Higher-Order Testing

Conrad Hughes
School of Informatics

Slides thanks to Stuart Anderson
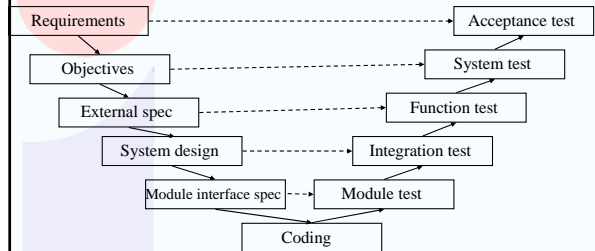
informatics

The V-Model and Defining Higher Order Tests

Requirements - - - - - - - - - → Acceptance test

Objectives - - - - - - - - - → System test

External spec - - - - - - - - - → Function test

System design - - - - - - - - - Integration test

Module interface spec — Module test

Coding

V-Model Stages

- Meyers version of the V-model has a number of stages that relate to distinct testing phases all of which are useful:
  1. Requirements – seen as inherently informal for Meyers so the acceptance test must engage with the user in agreeing the requirements have been met.
  2. Objectives are more tightly specified and quantitative issues of costs, sizes, speeds together with tradeoffs have been established. The system test should be directed to see that objectives have been met (objectives have clear criteria to decide whether they have been met).
  3. The external specification is a black-box specification of the functionality of the system and thus the function test is directed to checking the implementation of the specified function.
  4. System design provides a structure that can be tested by Integration testing (inherently dependent of the structure of the system).
  5. Module interface specifications are the basis for rigorous unit test (both black-box and white-box).

Test Objectives

- Given there are many different types of test there is the possibility of considerable redundancy across tests and the potential to waste effort.
- One way of managing this is to set a test objective for each level of test. Ideally an objective should be dealt with at the lowest level testing.
- For example we might take the following objectives:
  - Module test aims to find discrepancies between a module's behaviour and its interface specification.
  - Function test aims to demonstrate that the program does not conform to its external specification.
  - System test aims to show the product is inconsistent with the original objectives (e.g. that some quantitative aspect of the system is not delivered by the product).

Critique of the V-Model

- The V-model is often criticised because it provides a very highly constrained model of the interaction of test and development and of the sequencing of testing activities.
- However, the model is useful in identifying different kinds of test that need orchestrating in any development process.
- More iterative or agile processes will have all these types of test in one iteration that delivers new functionality.
- For some of the software lifecycles it may be that there is no independent notion of e.g. requirements and so that particular type of testing will be de-emphasised in that approach.
- Increasingly, as products become longer-lived, there is the issue of how to use test to evolve fielded products and how to use field data to improve product quality (e.g. beta testing exploits the ability to use a range of test sites to test systems).

Translating the V-Model to Other Processes

- If we consider an XP approach to development and consider the V-model classification of testing we can see that:
  - Module, Integration, and Function test are carried out against module specifications and external specifications of the system.
  - System and acceptance testing is moved more to the user who may be the embedded customer along with the team. The use of users as beta-testers helps identify real requirements issues as soon as significant functionality is available.
  - Regression testing is linked to refactoring as a means of ensuring no change in the delivered behaviour.
  - It may be that the system has no well-documented requirements and so it is impossible to carry out a system test. It may also be that requirements are acquired iteratively as the system evolves.

Module/Unit/Integration Testing

- It is important that this activity is appropriately targeted to avoid duplication of effort from other high-level test activity.
- The focus here should be on verifying that the behaviour of the components satisfies the interface specifications.
- In XP it may be that the collection of tests serve as a specification for the system.

Functional Testing

- Functional testing is aimed at discovering discrepancies between the behaviour of the system and its external specification.
- An external specification is a precise description of the required behaviour form a users point of view.
- Typically systems are seen as delivering some key functionality and functional testing is directed to uncovering failures to provide the functions of the system.
- Functional testing techniques are black-box and use the approaches we have discussed earlier in the course e.g. category partition, common error lists, boundary-value testing.

System Testing 1

- System testing is dependent on their being a quantified set of objectives for the system – these may be derived iteratively but we need to capture them to allow effective systems testing.
- Focus of system testing is to find discrepancies in the translation of the objectives into the external specifications.
- So the system test should be related to the objectives of the system – these should be captured in user-oriented term in the user documentations so we use the user document to help design he system tests.
- The problem in creating system tests is that there is no exhaustive list of the kinds of test that might be necessary to test the objectives of a system.
- Meyers provides a list of 15 different kinds of test that might be involved in testing the objectives of a system – but this is not exhaustive – it may be necessary to add some other testing approaches depending on the objectives

System Testing 2

- Facility Testing: this is checking that user accessible facilities (or functions) are implemented. This is an all-or-nothing test – either the system attempts to provide a facility or not – this might take the form of a visual inspection of the system.
- Volume Testing: objectives will usually have requirements on the volume of objects that a system can handle. E.g. a simulator might have a specification that it is capable of handling up to 100,000 logic gates or a database system might have a lower bound on the number of records handled.
- Stress Testing: systems involving real-time constraints will almost always have objective in terms of how rapidly they can deal with events (e.g. 10,000 transactions per hour, the VISA payments system supports more than 5000 card transactions per second – with very high reliability). In addition many system have multiple inputs and stress testing might investigate how well the system handles many input parameters changing simultaneously.

System Test 3

- Usability Testing: often carried out with real usability testers who can discover serious usability flaws even in well-implemented systems. Typical issues include:
  1. Appropriateness of interface to the user group
  2. Meaningful interactions, warnings etc
  3. Use of consistent styles, metaphors etc
  4. Is redundancy/diversity of interfaces used where accuracy is essential?
  5. Is the interface too configurable?
  6. Is it clear that the system is always live?
  7. Is it easy to get the system to do what the user wants?

System Test 4

- Security Testing: The external specification will usually have been derived from the objectives by developing a security model that constrains the flow of information from one area of the system to another. System testing should identify any flaws in moving from objectives to a behavioural specification. In addition the objectives may include a list of common threats that we should demonstrate are dealt with in the external specification.
- Performance Testing: many systems will have throughput objectives. These may be express statistically in terms of rates and likelihood of failure. The system should be instrumented to allow these objectives to by subjected test when the system is in the field.
- Storage Testing: systems will claim to run happily in a given amount of storage – we should check this.

System Test 5

- Configuration Testing: many systems are highly configurable (e.g. SAP and Oracle). A good example of this is a multi-platform system where considerable configuration effort might be required to get a system to run on a particular platform
- Compatibility/Compliance Testing: Objectives will often include compliance statements that the system is compliant with a standard – this style of testing aims to cover a significant subset of the range of different configurations.
- Installability Testing: Is the system easy to install correctly?
- Reliability Testing: is the mean time to failure long enouh
- Recovery Testing: assessing the mean time to repair a fault is often a critical objective of a system
- Serviceability Testing: how easy is it to upgrade the system?
- Documentation Testing: how well does the documentation match the actual system behaviour
- Procedure Testing: many systems involve complex interations between computers and human procedures – are the procedures and the computer system compatible?

System Testing Process/Management

- System testing should not be performed by the developers – ideally it should be an independent organisation.
- Seemingly independent objectives have dependencies and the planning for system test has to take account of these dependencies (in order to get better coverage).

Acceptance Testing

- Carried out by the customer on their premises.
- Testing is carried out independent of the development organisation.
- The acceptance test marks the transition from development before use to development in use.

Installation Testing

- Does the proposed software run on the proposed users configuration?
- Many vendors now offer to integrate systems offsite and bring them to the users site.
- Issues like versions of software, library versions, hardware, networking all impact on the runnability of a system.
- Installation testing should aim to characterise the hardware/software combination that ate

Test Planning and Control

- Even for modest-sized systems, there is a considerable management load in ensuring the tests are well-managed.
- Plans should be realistic and should allow for the discovery and repair of the most serious classes of error.
- A good test plan should:
  1. Have clear objectives for each phase of testing.
  2. Have clear completion criteria for each stage of test.
  3. Have a project plan with clear timings on activities and the dependency between activities.
  4. Testing responsibilities should be clearly allocated.
  5. Large projects should systematically manage resources.
  6. Appropriate hardware may need to be ordered/configured/secured.
  7. Good tracking of bugs is essential as is the tracking of all testing activity.
  8. Good debugging procedures.
  9. The capacity to regression test quickly after a fault has been repaired.

Test Completion Criteria

- Bad criteria include:
  - Stop when we run out of time.
  - Stop when test cases fail to find errors.
- One possible approach is to target a particular residual defect density.
- We have estimates of the effectiveness of bug finding for different parts of the system and different approaches to bug finding.
- Providing a target RDD is a good way of structuring the completion of particular stage in the system test.

Summary

- Higher-order testing is a complex business.
- There is very little hard advice to give since most of the approaches vary depending on the class of system.
- Not all approaches to system testing are necessary for all systems – deciding on a suitable collections of system tests can be difficult.