Software Engineering Essentials

Testing

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics

Learning Goals

1. Understand the difference between fault, failure and error
2. Understand the taxonomy for fault handling techniques
3. Explain testing activities: unit testing,
   integration testing and system testing

Faults are everywhere

Example: **F-16: Crossing equator using autopilot**

- Result: plane flipped over
- Reason: reuse of autopilot software from a rocket

- More examples: http://www5.in.tum.de/~huckle/bugse.html

Taxonomy for Fault Handling Techniques

More about this in the video **Software Configuration Management**

Fault Handling
Fault Avoidance
Fault Detection
Fault Tolerance
Methodology
Configuration Management
Verification
Testing
Debugging
Exception Handling
Modular Redundancy
Unit Testing
Integration Testing
System Testing

Testing requires creativity

To write effective tests, a tester needs:

- Detailed understanding of the system
- Application and solution domain knowledge
- Knowledge of testing techniques
- Skill to apply these techniques

Developers and testers should be different persons:

- Developers often develop a mental attitude that the program should behave in a certain way when in fact it does not
- Developers often stick to the data set that makes the program work
- A program often does not work when tried by somebody else

Common words: “On my machine it works”

What constitutes successful testing?

The purpose of testing is the generation of failures.

There are two ways to express the success of a test:

(A) The test was successful, because it generated a failure

(B) The test was successful, because it did not generate a failure

Test Model

The test model consolidates all test related decisions and components into one package

also called **tests**

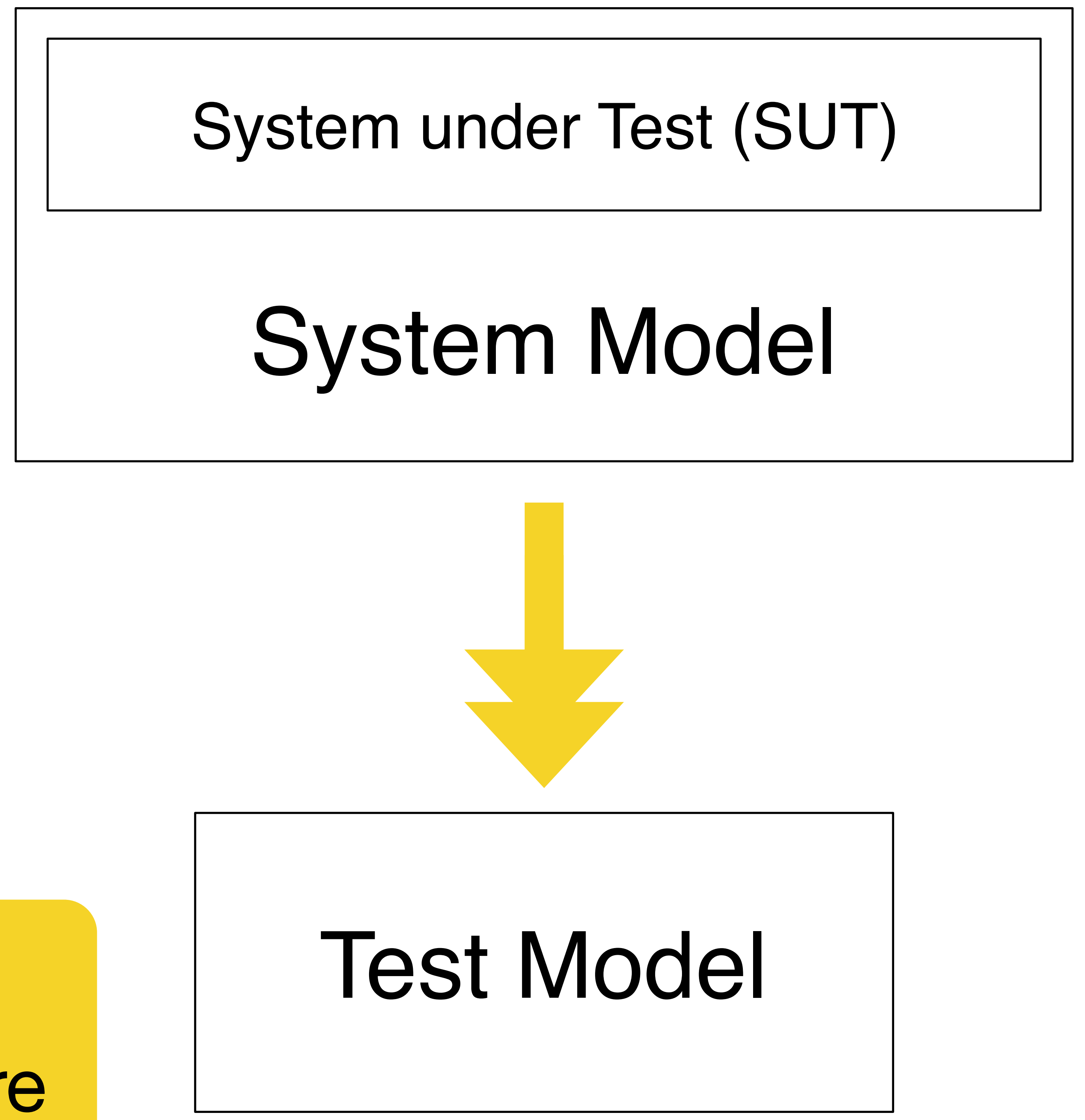
- Test cases: description of the testing activities to be performed, derived from scenarios and use cases
- Test driver: program that is executing the test cases
- The input data is the data needed for testing
- The oracle compares the expected output with the actual test output from the test
- Test harness: Software components or a framework that allows to run the tests under varying conditions and monitor their behavior and outputs

also called **testing framework**

Model-Based Testing

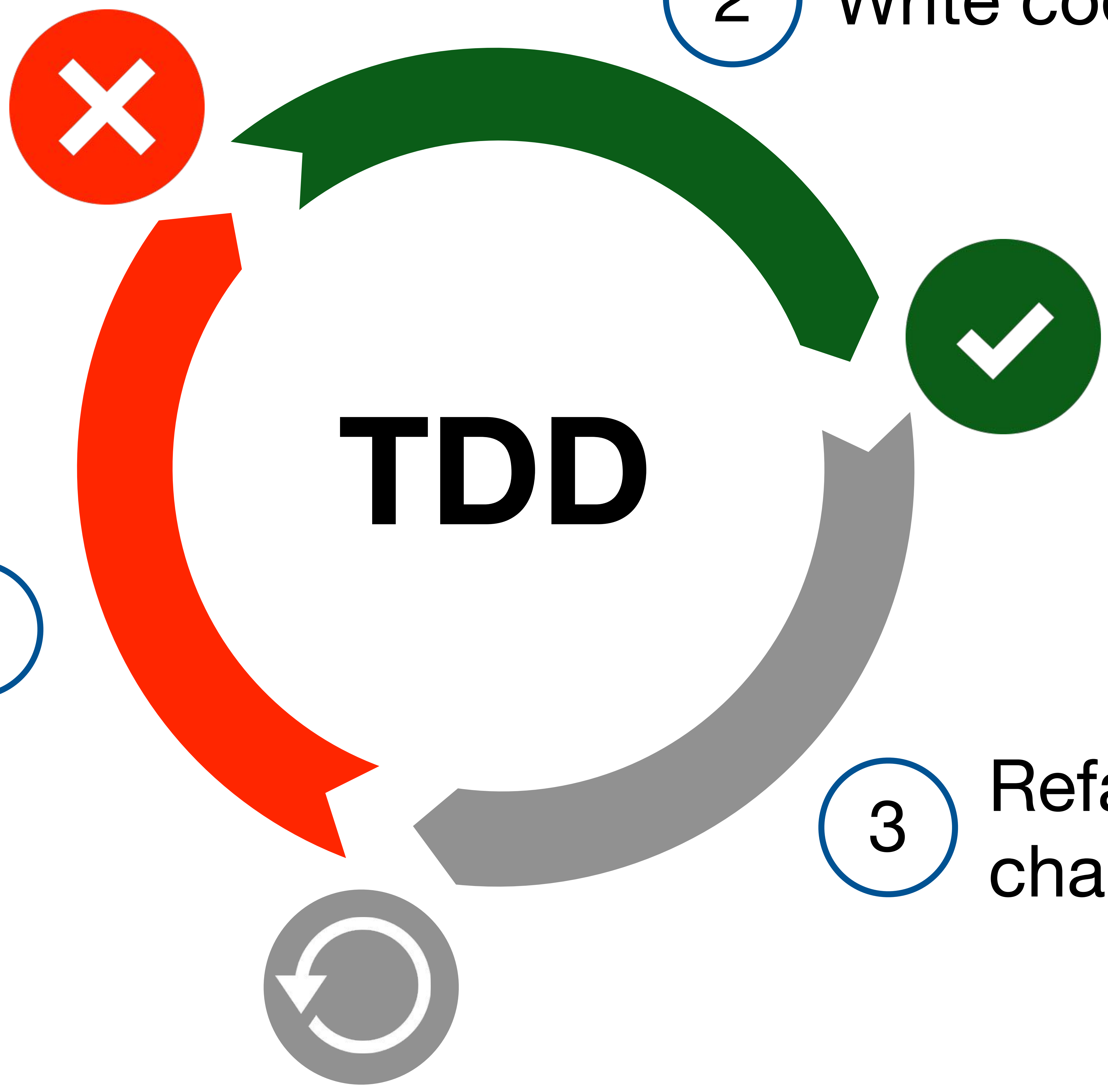
- Model-Based Testing is a technique, where the system is used for the generation of the test model
- System under Test (SUT) is the part of the system model which is being tested

- **Advantages** of Model-Based Testing
  - + Increase effectiveness of testing
  - + Decreased costs, better maintenance
  - + Reuse of artifacts, such as analysis and design models
  - + Traceability of requirements



Test-driven development
"Construct the test model first, before the system model"

Test-Driven Development



1 Turn requirements into specific test cases

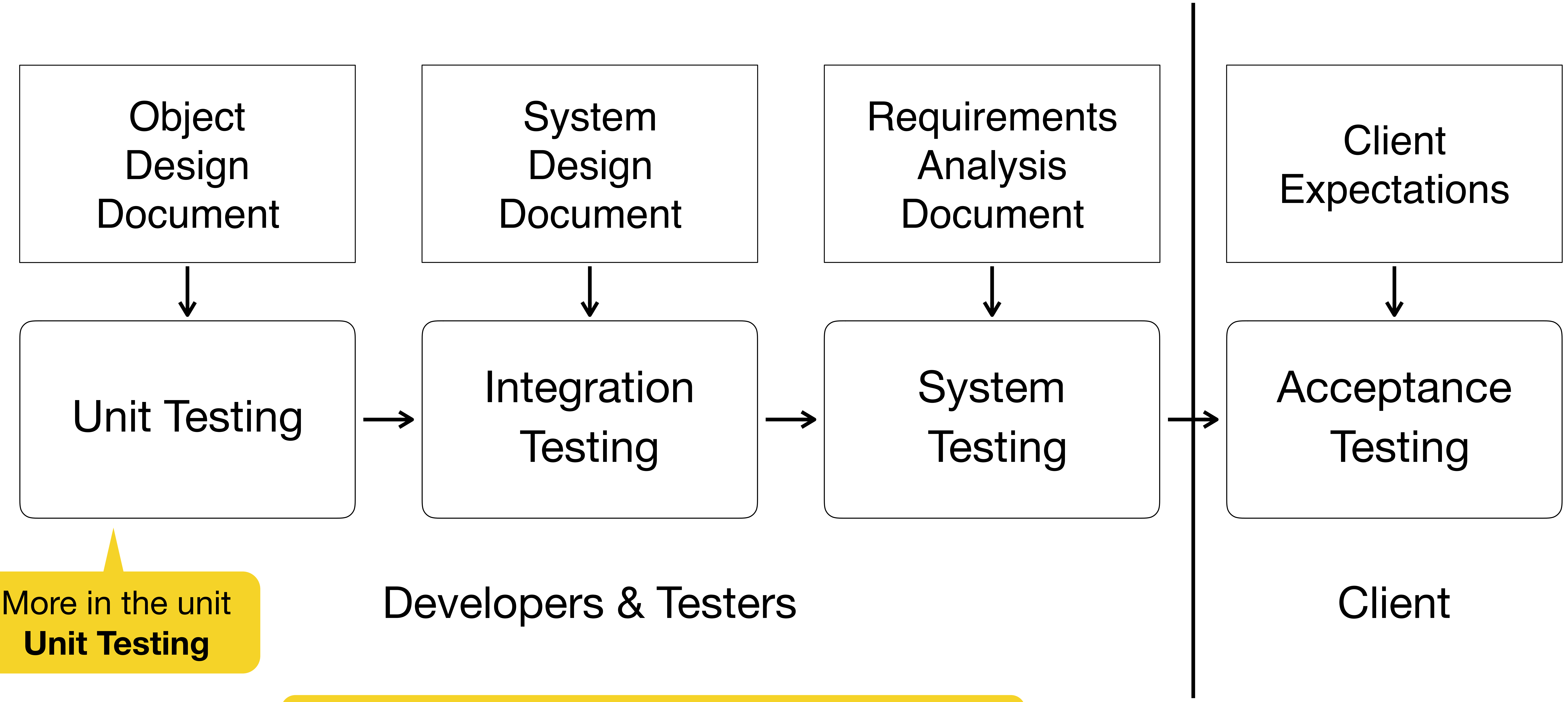
2 Write code to pass the tests

3 Refactor the code, without changing its behavior

Taxonomy for Fault Handling Techniques

Fault Handling
Fault Avoidance
Fault Detection
Fault Tolerance
Methodology
Configuration Management
Verification
Testing
Debugging
Exception Handling
Modular Redundancy
Unit Testing
Integration Testing
System Testing

Model-based Testing Activities



More in the unit
Unit Testing

Regression Testing: Testing after each change