FACULDADE ENGENHARIA

Departamento de Informática

Systematic Testing

(adapted from lecture notes of the CSCI 3060U - Software Quality Assurance unit, J.S. Bradbury, J.R. Cordy, 2018)

FACULDADE ENGENHARIA

Departamento de Informática

Introduction to Systematic Testing

Outline

- Today we begin a long, deep look at **software testing**
- We begin with:
  - Definitions: what is software testing?
  - Role of specifications
  - Levels of testing: unit, integration, system, acceptance
  - Testing in the life cycle
  - Test design and strategy
  - Test plans and procedures
  - Test results
  - Introduction to testing methods: black box & white box

What is Testing?

Testing

- Testing is the process of executing software in a controlled manner, to answer the question:

"does the software behave as specified?"

- Implies that we have a specification (or possibly the tests are it)
- (or) Implies that we have some property we wish to test for independently of the specification
  - e.g., *"all paths in the code are reachable (no dead code)"*
- Testing is often associated with the words validation and verification

FACULDADE ENGENHARIA

Departamento de Informática

Verification vs. Validation

Verification

▪Verification is the checking or testing of software (or anything else) for conformance and consistency with a given specification – answers the question:

"Are we doing the job right?"

Validation

▪Validation is the process of checking that what has been specified is what the user actually wanted – answers the question:

"Are we doing the right job?"

FACULDADE ENGENHARIA

Departamento de Informática

Verification vs. Validation

Verification

- Testing is most useful in verification – checking software (or anything else) for conformance and consistency with a given specification,
- However, testing is just one part of it analysis – inspection and measurement are also important

Validation

- Checking that what has been specified is what the user actually wanted usually involves meetings, reviews and discussions
- Testing is *less* useful in validation, although it can have a role

FACULDADE ENGENHARIA
Departamento de Informática

Testing vs. Debugging

Debugging is not Testing!

Debugging:
the process of analyzing and locating bugs when the software does not behave as expected.

Testing:
the process of methodically searching for and exposing bugs (not just fixing those that happen to show up) – much more comprehensive.

- Debugging therefore supports testing, but cannot replace it
- However, no amount of testing* is guaranteed to find all bugs

* except possibly exhaustive testing (where practical)

FACULDADE ENGENHARIA
Departamento de Informática

What is Systematic Testing?

Systematic Testing

- Systematic testing is an explicit discipline or procedure (a system) for:
  - choosing and creating test cases
  - executing the tests and documenting the results
  - evaluating the results, possibly automatically
  - deciding when we are done (enough testing)
- Because in general it is impossible to ever test completely, systematic methods choose a particular point of view, and test only from that point of view (the test criterion)
  - e.g., test only that every decision (if statement) can be executed either way

FACULDADE ENGENHARIA

Departamento de Informática

The Role of Specification

The Need for Specification

- Validation and verification activities, such as testing, cannot be meaningful unless we have a specification for the software
- The software we are building could be a single module or class, or could be an entire system
- Depending on the size of the project and the development methods, specifications can range from a single page to a complex hierarchy of interrelated documents

Departamento de
Informática

Levels of Specifications

- There are usually at least **three levels** of software specification documents in large systems:
  1. Functional specifications (or requirements) give a precise description of the required behaviour of the system – *what the software should do, not how it should do it – may also describe constraints on how this can be achieved*
  2. Design specifications describe the architecture of the design to implement the functional specification – *the componentes of the software and how they are to relate to one another*
  3. Detailed design specifications describe how each componente of the architecture is to be implemented – *down to the individual code units*

FACULDADE ENGENHARIA

Departamento de Informática

Levels of Testing

- Given the hierarchy of specifications, it is usual to structure testing into **three** (or more) corresponding levels:

3. Unit testing addresses the verification that individual components meet their detailed design specification
2. Integration testing verifies that the groups of units corresponding to architectural elements of the design specification can be integrated to work as a whole
1. System testing verifies that the integrated total product has the functionality specified in the functional specification

- To these levels it is usual to add the additional test level:

0. Acceptance testing, in which the actual customers validate that the software meets their real intentions as well as what has been functionally specified, and accept the result

FACULDADE ENGENHARIA

Departamento de Informática

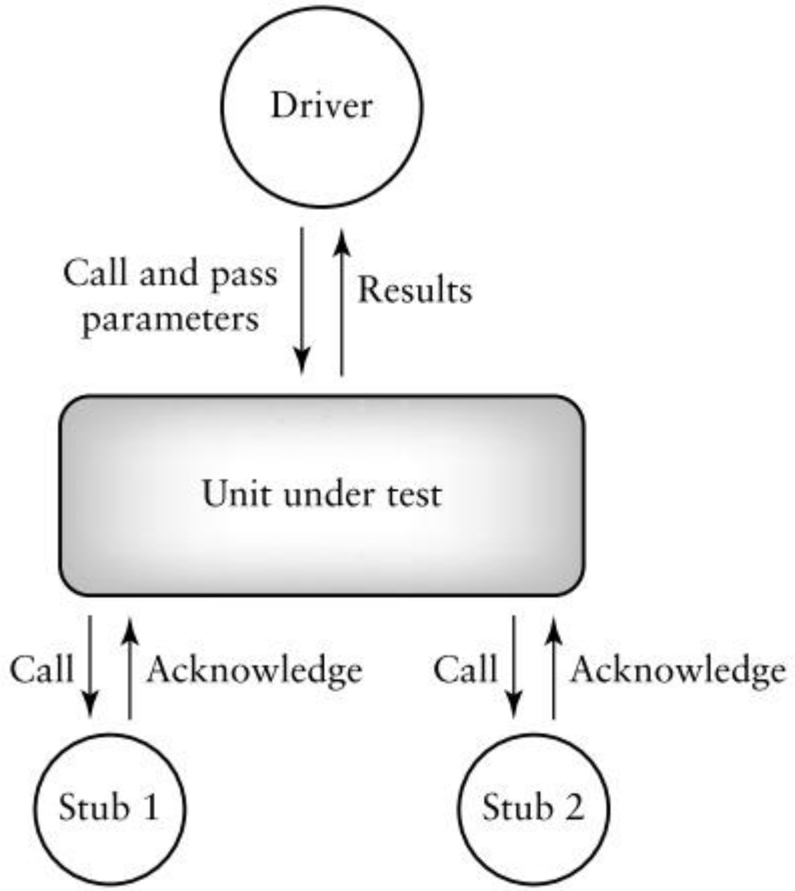
Unit Testing

- Unit testing is a level of software testing where individual units/ components of a software are tested.
- The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.
- In procedural programming, a unit may be an individual program, function, procedure, etc.
- In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module).
- Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing (e.g. White Box testing methods can be used).

FACULDADE ENGENHARIA

Departamento de Informática

Unit Testing



Unit Testing

When a unit fails a test there may be several reasons for the failure. The most likely reason for the failure is a fault in the unit implementation (**the code**).

Other likely causes that need to be carefully investigated by the tester are the following:

- a fault in the test case **specification** (the input or the output was not specified correctly);
- a fault in test procedure **execution** (the test should be rerun);
- a fault in the test **environment** (perhaps a database was not set up properly);
- a fault in the unit **design** (the code correctly adheres to the design specification, but the latter is incorrect).

FACULDADE ENGENHARIA

Departamento de Informática

Unit Testing - Advantages

- Unit testing increases confidence in changing/ maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.

- Codes are more reusable. In order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.

- Unit testing are more reliable than ‘developer tests’.

- The effort required to find and fix defects found during unit testing is very less in comparison to the effort required to fix defects found during system testing or acceptance testing.

- The cost (time, effort, destruction, humiliation) of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels.

- Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.

FACULDADE ENGENHARIA
Departamento de Informática

Integration Testing

- Integration testing is a level of software testing where individual units/ are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in this level of testing (e.g. any Black Box, White Box, or Gray Box Testing methods can be used).

FACULDADE ENGENHARIA

Departamento de Informática

Integration Testing - Approaches

- Big Bang is an approach where all or most of the units are combined together and tested at one go. This approach is taken when the testing team receives the entire software in a bundle. The difference between Big Bang Integration Testing and System Testing is that the first one tests only the interactions between the units while the latter tests the entire system.

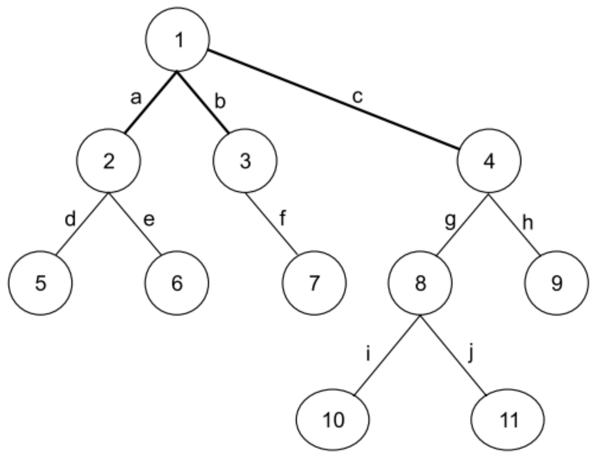
- Top Down is an approach where top-level units are tested first and lower level units are tested step by step after that. This approach is taken when top-down development approach is followed. Test Stubs are needed to simulate lower level units which may not be available during the initial phases.

- Bottom Up is an approach where bottom level units are tested first and upper-level units step by step after that. This approach is taken when bottom-up development approach is followed. Test Drivers are needed to simulate higher level units which may not be available during the initial phases.

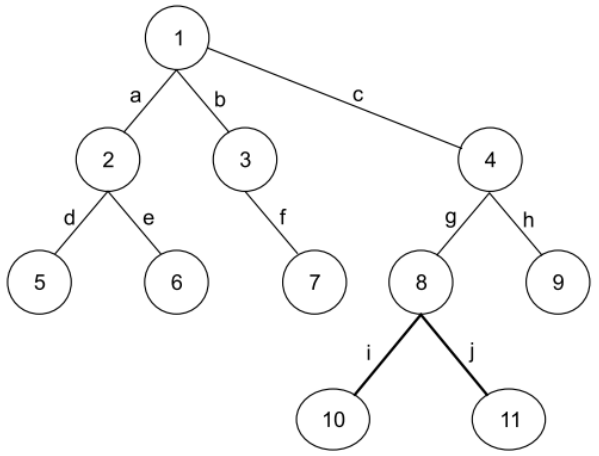
- Sandwich/Hybrid is an approach which is a combination of Top Down and Bottom Up approaches.

FACULDADE ENGENHARIA
Departamento de Informática

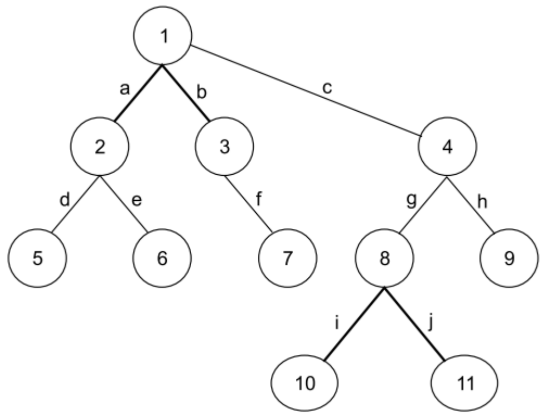
Integration Testing - Approaches



Top down integration (focus starts from edges a, b, c and so on)



Bottom up integration (focus starts from edges i, j and so on)



Sandwich integration (focus starts from a, b, i, j and so on)

FACULDADE ENGENHARIA

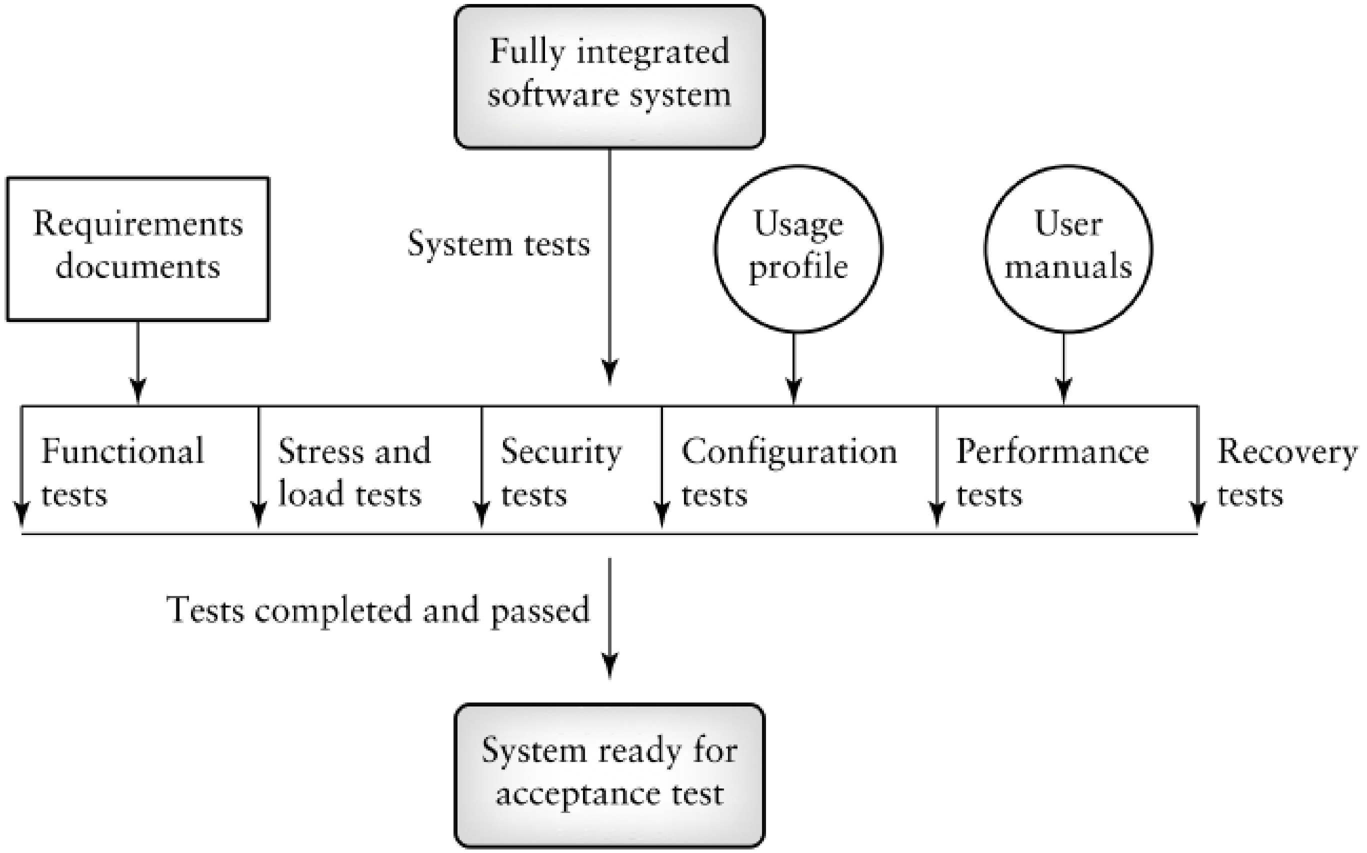
Departamento de Informática

System Testing

- System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements (e.g. Black Box testing methods can be used).

Departa
Informa

System Testing - Example



FACULDADE ENGENHARIA

Departamento de Informática

System Testing – Meet the Requirements

- Functional tests at the system level are used to ensure that the behaviour of the system adheres to the requirements specification. All functional requirements for the system must be achievable by the system.
- Functional tests are black box in nature. The focus is on the inputs and proper outputs for each function.
- This is the only phase of testing which tests both functional and non-functional requirements of the system (e.g. usability, performance, security, reliability, stress and load, …).

Departamento de
Informática

Acceptance Testing

- System testing is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery (e.g. Black Box testing methods can be used).

FACULDADE ENGENHARIA
Departamento de Informática

Acceptance Testing

- Internal Acceptance Testing (aka *Alpha Testing*) is performed by members of the organization that developed the software but who are not directly involved in the project (Development or Testing). Usually, it is the members of Product Management, Sales and/or Customer Support.

- External Acceptance Testing is performed by people who are not employees of the organization that developed the software.
  - Customer Acceptance Testing is performed by the customers of the organization that developed the software. They are the ones who asked the organization to develop the software. [This is in the case of the software not being owned by the organization that developed it.]
  - User Acceptance Testing (aka *Beta Testing*) is performed by the end users of the software. They can be the customers themselves or the customers' customers.

FACULDADE ENGENHARIA
Departamento de Informática

Tests as Goals

An Integral Task

- Once each level of specification is written, the next step is to write the tests for that level
(XP speeds this by making the tests themselves the specification)
- It is important that the tests be designed without knowledge of the software implementation (e.g., in XP, before implementation)
- Otherwise we are tempted to simply test the software for what it actually does, not what it should do

Using Tests

Evaluating Tests

▪Within each level of testing, once the tests have been applied, test results are evaluated

▪If a problem is encountered, then either:

a) the tests are revised and applied again, if the tests are wrong, or

b) the software is fixed and the tests are applied again, if the software is wrong

▪In either case, the tests are applied again, and so on, until no more problems are found.

▪Only when no more problems are found does development proceed to the next level of testing

FACULDADE ENGENHARIA

Departamento de Informática

Test Evolution

Tests Don’t Die!

- As we have already seen with XP, testing does not end when the software is accepted by the customer
- Tests must be repeated, modified and extended to insure that no existing functionality has been broken, and that any new functionality is implemented according to the revised specifications and design
- Maintenance of the tests for a system is a major part of the effort to maintain and evolve a software system while retaining a high level of quality
- In order to make this continual testing practical, automation plays a large role in software testing methods

Departamento de Informática

Testing in the Software Life Cycle

Kinds of Tests

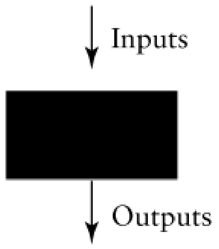
- Testing has a role at every stage of the software life cycle
- As we have seen, tests play a role in:
  - the development of code (unit testing),
  - the integration of the units into subsystems (integration testing) and
  - the acceptance of the first version of the software system (system testing)

Departamento de Informática

Testing in the Software Life Cycle

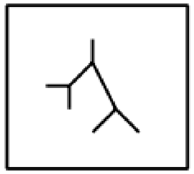
Kinds of Tests

- We will divide these tests into:



Black Box Testing Methods

- Black box methods – cannot see the software code (it may not exist yet!) – can only base their tests on the requirements or specifications



White Box Testing Methods

- White box (aka glass box) methods – can see the software's code – can base their tests on the software's actual architecture or code

FACULDADE ENGENHARIA

Departamento de Informática

Testing in the Software Life Cycle

Regression Tests

- In addition, as the system is maintained, other kinds of tests based on past behaviour come into play
- Once a system is stable and in production, we build and maintain a set of regression tests to insure that when a change is made the existing behaviour has not been broken
- These often consist of a set of actual observed production inputs and their archived outputs from past versions of the system

FACULDADE ENGENHARIA

Departamento de Informática

Testing in the Software Life Cycle

Failure Tests

- As failures are discovered and fixed, we also maintain a set of failure tests to insure that we have really fixed the observed failures, and to make sure that we don’t cause them again
- These consist of a set of actual observed inputs that caused the failures and their archived outputs after the system is fixed

FACULDADE ENGENHARIA

Departamento de Informática

Test Design

Design of Tests

- The design of tests for a system is a difficult and tricky engineering problem as important as design of the software itself
- The design of effective tests requires a set of stages from an initial high level test strategy down to detailed test procedures
- Typical test design stages are:
  - test strategy, test planning, test case design, test procedure

FACULDADE ENGENHARIA

Departamento de Informática

Test Design

(1) Test Strategy

- ▪A test strategy is a statement of the overall approach to testing for a software development organization
- ▪Specifies the levels of testing to be done as well as the methods, techniques and tools to be used
- ▪Part of the project's overall quality plan, to be followed and reported by all members of the project

FACULDADE ENGENHARIA
Departamento de Informática

Test Strategy Examples

Big Bang Testing Strategies

- Test the entire software once it is complete.

Incremental Testing Strategies

- Test the software in phases (unit testing, integration testing, system testing)
- This testing strategy is what we use in XP
- Incremental testing can occur bottom-up (using drivers) or top-down (using stubs)
- In general bottom-up is easier to perform but means the whole program behavior is observed at a later stage of development

FACULDADE ENGENHARIA

Departamento de Informática

Test Strategy Examples

Big Bang vs. Incremental

- Big bang testing only works with a very small and simple program
- In general, incremental testing has several advantages:
  - Error identification ✓
    - Easier to identify more errors
  - Error correction ✓
    - Simpler and requires less resources

Test Design

(2) Test Plans

- A test plan for a development project specifies in detail how the test strategy will be carried out for the project
- In particular, it specifies:
  - the items to be tested
  - the level they will be tested at
  - the order they will be tested in
  - the test environment
- May be project wide, or may be structured into separate plans for unit, integration, system and acceptance testing

Departamento de Informática

Test Design

(3) Test Case Design

- Once we have a plan, we need to specify a set of test cases for each item to be tested at each level
- Each test case specifies how the implementation of a particular functional requirement or design unit is to be tested and how we will know if the test is successful
- It is important to include test cases to test both that the software does what it should (positive testing) and that it doesn't do what it shouldn't (negative testing)
- Test cases are specified separately at each level: unit, integration, system and acceptance – and their documentation forms a test specification for the level

FACULDADE ENGENHARIA

Departamento de Informática

Test Design

(4) Test Procedures

- The final stage of test design is the test procedure, which specifies the process for conducting test cases
- For each item or set of items to be tested at each level of testing, the test procedure specifies the process to be followed in running and evaluating the test cases for the item
- Often this includes the use of test harnesses (programs written solely to exercise the software or parts of it on the test cases), test scripts (automated procedures for running sets of test cases), or commercial testing tools

FACULDADE ENGENHARIA
Departamento de Informática

Test Reports

Documenting Test Results

- Output of test execution should be saved in a test results file, and summarized in a readable report
- Test reports should be designed to be concise, easy to read and to clearly point out failures or unexpectedly changed results
- Test result files should be saved in a standardized form, for easy comparison with future test executions

Systematic Testing Methods

Systematic Methods Recap

- Recall that to be a systematic test method, we must have
  - a system (rule) for creating tests
  - a measure of completeness
- Need an easy, systematic way to create test cases
  (to know for sure what to test)
- Need an easy, systematic way to run tests
  (to know how to test)
- Need an easy, systematic way to decide when we're done
  (to know when we have enough tests)

Summary

- Testing addresses primarily the verification that software meets it specifications – without some kind of specification, we cannot test
- Testing is done at several levels, corresponding the levels of functional, design, and detailed design specifications in reverse order
- Testing remains for the life of the software system
- Testing is not just a one time task, it is a continuous process that lasts throughout the software life cycle
- Effective testing requires careful engineering, similar and parallel to the process for design and implementation of the software itself
- An overall test strategy drives test plans, test case design, and test procedures for a project
- Two classes of systematic test methods, black box and white box

WHY
WHERE
WHEN
WHY
WHAT
WHEN
WHAT
HOW
WHO
WHERE