A testing perspective

1 Why test software?

Software is an ISP, so no definitive specification, many possible solutions and no definitive way of knowing how good a solution is. Hence we need to test a solution to see how well it fits. And we can regard the process of testing as being an ISP, needing a design approach.

Alternate answer:

- To discover problems (a developer-centric view): does the component/system behave as expected?
- To assess quality (a customer/market-led view): is the system acceptable to the end users?
- Plan driven forms of development tend to put early emphasis on the developer perspective, agile forms tend to emphasise both perspectives throughout.

2 Limitations

Because of the characteristics of an ISP, testing can't demonstrate that software is free of defects or that it will always behave as specified.

In no sense does testing "prove" anything, it can only be a means of providing confidence about software

3 Where's the challenge?

- Many people find designing tests offers an intellectual challenge, regardless of perspective
- In some ways it is a bit like a game, you are pitting your wits against the designer of the software
- The job of the tester is to "think outside of the box" and try to avoid making the same (possibly wrong) assumptions that the developer might have done

4 Definitions

Error - Mistakes made when designing (or coding), which may be of logic (using the wrong operator) or of interpretation (such as misunderstanding a requirement). Often terms bugs and not usually detectable by a compiler

Fault - The result of an error (or its representation). Faults may be of commission(something that should be different, such as the wrong operator) or omission(something absent). The latter tend to be harder to detect and resolve

Failure - Occurs when a fault executes (or doesn't in the case of omission). Might not occur immediately, the fault may seed a later system crash

Test - Exercising software with test cases to check correctness or find faults

Test case - A set of inputs together with the expected outputs

5 The Test Oracle

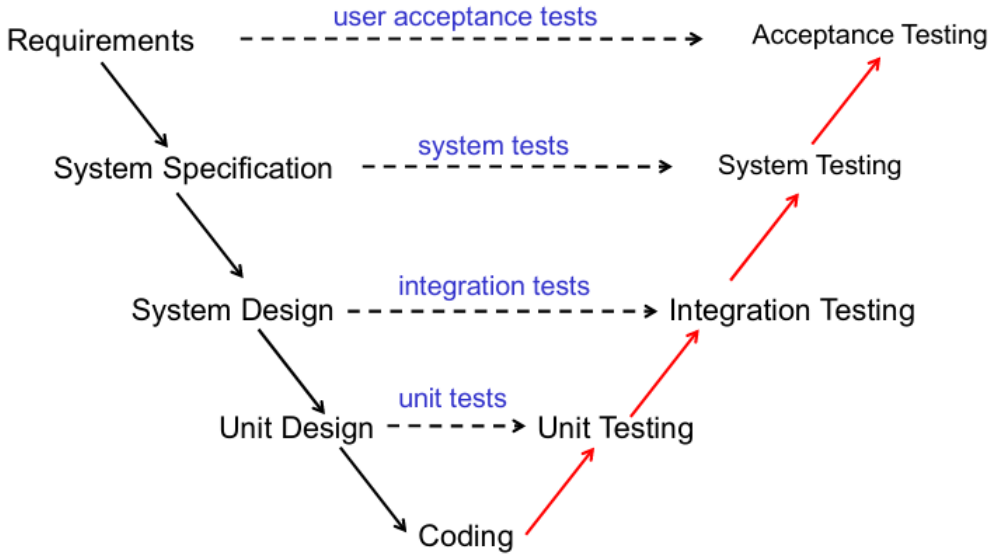
The key action when testing is to compare the value(s) or state changes that are output from our test case with some predicted value(s) or state changes.

The predictions are produced by the test oracle, a mythical being that "knows all the answers"

So designing test cases involves specifying both input and output values/states. For more complex stages, such as acceptance testing, the oracle may actually take the form of a set of expert users who "judge" whether or not the outputs are acceptable.

6 Testing in the Life Cycle

- Testing is one of the activities that is organised quite differently within waterfall and agile development processes, although the actual test details may not be very different
- Development usually has a hierarchy of test activities corresponding with particular stages of development, regardless of how they are organised.
- With agile forms, testing may be more interpolated with development activities, depending on the form. In particular, with XP, there is the practice of writing the testes before writing the code



7 Regression Testing

If we make changes to the elements of the system then we need to test everything again. This process is termed regression testing and is used to ensure that the changes have not destabilised code that previously worked

8 Canary Release

Even after all the testing, releasing a new version of software to users might reveal errors that the testing didn't uncover. One technique that can help here is to initially release it to selected users before making it publicly available.

9 The two major test strategies

Functional testing - Black box testing where we have no details of the inner workings of the parts of the system being tested. Helps to establish confidence

Structural testing - White box/clear box testing, for which the implementation is visible and can be used to generate test cases

10 Error and Fault taxonomies

Not all errors are of equal importance in terms of their effect

- A system may well be able to continue functioning in the presence of errors, although not fully correctly
- Some errors may only occur on very rare occasions

This is one example of a scale for severity:

	Level of severity	Example
1.	Mild	Misspelled word in output
2.	Moderate	Provides misleading or redundant information
3.	Annoying	Truncated names, bills for £0.00
4.	Disturbing	Some transactions are not processed
5.	Serious	Loses a transaction
6.	Very serious	Incorrect transaction execution
7.	Extreme	Frequent 'very serious' errors
8.	Intolerable	Database corruption
9.	Catastrophic	System shut down
10.	Infectious	Shut down that spreads to others

11 Implications

All of these forms of testing are relevant. Since regression testing is almost impossible to avoid, it is worth planning for it.

One way to do this is to ensure that, wherever possible, you store test cases (inputs and predicted outputs) in files, for ease of reuse.

You should aim to record the outcomes of tests in a systematic matter, adding them to a file or database