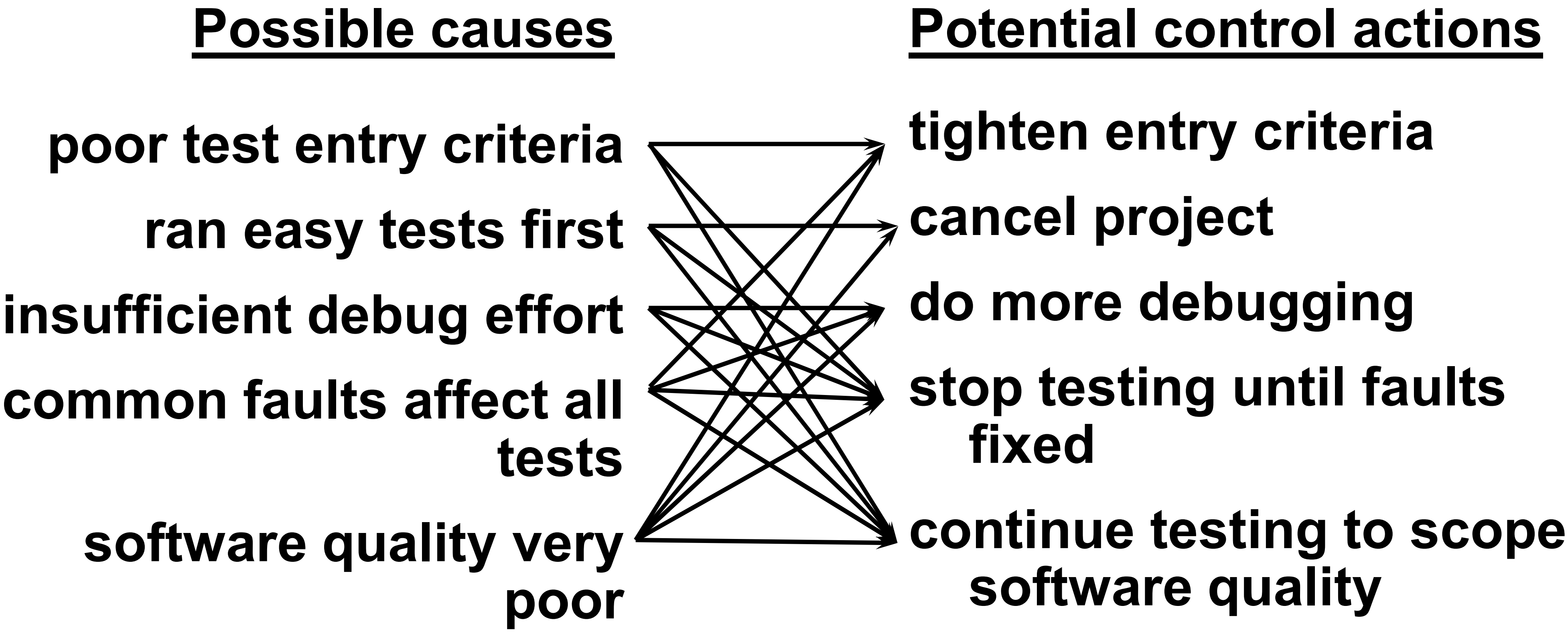
Fault analysis & reporting

Mike Royce: suspension criteria: when testers spend > 25% time on faults

Measuring test execution progress 1



Diverging S-curve



Possible causes

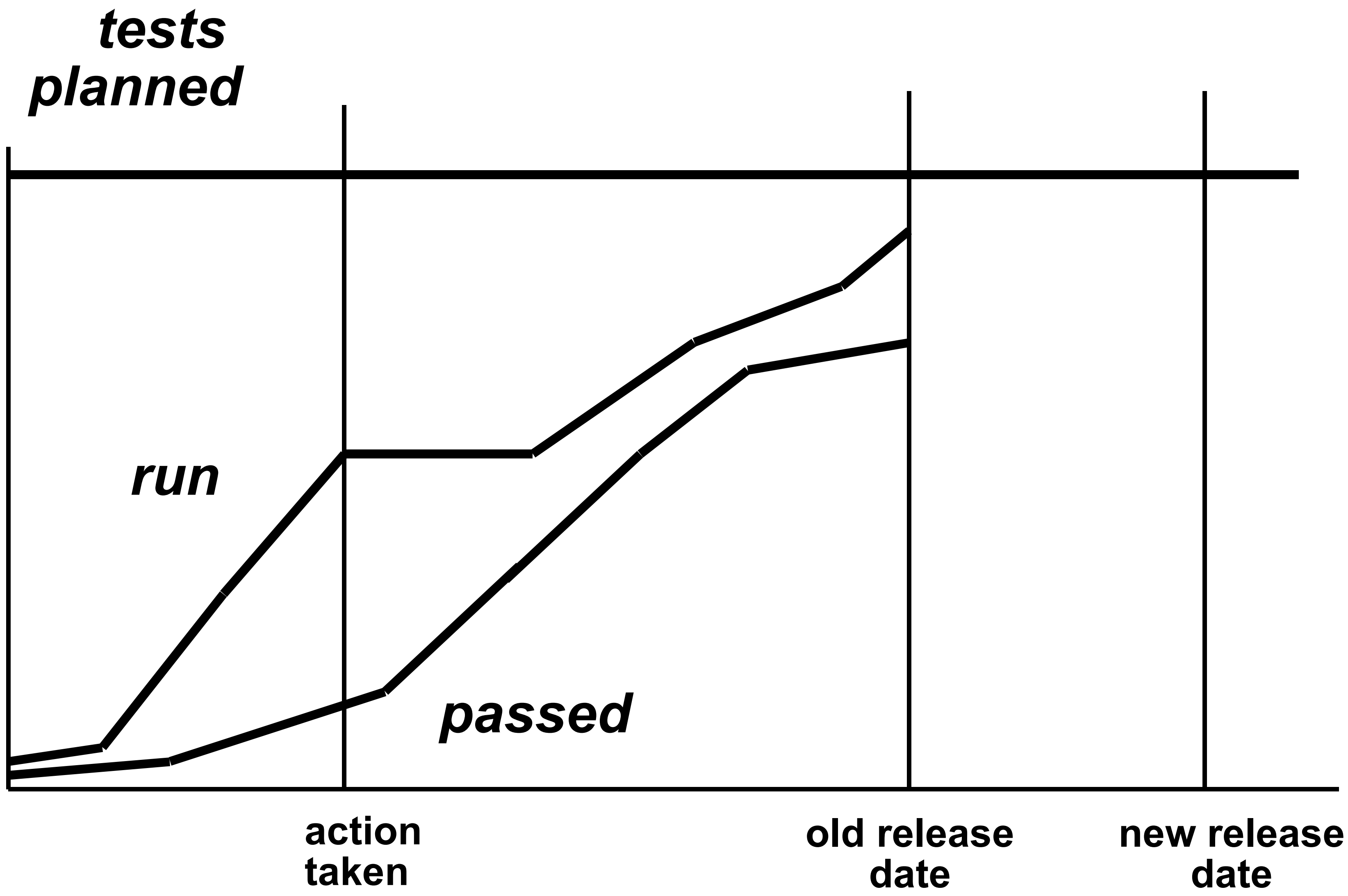
- poor test entry criteria
- ran easy tests first
- insufficient debug effort
- common faults affect all tests
- software quality very poor

Potential control actions

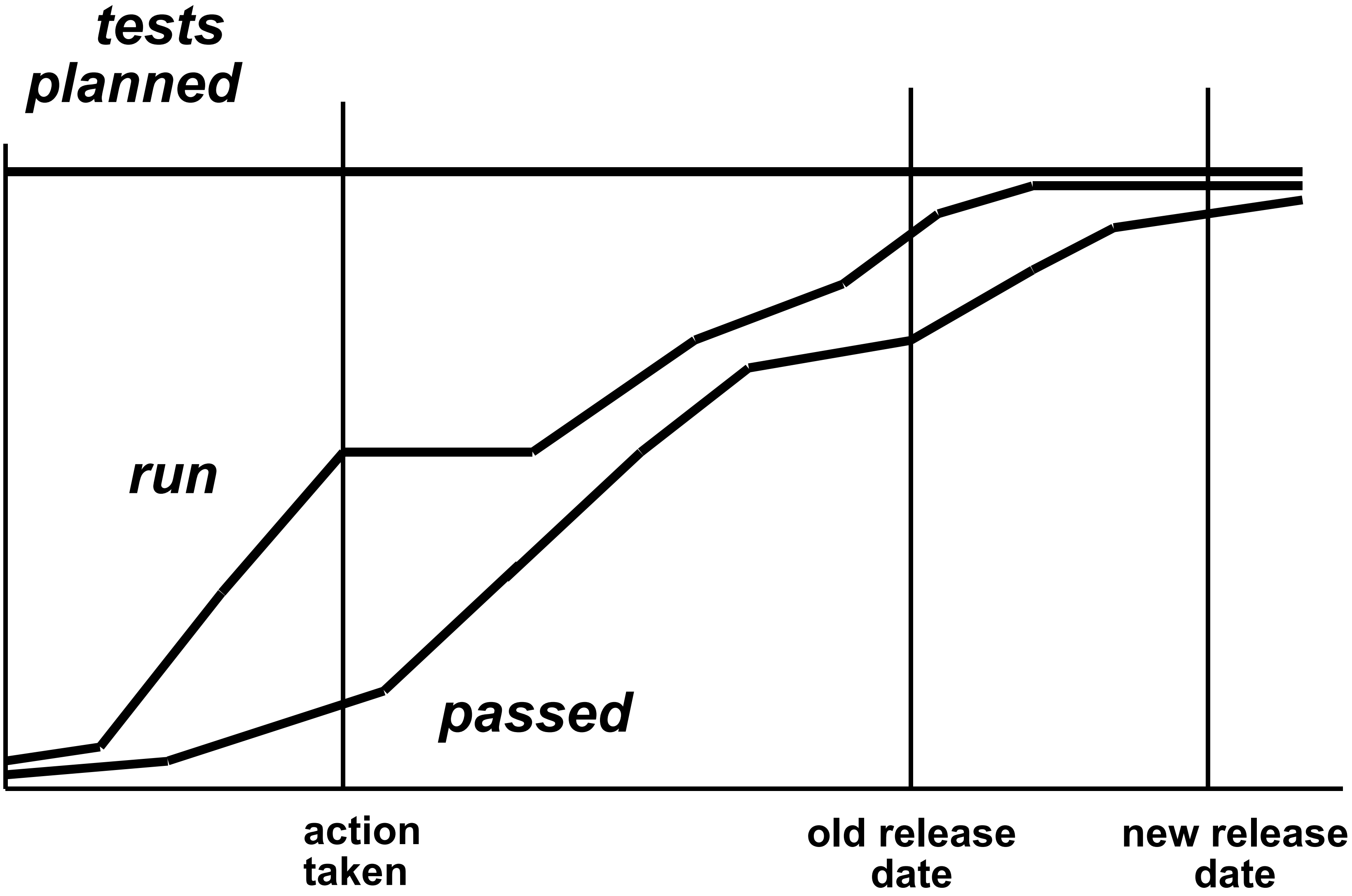
- tighten entry criteria
- cancel project
- do more debugging
- stop testing until faults fixed
- continue testing to scope software quality

Note: solutions / actions will impact other things as well, e.g. schedules

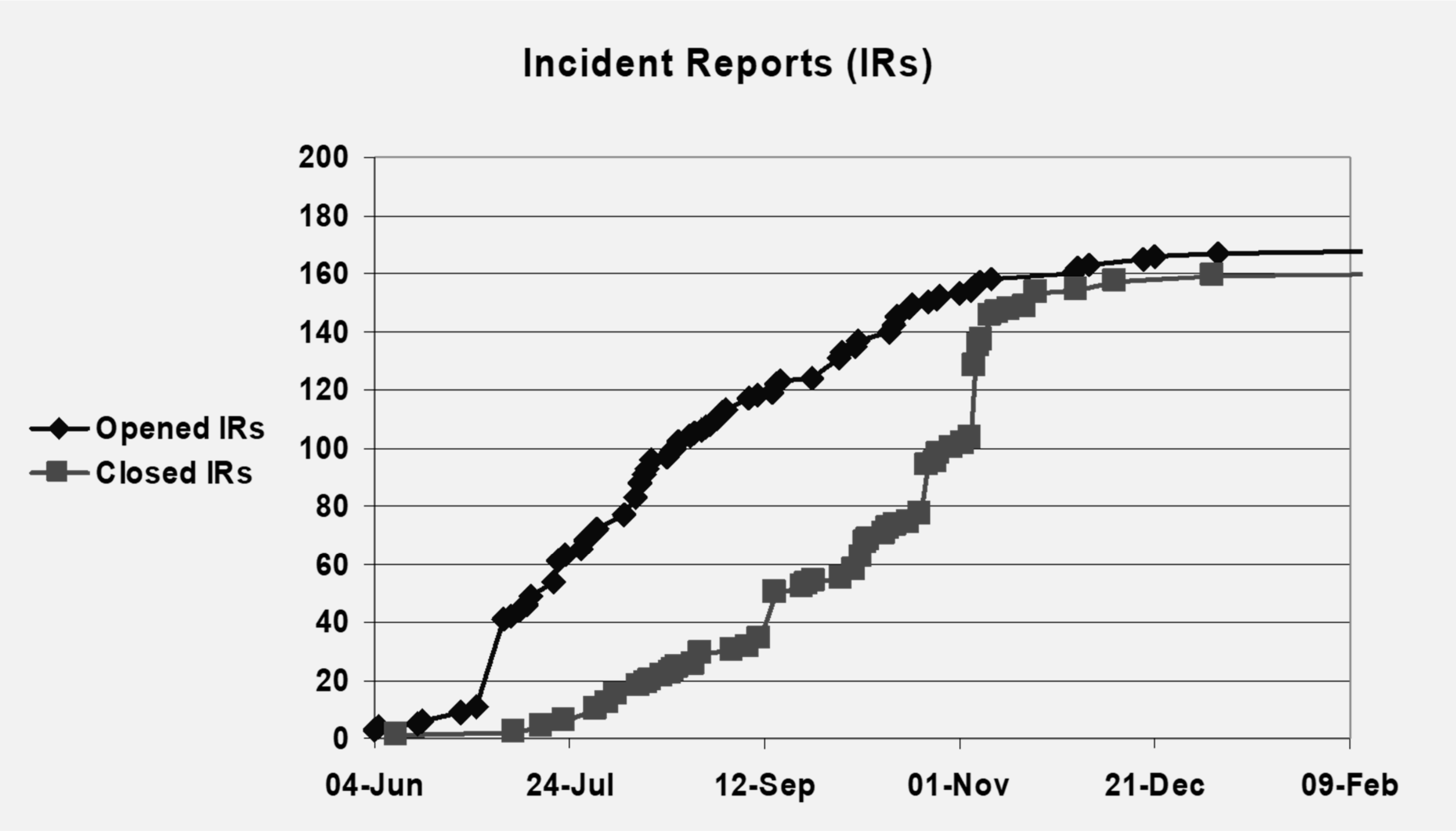
Measuring test execution progress 2



Measuring test execution progress 3



Case history



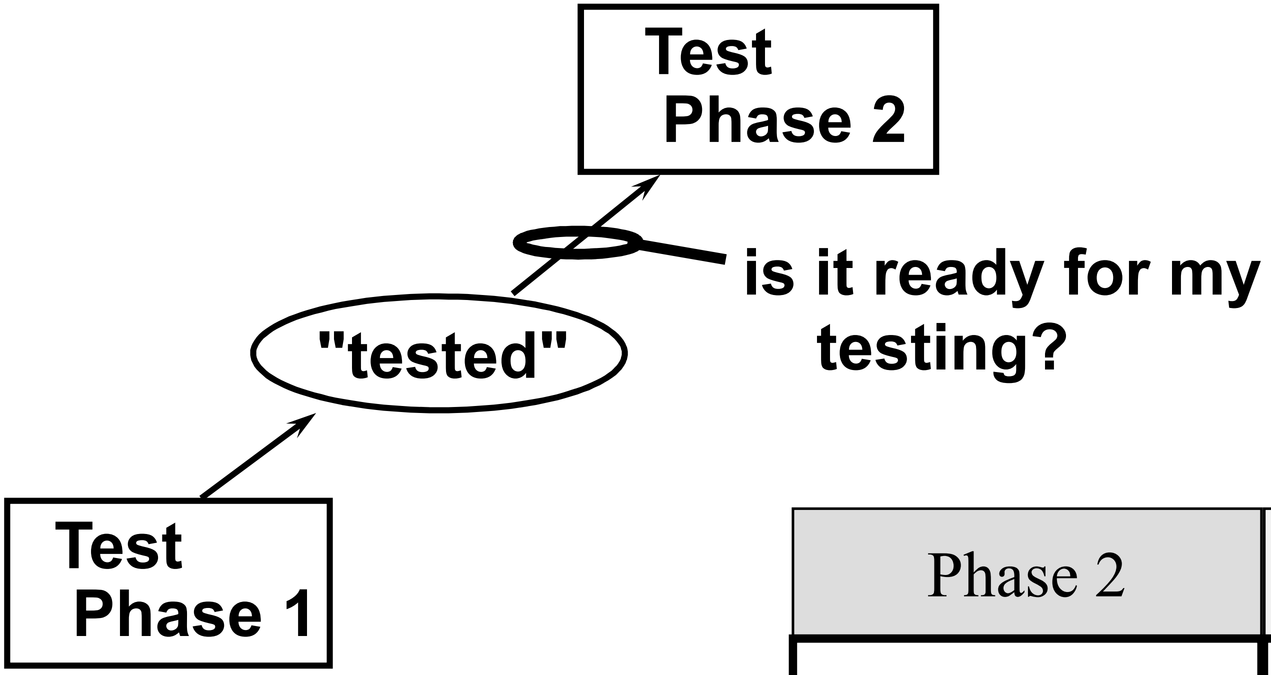
Source: Tim Trew, Philips, June 1999

Control

- **Management actions and decisions**
  - **affect the process, tasks and people**
  - **to meet original or modified plan**
  - **to achieve objectives**
- **Examples**
  - **tighten entry / exit criteria**
  - **reallocation of resources**

Feedback is essential to see the effect of actions and decisions

Entry and exit criteria



Test Phase 2

is it ready for my testing?

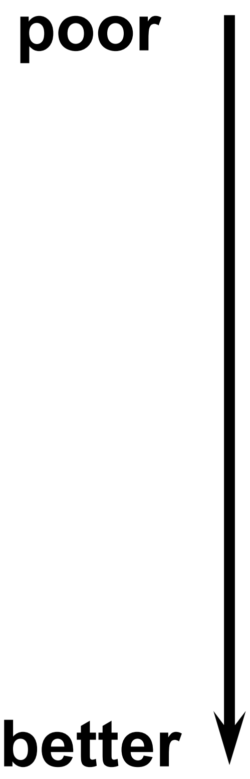
"tested"

Test Phase 1

Phase 2	Phase 1
Entry criteria	Exit criteria
Acceptance criteria	Completion criteria

Entry/exit criteria examples

- clean compiled
- programmer claims it is working OK
- lots of tests have been run
- tests have been reviewed / Inspected
- no faults found in current tests
- all faults found fixed and retested
- specified coverage achieved
- all tests run after last fault fix, no new faults



poor

better

What actions can you take?

- What can you affect?
  - resource allocation
  - number of test iterations
  - tests included in an iteration
  - entry / exit criteria applied
  - release date

- What can you not affect:
  - number of faults already there
- What can you affect indirectly?
  - rework effort
  - which faults to be fixed [first]
  - quality of fixes (entry criteria to retest)

1	2	3
4	5	6

Test Management

ISTQB / ISEB Foundation Exam Practice

Contents

Organisation

Configuration Management

Test estimation, monitoring and control

Risk and Testing

Incident management

Risk and Testing

- **Risks and levels of risk**
  - It's the possibility of a negative or undesirable outcome.
- **Product risks**
  - The possibility that the system or software might fail to satisfy some reasonable customer, user, or stakeholder expectation
  - Risk-based testing is the idea that we can organize our testing efforts in a way that reduces the residual level of product risk when the system ships.

Risk and Testing

- **Project risks**
  - **For any risk, product or project, you have four typical options:**
    - Mitigate, Contingency, Transfer, Ignore
- **Tying it all together for risk management**

1	2	3
4	5	6

Test Management

Contents

Organisation

Configuration Management

Test estimation, monitoring and control

Risk and Testing

Incident management

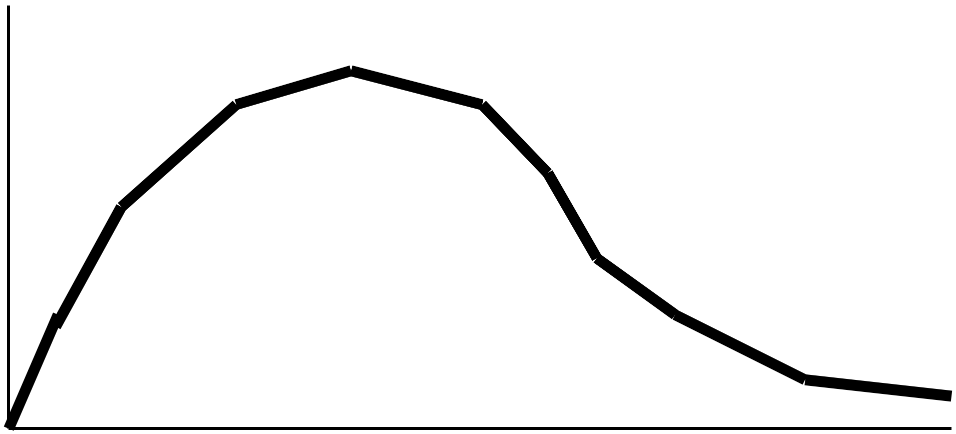
Incident management

- **Incident: any event that occurs during testing that requires subsequent investigation or correction.**
  - **actual results do not match expected results**
  - **possible causes:**
    - software fault
    - test was not performed correctly
    - expected results incorrect
  - **can be raised for documentation as well as code**

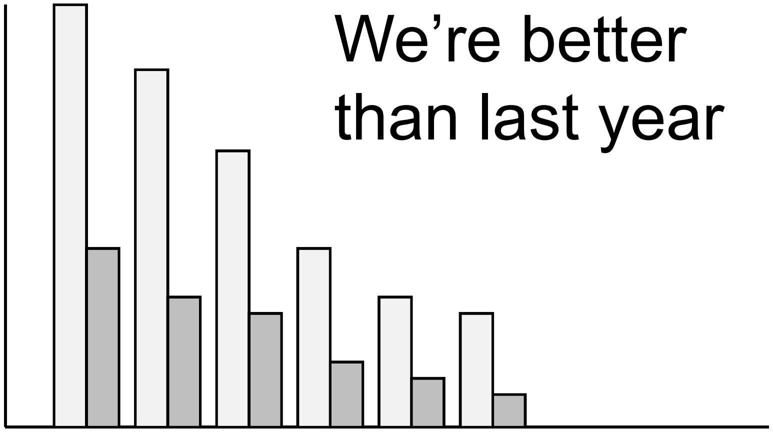
Incidents

- **May be used to monitor and improve testing**
- **Should be logged (after hand-over)**
- **Should be tracked through stages, e.g.:**
  - **initial recording**
  - **analysis (s/w fault, test fault, enhancement, etc.)**
  - **assignment to fix (if fault)**
  - **fixed not tested**
  - **fixed and tested OK**
  - **closed**

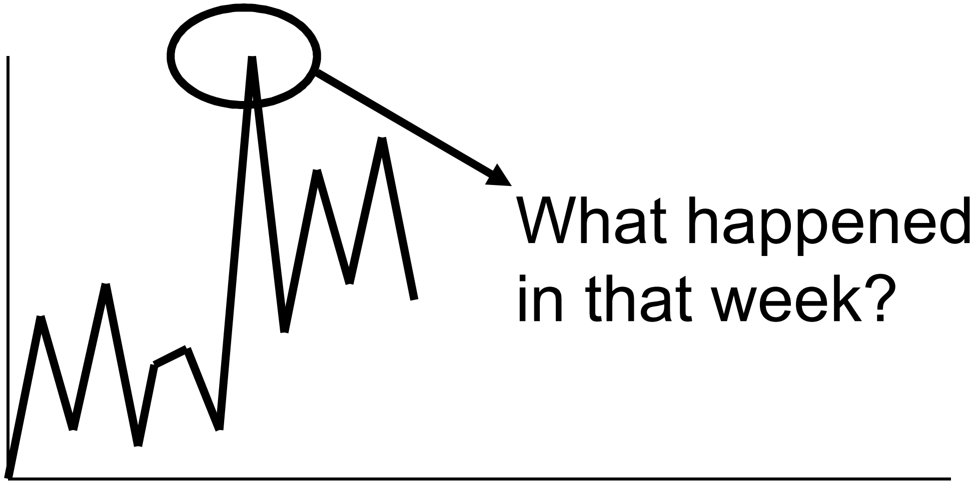
Use of incident metrics



Is this testing approach “wearing out”?



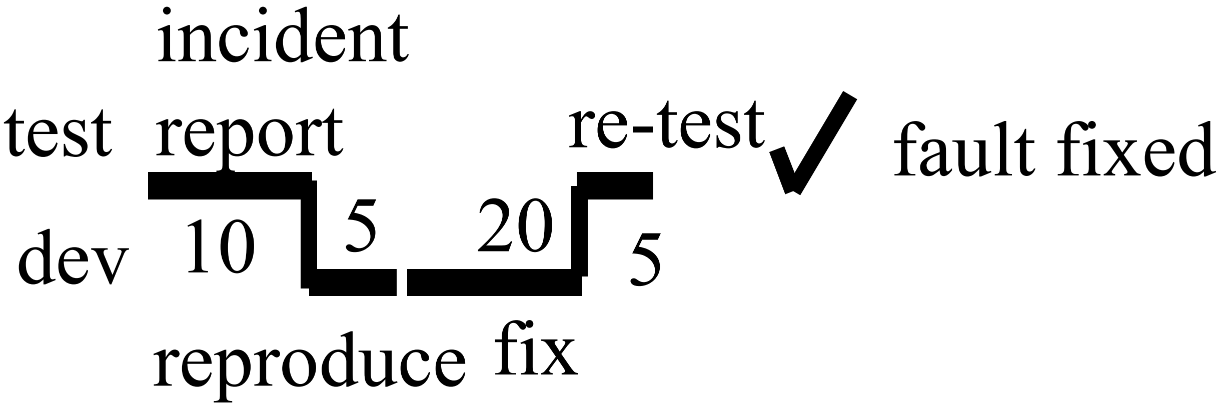
We’re better than last year



What happened in that week?

How many faults can we expect?

Report as quickly as possible?



incident report

test

dev

10

5

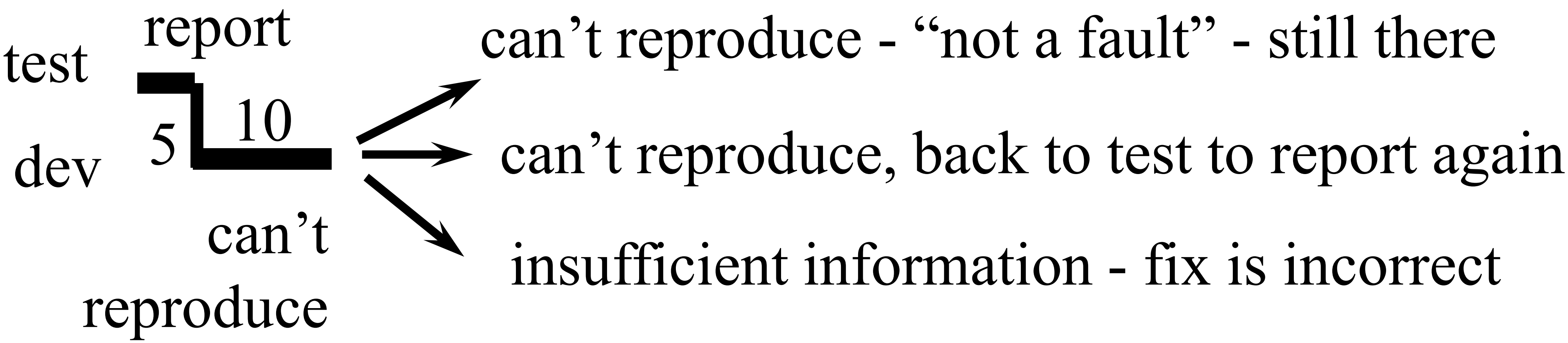
20

5

re-test ✓ fault fixed

reproduce

fix



test

report

dev

5

10

can't reproduce

can't reproduce - "not a fault" - still there

can't reproduce, back to test to report again

insufficient information - fix is incorrect

What information about incidents?

- Test ID
- Test environment
- Software under test ID
- Actual & expected results
- Severity, scope, priority
- Name of tester
- Any other relevant information (e.g. how to reproduce it)

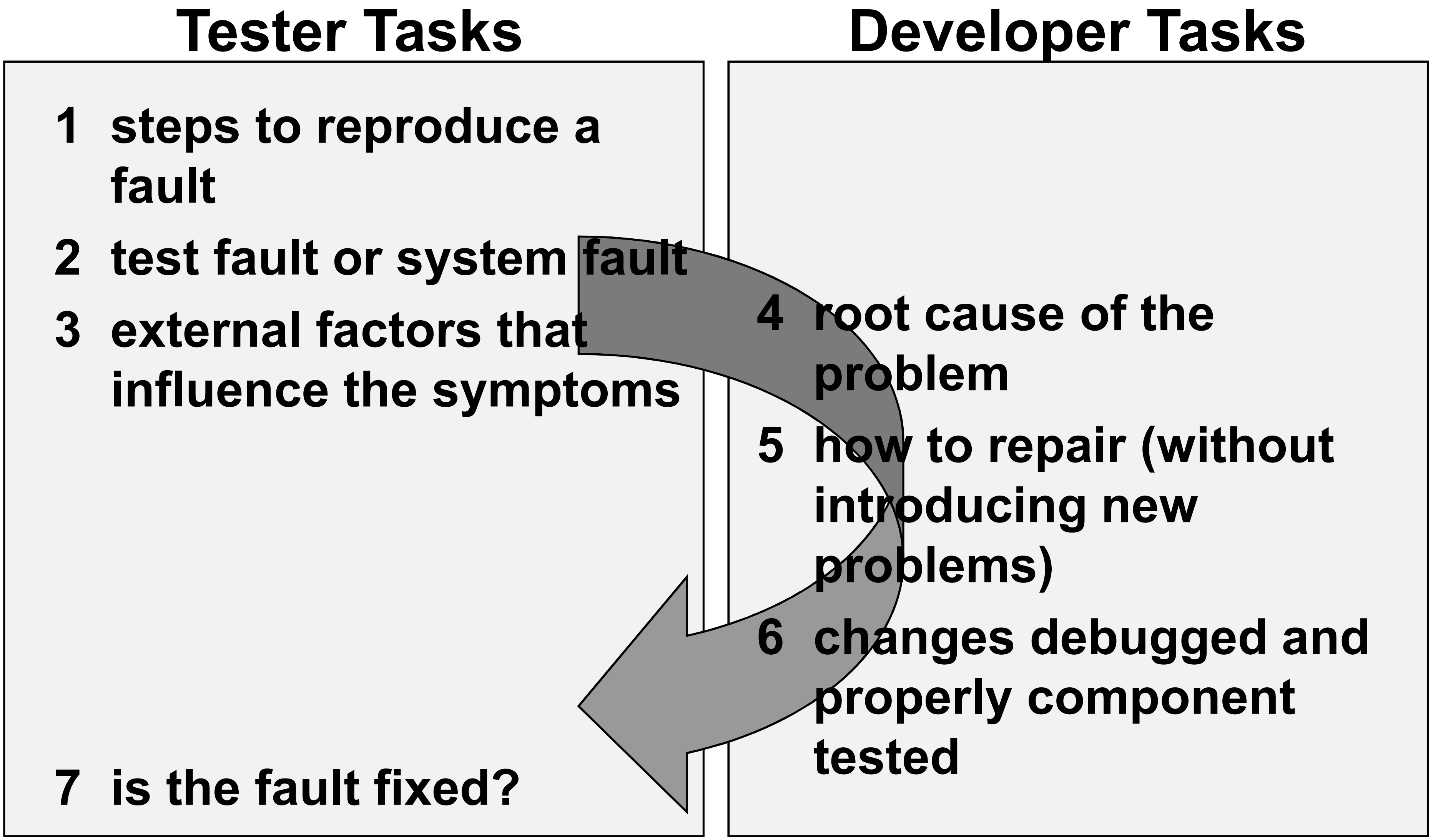
Severity versus priority

- **Severity**
  - **impact of a failure caused by this fault**
- **Priority**
  - **urgency to fix a fault**
- **Examples**
  - **minor cosmetic typo**
  - **crash if this feature is used**

company name,
board member:
priority, not severe

Experimental,
not needed yet:
severe, not priority

Incident Lifecycle



Tester Tasks

1 steps to reproduce a fault

2 test fault or system fault

3 external factors that influence the symptoms

7 is the fault fixed?

Developer Tasks

4 root cause of the problem

5 how to repair (without introducing new problems)

6 changes debugged and properly component tested

Source: Rex Black “Managing the Testing Process”, MS Press, 1999

Metrics Example GQM

- **Goal: EDD < 2 defects per KloC**
  - **Q1: What is the size of the software?**
    - M1.1: KloC per module
  - **Q2: How many defects in code?**
    - M2.1: Estimation of # defects
  - **Q3: How many defects found?**
    - M3.1: # defects in Review and Inspection
    - M3.2: # defects in subsequent tests
  - **Q4: What is the yield of the tests done?**
    - M4.1: # defects (M3) divided by estimation (M2)

Metrics Exercise

- **Goal: In ST, do an optimal check in minimum time based on the 3 customers for Reger**
  - **Priority of processes used by customers**
  - **Coverage of the processes**
  - **Incidents found**
  - **Severity of incidents**
  - **Time planned and spent**

1	2	3
4	5	6

Test Management

Summary: Key Points

Independence can be achieved by different organisational structures

Configuration Management is critical for testing

Tests must be estimated, monitored and controlled

Risk and Testing

Incidents need to be managed