POWERING
POTENTIAL

Software Developer

Agile Software Development

Software Testing

PRINCIPLES

Discuss:

- What is software testing?
- Why do we need to test software?
- What happens if we don’t?

Software failures are estimated to cost the UK economy at least **£12BN**

What is the cost of failed software?

- **Financial loss**
  business unable to sell product, or customers leave.

- **wasted time** – time to fix and recover from failure.

- **loss of reputation** –
  bad press or negative market reaction.

- **Injury or death** – safety control system failure.

Testing

Reduces the risk of failure.

Improves the quality of software by:

- Removing defects during development.
- Preventing defects.
- Ensuring existing systems are not affected by changes.

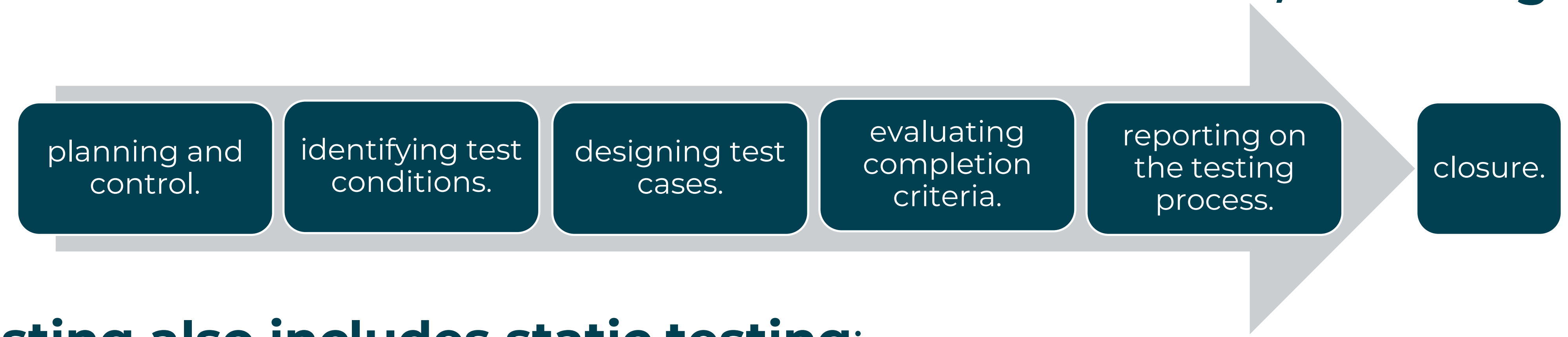
- Confirming **functional** and **non-functional** requirements.
- Ensuring reliability, availability, or performance.

- Meeting business requirements, contractual, legal.

Test Activities

Much more than simply executing software and checking results.

Test activities exist before and after test execution, including:


planning and control.
identifying test conditions.
designing test cases.
evaluating completion criteria.
reporting on the testing process.
closure.

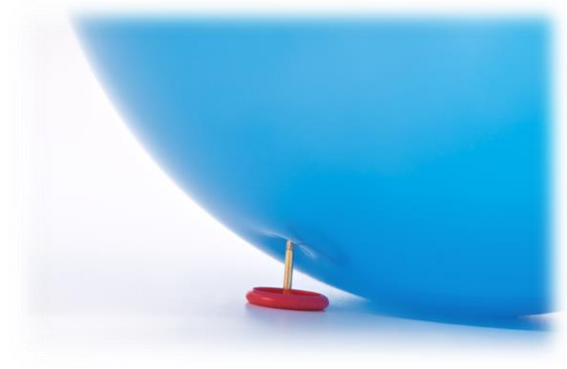
Testing also includes static testing:

- Reviewing documents or source code.

Software testing and quality

Typical quality measures are:

- number of defects found
- number of failures in a given time period (reliability)
- usability rating
- maintainability



Improve quality by

- learning from each project
- understanding root causes of defects
- integrating testing with other activities
- training, development standards

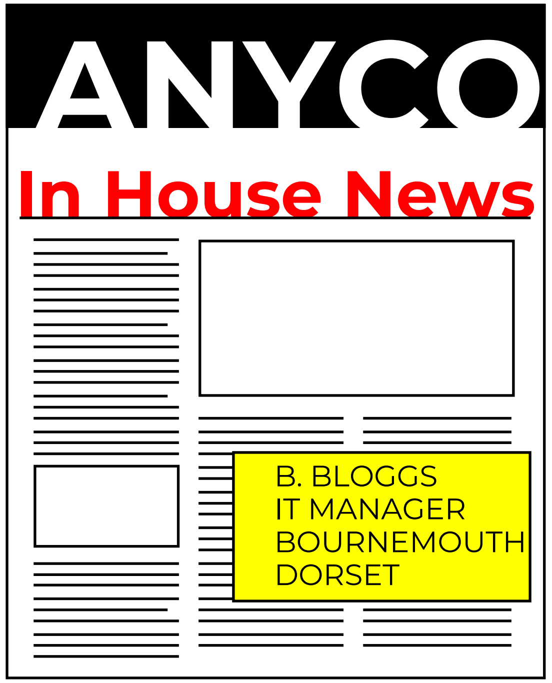
How much software testing is enough?

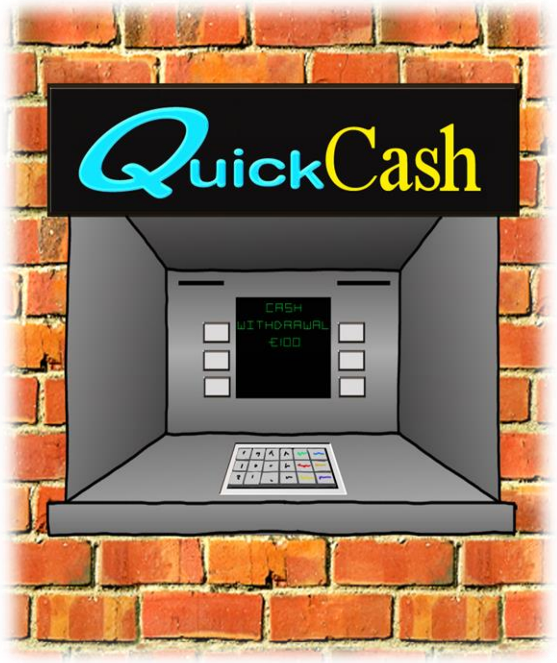
Systems vary in criticality.

The amount of testing, the approach taken, and techniques used will depend on:

→ the type of business system.

→ the risk of failure.





Seven testing principles

- **Testing shows the presence of defects**:
  - But cannot find 100% of defects.
- **Exhaustive testing is impossible unless you've unlimited time and resources!**
  - Infinite Input Combinations
  - Dynamic Nature of Software (number of states and CI).
  - Complexity of Interactions between components
  - Unpredictable Environments
- **Early testing**:
  - Start testing activities early in SDLC and parallel to coding.
- **Defect clustering**:
  - A small number of modules tend to contain the most defects.

Seven testing principles

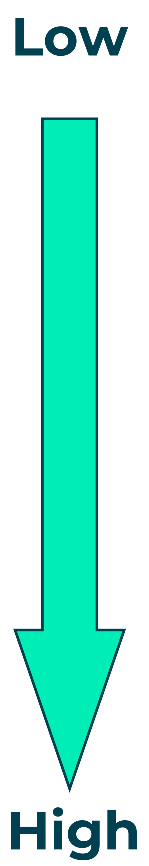
- **Pesticide paradox:**
  - Rerun existing test but always add new ones.

- **Testing is context-dependent:**
  - Website, safety-critical application, batch payroll system.

- **Absence-of-errors fallacy:**
  - Lack of discovered errors does not imply perfection!

Testing requires independence

Levels of independence:



→Tests designed by the person(s) who wrote the software under test.

→Tests designed by another person(s) within the development team.

→Tests designed by a different organisational group or specialist.

→Tests designed by a different organisation or company.

Functional Testing

„Testing based on an analysis of the specification of the functionality of a component or system„

ISTQB® Glossary, 2010

Tests WHAT the system does

- As described in business requirements, functional specifications, Use Cases, etc.
  Considers the external behavior of the software – Black Box

- Component specification is a source of functional tests

Non-Functional Tests

- Performance
- Security
- Reliability
- Load
- Stress
- Volume
- Usability
- Accessibility
- Maintainability
- Portability

LEVELS

Please answer the questions.

- List and briefly describe the various levels of testing:
  - Unit
  - Integration
  - System
  - Acceptance

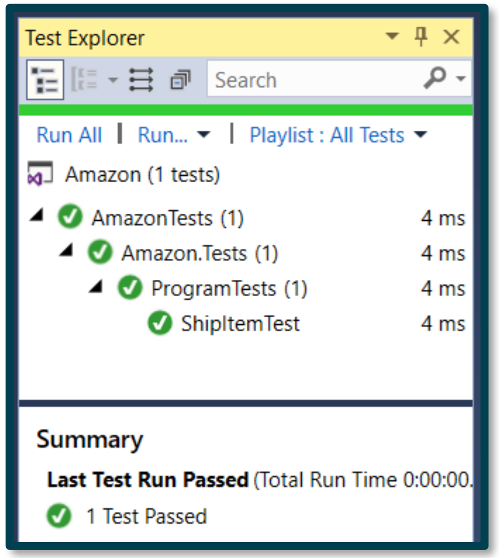
Unit (component) testing

Objectives:

- Remove defects
- Verify component specification, design, or data model
- Check code coverage
- May check non-functional

Test objects:

- Software modules, programs, objects, classes



```
Amazon (1 tests)
◢ AmazonTests (1)            4 ms
  ◢ Amazon.Tests (1)         4 ms
    ◢ ProgramTests (1)       4 ms
        ShipItemTest         4 ms
```

Summary
Last Test Run Passed (Total Run Time 0:00:00.
1 Test Passed

Unit Testing

Typical defects and failures:

- Incorrect logic, definition, or use of variables
- Incorrect data types
- Misinterpretation of specifications

Tool support:

- IDE, debugging tools
- Unit test framework

Responsibilities:

- Conducted by developers, not testers

Integration testing

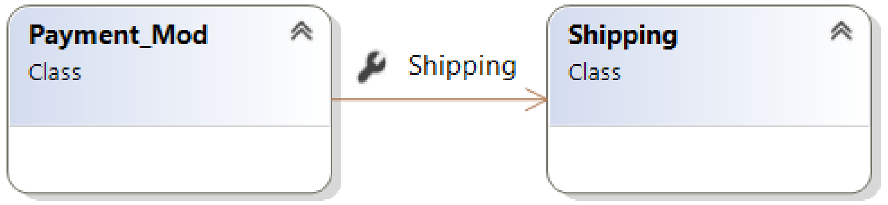
Tests interfaces and interactions between components.
Involves functional and some non-functional testing.

Test objects:

Software components. Internal interfaces.

Typical defects and failures:

Communication failures between components.
Parameter mismatches.



Responsibilities:

Conducted by developers

System testing

Objectives:

- Test the functional and non-functional behaviours of the whole system.

Test basis:

- System requirement specification and behaviour
- Business processes.
- Interactions with OS and system resources.

Typical defects and failures:

- Likely to include non-functional issues – load, volume, number of users....
- Incorrect system design specification.

Acceptance testing

Objectives are to ensure:

- confidence in the system, or specific non-functional characteristics.
- the system is fit for purpose.
- the system's readiness for deployment.
- the above is done before deployment and large-scale integration.

Test basis:

- User requirements.
- System requirements.
- Use cases.
- Business processes.
- Risk analysis reports.

Types of Acceptance Testing

User acceptance testing:

- Verify the system is fit for use by the business users.

Operational acceptance testing by system administrators:

Backup / restore.

Disaster recovery.

User management.

Maintenance tasks.

Data load and migration tasks.

Periodic checks of security vulnerabilities.

Contract acceptance testing:

- Third party apps must meet acceptance criteria defined in the contract
- System complies with government, legal, safety, and other regulations.

Types of acceptance testing

Alpha and beta testing:

- ➢Applies to the developers of COTS software.
- ➢Gets feedback from customers before commercial sale.

Alpha testing:	• Factory acceptance testing. • Done by customers at the developing organisation site.
Beta testing:	• Field or site acceptance testing. • Done by customers at the customers' site.

Test-first / test-driven development

One approach is to prepare and automate test cases before coding:

→ Test-first or test-driven development.

Highly iterative process:

→ Write component test cases first.

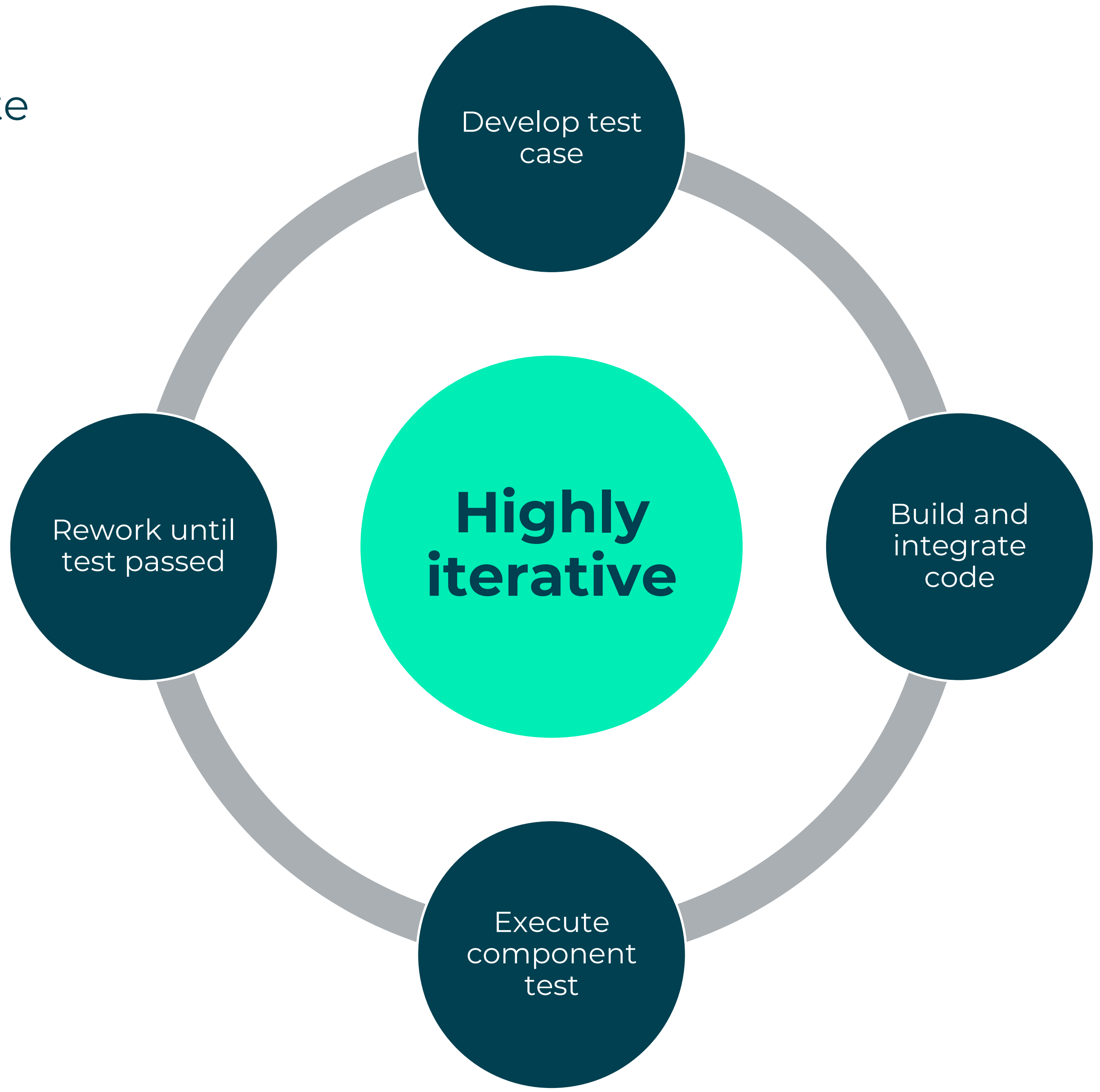
→ Build and integrate code.

→ Run test cases.

→ Correct defects.

Used in Agile models.

Will investigate TDD in another course



Static Testing Techniques

Informal Reviews

- Informal peer review – Documentation optional
- Useful and cheap – Not as effective as formal review

Walkthroughs

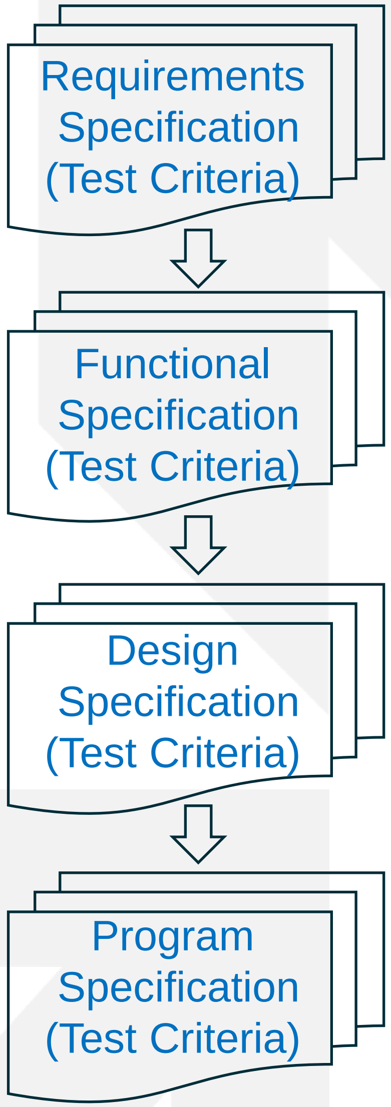
- Peer group review led by author – Scenarios

Technical Reviews

- Documented, defined fault-detection process
- Includes peers and technical experts
- Check lists - ‘What if’ activities

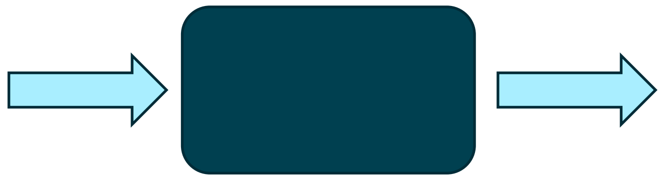
Inspections

- Formal review – Roles and objectives
- Defined roles – Inspectors, Scribe, Moderator


Requirements Specification (Test Criteria)
Functional Specification (Test Criteria)
Design Specification (Test Criteria)
Program Specification (Test Criteria)

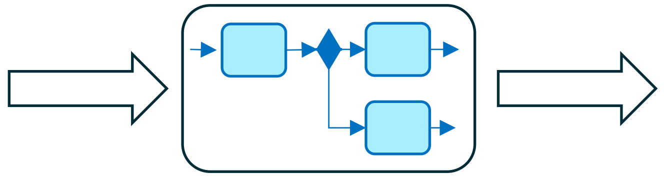
Dynamic Testing – Techniques

Black-box



Concerned with the correctness of the outputs based on the inputs
Not concerned with how the program works

White-box



Concerned with *how* the program works
Exercises the program logic and data processing

review

QA POWERING POTENTIAL

QUESTIONS AND FEEDBACK