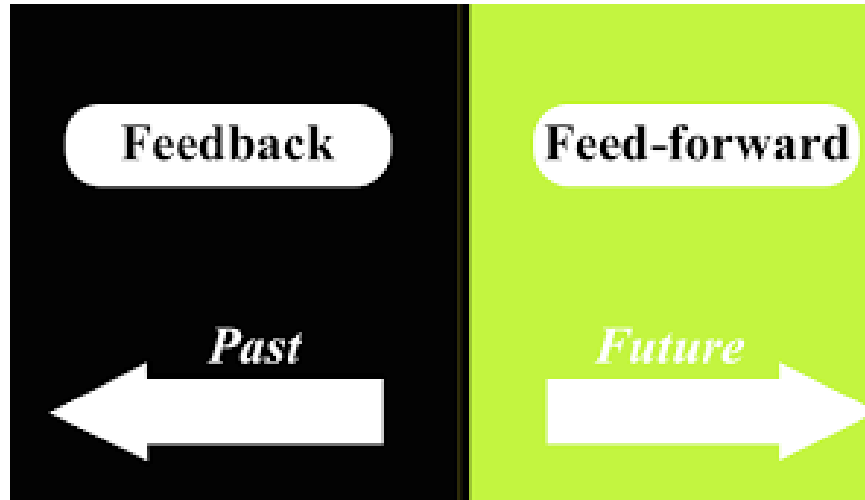# Software Systems Verification and Validation

Assoc. Prof. Andreea Vescan

Babeș-Bolyai University

Cluj-Napoca

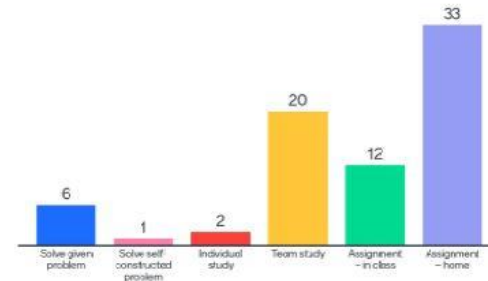2020-2021

Lecture 2a: Testing

Feedforward!

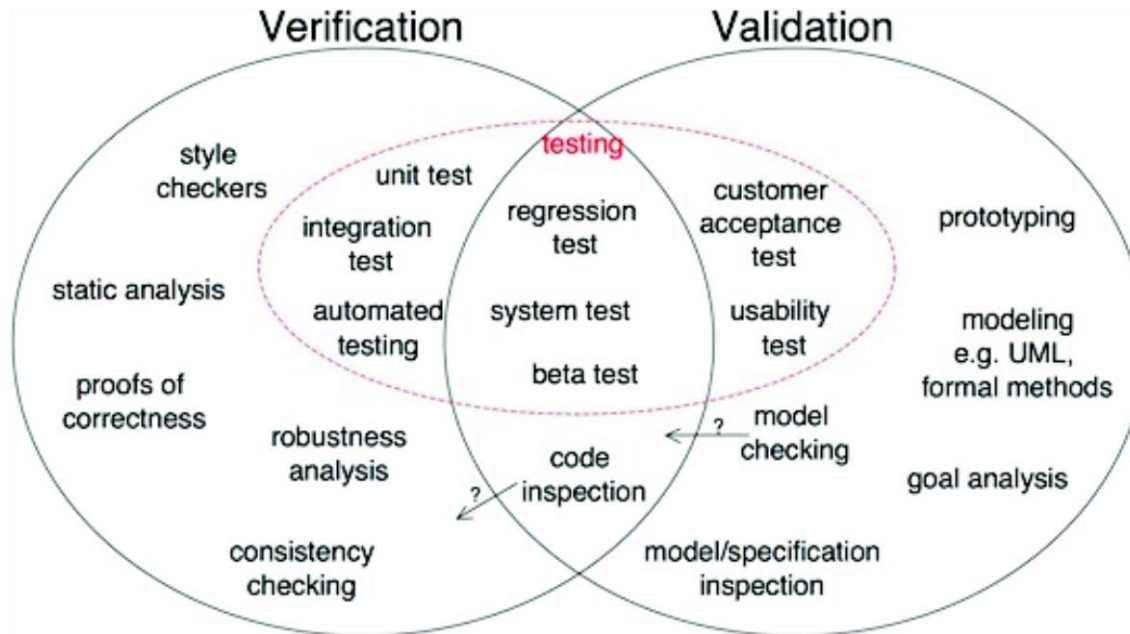

How would you like the Laboratories to be carry out?

Grading - Normal/retake session - % Activities (Lecture,Lab,Sem,Exam)

10% 30% 10% 50%	0%, 40%, 20%, 40%	60% lab, 20%sem, 20%exam
50% lab, 50%exam	60 % Exam	10% 20% 20% 50%
40% lab 20% sem 40% exam +5% lectures	60% Lab, 10% Seminar, 30% Lecture	50 % exam 30 % lab 20 % sem

https://www.youtube.com/watch?v=I_9I3LGS8I8&ab_channel=QuixotKids-Story

What you will learn!

Outline

- Testing – fundamental questions (why, how, when, enough?)
- Software testing
  - Definitions (why)
- Testing strategy (how)
- Test planning (enough)
  - Test cases
  - Testing activities
  - Bug report
- Software testing (Required reading before Seminar 2)
  - Testing principles
  - Testing axioms
- When do we test?

Testing - fundamental questions

- What skills do we need to have to test?

- Why do we test?

- How do we test?

- How do we organize the process of testing?

- When have we tested enough?

- When we test?

What skills do we need to have to test?

Surprise!

Play … detective!

Playing detective (1)

Context 1.

A man went into a party and drank some of the punch. He then left early. Everyone at the party who drunk the punch subsequently died of poisoning. Why did the man not die?

Context 2.

A detective who was mere days from cracking an international smuggling ring has suddenly gone missing. While inspecting his last-known location, you find a note:

710	57735	34	5508	51	7718

Currently there are 3 suspects: Bill, John, and Todd. Can you break the detective's code and find the criminal's name?

Context 3.

There is a man found dead in a circular mansion. The detective interviews the cook, maid, and babysitter. The cook said he couldn't have done it because he was preparing the meal. The maid said she couldn't have done it because she was dusting the corners. The babysitter said she couldn't because she was playing with the children. Who was lying?

TABARA
DE TEST ARE

Playing detective (2)

Can you state (some) similarities
between **a detective** and **a tester**?

Subject of **Investigation**

Detectives

Crime

- the commission of an act that is forbidden
- activity that is against the law
- harmful act against the public

Testers

Computer program (CP)

- CP=Set of instructions for a computer
- CP=A communication among several humans and computers [BBST]

  - Defining a house
    - A set of construction materials put together according to a house-design pattern
    - Is built for people to live in

TABARA DE TESTARE

Playing detective (3)

Can you state (some) similarities
between **a detective** and **a tester**?

Used Tools

Detectives

- writing statements by witnesses/suspects
- writing up reports for superiors and team members
- recording equipment
- taking charge of all the information related to the case (keep track using folders)
- standard evidence kit (gloves, envelopes and/or plastic bags, tape measures, flashlights and scissors)

Testers

- Test plan
- Test management tools
- Bug reports
- Software tools (developer, tester)
- Logs
- Screen capture tools
- Oracles
- Testing techniques

TABARA DE TESTARE

Playing detective (4)

Can you state (some) similarities
between **a detective** and **a tester**?

Job duties

Detectives

- Researching and Analyzing Crimes
- Identifying and Interrogating Suspects
- Testifying in Court

Testers

- Researching for Information
- Interrogating

- Bug advocacy

- **Remark**: They don't even know what crime has been committed yet.

TABARA
DE TESTARE

Why testers are hunters and developers are farmers

- https://qubiz.com/blog/testers-are-hunters-and-developers-are-farmers/
- Tester's skills

IT firm – Altom

Lecture invitation

Testing - fundamental questions

- What skills do we need to have to test?

- Why do we test?

- How do we test?

- How do we organize the process of testing?

- When have we tested enough?

- When we test?

Why do we test?

Software testing - DEFINITION

- **Software testing [BBST]** - is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or service under test
- **False/incomplete definitions**
  - Testing is the process of demonstrating that errors are not present. "Testing can only reveal the presence of errors, never their absence." [Dij75]
  - The purpose of testing is to show that a program performs its intended functions correctly.
  - Testing is the process of establishing confidence that a program does what it is supposed to do.
- **Definition**
  - Testing is the process of executing a program with the intent of finding errors. [Mye04]
    - Human beings tend to be highly goal-oriented.
    - Testing is a destructive process.
    - Successful and unsuccessful.

    - Error: the program does not do what it is supposed to do, BUT also if the program does what it is not supposed to do.

Software testing

Why do we test?

    – We test a product to learn about its quality. [BBST]

- **Everyone tests in context [BBST]**
  - **Harsh constraints**
    - **Complete testing is impossible**
    - **Finite project schedules and budget**
    - **Limited skills of the test group**
  - **Do testing – before/during/after – release**
  - **Improvement of product – might/might not be an objective of testing**
  - **Test in behalf of stakeholders**
    - **Project manager/customer/programmer/attorney**

- **A search for information [BBST]**
  - **Question:** What kind of information are we most concerned about learning from this testing?
  - **Answer:** Our information objectives.

TESTING IS ALWAYS A SEARCH FOR INFORMATION
- Find important bugs
- Assess the quality of the product
- Help managers assess the progress of the project
- Help managers make release decisions
- Block premature product releases
- Help predict and control product support costs
- Check interoperability with other products
- Find safe scenarios for use of the product
- Assess conformance to specifications
- Certify the product meets a particular standard
- Ensure the testing process meets accountability standards
- Minimize the risk of safety-related lawsuits
- Help clients improve product quality & testability
- Help clients improve their processes
- Evaluate the product for a third party

Different objectives require different testing tools and strategies and will yield different tests, test documentation and test results.

Black Box Software Testing   Copyright © 2000-10   Cem Kaner & James Bach   24

Testing ... in context

Context 1.

The teacher asked a group of students (development team) to design and implement a Java application for the given statement problem. The teacher just want to verify the students 's design and implementation skills.

You are working as a tester and you are assigned to the development team early in the project. The project manager (i.e., the teacher) wants you to test along with the programmers as they write code. She wants you to help the programmers deliver clean code and she wants to help her and her staff identify, understand and control the implementation.

Context 2.

The teacher asked a group of students (development team) to design and implement a Java application for the given statement problem. The teacher just want to verify the students 's design and implementation skills.

You are working as a tester and you are assigned to the development team when all the required stated functionalities were implemented and individually tested. The project manager (i.e., the teacher) wants you to test the integration of the implemented functionalities.

Context 3.

The teacher asked a group of students (development team) to design and implement a Java application for the given statement problem. The resulted application will be used by the doctors in their university and by other family doctors.

You are working as a tester and you are assigned to the development team close to the project's release date. The doctors expected it to work and they needed to work well. The project manager (i.e., the teacher) expects you to test the product in ways that will help her understand whether the product is ready to release or not.

TABARA
DE TESTARE

Testing - fundamental questions

- What skills do we need to have to test?

- Why do we test?

- How do we test?

- How do we organize the process of testing?

- When have we tested enough?

- When we test?

How do we test?

Strategy/Approach/Technique

- **Strategy definition** (https://en.oxforddictionaries.com/definition/strategy)
  - A plan of action designed to achieve a long-term or overall aim.
- **Approach definition** (http://www.dictionary.com/browse/approach)
  - the method used or steps taken in setting about a task, problem
- **Technique definition**   (http://www.dictionary.com/browse/approach)
  - technical skill; ability to apply procedures or methods so as to effect a desired result

- **Testing strategy =** guiding framework for deciding what tests (what test techniques) are best suited to your product.
- **Approach = a way of   thinking   about …**
- **Technique = methods used to …**

Testing strategies [BBST]

- **Testing strategy is:**
  - The guiding framework for deciding what tests (what test techniques) are best suited to your product.
- **Context** and **information objectives** are (or should be) the drivers of any **testing strategy.**

- **Examples of context factors that drive and constrain testing**
  - Who are the stakeholders with influence?
  - What are the goals and quality criteria for the project?
  - What skills and resources (time, money, tools, data, etc) are available?
  - Potential consequences of potential failures?
  - How could it fail?

- **Common information objectives**
  - Find important bugs
  - Asses the quality of the product
  - Help managers make release decisions
  - Block premature product releases
  - Assess conformance to the specification
  - Find safe scenarios for the use of the product
  - Evaluate the product for a third party

Strategy/Approach/Technique (revisited)

- **Context** and **information objectives** are (or should be) the drivers of any **testing strategy.**

Selecting the Testing techniques ?

Techniques differ in core.   *Attributes* *of "good" tests*

- Power
- Valid
- Value
- Credible
- Representative
- Non-redundant
- Motivating
- Reusable
- Performable
- Maintainable

- Information value
- Coverage
- Easy to evaluate
- Support troubleshooting
- Appropriately complex
- Accountable
- Affordable
- Opportunity cost

Strategy/Approach/Technique (revisited)

- **Context** and **information objectives** are (or should be) the drivers of any **testing strategy.**

Selecting the Testing techniques ?

http://www.testingeducation.org/BBST/testdesign/

- *http://www.testingeducation.org/BBST/testdesign/BBSTTestDesign2011pfinal.pdf*
- *Slides 67 to 72*
- COVERAGE-BASED TECHNIQUES FOCUS ON WHAT GETS TESTED
- TESTER-BASED TECHNIQUES FOCUS ON WHO DOES THE TESTING
- RISK-BASED TECHNIQUES FOCUS ON POTENTIAL PROBLEMS
- ACTIVITY-BASED TECHNIQUES FOCUS ON HOW YOU DO THE TESTING
- EVALUATION-BASED TECHNIQUES FOCUS ON YOUR ORACLE
- DESIRED-RESULT TECHNIQUES FOCUS ON A SPECIFIC DECISION OR DOCUMENT
- THERE ARE ALSO GLASS-BOX TECHNIQUES

Testing - fundamental questions

- What skills do we need to have to test?

- Why do we test?

- How do we test?

- How do we organize the process of testing?

- When have we tested enough?

- When we test?

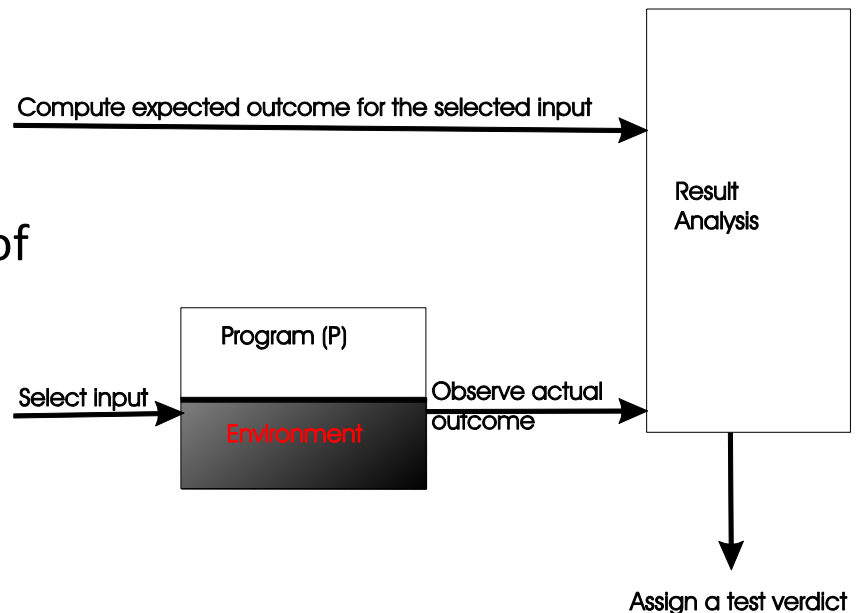# How do we organize the process of testing?

Test planning

- Test plan?
  - testers communicate what they intend to do.
  - takes the form of a written document, not only the creation of the document, but also planning the testing tasks.
  - The ultimate goal of test planning process is communicating the software test team's intend, expectations, understanding.
- Test planning topics
  - A test plan template? Important topics?
    - The test team's high-level expectations
    - People, places and things, Definitions
    - Inter-group responsibilities
    - What will and won't be tested
    - Test phases, test strategy, Resource requirements
    - Tester assignments
    - Test schedule, Test cases
    - Bug reporting

- Reasons for planning test cases
  - Organization
  - Repeatability
  - Tracking
  - Proof of testing

- VerVal matrix

Test case

- Test case < i , r >
  - i domain D
  - r domain R.
  - for input i the expected result is r.
- Test case: " A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement." [IEE90]
- "Good" test case attributes:
  - High probability of finding an error.
  - Is not redundant.
  - "Best of breed".
  - Neither too simple nor too complex.

Testing activities [NT05]

- Identify an objective to be tested.
- Select inputs.
- Compute the expected outcome.
- Set up the execution environment of the program.
- Execute the program.
- Analyze the test result.

Compute expected outcome for the selected input

Result Analysis

Program (P)

Select Input

Environment

Observe actual outcome

Assign a test verdict

Reporting a bug

- Principles for reporting a bug [Pat05]
  - Report bugs as soon as possible
  - Effectively describe the bugs
  - Be nonjudgmental in reporting bugs
  - Follow up on your bug reports
- Isolating and reproducing bugs [Pat05] - suggestions in isolating a bug
  - Don't take anything for granted
  - Look for time-dependent and race condition problems
  - White-box issues of boundary condition bugs, memory leaks, data overflows.
  - State bug
  - Resource dependencies and interactions with memory, network, hardware sharing.
  - Don't ignore the hardware

A bug's Life Cycle [Pat05]
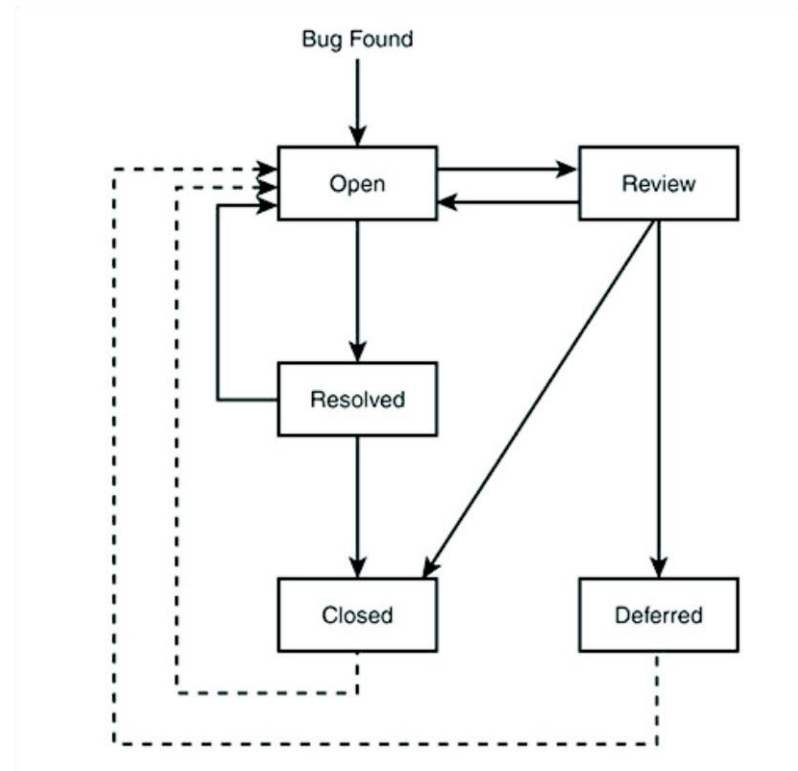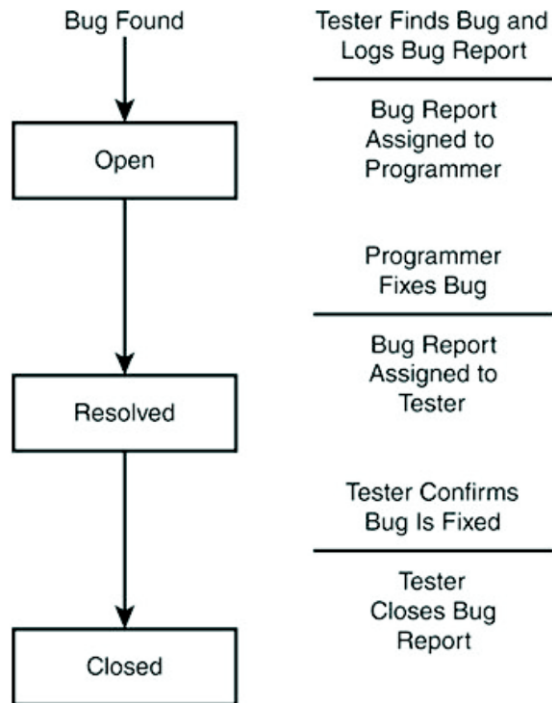
Testing - fundamental questions

- What skills do we need to have to test?

- Why do we test?

- How do we test?

- How do we organize the process of testing?

- When have we tested enough?

- When we test?

When have we tested enough?

Program under test

Types of testing [Fre10]

- Program P [Fre10]

- $P : D \rightarrow R$, where:
  - D - set of input data;
  - R - set of output data.

- Exhaustive testing - all the possible inputs.
  - if D is finite, then P is executed for all possible inputs.
- Selective testing
  - if D is not finite, then we choose inputs i from S (included in D).

- Cem Kaner, The Impossibility of Complete Testing,
 http://www.testingeducation.org/BBST/foundations/Kaner_impossibility.pdf

Required reading (before Seminar 2)

Testing principles [Mye04]

- Definition of the expected output or result;
- Avoid testing your own program.
- A programming organization should not test its own programs.
- Thoroughly inspect the results of each test.
- Test cases for valid/invalid input conditions.
- Test if the program does not do what it is supposed to do, AND if the program does what it is not supposed to do.
- Do not throwaway test cases.
- Plan testing assuming errors will be found.
- The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section.
- Testing is an extremely creative and intellectually challenging task.

- **Remark:** Principles rewritten in [CB03].

Testing axioms [Pat05]

- It is impossible to test a program completely.
- Software testing is a risk-based exercise.
- Testing can't show that bugs don't exist.
- The more bugs you find, the more bugs there are.
- The pesticide paradox.
- Not all the bugs you find will be fixed.
- When a bug's a bug it is difficult to say.
- Product specification are never final.
- Software testers aren't the most popular member of a project team.
- Software testing is a disciplined technical profession.
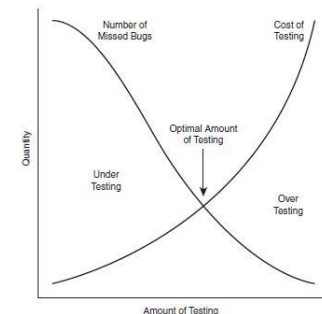
When we test?

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

End User

↓

Requirements

↓

Objectives

↓

External Specification

↓

System Design

↓

Program Structure Design

↓

Module Interface Specifications

↓

Code
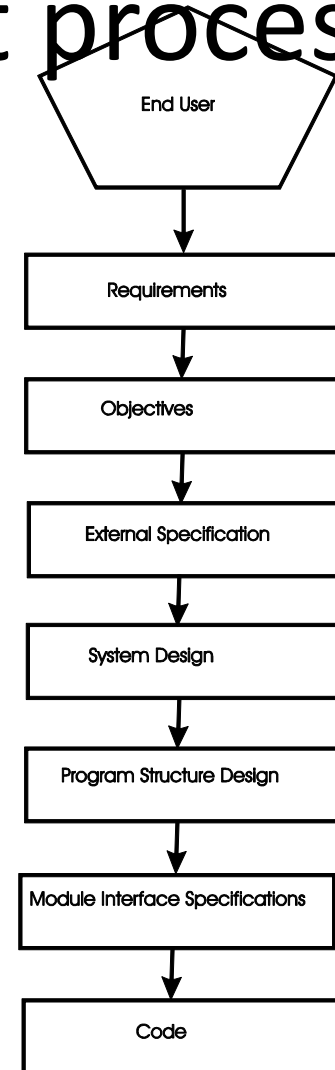
Development and testing processes

- 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.
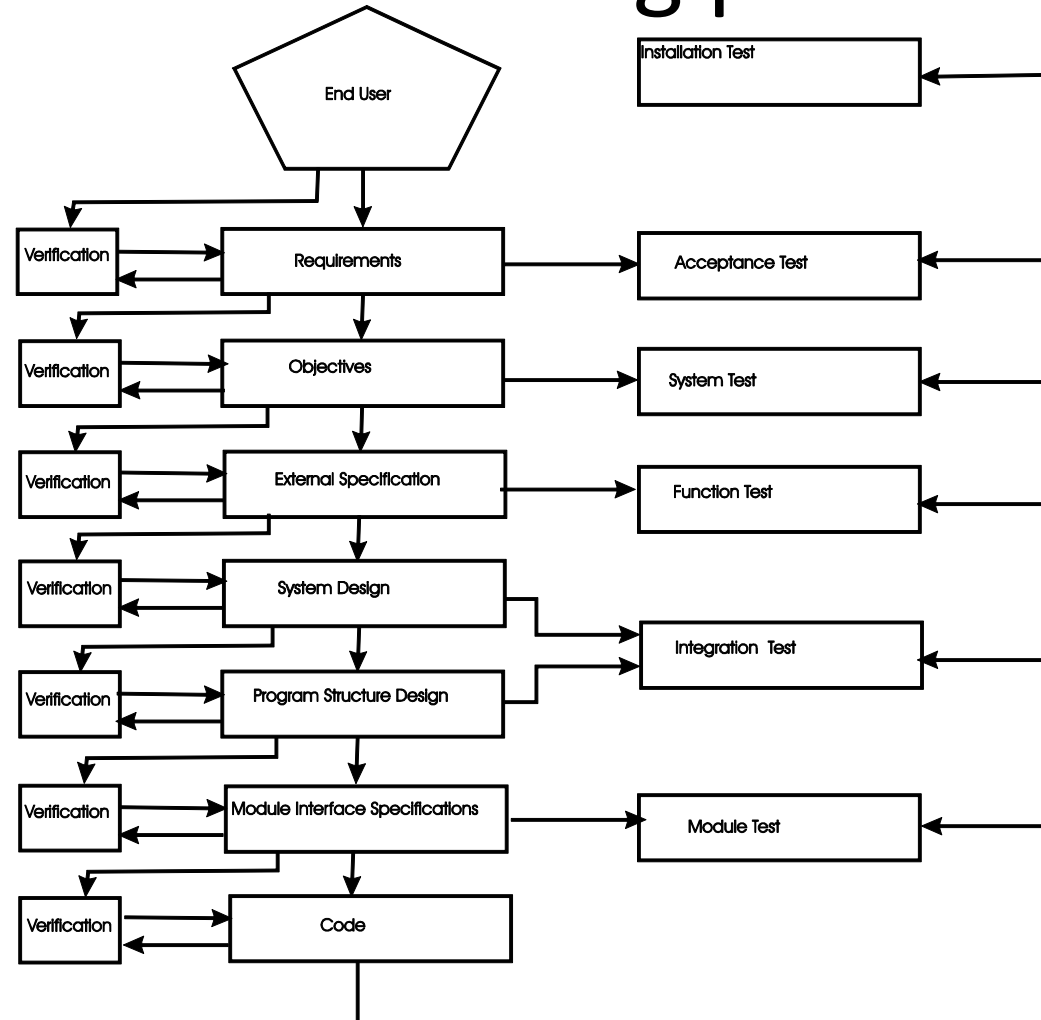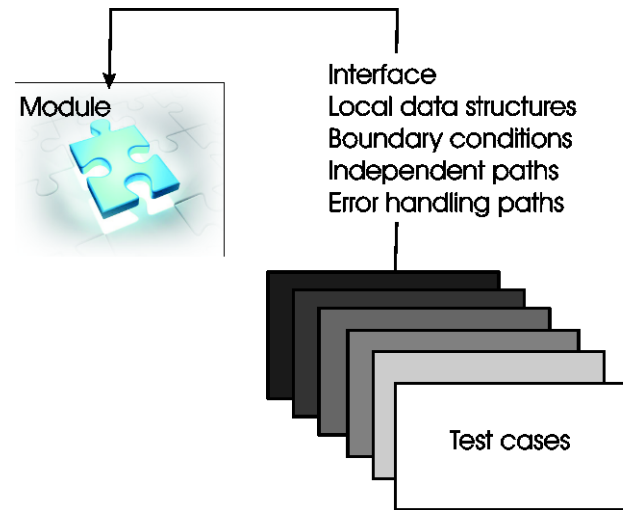
Levels of testing

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

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.
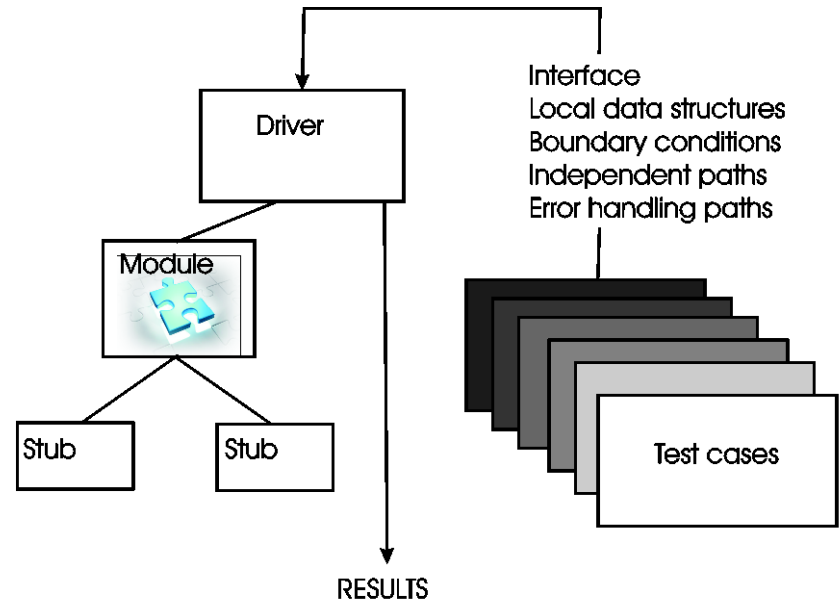 - Applying black-box methods to the module's specification.

Levels of testing
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.

Interface
Local data structures
Boundary conditions
Independent paths
Error handling paths

Driver

Module

Stub   Stub

Test cases

RESULTS

Next Lecture (Still today!)

- Domain testing (black-box testing)

Questions

- Thank You For Your Attention!

References

- [Pat05] R. Patton. *Software Testing*. Sams Publishing, 2005.

- [Mye04] Glenford J. Myers, *The Art of Software Testing,* John Wiley & Sons, Inc., 2004

- [CB03] Jean-Francois Collard and Ilene Burnstein. *Practical Software Testing*. Springer-Verlag New York, Inc., 2003.

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

- [Fre10] M. Frentiu, Verificarea si validarea sistemelor soft, Presa Universitara Clujeana, 2010

- [BBST] BBST Testing course, http://testingeducation.org/BBST/
  - **Foundations of Software Testing,**
    - **Lecture 2: Strategy**
    - **Lecture 5: The Impossibility of Complete Testing**
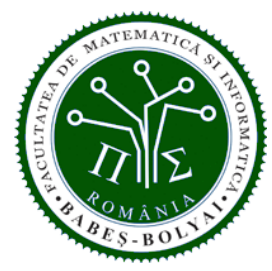
Software Systems Verification and Validation

Assoc. Prof. Andreea Vescan

Babeș-Bolyai University

Cluj-Napoca

2020-2021

Lecture 2b: BBT

Outline

- Testing – fundamental questions
- Black-box testing (domain testing)
  - Equivalence partitioning (EP)
  - Boundary-value analysis (BVA)
  - Advantages/Disadvantages
- Example - black-box testing
  - Example – EP+BVA
- Test management tool – Testlink
- Maven
- Continous integration tool - Jenkins

Domain testing

- The problem you have to solve
  - You cannot afford to run every possible test.
  - You need a method for choosing a few powerful tests that will represent the rest.
- Domain testing
  - Provides **a sampling strategy**
  - It provides **heuristics** for choosing a small number of tests that are powerful enough to represent the larger domain.
    - Equivalence
    - Boundaries
- Input versus output variables
  - Domain definitions
    - Specification:
      - Data (X) + preconditions
      - Result (Z)+ postcondition
- Primary dimension versus secondary dimension [BBST]
  - Primary dimension - reflects the reason you are entering data into the field
  - Secondary dimension - reflects the other ways the input can vary

Domain testing (Black-box testing)

- Equivalence partitioning
- Boundary-value analysis

Equivalence partitioning

- Equivalence class (EC) - definition [Mye04]
  - a partition of input domain of a program.
- Equivalence partitioning
  - to partition the input domain of a program into a finite number of equivalence classes such that you can reasonably assume that a test of a representative value of each class is equivalent to a test of any other value.

Test-case design by equivalence partitioning - steps

- Identifying the equivalence classes (EC)
  - Valid equivalence classes.
  - Invalid equivalence classes.
  - Default, empty, blank, null, zero, none.
  - Invalid, wrong, incorrect, garbage date.
- Defining the test cases
  - assign a unique number to each equivalence class;
  - Until all valid/invalid equivalence classes have been covered by (incorporated into) test cases:
    - write a new test case covering as many of the uncovered valid equivalence classes as possible;
    - write a test case that covers one, and only one, of the uncovered invalid equivalence classes.

Test-case design by equivalence partitioning - guidelines

- An input condition specifies a range of values [a,b].
  - ➔ 1 valid EC, 2 invalid EC
- An input condition specifies the number of values "1 to 3 possibilities".
  - ➔ 1 valid EC and 2 invalid EC
- An input condition specifies a set of input values.
  - ➔ 1 valid EC for each element in the set, 1 invalid EC
- An input condition specifies a must be situation.
  - ➔1 valid EC, 1 invalid EC
- If there is any reason to believe that the program does not handle elements in an equivalence class identically, split the equivalence class into smaller equivalence classes.

Boundary-value analysis

- Boundary-value analysis - definition [Mye04]
  - focuses on the boundary areas of a programs input domain
- Boundary conditions
  - Situations directly on, above, and beneath the edges of input EC and output EC.
  - One or more elements should be selected such that each edge of the EC is the subject of a test.
  - BVA explores situations on and around the edges of the EP.

Test-case design by boundary-value analysis -guidelines

- An input condition specifies a range of values [a,b].

  ➔ the ends of the range, situations just beyond the ends;

- An input condition specifies the number of values "1 to 3 possibilities".

  ➔ the minimum and maximum number of values, one beneath and beyond these values;

- An input condition specifies an ordered set of input values.

  ➔ the first, the last elements of the set;

- The above rules are applied to the output conditions.

Domain testing (Black-box testing)

Advantages

- No knowledge of implementation.
- Tester independent of programmer.
- User's point of view.
- Ambiguities in spec.
- After specifications is completed.

Disadvantages

- A small number of inputs.
- No clear spec.
- Hard to design.
- Unnecessary repetition of test.
- Many program paths untested.
- Specific segments of code?

Example

- **Problem statement**: Compute the number of participants with the maximum score (0 to 100 points possible) at a competition.

- Applied:
  - EP
  - BVA
- See example files
- **Lecture02_Demo**

Maven

- goal - to allow a developer to comprehend the complete state of a development effort in the shortest period of time
- https://maven.apache.org/what-is-maven.html
- Maven
  - The Failsafe Plugin is designed to run integration tests.
  - Maven 2 Integration

Testlink

- Test management tool

Jenkins

- Continuous integration tool

Lecture 2: Required reading (before Seminar 2)

- Assignment 2
  - Laboratory 2 – BBT
- Assignment 3
  - Laboratory 3 - WBT

- Testing – Black-box testing/Domain testing (EC+BVA)
  - In class assignments
  - Homework assignments

References

- [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

- [Fre10] M. Frentiu, Verificarea si validarea sistemelor soft, Presa Universitara Clujeana, 2010

- [BBST] BBST Testing course, http://testingeducation.org/BBST/
  - **Test design,**
    - **Lecture 5: Domain testing**
- Tutorials
  - Module Tutorials in canvas

Questions

- Thank You For Your Attention!

Next Lecture

- White-box testing