

CSC 4181 – Compiler Construction Software Engineering Lectures

Part 3

Dr. Tom Way

CSC 4181

Slide 1

Verification and Validation

Dr. Tom Way

CSC 4181

Slide 2

Verification vs validation

- Verification:
"Are we building the product right".
- The software should conform to its specification.
- Validation:
"Are we building the right product".
- The software should do what the user really requires.

Dr. Tom Way

CSC 4181

Slide 3

The V & V process

- Is a whole life-cycle process - V & V must be applied at each stage in the software process.
- Has two principal objectives
 - The discovery of defects in a system;
 - The assessment of whether or not the system is useful and useable in an operational situation.

Dr. Tom Way

CSC 4181

Slide 4

V& V goals

- Verification and validation should establish confidence that the software is fit for purpose.
- This does NOT mean completely free of defects.
- Rather, it must be good enough for its intended use and the type of use will determine the degree of confidence that is needed.

Dr. Tom Way

CSC 4181

Slide 5

V & V confidence

- Depends on system's purpose, user expectations and marketing environment
 - Software function
 - The level of confidence depends on how critical the software is to an organisation.
 - User expectations
 - Users may have low expectations of certain kinds of software.
 - Marketing environment
 - Getting a product to market early may be more important than finding defects in the program.

Dr. Tom Way

CSC 4181

Slide 6

Static and dynamic verification

- Software inspections. Concerned with analysis of the static system representation to discover problems (static verification)
 - May be supplement by tool-based document and code analysis
- Software testing. Concerned with exercising and observing product behaviour (dynamic verification)
 - The system is executed with test data and its operational behaviour is observed

Program testing

- Can reveal the presence of errors NOT their absence.
- The only validation technique for non-functional requirements as the software has to be executed to see how it behaves.
- Should be used in conjunction with static verification to provide full V&V coverage.

Types of testing

- Defect testing
 - Tests designed to discover system defects.
 - A successful defect test is one which reveals the presence of defects in a system.
 - Covered in Chapter 23
- Validation testing
 - Intended to show that the software meets its requirements.
 - A successful test is one that shows that a requirements has been properly implemented.

The structure of a software test plan

- The testing process.
- Requirements traceability.
- Tested items.
- Testing schedule.
- Test recording procedures.
- Hardware and software requirements.
- Constraints.

The software test plan

The testing process
A description of the major phases of the testing process. These might be as described earlier in this chapter.

Requirements traceability
Users are most interested in the system meeting its requirements and testing should be planned so that all requirements are individually tested.

Tested items
The products of the software process that are to be tested should be specified.

Testing schedule
An overall testing schedule and resource allocation for this schedule. This, obviously, is linked to the more general project development schedule.

Test recording procedures
It is not enough simply to run tests. The results of the tests must be systematically recorded. It must be possible to audit the testing process to check that it has been carried out correctly.

Hardware and software requirements
This section should set out software tools required and estimated hardware utilisation.

Constraints
Constraints affecting the testing process such as staff shortages should be anticipated in this section.

Software inspections

- These involve people examining the source representation with the aim of discovering anomalies and defects.
- Inspections not require execution of a system so may be used before implementation.
- They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.

Cleanroom software development

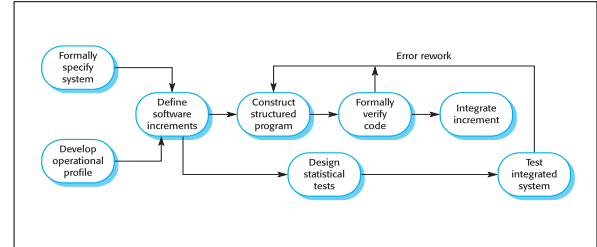
- The name is derived from the 'Cleanroom' process in semiconductor fabrication. The philosophy is defect avoidance rather than defect removal.
- This software development process is based on:
 - Incremental development;
 - Formal specification;
 - Static verification using correctness arguments;
 - Statistical testing to determine program reliability.

Dr. Tom Way

CSC 4181

Slide 13

The Cleanroom process



Dr. Tom Way

CSC 4181

Slide 14

Software testing

Dr. Tom Way

CSC 4181

Slide 15

The testing process

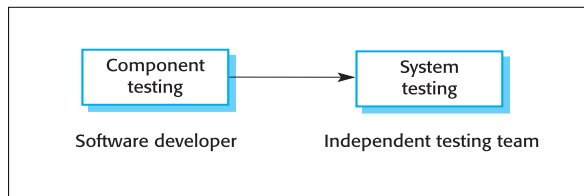
- Component testing
 - Testing of individual program components;
 - Usually the responsibility of the component developer (except sometimes for critical systems);
 - Tests are derived from the developer's experience.
- System testing
 - Testing of groups of components integrated to create a system or sub-system;
 - The responsibility of an independent testing team;
 - Tests are based on a system specification.

Dr. Tom Way

CSC 4181

Slide 16

Testing phases



Dr. Tom Way

CSC 4181

Slide 17

Defect testing

- The goal of defect testing is to discover defects in programs
- A *successful* defect test is a test which causes a program to behave in an anomalous way
- Tests show the presence not the absence of defects

Dr. Tom Way

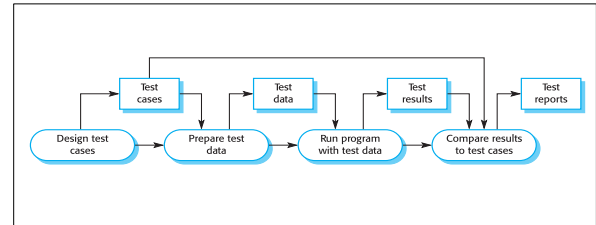
CSC 4181

Slide 18

Testing process goals

- Validation testing
 - To demonstrate to the developer and the system customer that the software meets its requirements;
 - A successful test shows that the system operates as intended.
- Defect testing
 - To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification;
 - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.

The software testing process



System testing

- Involves integrating components to create a system or sub-system.
- May involve testing an increment to be delivered to the customer.
- Two phases:
 - Integration testing - the test team have access to the system source code. The system is tested as components are integrated.
 - Release testing - the test team test the complete system to be delivered as a black-box.

Integration testing

- Involves building a system from its components and testing it for problems that arise from component interactions.
- Top-down integration
 - Develop the skeleton of the system and populate it with components.
- Bottom-up integration
 - Integrate infrastructure components then add functional components.
- To simplify error localisation, systems should be incrementally integrated.

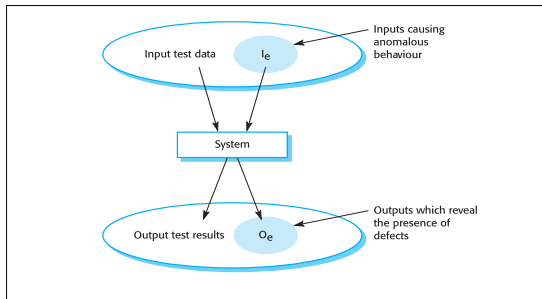
Testing approaches

- Architectural validation
 - Top-down integration testing is better at discovering errors in the system architecture.
- System demonstration
 - Top-down integration testing allows a limited demonstration at an early stage in the development.
- Test implementation
 - Often easier with bottom-up integration testing.
- Test observation
 - Problems with both approaches. Extra code may be required to observe tests.

Release testing

- The process of testing a release of a system that will be distributed to customers.
- Primary goal is to increase the supplier's confidence that the system meets its requirements.
- Release testing is usually black-box or functional testing
 - Based on the system specification only;
 - Testers do not have knowledge of the system implementation.

Black-box testing



Dr. Tom Way

CSC 4181

Slide 25

Testing guidelines

- Testing guidelines are hints for the testing team to help them choose tests that will reveal defects in the system
 - Choose inputs that force the system to generate all error messages;
 - Design inputs that cause buffers to overflow;
 - Repeat the same input or input series several times;
 - Force invalid outputs to be generated;
 - Force computation results to be too large or too small.

Dr. Tom Way

CSC 4181

Slide 26

Use cases

- Use cases can be a basis for deriving the tests for a system. They help identify operations to be tested and help design the required test cases.
- From an associated sequence diagram, the inputs and outputs to be created for the tests can be identified.

Dr. Tom Way

CSC 4181

Slide 27

Performance testing

- Part of release testing may involve testing the emergent properties of a system, such as performance and reliability.
- Performance tests usually involve planning a series of tests where the load is steadily increased until the system performance becomes unacceptable.

Dr. Tom Way

CSC 4181

Slide 28

Stress testing

- Exercises the system beyond its maximum design load. Stressing the system often causes defects to come to light.
- Stressing the system test failure behaviour.. Systems should not fail catastrophically. Stress testing checks for unacceptable loss of service or data.
- Stress testing is particularly relevant to distributed systems that can exhibit severe degradation as a network becomes overloaded.

Dr. Tom Way

CSC 4181

Slide 29

Component testing

- Component or unit testing is the process of testing individual components in isolation.
- It is a defect testing process.
- Components may be:
 - Individual functions or methods within an object;
 - Object classes with several attributes and methods;
 - Composite components with defined interfaces used to access their functionality.

Dr. Tom Way

CSC 4181

Slide 30

Interface testing

- Objectives are to detect faults due to interface errors or invalid assumptions about interfaces.
- Particularly important for object-oriented development as objects are defined by their interfaces.

Requirements based testing

- A general principle of requirements engineering is that requirements should be testable.
- Requirements-based testing is a validation testing technique where you consider each requirement and derive a set of tests for that requirement.

Structural testing

- Sometime called white-box testing.
- Derivation of test cases according to program structure. Knowledge of the program is used to identify additional test cases.
- Objective is to exercise all program statements (not all path combinations).

Path testing

- The objective of path testing is to ensure that the set of test cases is such that each path through the program is executed at least once.
- The starting point for path testing is a program flow graph that shows nodes representing program decisions and arcs representing the flow of control.
- Statements with conditions are therefore nodes in the flow graph.

Test automation

- Testing is an expensive process phase. Testing workbenches provide a range of tools to reduce the time required and total testing costs.
- Systems such as Junit support the automatic execution of tests.
- Most testing workbenches are open systems because testing needs are organisation-specific.
- They are sometimes difficult to integrate with closed design and analysis workbenches.