Regression Testing

Conrad Hughes
School of Informatics

Slides thanks to Stuart Anderson

Regression Testing

- Regression testing is applied to code immediately after changes are made.
- The goal is to assure that the changes have not had unintended consequences on the behaviour of the test object.
- We can apply regression testing during development and in the field after the system has been upgraded or maintained in some other way.
- Good regression tests give us confidence that we can change the object of test while maintaining its intended behaviour.
- So, for example, we can change to a new version of some piece of infrastructure in the environment, make changes to the system to take account of that and then ensure the system behaves as it should.
- Regression testing is an important way of monitoring the effects of change.
- There are many issues but the balance of confidence against cost is critical.

Why Use Regression Tests?

- Good reasons:
  - Bug fixes often break other things the developer isn't concentrating on.
  - Sometimes bug fixes don't fix the bug.
  - Checking software still runs after making a change in the infrastructure.
  - Discovering faulty localisation.
  - Errors in the build process (e.g. wrong parameters).
  - Conforming to standards or regulators.
- Bad reasons:
  - Arguments in terms of replicability of results (i.e. scientific analogy).
  - Arguments in terms of quality in analogy with a production line (i.e. a manufacturing analogy).

Risks of Change

- **Bug regression testing**: checks that a bug fix has removed the symptoms of the bug that have been identified.
- **Old fix regression:** checks that a new fix has not broken an old fix: refactoring should limit this as old fixes are refactored into the code.
- **Functional regression:** new code or fix has not broken previously working code.
- **Incremental Regression testing:** regression testing as we develop.
- **Localisation Testing:** tests if a product has been correctly localised for a particular market.
- **Build Testing:** has an error been introduced in the field that means the system will not build correctly.

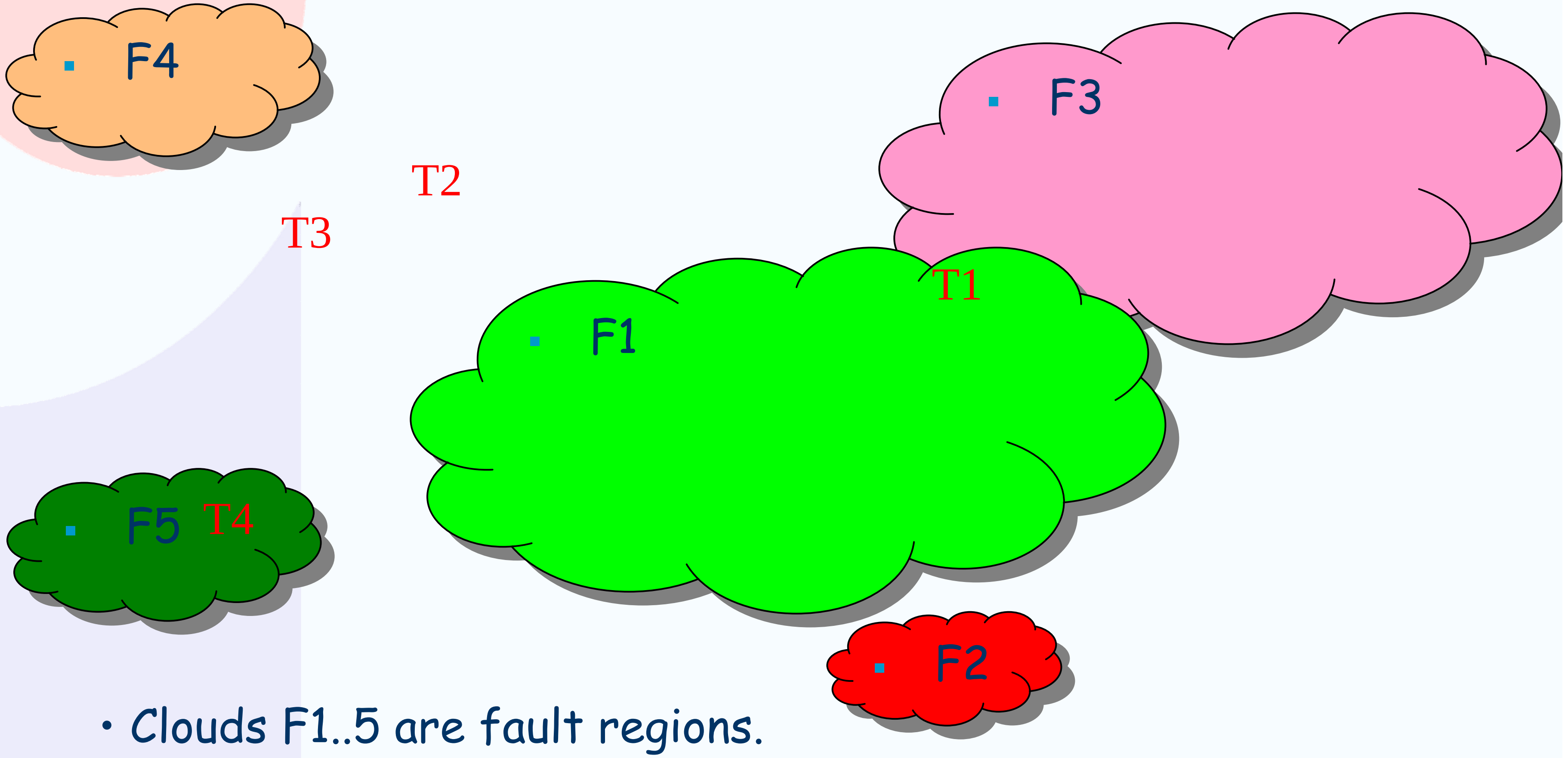
Motivation for Reusing Tests

- Motivations vary depending on the context:
  - In development (e.g. XP) tests play the role of specifications so we want to keep them fixed and reduce the cost of regression.
  - In an established product:
    - Using the same tests may help us manage risk since we can focus tests on mitigating a particular risk.
    - Some tests are good at uncovering likely errors so we want to reuse.
    - There may be economic motivations:
      - Automated retest (replay or oracle).
      - Replay with human inspection may reduce the need for specialist technical time (e.g. in GUI testing – this is a particularly common approach). The aim is to routinise repeat testing.

Key Questions about Reuse

- Which tests should we reuse (for a particular situation – this may vary if tests are expensive to carry out)?
  - The "goodness" of a test is context sensitive, so in a development situation it may be good to concentrate on the core functionality – but later in the cycle this may be less important.
- What is the cost of maintaining tests?
  - Complex tests may be make extensive use of the environment and may be complex to maintain.
  - This is not an argument against using complex tests but it is an argument in favour of developing test architecture to support tests.
  - Specific architectures have corresponding test architectures e.g. Web Services.
- What is the cost of applying tests?
- What is the benefit of applying regression tests?

A "Model"

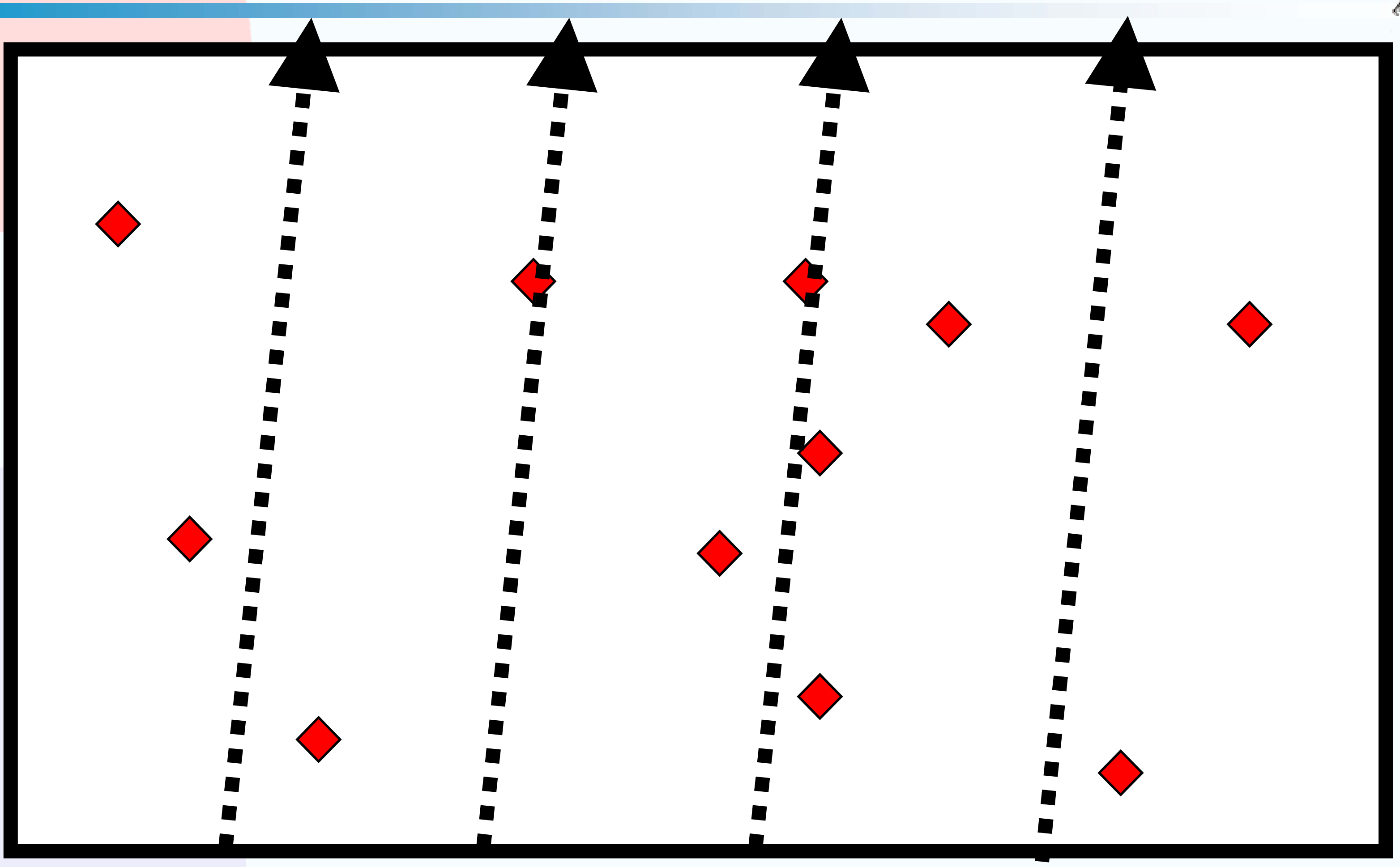


- Clouds F1..5 are fault regions.
- T1..4 are point tests.

Fault Region Model

- Systems have fault regions where their behaviour is does not conform to the requirements.
- Tests are point executions of the system.
- Test specifications may specify a region in the input space
- We still have to execute on test (unless we can do symbolic execution).
- Faults come in all shapes and sizes and may overlap or be intertwined.
- When a test hits a fault region we discover an error.
- At that point – we change the system so:
  - The clouds can move,
  - A cloud can disappear,
  - One of an overlapping pair can disappear
  - Clouds can break into fragment or amalgamate,
  - Clouds can appear.
- So retest can be valuable – approx 15% of errors are discovered by regression test – these are often critical to product quality.

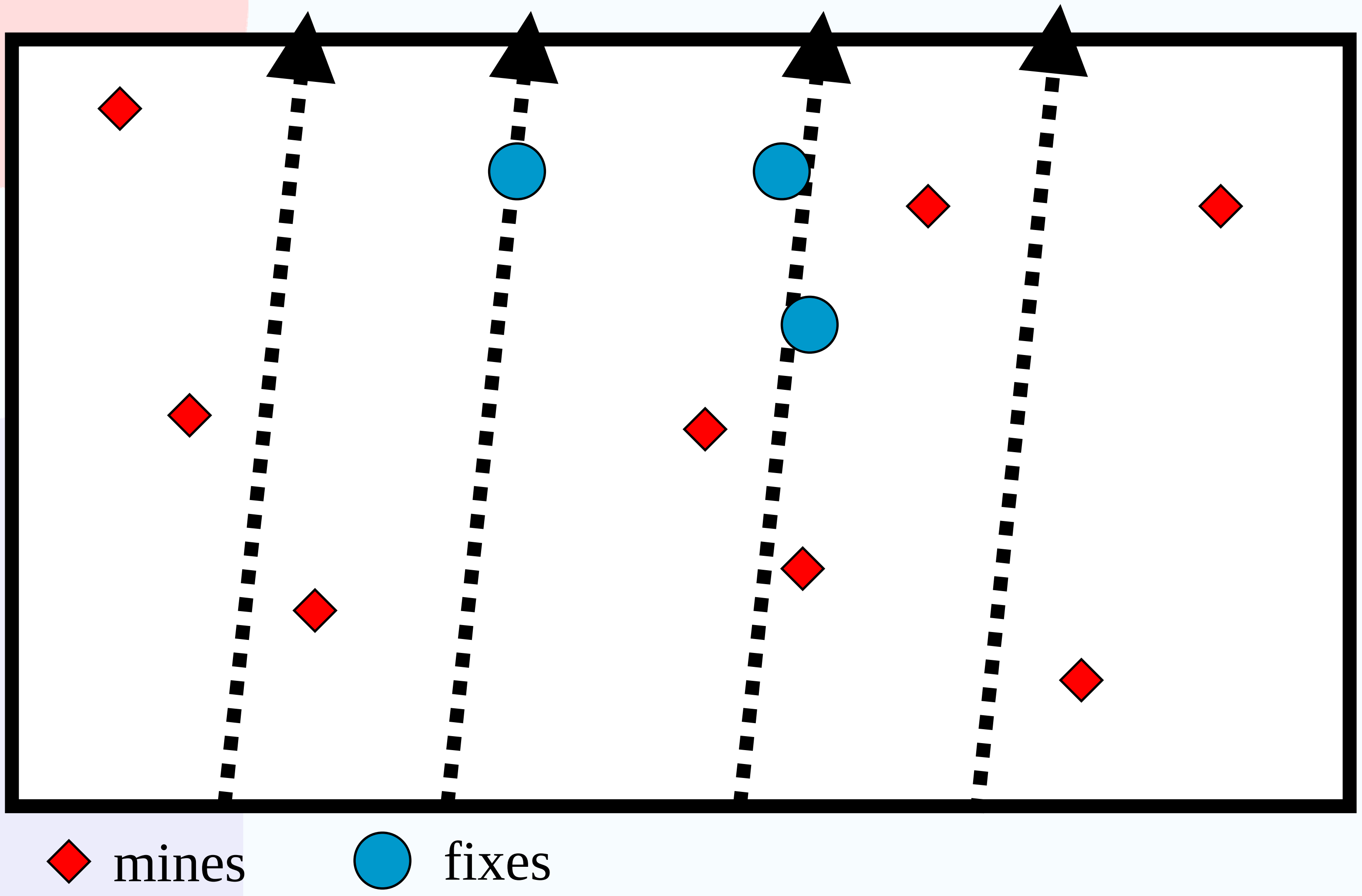
An analogy: Clearing mines (Bach)



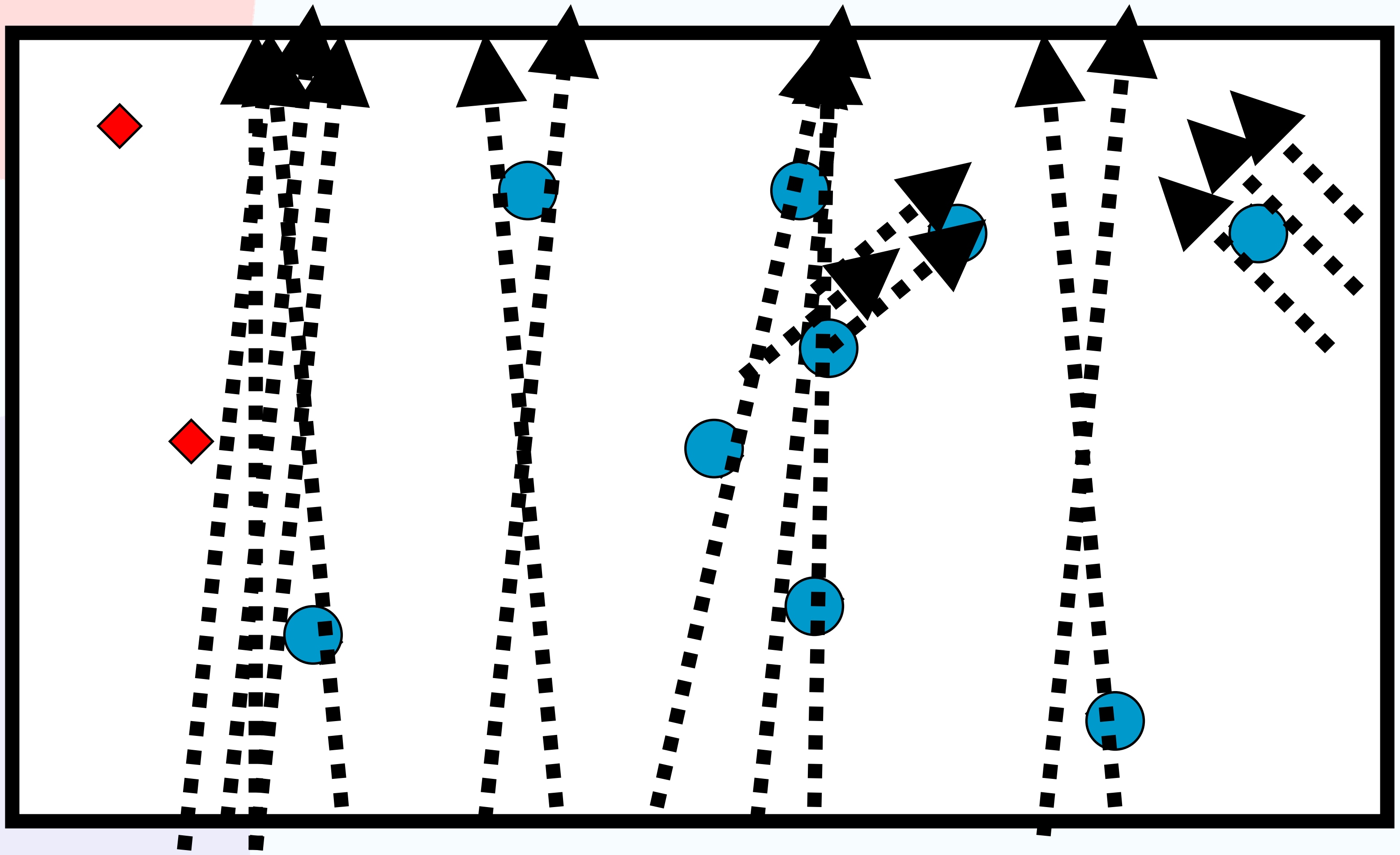
◆ mines

This analogy was first presented by Brian Marick.
These slides are from James Bach..

Totally repeatable tests won't clear the minefield (Bach)



Variable Tests are Often More Effective (Bach)



◆ mines ● fixes

Automated GUI regression (Bach)

- Look back at the minefield analogy
- Are you convinced that variable tests will find more bugs under *all* circumstances?
  - ***If so, why would people do repeated tests?***
  - ***Are bugs like clouds or mines?***

Generate 10 counter-examples to the minefield analogy.

... i.e. arguments why repeating the same tests could be valuable.

Economic Perspective

- What is the best way to improve product quality:
  - Maintain a regression test set
  - Develop new tests
  - Is it possible to develop new tests for low value events (e.g. patch bundles)
- What is the benefit of reusing tests:
  - Tends to focus on core functionality of the system
  - Perhaps takes a narrow view of the functionality.
- Costs:
  - How much does it cost to maintain tests?
  - How much does it cost to create tests?

Support for Refactoring

- Tests act as an executable specification.
- Tools like JUnit reduce the cost to the developer.
- Tendency to focus on unit level behaviour.
- Tendency to focus on function over resource use.
- Issues about how to integrate many unit level test sets that have been created individually.

Risk management

- Tests target critical behaviour – the main hazards.
- For embedded systems we have good specifications and it may be possible to infer more from a test result.
- We can use combinations of old tests to exercise the system more extensively on retest:
  - More tests.
  - More combinations of test.
  - More variants.
  - With a good specification we can see how the tests cover the different behaviours of the system.
  - We provide independent testers with a large armoury of possible weapons to break the system.

Summary

- Regression testing provides a tool for managing change.
- Regression testing can be used throughout the lifecycle.
- It can reduce the cost of applying tests (by storing the expected result).
- It is a tool in helping to provide stability in the face of code change.
- Costs of test maintenance and test reuse are very variable but in some environments it is affordable.
- Standards and regulation often require regression testing.
- The analogy between a manufacturing environment and a software production environment is very weak.
- The role of testing in the two environments is quite different.