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 SO 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

PROBLEMS RESULTING FROM POOR CONFIGURATION MANAGEMENT

can't reproduce 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

PRODUCTS FOR CM IN TESTING

test plans

test designs

test cases:

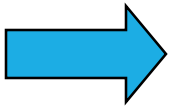
- test input
- test data
- test scripts
- expected results

actual results

test tools

CM is critical for controlled testing

What would not be under configuration management?



Live data!

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?

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

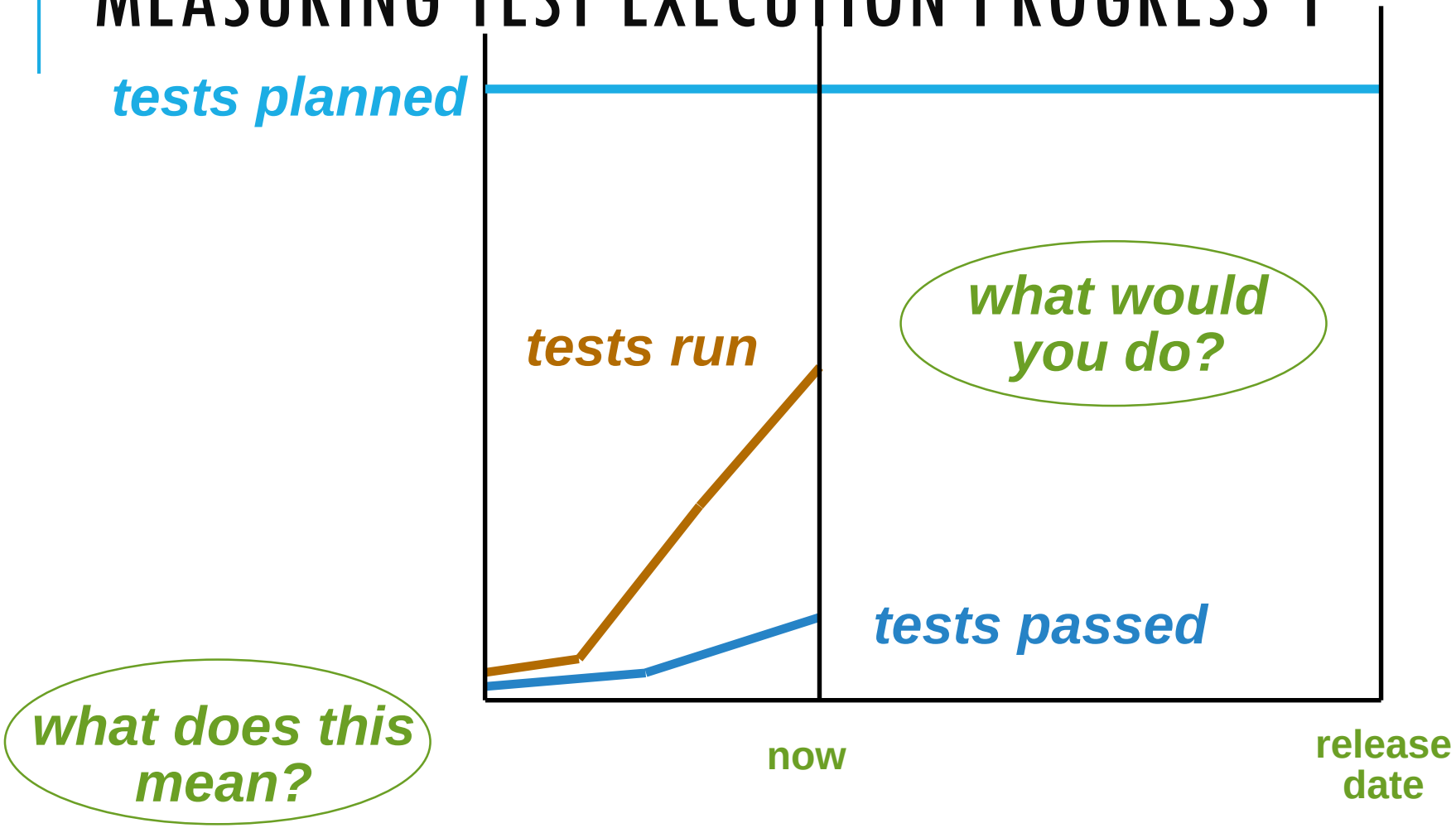
how much in each iteration?

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.

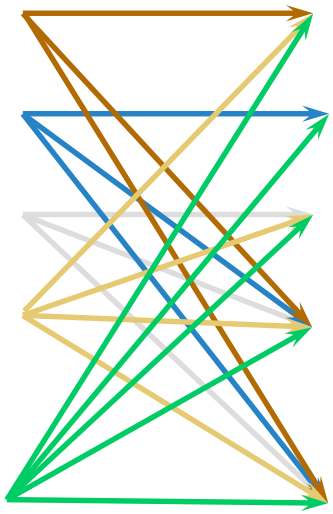
MEASURING TEST EXECUTION PROGRESS 1



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

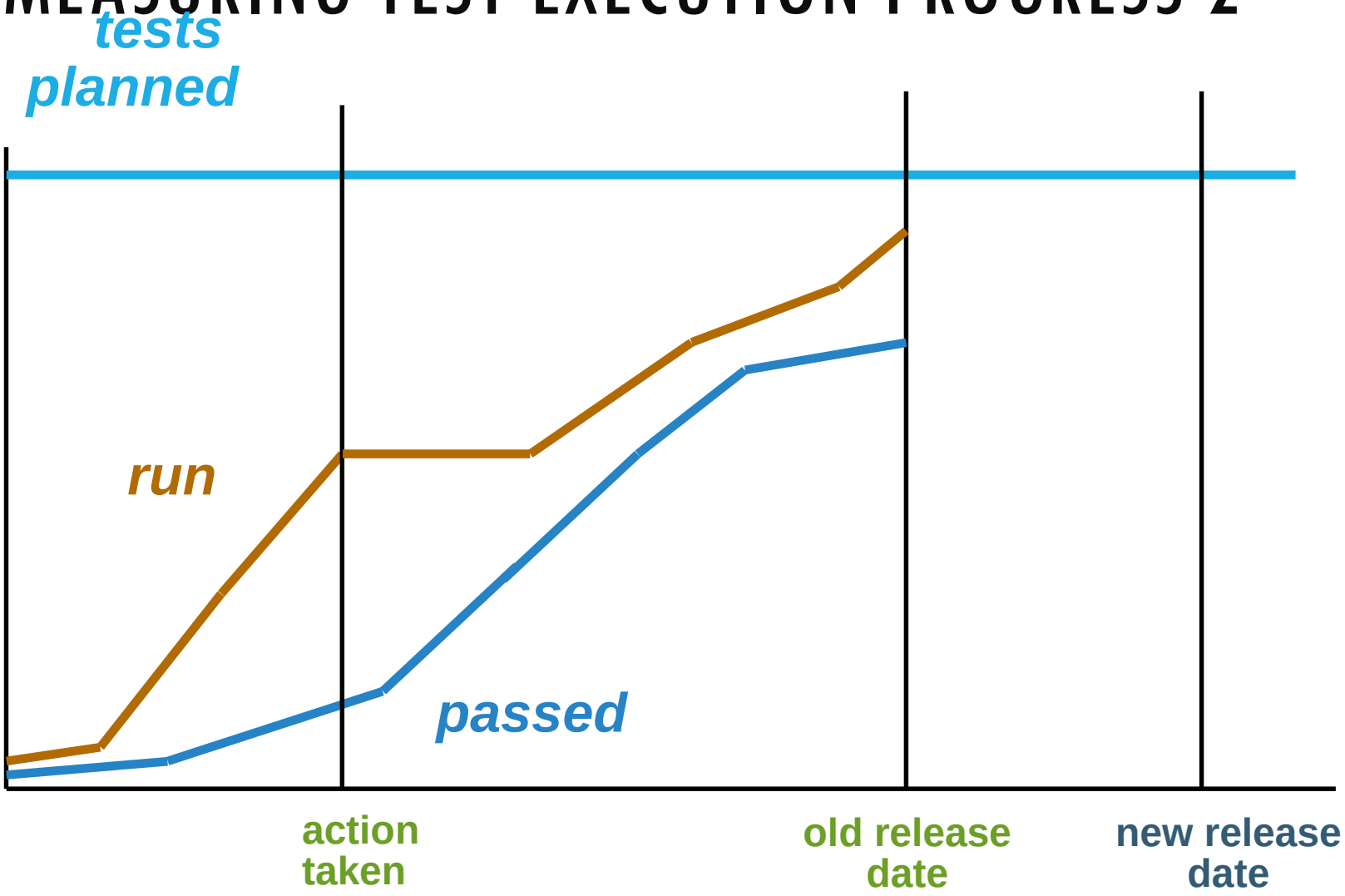


Potential control actions

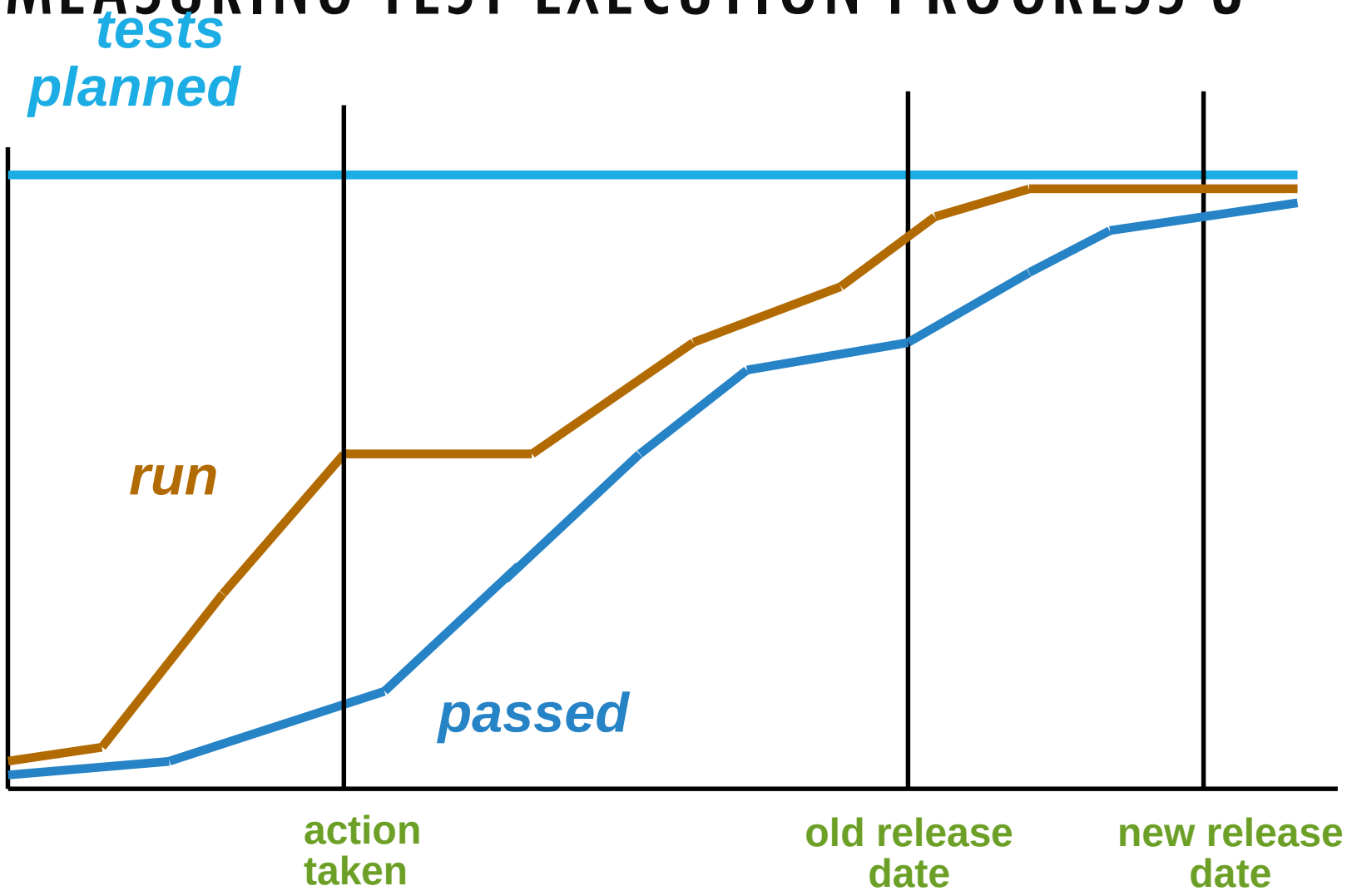
- tighten entry criteria
- cancel project
- do more debugging
- stop testing until faults fixed
- continue testing to scope software quality

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

MEASURING TEST EXECUTION PROGRESS 2



MEASURING TEST EXECUTION PROGRESS 3



CONTROL

Management actions and decisions

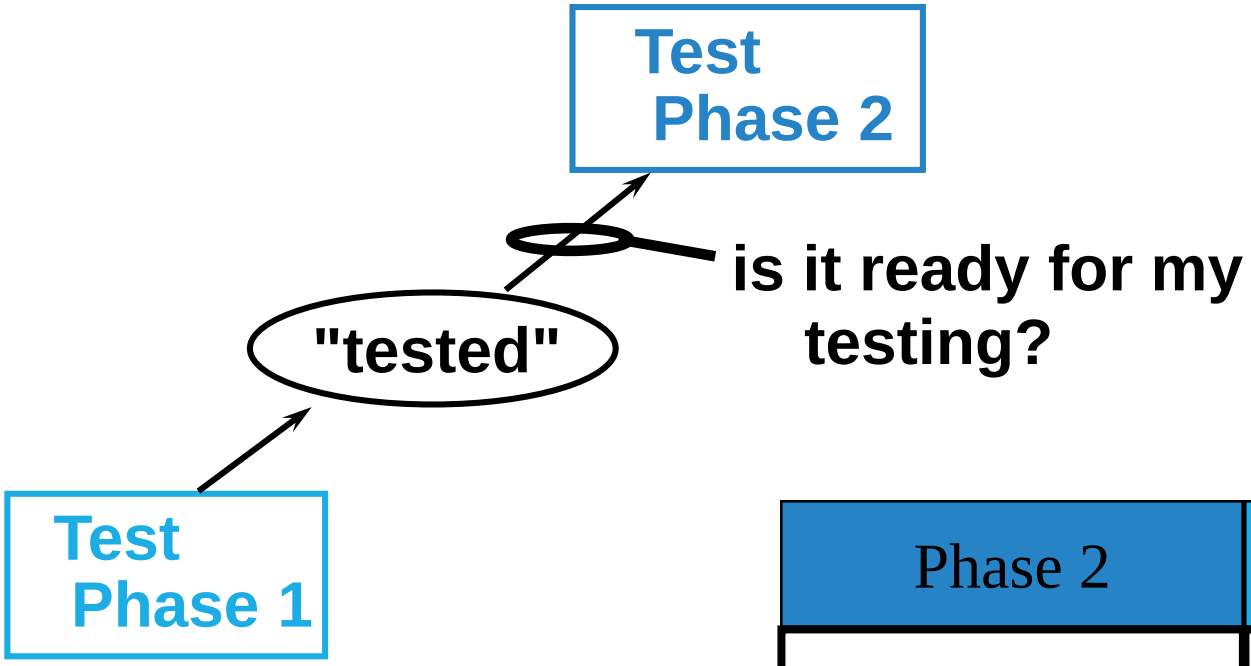
- affect the process, tasks and people
- to meet original or modified plan
- to achieve objectives

Examples

- tighten entry / exit criteria
- reallocation of resources

Feedback is essential to see the effect of actions and decisions

ENTRY AND EXIT CRITERIA



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

ENTRY/EXIT CRITERIA EXAMPLES

poor

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

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)

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

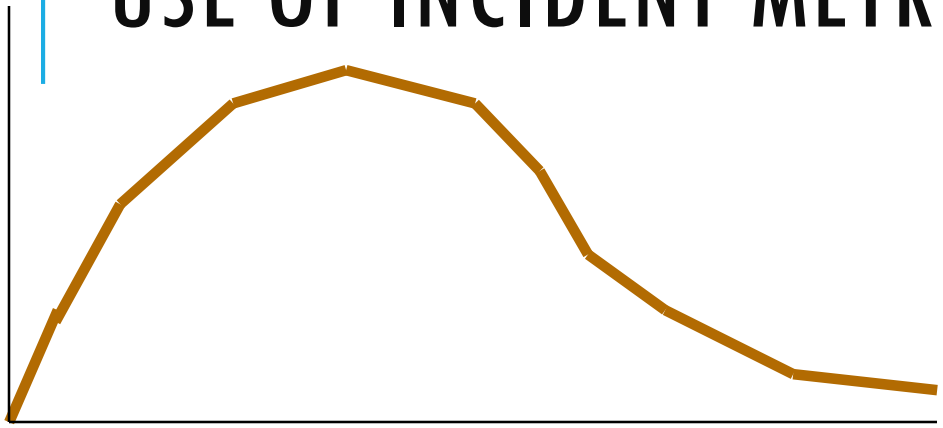
May be used to monitor and improve testing

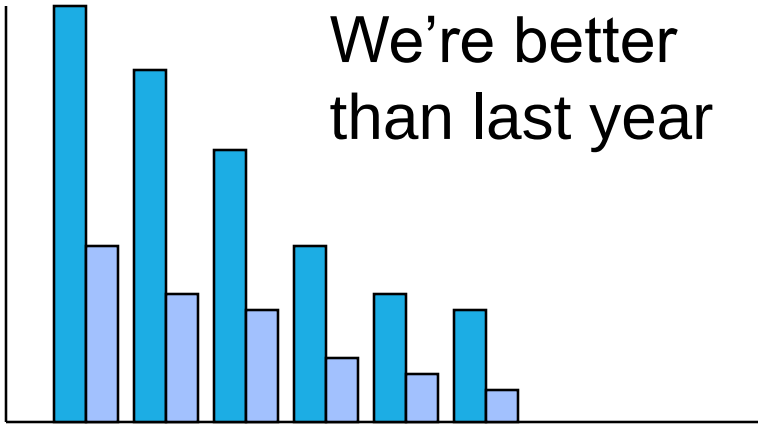
Should be logged (after hand-over)

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

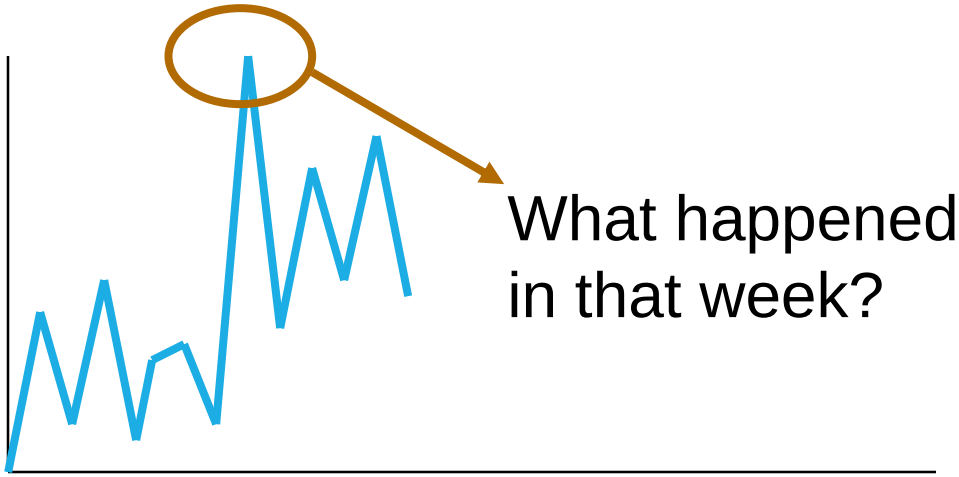
USE OF INCIDENT METRICS





We're better than last year

Is this testing approach "wearing out"?



What happened in that week?

How many faults can we expect?

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

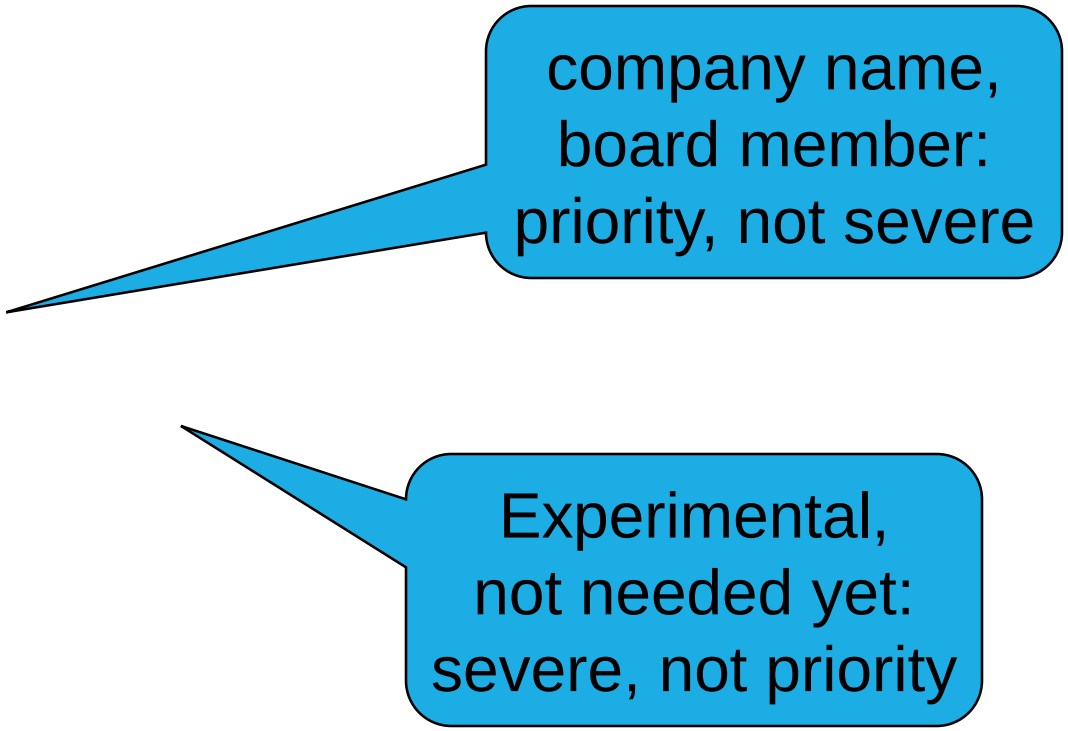
- impact of a failure caused by this fault

Priority

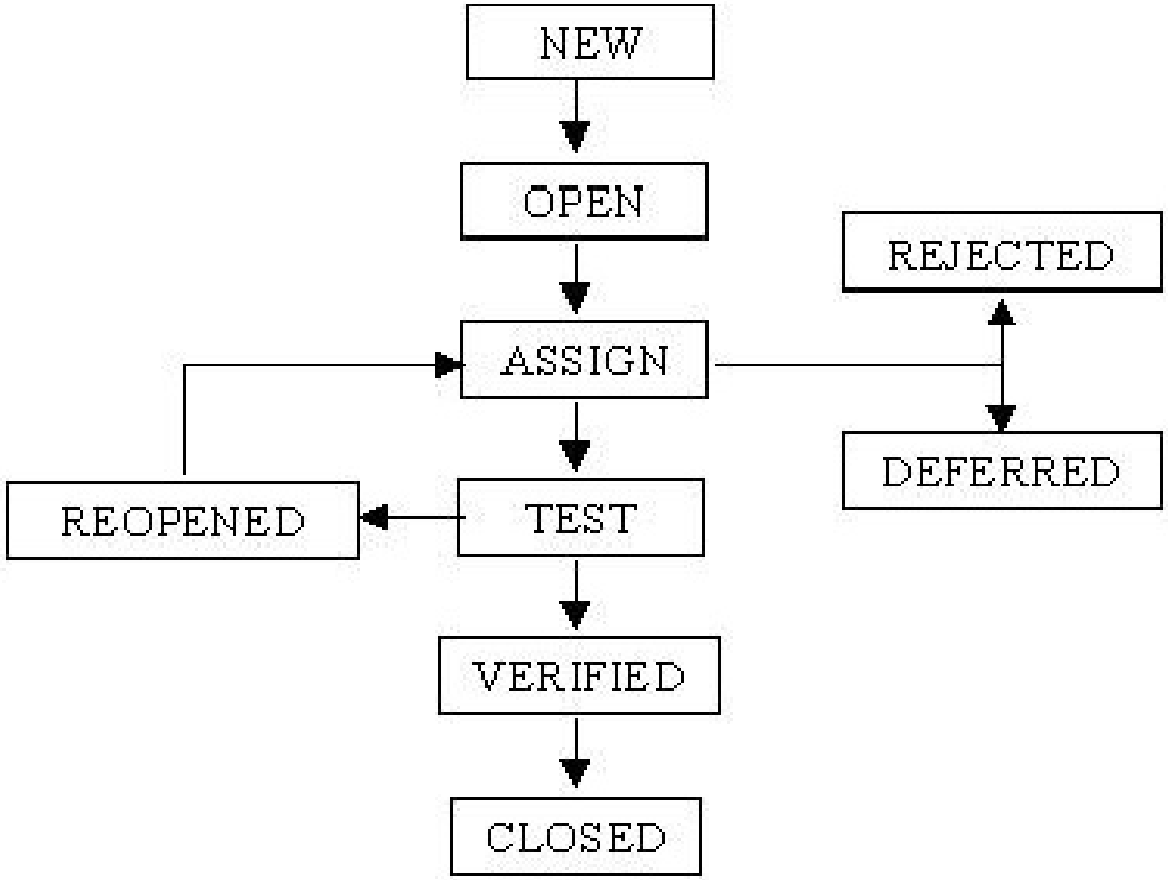
- urgency to fix a fault

Examples

- minor cosmetic typo
- crash if this feature is used



INCIDENT LIFECYCLE



INCIDENT LIFECYCLE