UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

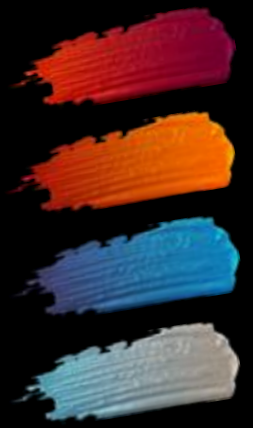
SERLAB
Software Engineering Research

Integrazione e Test di Sistemi Software

Software testing: preliminary definitions

Azzurra Ragone

Dipartimento di Informatica - Università degli Studi di Bari
Via Orabona, 4 - 70125 - Bari
Tel: +39.080.5443270 | Fax: +39.080.5442536
serlab.di.uniba.it



Software testing

Process consisting of any activity concerned with planning, preparation and evaluation of software products to:

1. Determine that they satisfy specified requirements
2. Demonstrate that they are fit for purpose
3. Detect defects

Slide courtesy of Simone Romano

Terminology

- There is a confusing terminology in the testing literature
  - E.g., errors are **wrongly** used as synonyms of bugs
- The International Software Testing Qualification Board (ISTQB) has provided an extensive glossary of testing terms
- http://glossary.istqb.org/
- ISTQB terminology is compatible with that of the Institute of Electronics and Electrical Engineers (IEEE) Computer Society

Slide courtesy of Simone Romano

Terminology

Error: People make errors (i.e., mistakes)

E.g., when people make mistakes while coding, we call the result of those mistakes bugs

Bug (aka defect or fault): Result of an error

More precisely, a bug is the representation of an error in narrative text, UML diagrams, source code, etc.

Failure: Occurs when the code corresponding to a bug executes

Incident: Symptom that a failure has occurred. Something that deserves an investigation

Slide courtesy of Simone Romano

Terminology

Test: The act of exercising a software system with a test case (or more test cases)

- Its goal is to break the system or demonstrate its correct execution

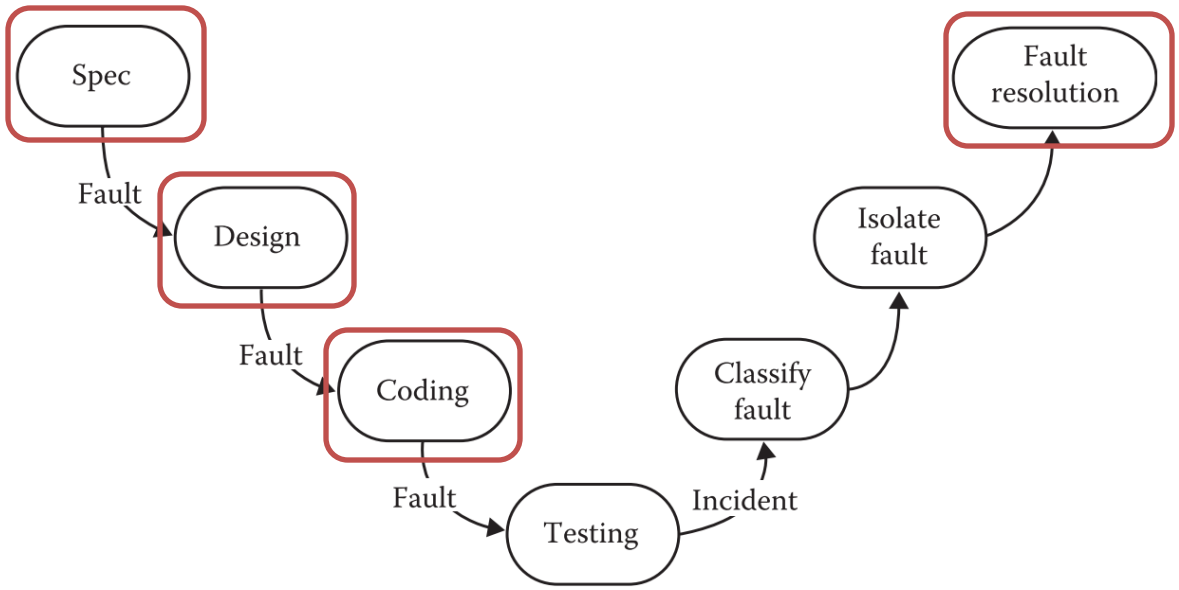
Test case: a recognized work product

- It is associated to a system behavior
- It has a set of inputs and expected outputs (i.e., **oracle**)

Test suite: Set of test cases

Slide courtesy of Simone Romano

Life cycle model for testing



In the dev phases (left side), three opportunities arise for introducing bugs

Fault resolution represents another opportunity for introducing bugs: modifying an existing program is a process that is more error prone than writing a new program.

Slide courtesy of Simone Romano

Test case

- A complete test case description contains:
  - An identifier
  - A test case description (statement of purpose)
  - A description of preconditions
  - Inputs
  - Expected outputs
  - A description of expected postconditions
  - Execution history
    - Date of its execution
    - People who has run it
    - System version
    - Pass/fail result

Slide courtesy of Simone Romano

Test case template

Module Name:-	Login
Test Case ID	gfg_01
Tester Name	Geek
Test Case Description	To check login functionality of geeksforgeeks website.
Prerequisites:	1. Stable internet connection. 2. Browser(Chrome, Firefox, internet explorer, etc.)
Tester's Name	Geek
Enviromental Information:-	1. OS: Windows/Linux/Mac 2. System: Laptop/Desktop
Test Scenario	Checking that after entering the correct username and password, the user can login.

Test Case ID	Test Steps	Test Input	Expected Results	Actual Results	Status	Comments
1.	1. Enter Username	Username: geeksforgeeks	Welcome to geeksforgeeks!	Welcome to geeksforgeeks!	Pass	No issues found.
	2. Enter Password	Password: geek123				
	3. Click on login					

Test case

Executing it entails:

1. Establishing necessary preconditions (if any)
2. Providing test case inputs
3. Observing outputs
4. Comparing observed with expected outputs
5. Ensuring that expected postconditions exist (if any)

Slide courtesy of Simone Romano

Homework 0

- Find a good template for test cases
- There are different templates depending on the type of tests we are performing (Unit, Integration, System)

Identifying test cases

The **input domain** of a system consists of all the possible inputs to that system

Even for a small system, the input domain is so large that it is practically **impossible** to test all the inputs

To identify test cases (i.e., to select a finite set of values from the input domain with which to exercise the system), two main strategies can be followed:

- **Functional / Specification-based** testing
- **Structural / Code-based** testing

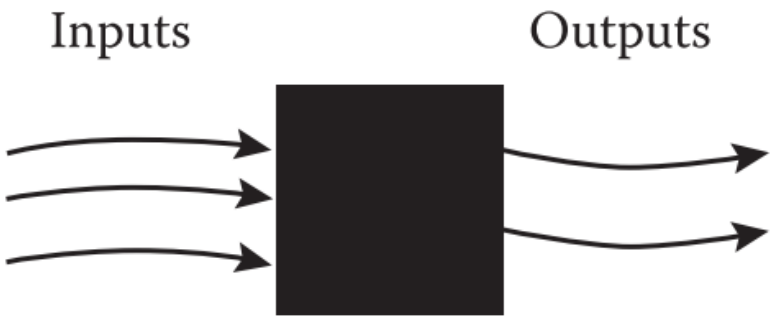
Slide courtesy of Simone Romano

Specification-based testing (1/2)

Systems are considered as black boxes

The content (implementation) of the black box is not known but the function of the black box is understood completely in terms of its inputs and outputs

It is also known as **black-box** testing (ex. Driving a car)



Slide courtesy of Simone Romano

Specification-based testing (2/2)

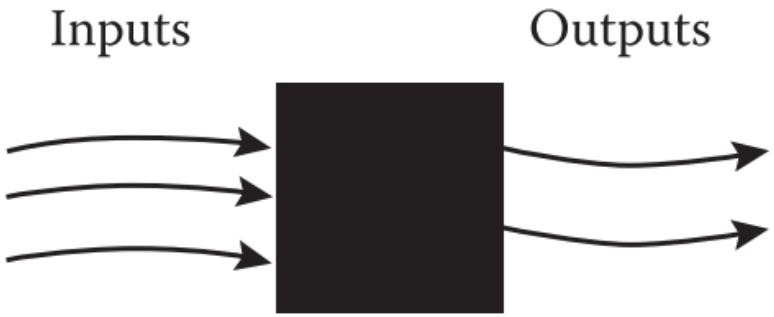
The only information used is the specification of the software

PLUS

- Tests are independent of how the software is implemented (if the implementation changes, we do not have to change tests)
- Test case development and implementation can occur in parallel

MINUS

- Gaps of untested software
- Redundancies



Structural testing (code-based)

In this case, the implementation of the box is known and used to identify test cases

See inside the box allows the tester to identify test cases on the basis of how the function is actually implemented

It is also known as **white-box** testing or **code-based** testing

Test coverage metrics explicitly state the extent to which a software item has been tested

Specification-based vs Structural testing

No approach by itself is sufficient (remind *The pesticide paradox*):

1) If some specific requirements have not been implemented structural testing (code-based) will never be able to recognize this
2) If the program implements behaviors that have not been specified, this will never be revealed by specification-based test cases

You need a combination of the two approaches

Error & Fault

Errors and faults are linked to the distinction between process and product:

- **process** refers to how we do something
- **product** is the end result of a process

Software Quality Assurance (SQA) tries to improve the product by improving the process, reducing errors endemic in the development process

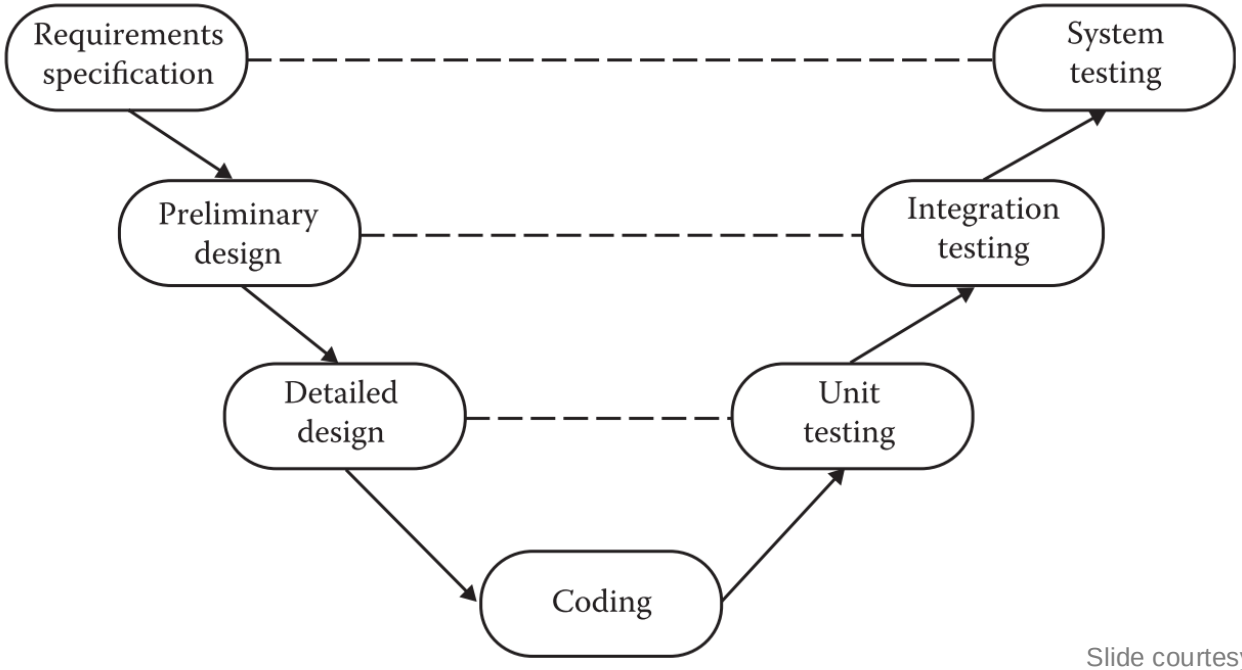
Testing is product-oriented as is concerned with discovering faults in a product

Levels of testing (V- model)

Unit, **integration**, and **system** testing

Acceptance testing is considered the fourth level of testing

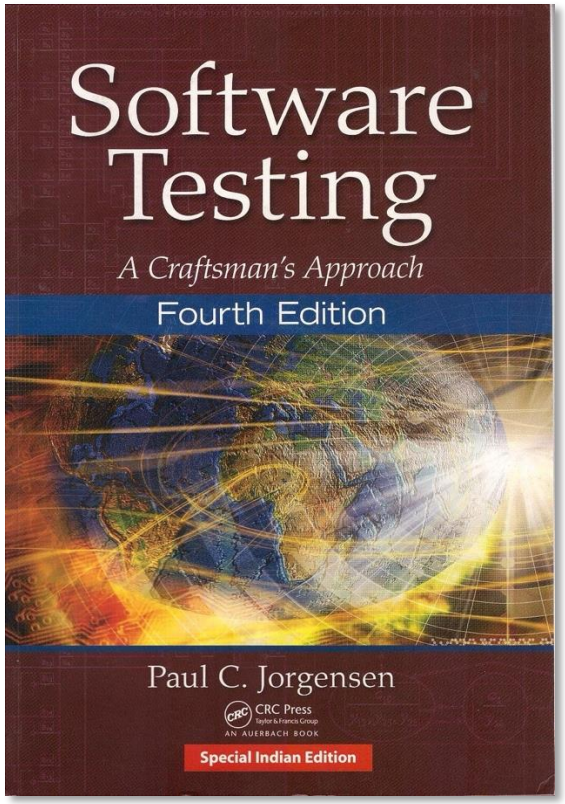
They echo the phases of the waterfall model



Slide courtesy of Simone Romano

Reference book:

Software Testing (A Craftsman's approach). Paul C. Jorgensen. Ed. CRC Press (pdf available online) – **Chapter 1 – pdf available online**



Azzurra Ragone

Department of Computer Science- Floor VI – Room 616
Email: azzurra.ragone@uniba.it