

Software Systems Verification and Validation

Assoc. Prof. Andreea Vescan

Babeş-Bolyai University
Cluj-Napoca
2020-2021

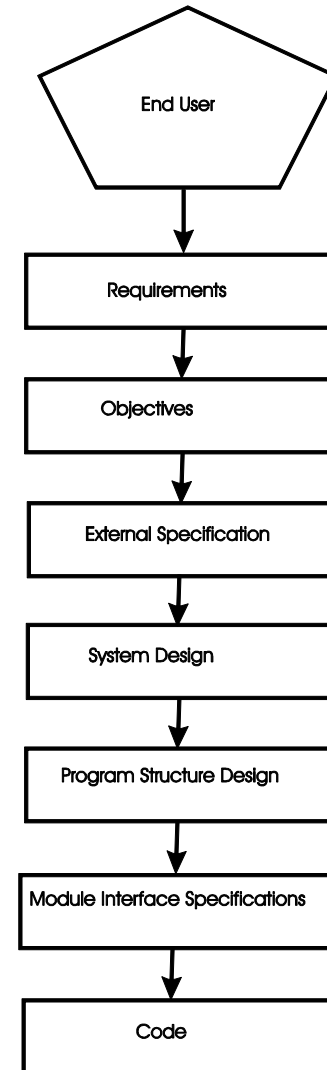
Lecture 4: Levels of testing

Outline

- Software development process
 - Software development process
 - Development and testing processes
- Levels of testing
 - Unit testing
 - Integration testing
 - Function testing
 - System testing
 - Acceptance testing
- Retesting vs regression testing
- Next lecture:
 - **EVOZON Invited Lecture**
- Questions

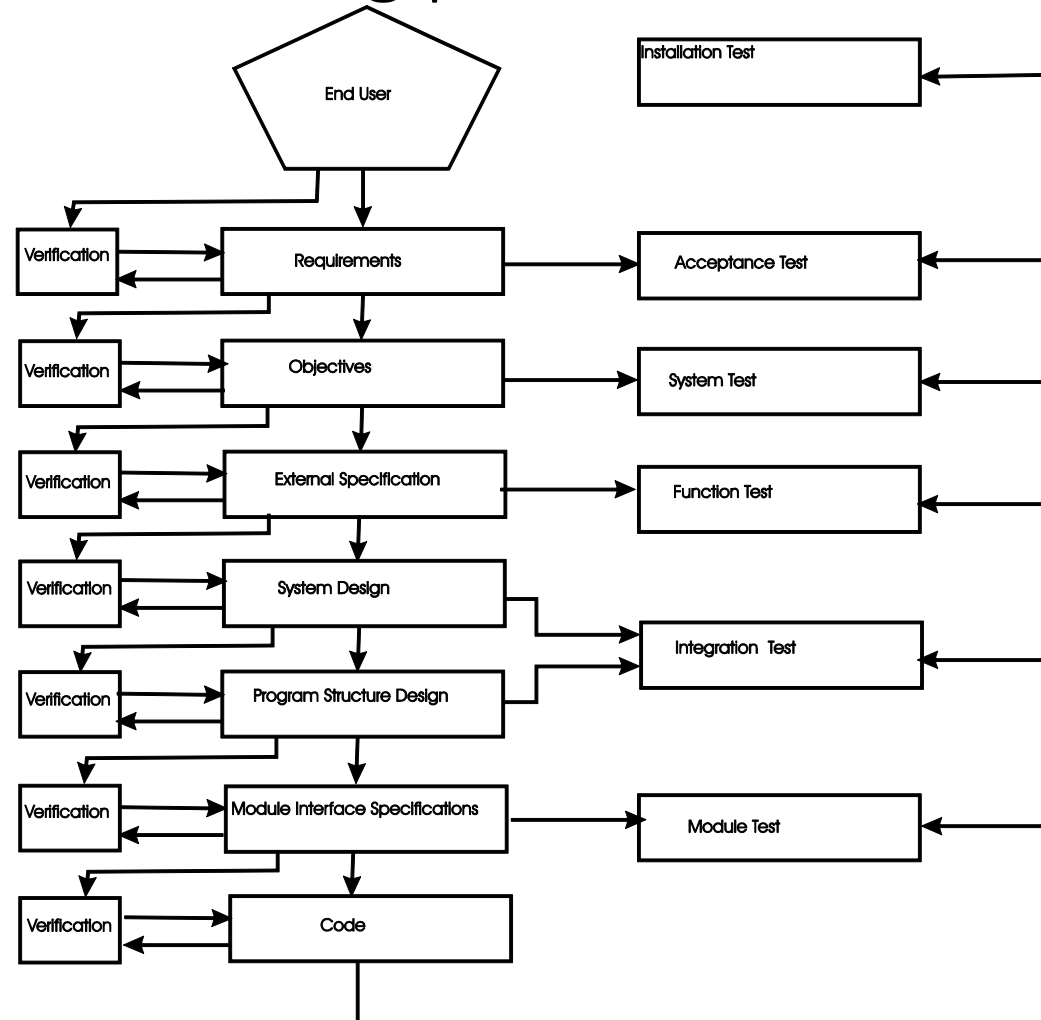
Software development process

- 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.
 - Orient distinct testing processes toward distinct development processes.



Outline

- Software development process
 - Software development process
 - Development and testing processes
- Levels of testing
 - Unit testing
 - Integration testing
 - Function testing
 - System testing
 - Acceptance testing
- Retesting vs regression testing
- Next lecture:
 - Agile testing
- Questions

Levels of testing

1. Unit testing

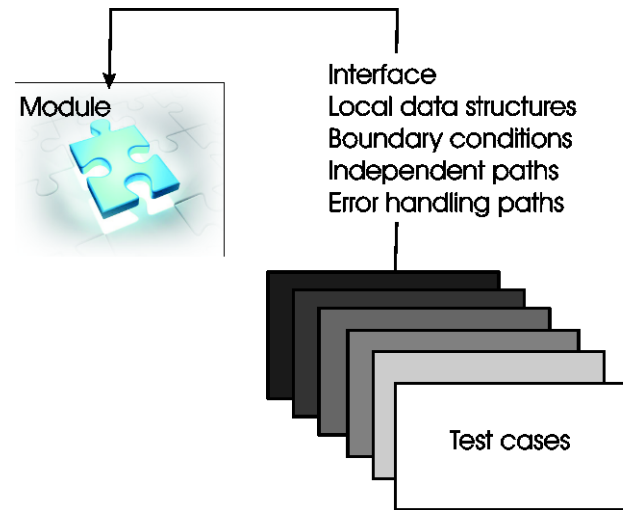
- Testing individual subprograms, subroutines, procedures, the smaller building blocks of the program.
- **Motivations:**
 - Managing the combined elements of testing.
 - Module testing eases the task of debugging.
 - Module testing introduces parallelism into the program testing process.
- **Points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

Levels of testing

1. Unit testing (cont)

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.

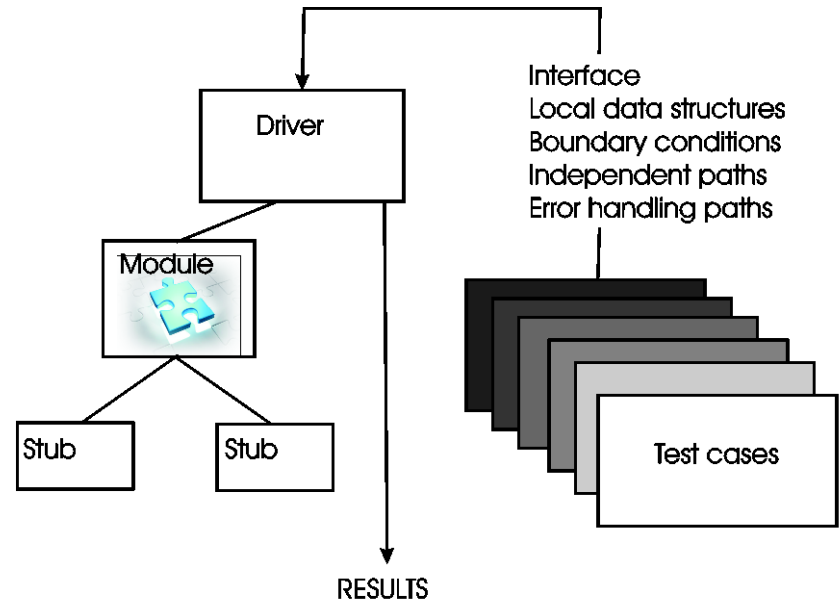


Levels of testing

1. Unit testing (cont)

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.



Levels of testing

2. Integration testing

- Constructing the program structure while at the same time conducting tests to uncover errors associated with *interfacing*.
- **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.
- 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.
- Reference: [NT05] (chapter 7).

Levels of testing

2. Integration testing (cont)

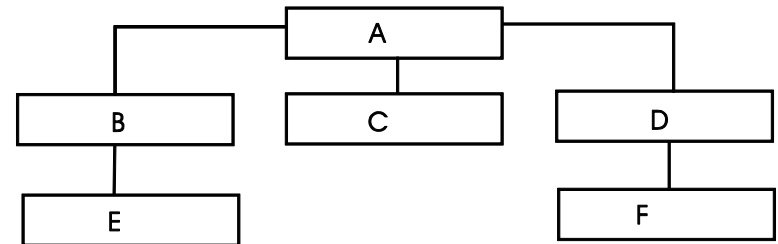
Techniques [Fre10]

- Big-bang
- Incremental
 - Top-down.
 - Bottom-up.
- Sandwich.

Levels of testing

2. Integration testing (cont)

Big-bang testing



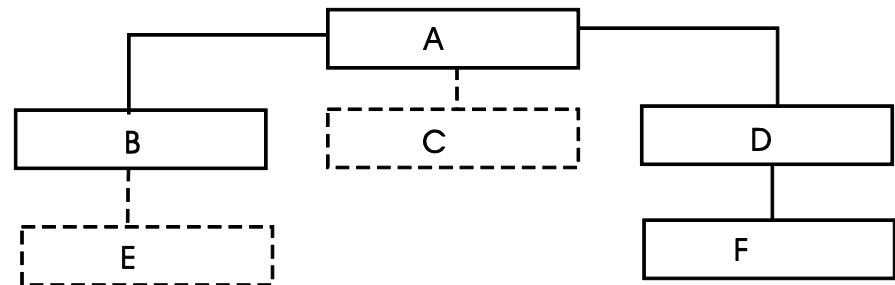
- Big-bang procedures:
 - Module test for each individual unit;
 - A driver module;
 - Several stub modules.
 - The modules are combined to form the program.
- **Observations**
 - more work for big-bang
 - mismatching interfaces/incorrect assumptions among modules - detected earlier with incremental
 - Debugging easier - incremental
 - Big-bang - appears to use less machine time
 - parallel activities – opportunity for big-bang

Levels of testing

2. Integration testing (cont)

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 \leftarrow actual components;
 - tests are conducted as each component is integrated;
 - on completion of each set of tests, another stub \leftarrow real component;
 - regression testing may be conducted.

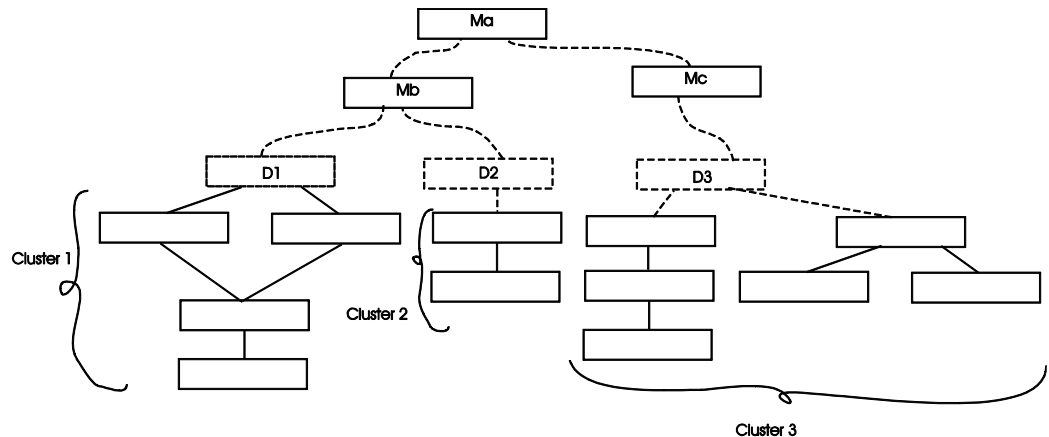


Levels of testing

2. Integration testing

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.

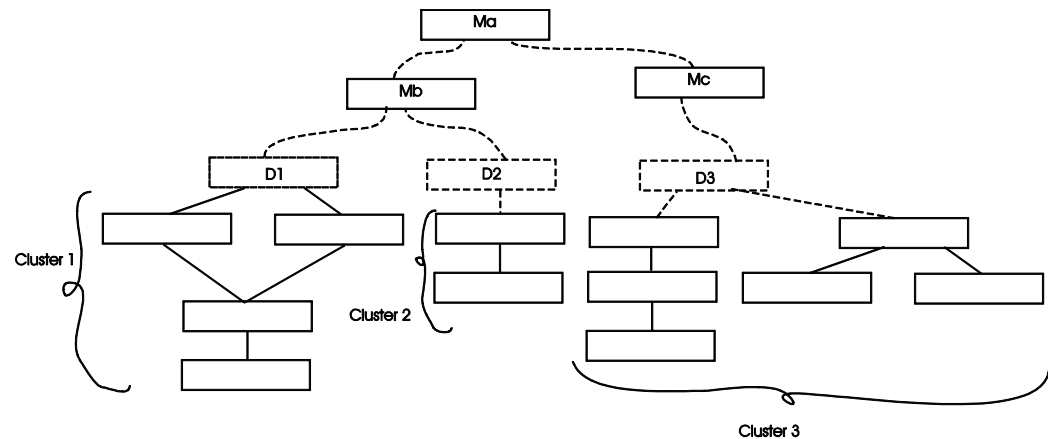


Levels of testing

2. Integration testing

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.



Levels of testing

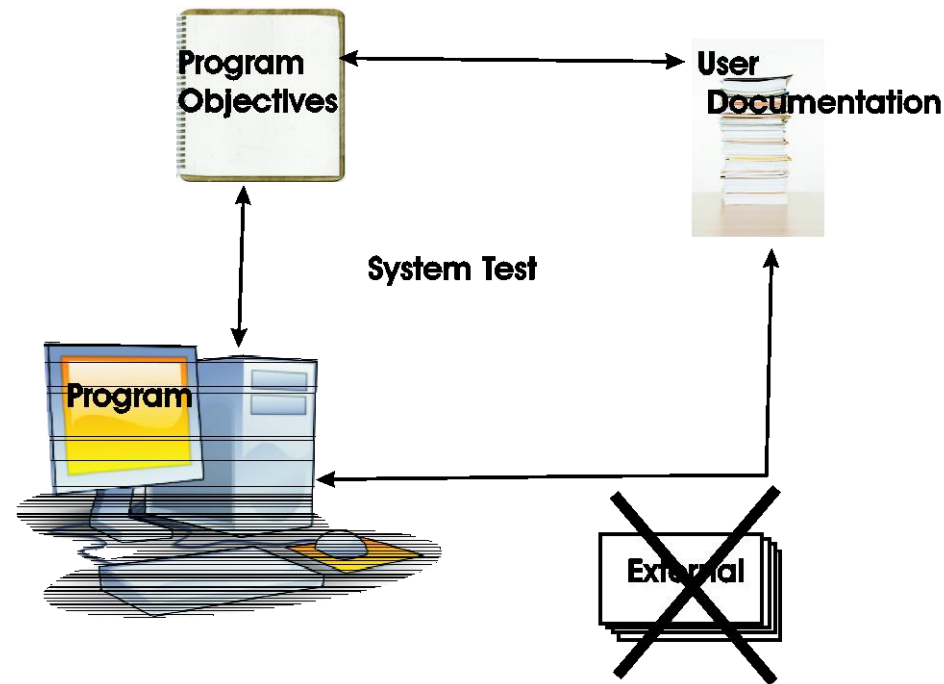
3. Function testing

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

Levels of testing

4. System testing

- compare the program - original objectives.
 - Use external specification? no, 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).



Levels of testing

4. 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
- References:[Mye04] (Chapter 6), [NT05] (chapter 8), [PY08] (chapter 22).

Levels of testing

4. System testing (cont)

System testing types

- In [Mye04] (Chapter. 6) there are 15 types of system testing:
 - Facility testing
 - Volume testing
 - Usability testing
 - Recovery testing
 - **Security testing – Details in next lectures**
 - Stress testing
 - **Performance testing – Details – IT firm EVOZONE – Lecture 5 invitation**
 - Storage testing
 - Configuration testing
 - Compatibility testing
 - Instability testing
 - Reliability testing
 - Serviceability testing
 - Documentation testing
 - Procedure testing

Levels of testing

5. Acceptance testing

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

Outline

- Software development process
 - Software development process
 - Development and testing processes
- Levels of testing
 - Unit testing
 - Integration testing
 - Function testing
 - System testing
 - Acceptance testing
- Retesting vs regression testing
- Next lecture:
 - **EVOZON Invited Lecture**
- Questions

Testing level vs. Testing type

Testing level

- set of activities that are associated to a phase of the software development product

Testing type

- the mean by which an objective of a testing level is achieved

Examples

- Testing a function – unit level or integration level by bbt(domain)/wbt
- Testing of a non-functional characteristic – at system level by performance testing or usability testing
- Testing after eliminating a bug – at any level after debbuging/correcting the bug by appying retesting, confirmation testing
- Testing relating to eliminating a bug – at any level by regression testing to verify if the elimination of the bug doesn't have side-effects

Retesting (confirmation testing)

- Retesting
 - Execution of the test cases that revealed a bug that was reported
 - Goal – confirmation that the bug was eliminated
- Test cases – are the same with those already executed

Regression testing

- Regression testing - **the re-execution of some subsets of tests that have already been conducted** to ensure that changes have not propagated unintended side effects.
- 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.
- Reference: [PY08] (chapter 22).
- Regression testing – **new test cases** (Cem Kaner) [BBST]



Le pont durable



- You can now apply **your values about sustainability** with real results **for you** and **those around you** through your involvement in the "Le pont durable" project!
- I invite you to participate in 4 important events (every 3 hours) about teamwork, sustainability and social entrepreneurship!
- *You will learn* new things about **yourself** and **how to communicate better in the team!**
- *You will integrate* values about **sustainability** and build an **online platform** to help the community uphold these values!
- *You will find* out the opinion of **younger colleagues** about **social entrepreneurship**, solutions offered by them and you will be able to integrate some of them in the online platform.
- Important:
 - **The workshops will be held in Romanian.**
 - **Places are limited.**
 - Please respond to the invitation by Friday, March 19, 2021, 12:00.
- More information about workshops you can discover events online here:
 - **Facebook event:** <https://fb.me/e/1c7KMyJg7>

Surprise!





Having fun learning about testing During Lecture 4 (second hour)

1. Guerilla testing

<https://jamboard.google.com/d/1nSi-nTkc1fGTTZahkZBtb8hgi5ZN9H9Uy-XV1pvSoa0/edit?usp=sharing>

2. Dumb monkey testing

<https://jamboard.google.com/d/1MmmBbUIIKPX8Y41nvwH3yf4aMMbC04-oS92eP1HIVA/edit?usp=sharing>

3. Smoke testing

<https://jamboard.google.com/d/1FSVURrJld147nXcotqH91QptA7MtvFtT2czBZb7JHu0/edit?usp=sharing>

4. Bug bashes (in testing)

<https://jamboard.google.com/d/1aMYnwYLLELttINKi3XQSHzbi0DZMSf-atbAGMUJbOI/edit?usp=sharing>

1 A4 page information

Work in teams

- Definition/description
- Characteristics (5 to 10 points)
- Levels of testing
- Example = simple + real world application
- Interesting fact(s)

You are **NOT ALLOWED** to include “guest names.” Every person listed as a collaborator must contribute.

- **Poster - creating 50 XP**
 - 15 minutes to Create the poster
 - Each Team presents in class in 2 minutes
 - 4teams*2minutes=8 minutes
 - Submitted in Canvas in BonusLecture4 by
 - 20:00 on 17March2021
- **Poster – discussion Debriefing**
 - 10 minutes
 - Learning
 - Individual/Team

Outline

- Software development process
 - Software development process
 - Development and testing processes
- Levels of testing
 - Unit testing
 - Integration testing
 - Function testing
 - System testing
 - Acceptance testing
- Retesting vs regression testing
- Next lecture:
 - **EVOZON Invited Lecture**
- Questions

Questions

- Thank You For Your Attention!

References

- [Pat05] R. Patton. *Software Testing*. Sams Publishing, 2005.
- [PY08] M. Pezzand and M. Young. *Software Testing and Analysis: Process, Principles and Techniques*. John Wiley and Sons, 2008.
- [Mye04] Glenford J. Myers, *The Art of Software Testing*, John Wiley & Sons, Inc., 2004
- [You79] E. Yourdon, *Structured Walkthroughs*, Prentice-Hall, Englewood Cliffs, NJ, 1979
- [NT05] K. Naik and P. Tripathy. *Software Testing and Quality Assurance*. Wiley Publishing, 2005.
- [CB03] Jean-Francois Collard and Ilene Burnstein. *Practical Software Testing*. Springer-Verlag New York, Inc., 2003.
- [Fre10] M. Frentiu, *Verificarea si validarea sistemelor soft*, Presa Universitara Clujeana, 2010
- [BBST] BBST Testing course, <http://testingeducation.org/BBST/>