UNIVERSITY OF MISKOLC
FACULTY OF MECHANICAL ENGINEERING
AND INFORMATICS

Software Technology Lab

GEIAL316-B2a

Basics of software testing

Tamas Tompa, PhD
assistant professor
Department of Information Technology

Miskolc, 2026

Introduction to SW testing

- **What is the software testing, why is it needed?**
  - **find existing errors/mistakes in the software** product before delivery it, **to increase the product quality** and reliability

  - Will there be a error/bug in the software?
    - **sure :) → developed by people**
    - **driving license exam analogy**

      - driver → software
        examiner → stester
        exam → testing process
      - nobody can guarantee that the driver (software) perform it without error
      - the examiner (tester) tries to find the errors

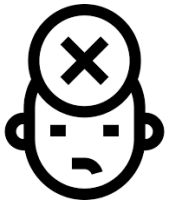
Introduction to SW testing

- **driving license exam analogy**
  - cannot be tested every situation
    - storm, sudden flat tire, etc.
  - goal : decrease the chance of errors, but no 100% guarantee
    - the examiner goal is not that everybody to fail ☺

- the software testing similar to driving license exam:
  - Errors can occur because people develop the software
    - and people can make mistakes
  - testers try to find errors as soon as possible
    - but they may not be able to cover every cases

Introduction to SW testing

- **What is testing, why is it needed?**
  - **program errors**
    - **failure** : the system's operation different from the expected result

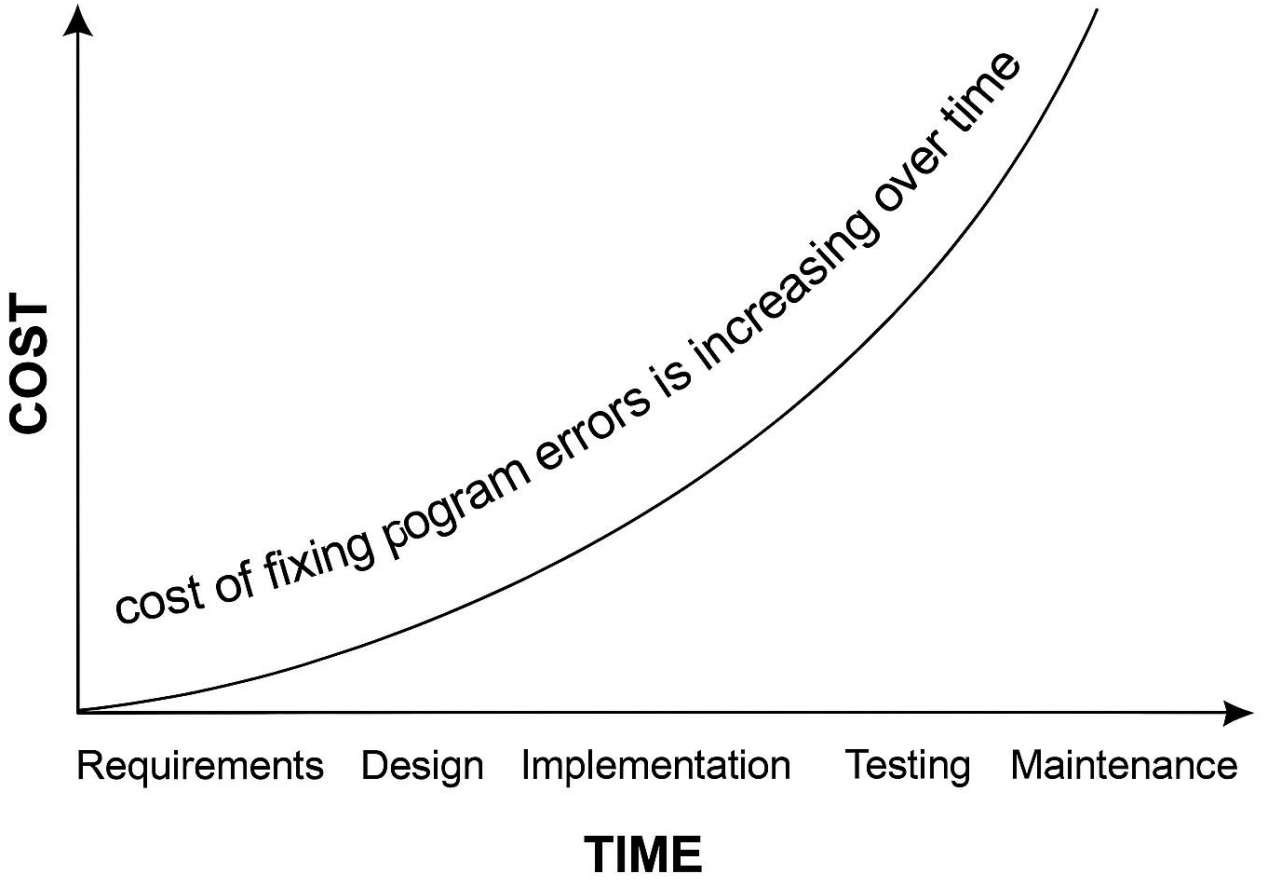


      - **human error**
        - incorrect usage, incorrect activity (*error, mistake*)
      - **software error**
        - internal error, not the expected behavior (*bug , defect, fault*)

Introduction to SW testing

- **A process, not a one-time activity**
  - tester performs
    - an expert who works to ensure that the customer uses a good/properly worked software and satisfied with it
- **It is related to all software development lifecycles**
  - a program error can occur at any level
  - found as soon as possible/early, the cheaper it is to fix
- Does software testing only mean testing the source code?
  - no…

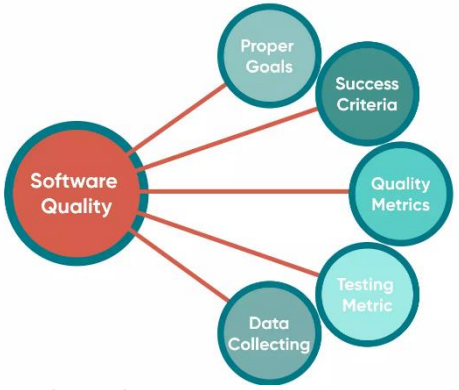
Introduction to SW testing



Source: Dorothy Graham, Erik van Veenendaal , Isabel Evans and Rex Black - Fundamentals of Software Testing

Software quality assurance

- software specification is defined
  - **set of requirements**



- Goal
  - **implement the software that way it fulfils the specification**
  - and it fulfils the requirements also

- **How do we know it fulfils the specification?**
  - **Testing**
    - verification of requirements
    - testing is more than just source code testing

Definition

- All software development processes, whether static or dynamic, that are related to the design, construction, and evaluation of software products to determine whether the software product meets specified requirements and is fit for purpose.

"Testing can detect the presence of defects, not the lack of defects."

Dijkstra

Meaning of the definition

- **Static, dynamic process**
  - searching for program errors by running the software code to display the results of the running test (dynamic) or not (static)
  - static involves reviewing documents including source code

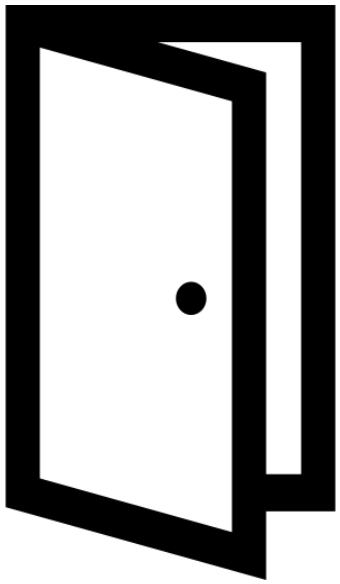
- Need to design
  - **testing needs to be managed**, need to plan what and how we want to do

Meaning of the definition

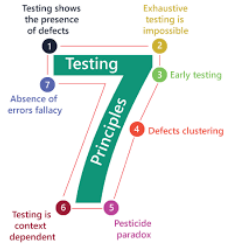
- Requirement
  - a condition or ability that a user needs to solve a problem or achieve a specific goal
- Testing is not meaning only the source testing
  - requirements
  - specifications
  - documents

Door example

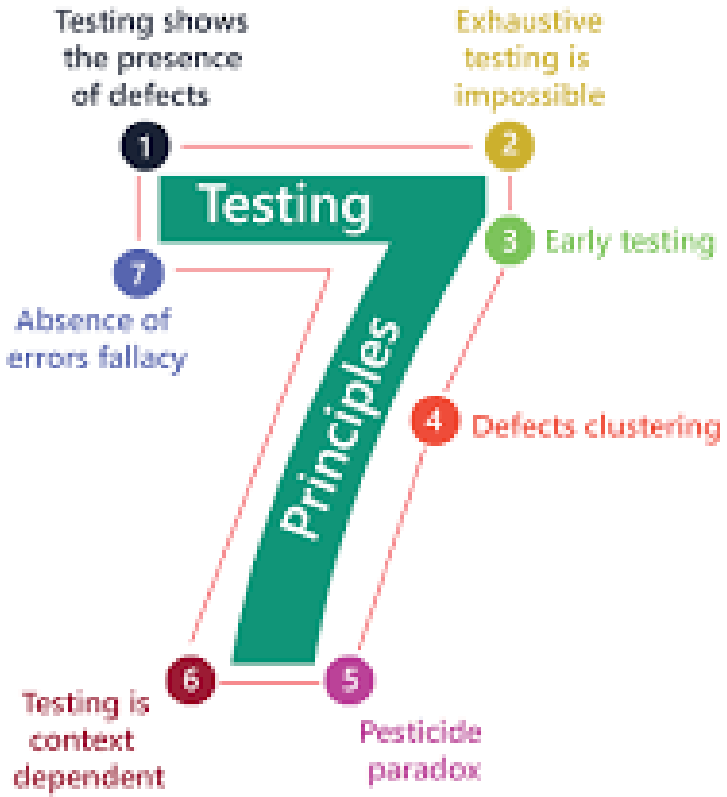
- The customer wants a door from a company
- **Software as a door**
  - every member of the team works on the door
  - What should the door be like?
  - What size should it be?
  - What kind of lock should it have?
    - hook, push , fingerprint reader, retina scanner, dns analyzer
  - Does every member of the team need to know what a door looks like?
    - no → management
  - what need to know
    - the code what write is part of the door and it **must work**
    - How do I know the code is working?
      - testing…
  - For example, developing and testing a lock
    - this is a sliding door :D → the lock is not good → missing information
    - the lock is a component → the lock works on its own (unit test - ok), but as part of the system? (integration test → failure)



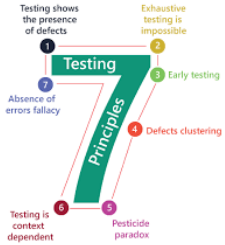
Testing principles



1. Testing show the presence of defect
2. Exhaustive testing is impossible
3. Early testing
4. Defects clustering
5. Worm paradox
6. Testing is context dependent
7. Fallacy of a defectless system

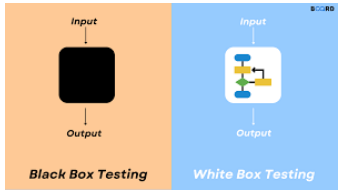


Testing principles



1. **Testing show the presence of defects**: it shows them, but it does not show that they do not exist
2. **Exhaustive testing is impossible**: it is impossible to test every input combination… Testing high-priority cases and trivial cases
3. **Early testing**: start early in the life cycle phase → decrease costs
4. **Defects clustering**: have a finite amount of time, need to focus on the modules where defects are most probably to occur
5. **Worm paradox**: always running the same test cases during retesting, after a while no new defects are found (the worm adapts to the test) → Expanding test levels
6. **Testing is context dependent**: nuclear power plant vs. semester assigment, 10 days vs. 1 night
7. **Fallacy of a defectless system**: unusable software is not worth testing, fixing bugs ↔ dissatisfied customer

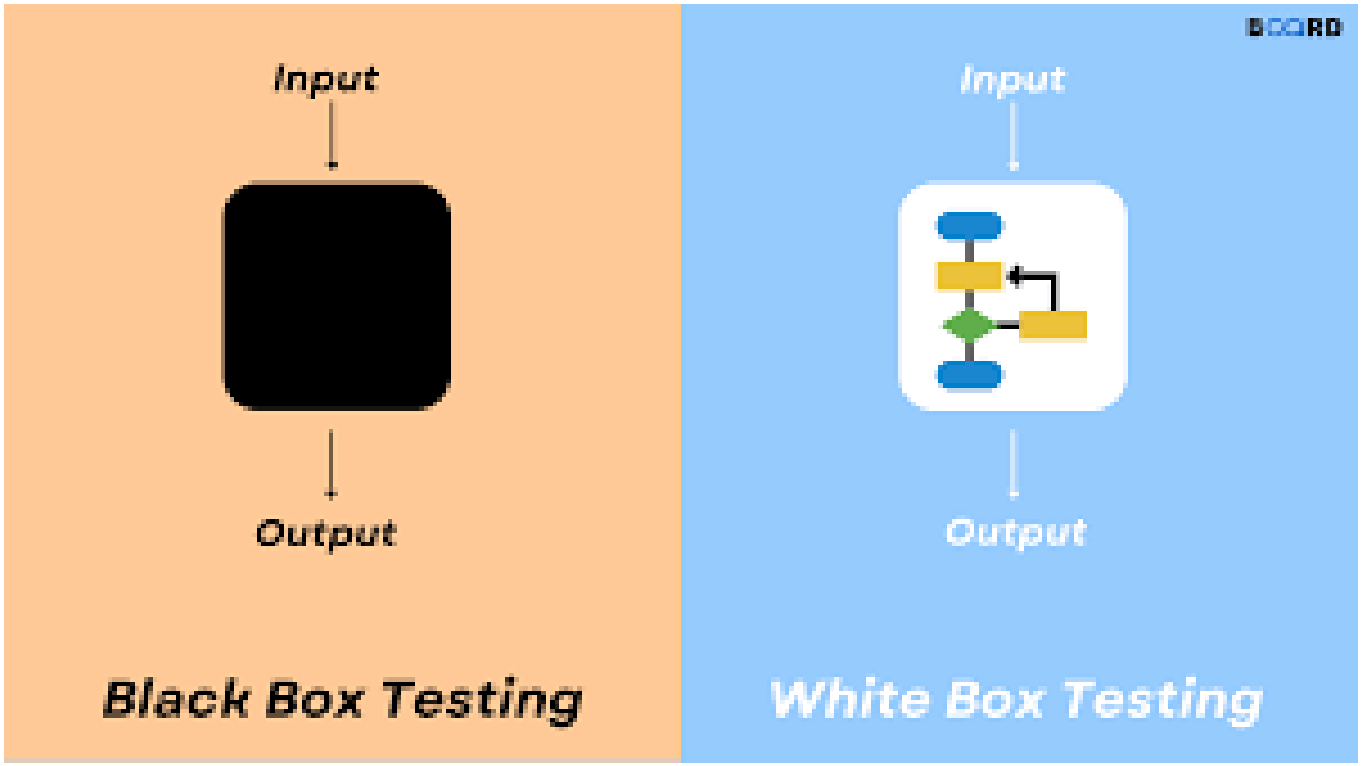
Testing techniques



- **Black-box**
  - specification-based, test cases **based on** the **specification**
  - without knowing the source code → functionality
  - executable software is needed
  - testing "input - output" pairs

- **White-box**
  - structural test, test cases are created **based on the source code**
  - the source code is known → the specific operation is readable
  - definition of coverage
    - what percentage of the structure can be test with existing test cases
    - structure: usually codes, methods, branches, classes, modules, functions, etc.
    - unit test for testing the structure

Testing techniques



Source: https://www.boardinfinity.com/blog/white-box-vs-black-box/

Testing levels

- **Developer test (performed by employees of the company)**
  - **Component/Unit test**
    - testing a component of the system (by itself)
    - usually white-box testing (source code known)
    - unit test (testing the methods)
    - module test (testing non-functional propertie: memory leaks, speed, bottlenecks)
  - **Integration test**
    - testing interfaces
    - component-to-component: interactions between components
    - system-to-system: interactions between a system and other systems
  - **System test**
    - testing the whole system, all components together
    - Does it fulfils: the requirements specification, the functional specification, the system design

Testing levels

- **Acceptance test (performed by end-users)**
  - testing the whole system but **from a user perspective**
  - **alpha test**
    - testing of a finished product at the development company, but not by the development team
    - millions of random mouse clicks
  - **beta test**
    - performed by a small group of end-users
    - gamer fans, sending for them before the relsease

Testing levels

- **Acceptance test (performed by end users)**
  - **user acceptance test**
    - almost all users got the software and use it in the production environment
    - installed and used, but not yet in production
  - **operator acceptance test**
    - system administrators verify that security features, such as backup and recovery, are working correctly or not

Testing activity

- **It does not only consist of creating and running tests, but also includes:**
  - **preparation of a test plan**
    - describes what, for what purpose, how to test, when the test is successful. It is usually part of the system design, including quality assurance (quality assurance, QA) belongs to the chapter
  - **designing test cases**
    - what test data should be performed the test object, what is the expected return value or behavior
  - **preparation for perform**
    - a test environment is needed, when designing the test environment, should strive to make it as similar as possible to the real environment

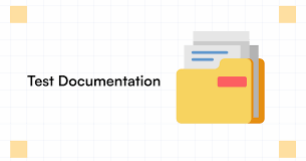
Testing activity

- It does not just consist of creating and running tests, but also includes:
  - **performing tests**
    - writing a test log that lists the steps taken and the results obtained. Based on the test log, the test must be repeatable
  - **examination of exit conditions**
    - after tests, necessary to check whether the exit condition has been successfully met. Comparing the expected result described in the test case with the actual result in the test log
  - **evaluation of results**
    - based on the results further tests prepraring
    - decide based on it: a component not worth any further to test, but a another more deeply must to test

Testing activity

- It does not just consist of creating and running tests, but also includes:
  - **reporting**
    - collecting information about testing → project management, sponsor, customers, stakeholders
    - finding defects in code → attacking testers → avoiding personal attacks
    - cause of error: short time, high pressure, etc.

Test plan document

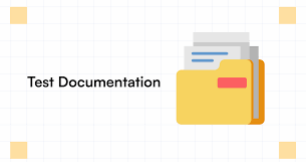


- **Important document which describes what, what aim and what way must test**

- The test purpose
  - to find the defects/errors,
  - increase reliability,
  - prevent the defects/errors,

- Describes the **subject of the test**
- Collects the requirements covered by the test **from the test base**
- Defines the **exit condition**
- The **test data** usually only the test case is defined, but often the test cases are also part of the test plan

Test plan document



- **The test subject**: The part of the system that under testing, this can be the whole system too.

- **Test base**: Set of documents which contain requirements about the subject of test.

- **Test data**: Data, with which performing the test subject. Usually it is known that what values should to give the test of the subject or what kind behavior should to produce. This the expected return value, or behavior. The real return value and behavior compare with the expected return value and behavior.

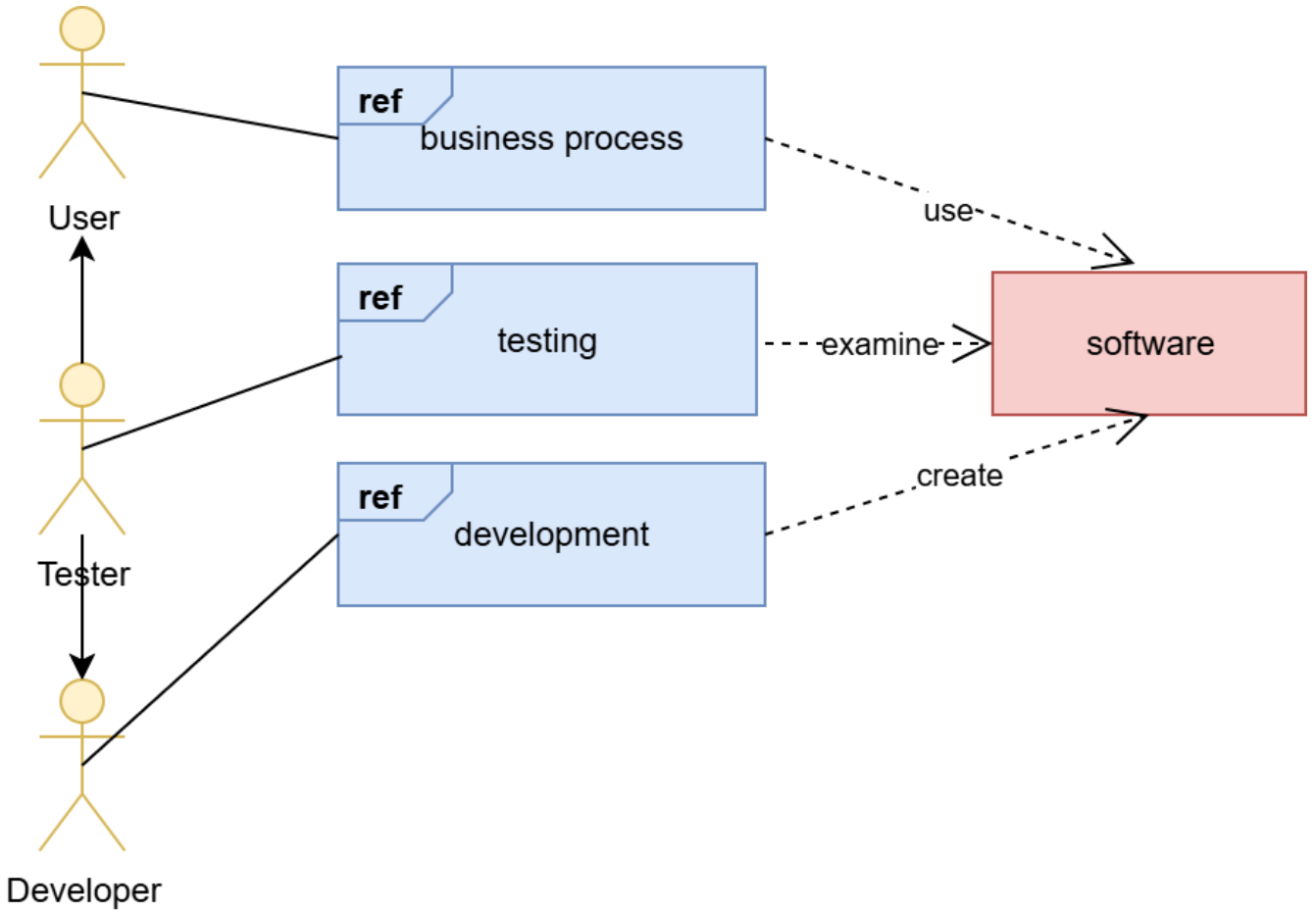
- **Exit condition**: Determine for all tests when can be considered the test can be closed . This called as exit condition. The exit condition usually is all test successfully executed, but it can also that the critic parts test coverage 100%

Software relationships throughout the life cycle

- **During the life cycle, different actors have different relationships with the software**

- **Actors**
  - **Developer**
    - **creates** the software by a development process
  - **Tester**
    - **examines** the software's quality by testing
  - **User**
    - **uses** the system through its various activities

Software connections through the life cycle



About testing controller and system softwares

- **Software that directly operates a hardware device**
- Hardware-specific operation: signal generation/processing
  - e.g. microcontrollers: control of washing machines, refrigerators, microwave ovens, TVs, cars, airplanes, etc.
- A software error can lead to hardware device failure
- **E.g. Airbus A320**
  - first aircraft which fully operate with a computer control system
  - the aircraft's rudders are not in direct contact with the control surfaces
  - joystick and pedals are used to send signals to the control software, which then operates the various hardware devices
  - system software constantly monitors the rudders and if it decides that the rudder deflection is "inappropriate", it overrides the pilot's decision
  - "appropriate": design based on vast experience
  - first prototype: the control software overrode the pilots, so instead of landing, the plane crashed into a field near the airport. Seven crew members died.

- **the topic of testing these devices is beyond the scope of this course**

Thank you for your attention !