Software Systems Verification and Validation

Lecture 04 - Levels of testing

Lect. dr. Andreea Vescan

Babeș-Bolyai University
Cluj-Napoca

2015-2016

Testing

- Testing is the process of executing a program with the intent of finding errors. [Mye04]
- Testing -fundamental questions
  - What do we test? What is our goal?
    $\Rightarrow$ Find bugs!
  - How do we organize the process of testing?
    $\Rightarrow$ Testing strategy problem!
  - When we have tested enough?
    $\Rightarrow$ Testing measurement problem!
  - Exhaustive testing?
    $\Rightarrow$ Input domain subset selection problem!
  - Testing strategies?
    $\Rightarrow$ Different testing techniques are appropriate at different points in time!

Test case design

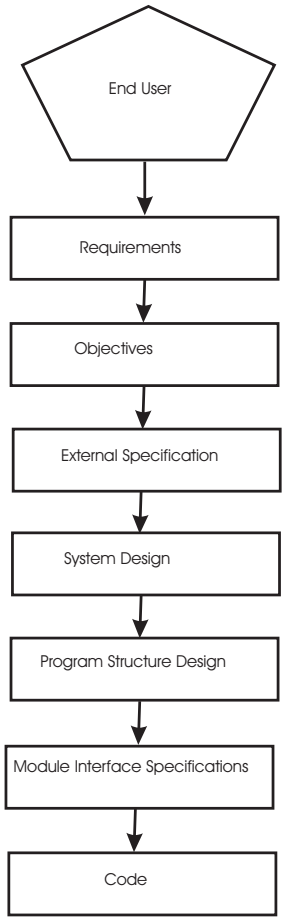
- Black-box testing $\Rightarrow$ software requirements

Test case design

- Black-box testing ⇒ software requirements
- White-box testing ⇒ internal program logic

Software development process

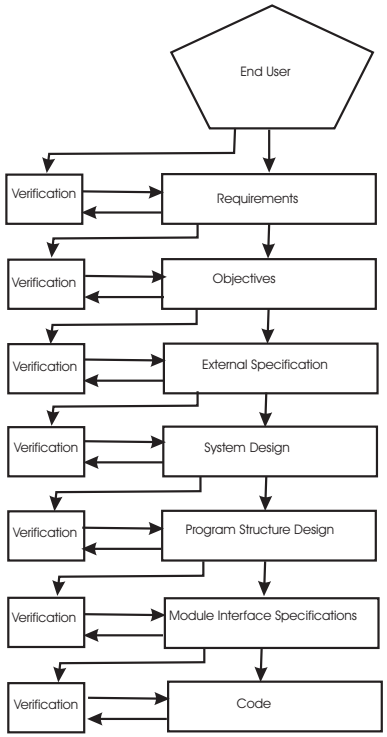
- user's needs are translated into requirements;
- requirements are translated into objectives;
- objectives are translated into external specification;
- system design;
- program structure design;
- module interface specification;
- code.



Development and testing processes

Approaches to prevent errors:

- More precision into the development process.
- Introduction of a verification step at the end of each process.

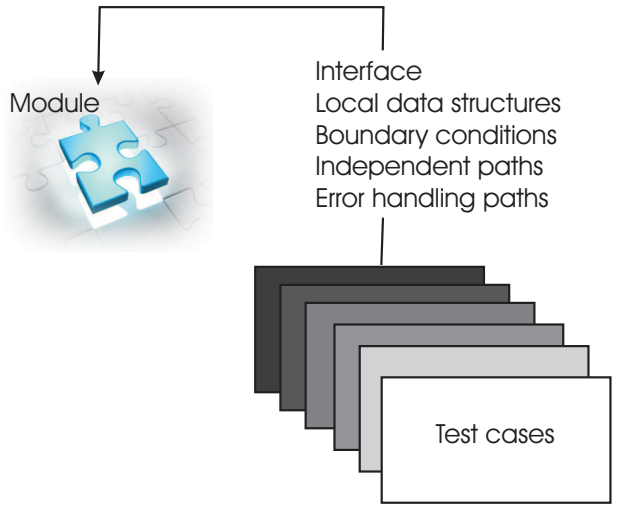


Unit testing

1. Testing individual subprograms, subroutines, procedures, the smaller building blocks of the program.
2. Motivations:
   - Managing the combined elements of testing.
   - Module testing eases the task of debugging.
   - Module testing introduces parallelism into the program testing process.
3. Points of view of model testing
   - The manner in which test cases are designed.
   - The order in which modules should be tested and integrated.
   - Advice about performing the test.
4. References: [Mye04] (chapter 5),[NT05] (chapter 3).

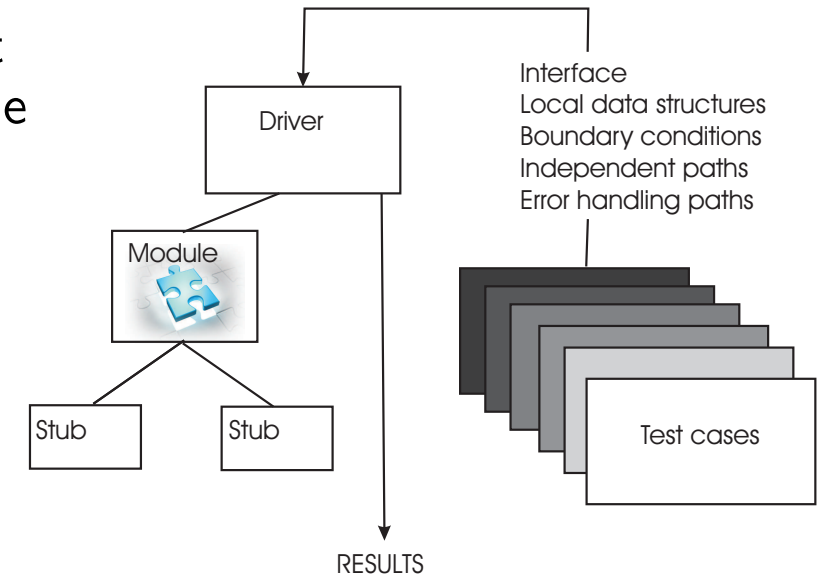
Test case design

- Information needed when designing test cases for a module:
  - specification of the module;
  - the module's source code.
- Test case design procedure for a module test is:
  - Analyze the logic of the module using white-box methods.
  - Applying black-box methods to the module's specification.



Unit test procedures

- Unit test environment
  - driver - a "main program" that accepts test case data, passes such data to the component to be tested and prints relevant results;
  - stub - serve to replace modules that are subordinate the component to be tested.
    - uses the subordinate module's interface
    - may do minimal data manipulation
    - prints verification of entry
    - returns control to the module undergoing testing.



Integration testing

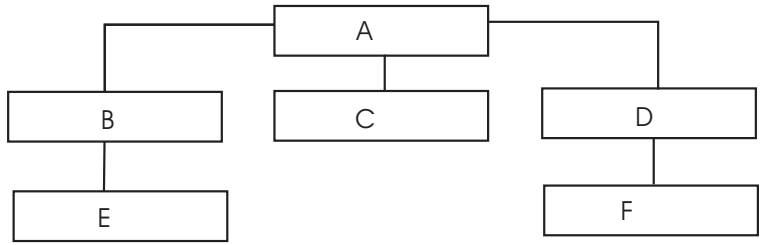
1. Constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing.
2. Importance of integration testing:
   - Different modules are generally created by groups of different developers.
   - Unit testing of individual modules is carried out in a controlled environment by using test drivers and stubs.
   - Some modules are more error prone than other modules.
3. Objectives:
   - putting the modules together in an incremental manner
   - ensuring that the additional modules work as expected without disturbing the functionalities of the modules already put together.
4. Reference: [NT05] (chapter 7).

Integration testing - Techniques

1. Big-bang
2. Incremental
   - Top-down.
   - Bottom-up.
3. Sandwich.

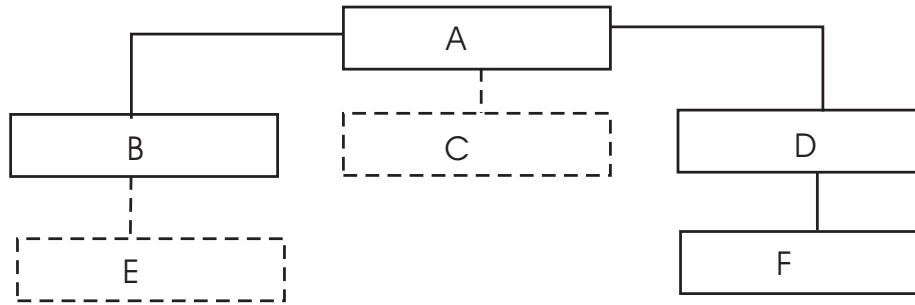
Big-bang testing

- Big-bang procedures:
  - Module test for each individual unit;
    - A driver module;
    - Several stub modules.
  - The modules are combined to form the program.
- Observations
  - more work;
  - mismatching interfaces/incorrect assumptions among modules;
  - debugging;
  - machine time;
  - parallel activities.



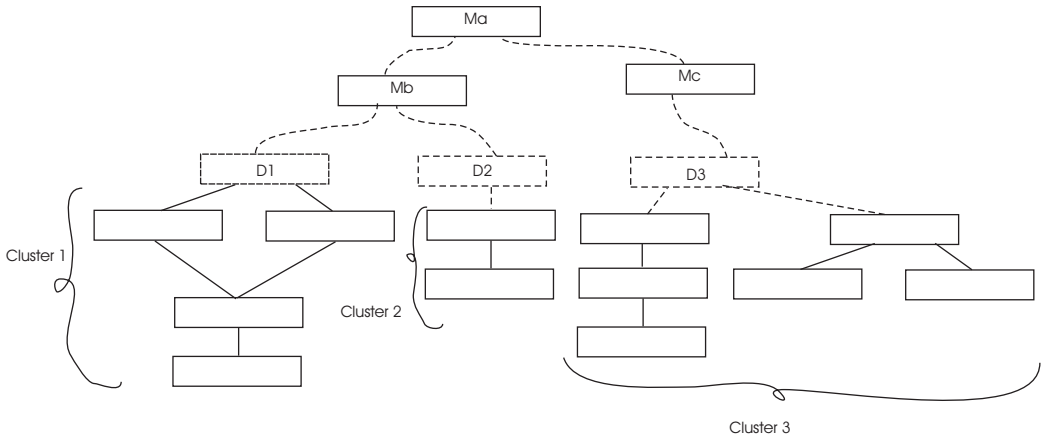
Top-down incremental testing

- Top-down integration manner:
  - Depth-first integration;
  - Breadth-first integration.
- Top-down integration process:
  - main control module = driver; stubs=substituted for all components directly subordinate;
  - subordinates stub $<-$ actual components;
  - tests are conducted as each component is integrated;
  - on completion of each set of tests, another stub $<-$ real component;
  - regression testing may be conducted.



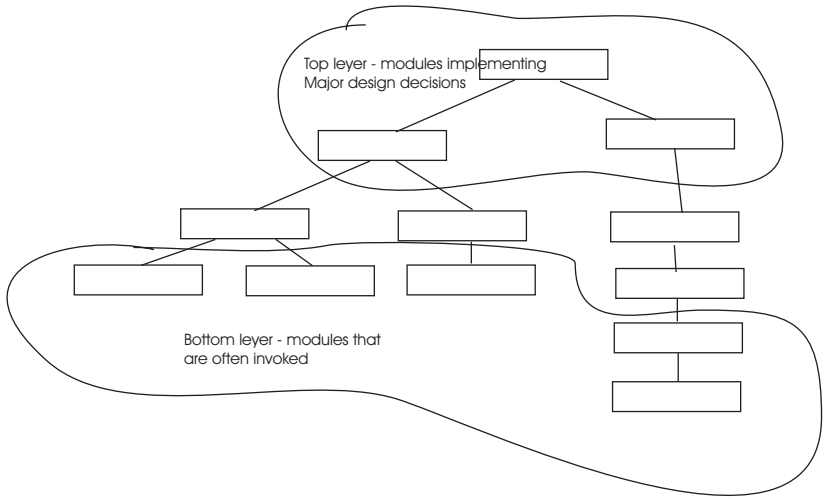
Bottom-up incremental testing

- Bottom-up integration process:
  - low-levels components are combined into clusters;
  - a driver is written to coordinate test case input and output;
  - the cluster is tested;
  - drivers are removed and clusters are combined moving upward in the program structure.



Sandwich testing

- Sandwich procedures:
  - mix of the top-down and bottom-up approaches;
  - layers of a hierarchical system:
    - bottom-layer - using bottom-up module integration;
    - top-layer - using top-down approach integration;
    - middle-layer - big-bang approach.



Regression testing

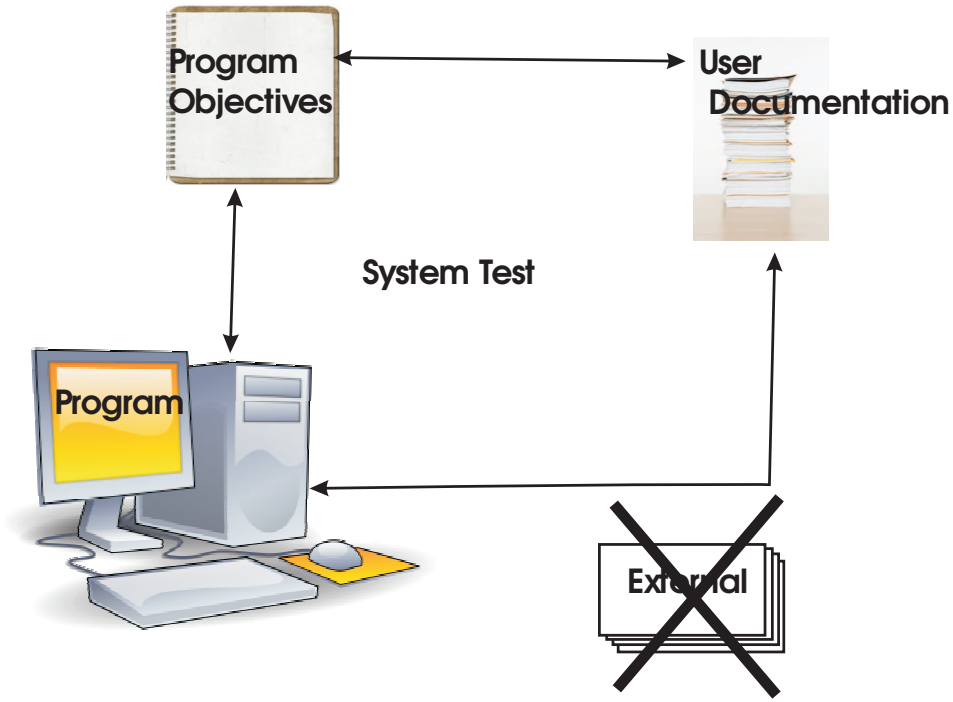
1. The reexecution of some subsets of tests that have already been conducted to ensure that changes have not propagated unintended side effects.
2. Regression test suits - classes of test cases:
   - Tests to exercise all software functions.
   - Tests that focus on software functions that are likely to be affected by the change.
   - Tests that focus on the software components that have been changed.
3. Reference: [PY08] (chapter 22).

Function testing

1. testing requirements described in the external specification of the system;
2. a process of attempting to find discrepancies between the program and the external specification.
3. A black-box activity;
4. Uses system specification.
5. References: [Mye04] (chapter 6), [NT05] (chapter 9), [PY08] (chapter 10).

System testing

- compare the program - original objectives.
- Use external specification? no, e may appear defects during the process of translating the objectives in external specifications;
- Use objectives documents? no, do not contain exact description of the external interfaces of the program;
- Use program's user documentation!
- References:[Mye04] (Chapter 6), [NT05] (chapter 8), [PY08] (chapter 22).



System testing (cont.)

- the objectives does not offer information about the functionality of the system (interfaces of the modules being tested);
- there is no methodology for created test cases in system testing;
- the process of creating test cases:
  - use imagination, creativity and experience.

System testing types

- In [Mye04] (Chapter. 6) there are 15 types of system testing:
  1. Facility testing
  2. Volume testing
  3. Usability testing
  4. Recovery testing
  5. Security testing
  6. Stress testing
  7. Performance testing
  8. Storage testing
  9. Configuration testing
  10. Compatibility testing
  11. Instability testing
  12. Reliability testing
  13. Serviceability testing
  14. Documentation testing

Acceptance testing

1. a process of comparing the program to its initial requirements and the current needs of its end user;
2. not the responsibility of the development organization;
3. the customer first performs an acceptance test to determine whether the product satisfies its needs.
4. References: [NT05] (chapter 14), [PY08] (chapter 22).

Example -unit testing and integration testing

- Problem statement: Compute the number of appearances of the maximum value in an array of natural elements.
- Applied techniques:
  - Test cases using BBT
  - Test cases using WBT
- See example files on SSVV lecture's homepage

Testlink tool

- Test management tools - Testlink. (Release 1.9.5)
- `http://www.scs.ubbcluj.ro/testlink/testlink-1.9.5/logir`
- Test cases using BBT - *BBT_TestPlan*
- Test cases using WBT - *WBT_TestPlan*

Jenkins tool

- Problem (Lectures on BBT and WBT)
- Maven Tutorial
- Jenkins Tutorial
- Jenkins `https://www.scs.ubbcluj.ro:9090/`

Bibliografie I

[Fre10] M. Frentiu.
Verificarea și validarea sistemelor soft.
Presa Universitară Clujeană, 2010.

[Mye04] G. Myers.
The Art of Software Testing, 2nd Edition.
John Wiley, 2004.

[NT05] K. Naik and P. Tripathy.
Software Testing and Quality Assurance.
Wiley Publishing, 2005.

[PY08] M. Pezzand and M. Young.
Software Testing and Analysis: Process, Principles and Techniques.
John Wiley and Sons, 2008.