Software Testing

Instructor
Battista Biggio

M.Sc. in Computer Engineering, Cybersecurity and Artificial Intelligence
University of Cagliari, Italy

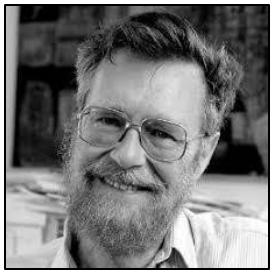
Do I Need to Do Testing?

- Everyone who develops software does testing—that's unavoidable
- The only question is whether testing is conducted in an unstructured manner by random trial-and-error, or systematically, with a plan
- Why does it matter? —The goal is to try as many "critical" input combinations as possible, given the time and resource constraints
  - i.e., to achieve as high coverage of the input space as is practical, while testing first the "highest-priority" combinations

Software Testing

- Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use
- When you test software, you execute a program using artificial data
- You check the results of the test run for errors, anomalies or information about the program's non-functional attributes

"Testing shows the presence, not the absence of bugs."
Edsger W. Dijkstra



Overview of Software Testing

- A **fault**, also called *defect* or *bug*, is an erroneous hardware or software element of a system that can cause the system to fail

- **Test-Driven Development (TDD)**
  - Every step in the development process starts with *a plan of how* to verify the results
  - The developer should not create a software artifact (a system requirement, a UML diagram, or source code) unless he/she knows how it will be tested

- A **test case** is a particular choice of input data to be used in testing a program

- A **test** is a finite collection of test cases

Program Testing Goals

- **Validation Testing:** *to demonstrate to the developer and the customer that the software meets its requirements*
  - For custom software, this means that there should be at least one test for every requirement in the requirements document
  - For generic software products, it means that there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release

- **Defect Testing:** *to discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification*
  - Defect testing is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations and data corruption

Validation and Defect Testing

- In validation testing, you expect the system to perform correctly using a given set of test cases that reflect the system's expected use

- In defect testing, the test cases are designed to expose defects
  - The test cases in defect testing can be deliberately "obscure "
  - They do not need to reflect how the system is normally used

Testing Process Goals

- Validation testing
  - To demonstrate to the developer and the system customer that the software meets its requirements
  - A successful test shows that the system operates as intended

- Defect testing
  - To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification
  - A successful test makes the system perform incorrectly and so exposes a defect in the system

Verification vs Validation

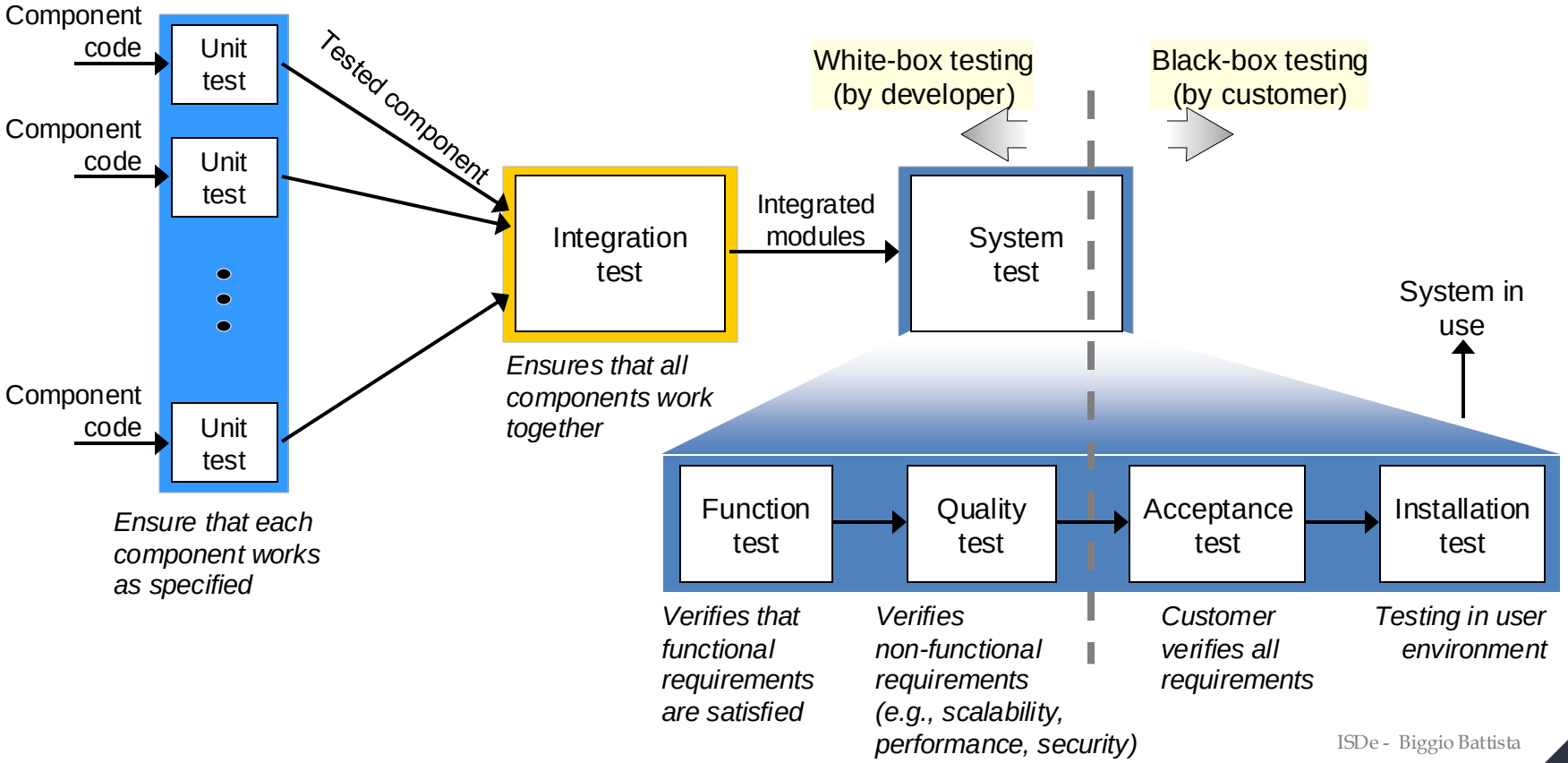
- Verification:
  - "Are we building the product right”
  - The software should conform to its specification

- Validation:
  - "Are we building the right product”
  - The software should do what the user really requires

Why Testing is Hard

- A key tradeoff of testing:
  - testing as many potential cases as possible while keeping the economic costs limited
- Our goal is to find faults as cheaply and quickly as possible.
  - Ideally, we would design a single “right” test case to expose each fault and run it
- In practice, we have to run many “unsuccessful” test cases that do not expose any faults

Logical Organization of Testing

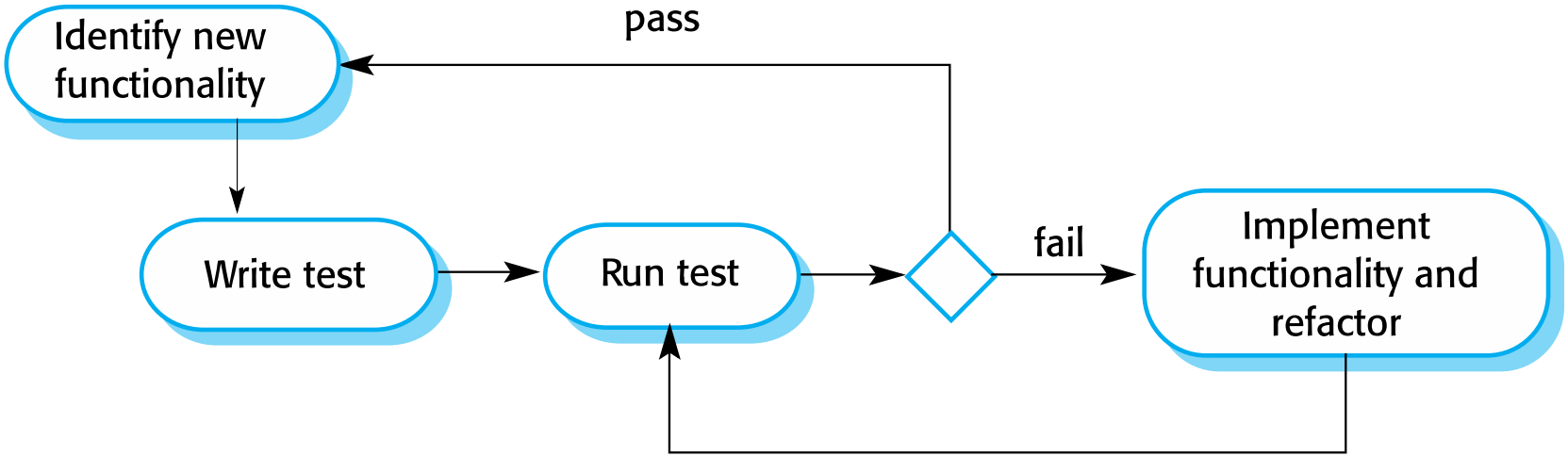


Test-Driven Development (TDD)

Test-Driven Development (TDD)

- Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development
- Tests are written before code and ‘passing’ the tests is the critical driver of development
- You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
- TDD was introduced as part of agile methods such as Extreme Programming, but it can also be used in plan-driven development processes

Test-driven development



TDD Process Activities

- Start by identifying the increment of functionality that is required. This should normally be small and implementable in a few lines of code.
- Write a test for this functionality and implement this as an automated test.
- Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail.
- Implement the functionality and re-run the test.
- Once all tests run successfully, you move on to implementing the next chunk of functionality.

Benefits of test-driven development

- Code coverage
  - Every code segment that you write has at least one associated test so all code written has at least one test.
- Regression testing
  - A regression test suite is developed incrementally as a program is developed.
- Simplified debugging
  - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- System documentation
  - The tests themselves are a form of documentation that describe what the code should be doing.

Regression testing

- Regression testing is testing the system to check that changes have not ‘broken’ previously working code.
- In a manual testing process, regression testing is expensive but, with automated testing, it is simple and straightforward. All tests are rerun every time a change is made to the program.
- Tests must run ‘successfully’ before the change is committed.