What is Software Testing Continued; Model-Driven Test Design

Dr. Salim Saay
Lecturer at the Department of Computer and Information System
Uuniversity of Limerick
Salim.saay@ul.ie

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

CS4004 Software Testing

Recap

Terminology

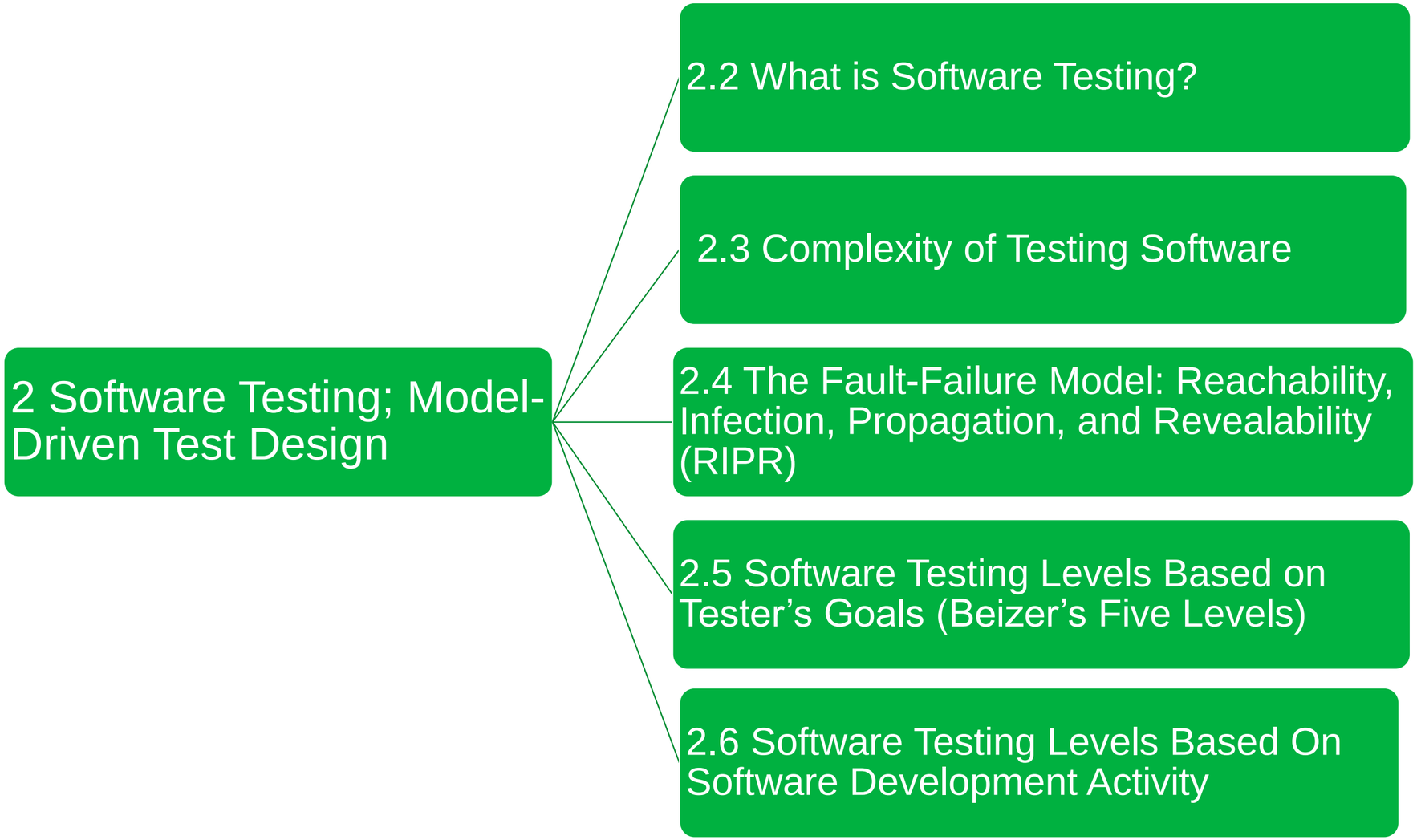
- ***Software Fault:*** A static defect in the software.
- ***Software Error:*** An incorrect internal state that is the manifestation of some fault.
- ***Software Failure:*** External, incorrect behavior with respect to the requirements or another description of the expected behaviour.

A word to the wise: there are many schools of thought in this but these definitions stand for this module.

Verification & Validation

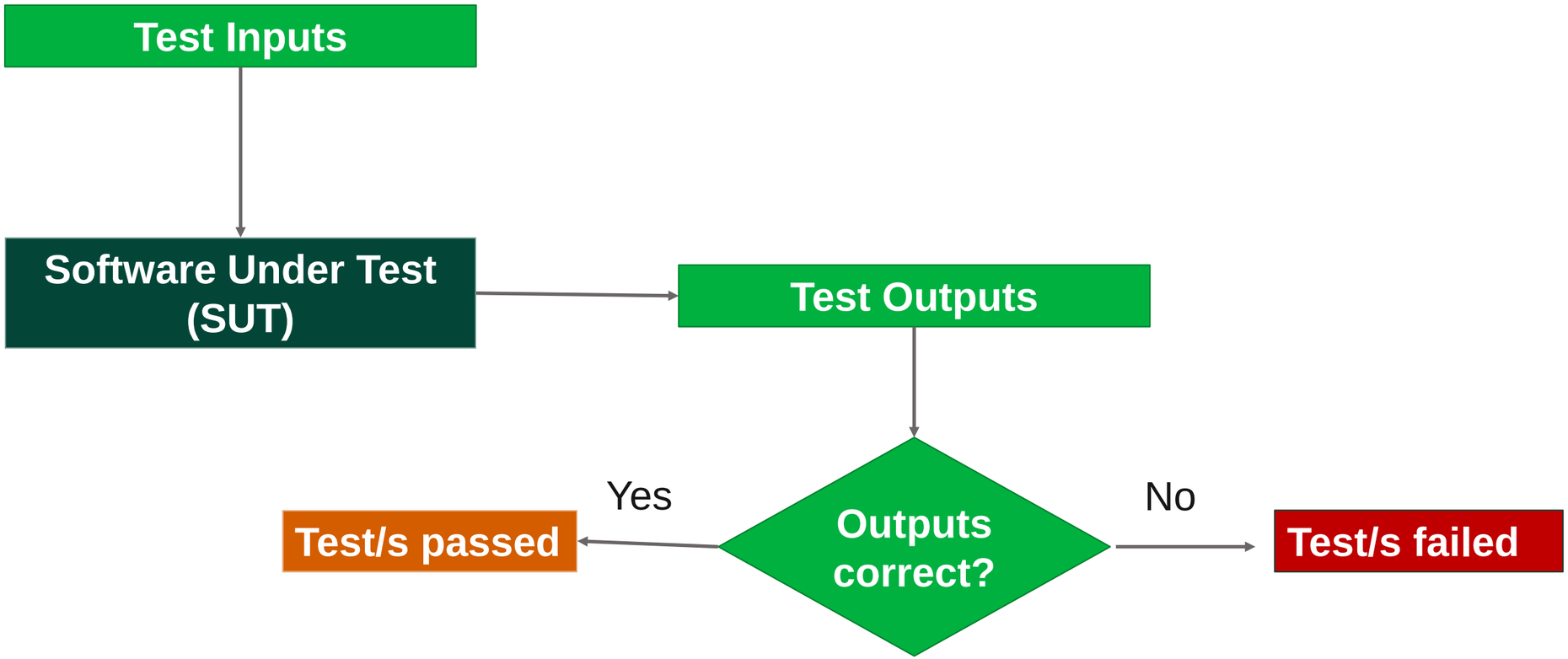
Criteria	Verification	Validation
Definition	The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase.	The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements.
Objective	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.	To ensure that the product actually meets the user's needs and that the specifications were correct in the first place. In other words, to demonstrate that the product fulfills its intended use when placed in its intended environment.
Question	Are we building the product *right*?	Are we building the *right* product?
Evaluation Items	Plans, Requirement Specs, Design Specs, Code, Test Cases	The actual product/software.

Overview

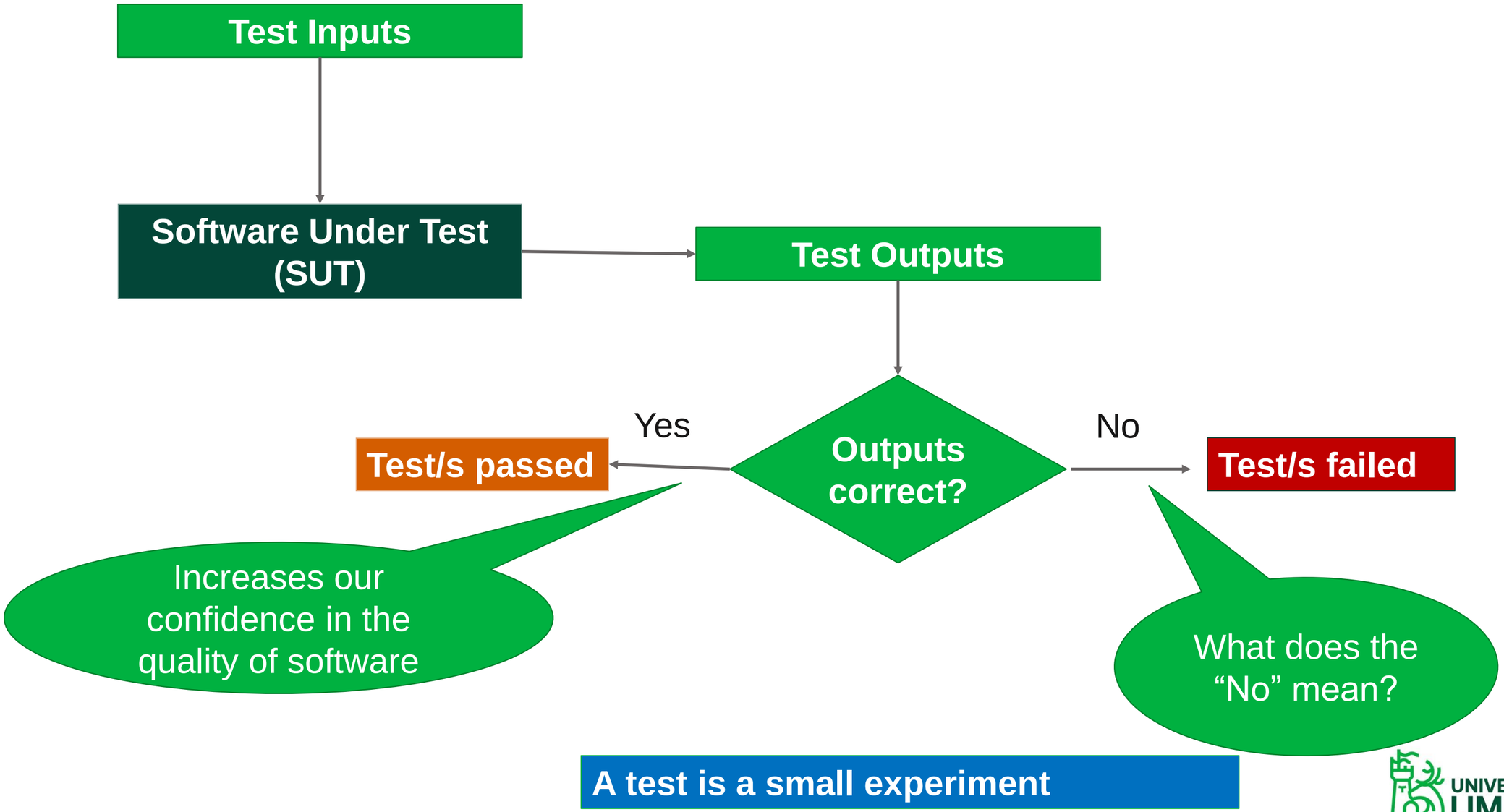


2.2 What is Software Testing?

What is software testing?

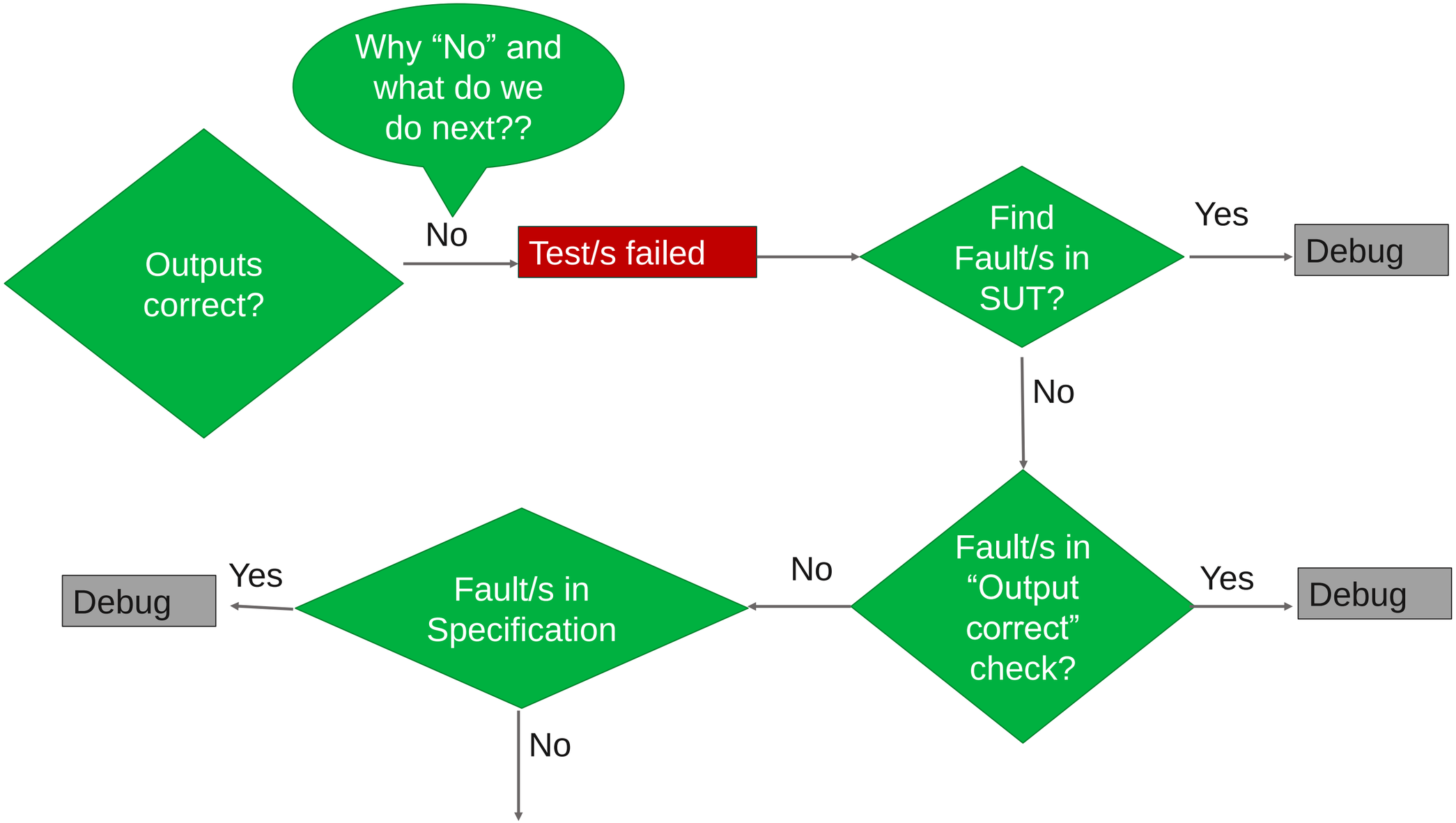


What is software testing?



A test is a small experiment

What is software testing?



… programming platform , etc.
???

Software Testing Foundations

- One of the most important facts that all software testers need to know:

Testing can only show the presence of failures

Not their absence

Testing v. Debugging

- *Definition* -- Testing: Evaluating software by observing its execution
- *Definition* -- Test Failure: Execution of a test that results in a software failure
- *Definition* -- Debugging: The process of finding a fault given a failure

2.3 Complexity of Testing Software

Complexity of Testing Software

- No other engineering field builds products as complicated as software
- What does correctness mean when we are building a piece of engineering as complicated as a large computer program?
- Do we expect correctness
  - In a building, car, road network?
  - What does correct mean in this context?

Complexity of Testing Software

- Instead of looking for "correctness" we can try to evaluate software's "behavior" to decide if it is acceptable from the points of:
  - Reliability
  - Safety
  - Maintainability
  - Security
  - Efficiency

How can we manage this Overwhelming Complexity?

- We do the same as all other engineers do:
  - We use mathematics to raise our level of abstraction to manage complexity
  - The Model-Driven Test Design (MDTD) process breaks testing into a series of small tasks that simplify test generation
  - The test designers isolate their task, and work at a higher level of abstraction to design tests.
  - Typically use mathematical engineering structures to design tests values independently of the details of the software.

How can we manage this Overwhelming Complexity?

- A key intellectual step in model-driven test design is test case design
  - Tests can be designed with a “human-based” approach, where a test engineer uses domain knowledge and her/his experience to design tests that will be effective at finding faults.
  - Tests can be designed to satisfy well-defined engineering goals such as coverage criteria

2.4 The Fault-Failure Model

Reachability, Infection, Propagation, and Revealability (RIPR) Model

Fault and Failure → Fault-Failure Model

The central issue is that for a given fault, not all inputs will "trigger" a failure.

It can be very difficult to map a failure to the respective fault.

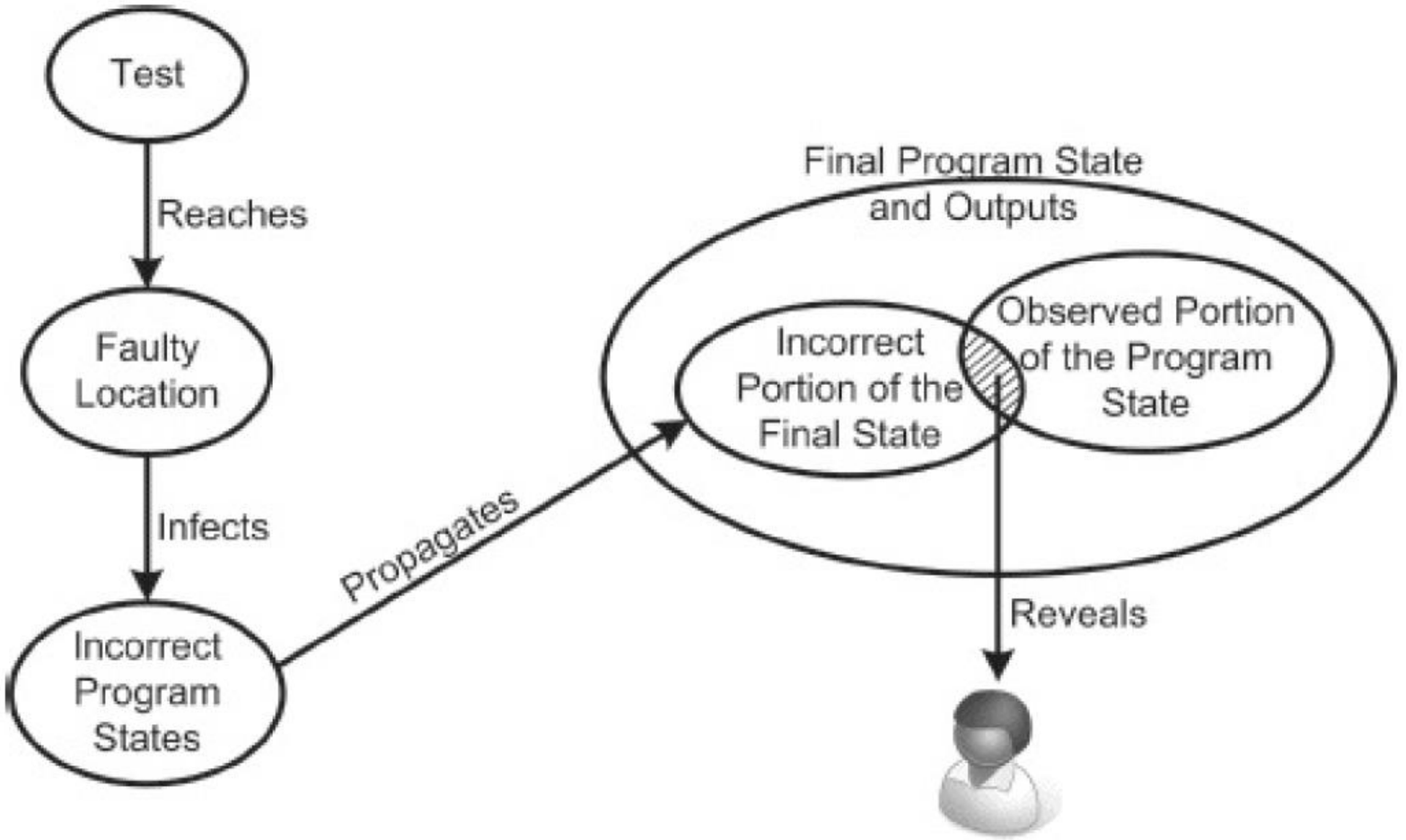
Analysing these ideas lead to the Fault-Failure model (The Reachability, Infection, Propagation, and Revealability Model (RIPR)) that states that four conditions are needed for a failure to be observed.

The Reachability, Infection, Propagation, and Revealability Model (RIPR)

Four conditions necessary for a failure to be observed:

RIPR Model

- **Reachability**
- **Infection**
- **Propagation**
- **Revealability**



UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

Important point on the RIPR model

- Does the RIPR model stand when the fault is that a piece of code is missing?

Important point on the RIPR model

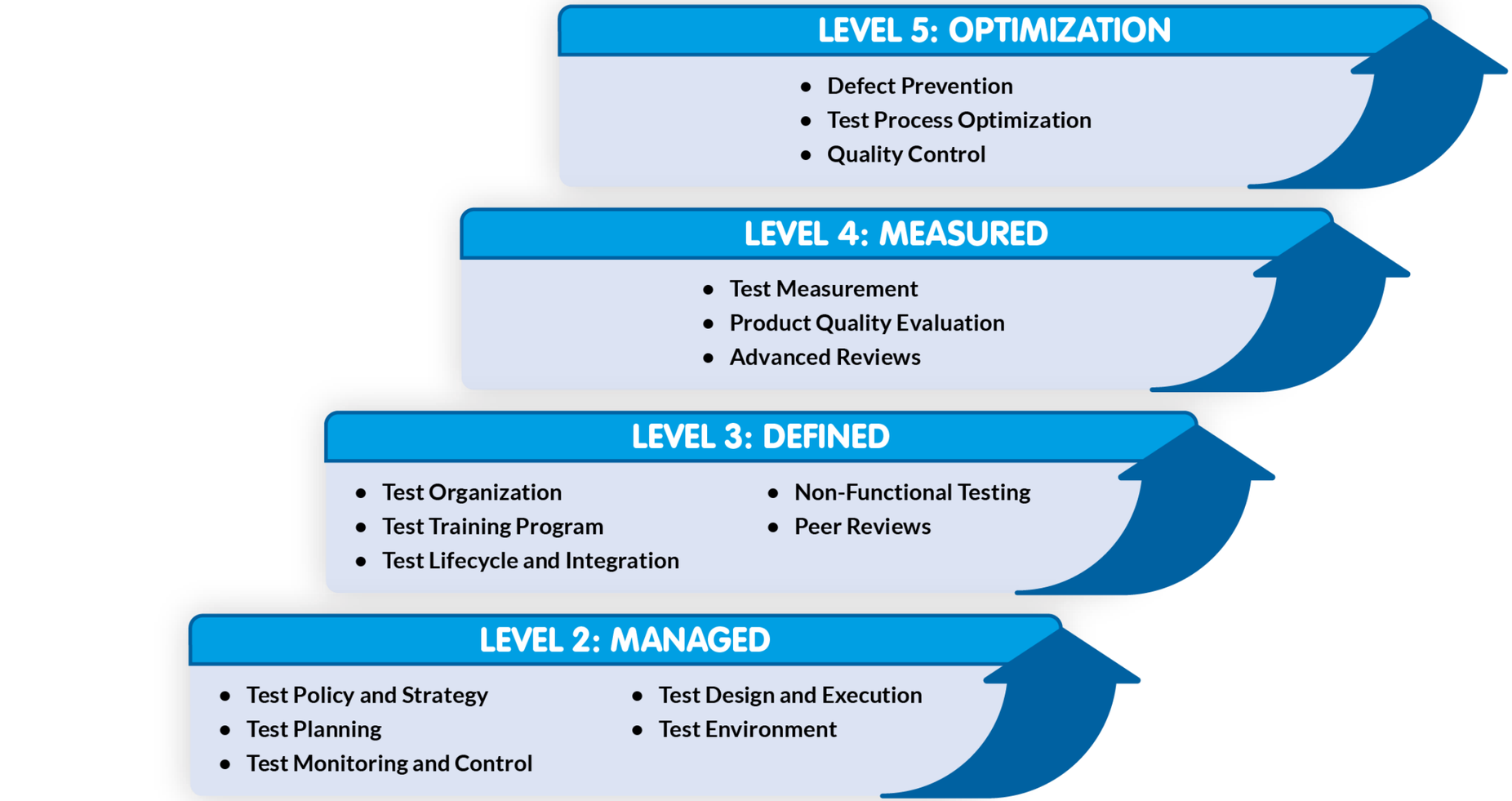
- The RIPR model stands even when the fault is that a piece of code is missing.
  - When execution passes through the location where the missing code should be, the PC (program counter), is part of the program state, and will have the wrong value

2.5 Software Testing Levels Based on Tester's Goals (Beizer's Five Levels)

Beizer's Five Testing levels

- **Boris Beizer's** discussed the goals of testing in terms of "**process maturity levels**" of an organization
- Beizer's levels are **characterized** by the **tester's goals**
- Beizer **defined 5 levels**

Maturity levels and goals of the Testing Maturity Model TMM



https://www.tmmi.org/tmmi-model/

Testing Goals Based on Test Process Maturity

- Level 1 : There's no difference between testing and **debugging**
- Level 2: The purpose of testing is to **show correctness**
- Level 3 : The purpose of testing is to show that the **software doesn't work**
- Level 4 : The purpose of testing is not to prove anything specific, but to **reduce the risk** of using the software
- Level 5 : Testing is a mental discipline that helps all IT professionals **develop higher quality software**

These are Beizer's five levels of testing as characterized by the tester's goals

Level 1 Thinking

- **This is the initial view of many undergraduate CS students**

❖ Testing is the same as debugging

❖ Does not distinguish between incorrect behaviour and mistakes in the program

❖ Does not help develop software that is reliable or safe

Level 2 Thinking

- This is typical view from hardware engineers

❖Purpose is to show correctness

❖Test engineers have no:

➢Strict goal

➢Real stopping rule

➢Formal test technique

Level 3 Thinking

- ❖Purpose is to show failures
- ❖Looking for failures is a negative activity
- ❖Puts testers and developers into an adversarial relationship

This describes a lot of software companies.

How can we move to a *team approach* ??

Level 4 Thinking

- Testing can only show the presence of failures
  - Whenever we use software, we incur some risk
  - Risk may be small and consequences unimportant
  - Risk may be great and consequences catastrophic
  - Risk is always there
  - Testers and developers cooperate to reduce risk

More and more software companies move to this level every year

Level 5 Thinking

A mental discipline that increases quality

- Now the testers and developers are on the same “team”
- Testing is only one way to increase quality
- Test engineers can become technical leaders of the project
- Primary responsibility to measure and improve software quality
- Their expertise should help the developers
- Spellchecker or Grammarly:
  - can be used to find misspelled words or bad grammar
  - best purpose is to improve our spelling & grammar

This is the way “traditional” engineering works

Levels 2- 5 Thinking

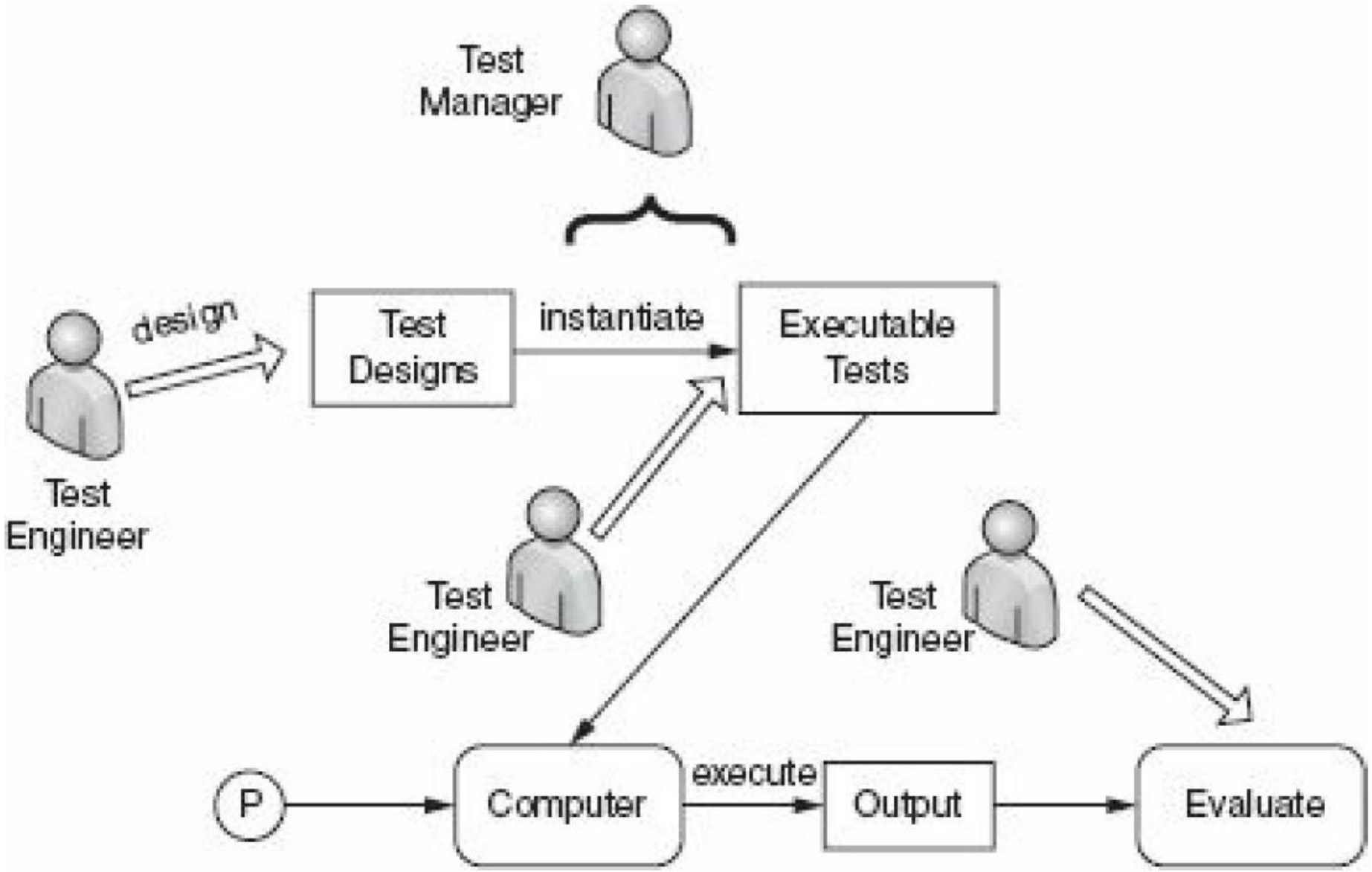
- These levels help us decide at a strategic level why we test
- At a more practical level we need to know why each test has been designed (or needs to be designed)

2.6 Software Testing Levels Based On Software Development Activity

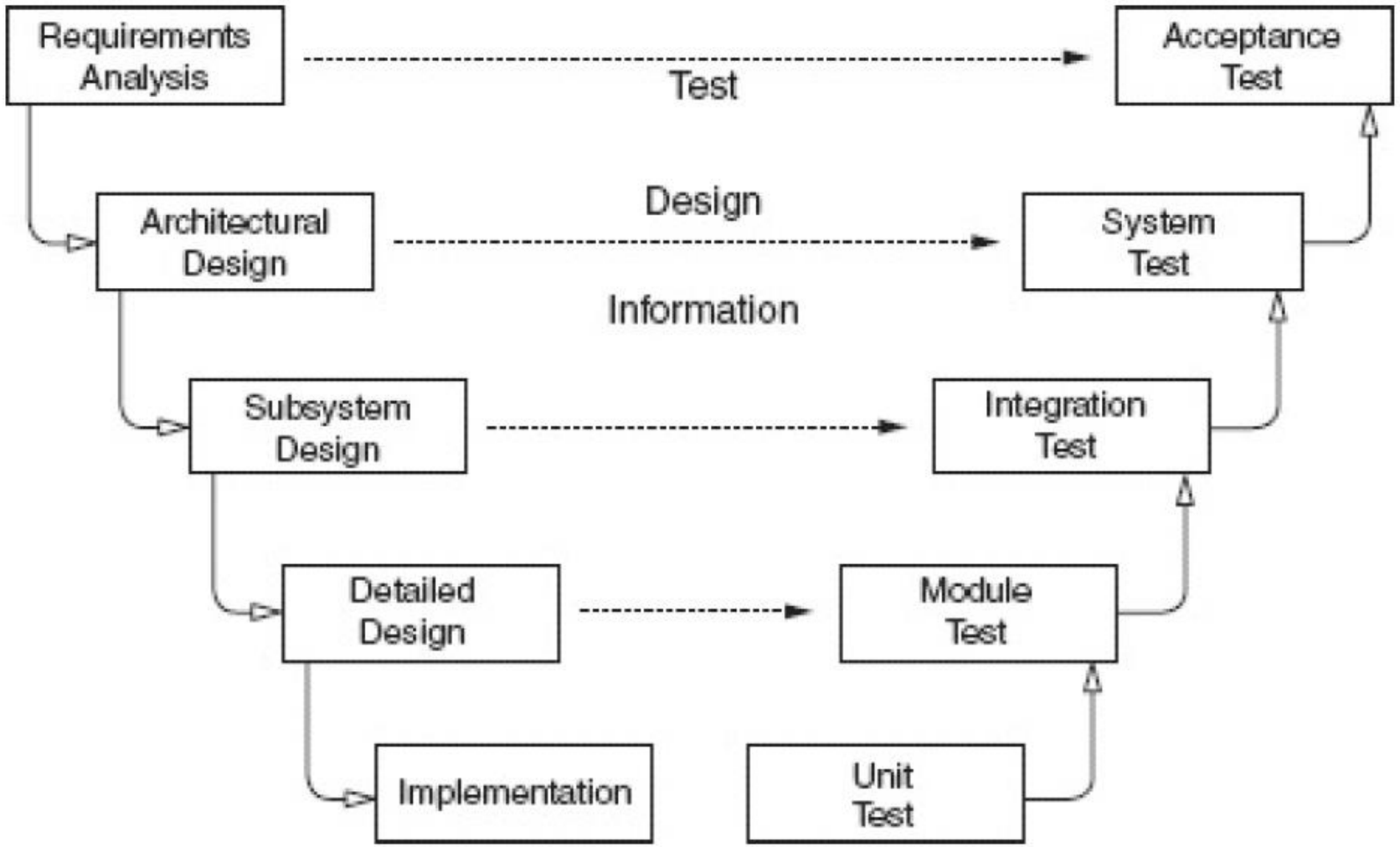
Software Testing Activities

- **Test Engineer:** An IT professional who is in charge of one or more technical test activities
  - Designing test inputs
  - Producing test values
  - Running test scripts
  - Analyzing results
  - Reporting results to developers and managers
- **Test Manager:** In charge of one or more test engineers
  - Sets test policies and processes
  - Interacts with other managers on the project
  - Otherwise supports the engineers

Key Activities of Software Engineer



Software development activities and testing levels – the "V Model"



It does not imply a waterfall process!

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

Testing Levels Based on Software Activity

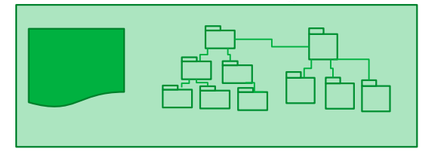
Acceptance Testing: assess software with respect to requirements or user's needs.

System Testing: assess software with respect to architectural design and overall behavior

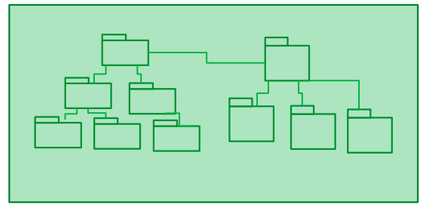
Integration Testing: assess software with respect to subsystem design.

Module Testing: assess software with respect to detailed design.

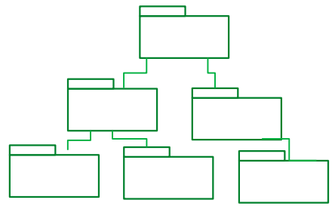
Unit Testing: assess software with respect to implementation



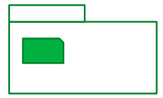
Acceptance Testing



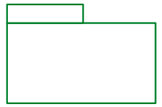
System Testing



Integration Testing



Module Testing

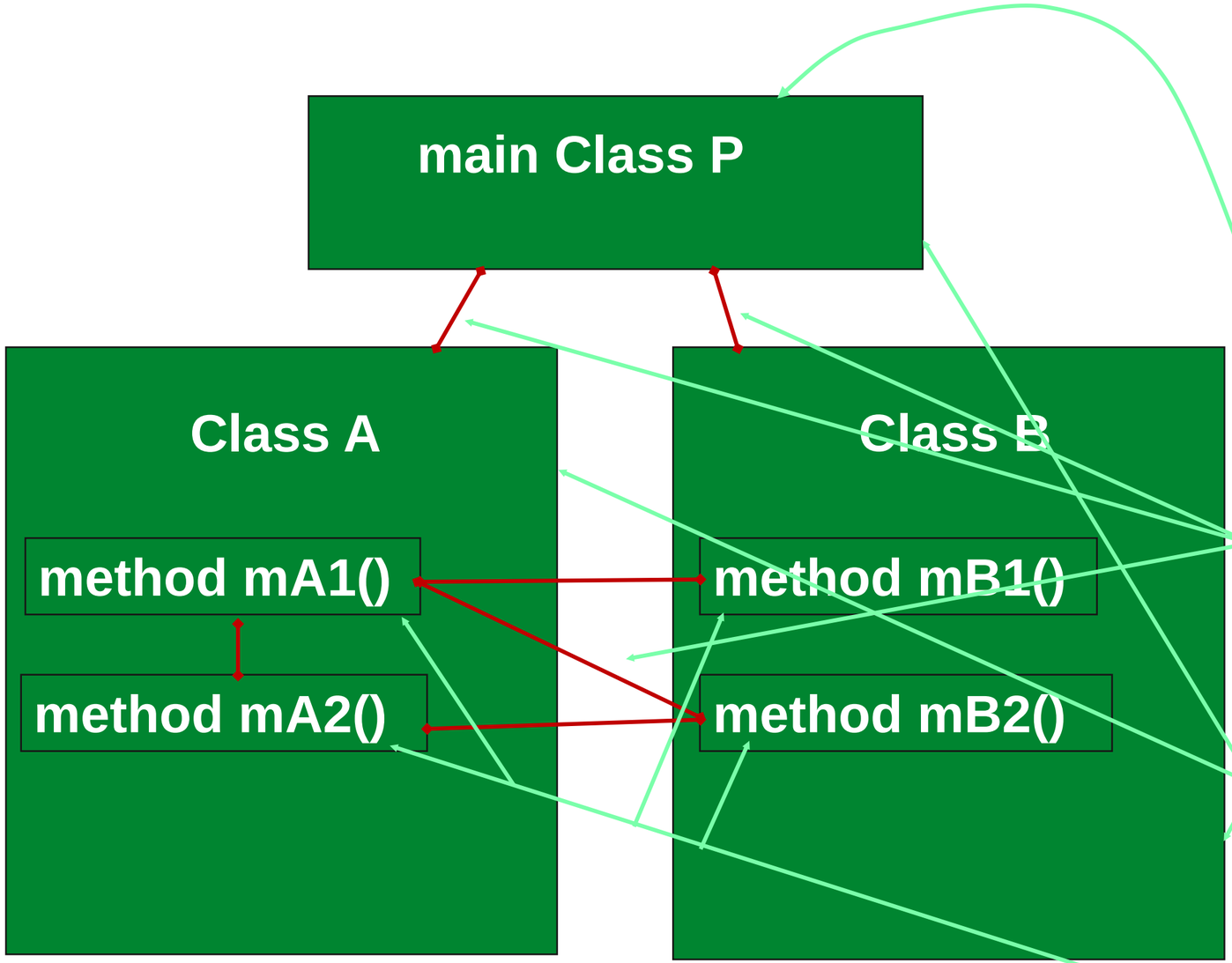


Unit Testing

Another view of these testing levels

- A lot of literature emphasizes these levels in terms of when they are applied (e.g. during integration)
- A more important distinction is on the types of faults we are looking for.
- The faults are based on the software artefact we are testing
  - E.g. unit and module tests are derived to test units and modules → here we typically try to find faults that can be found when executing the units and modules individually

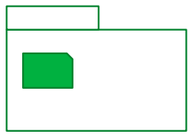
Traditional Testing Levels



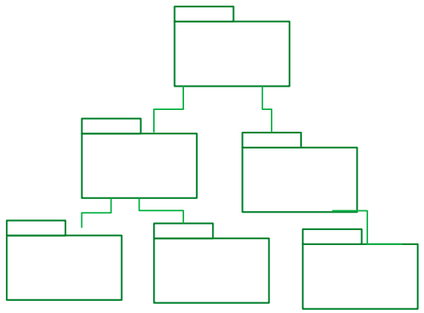
- Acceptance testing: Is the software acceptable to the user?
- System testing: Test the overall functionality of the system
- Integration testing: Test how modules interact with each other
- Module testing (developer testing): Test each class, file, module, component
- Unit testing (developer testing): Test each unit (method) individually

A Snapshot of Different Kinds of Testing

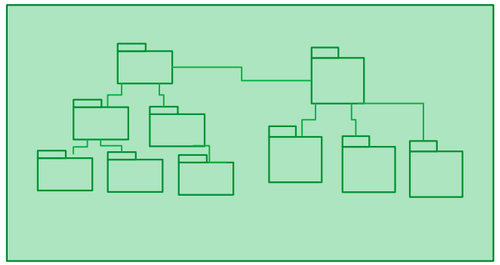
Unit Testing



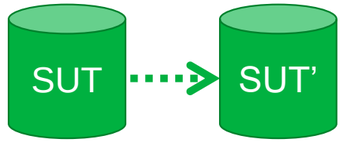
Module Testing



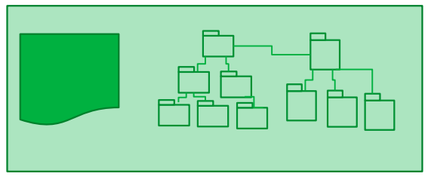
Integration Testing



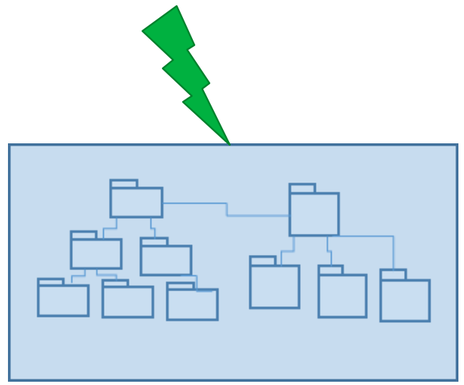
System Testing



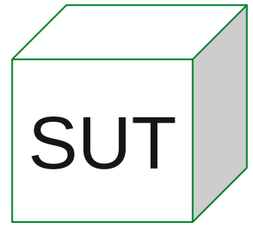
Regression Testing



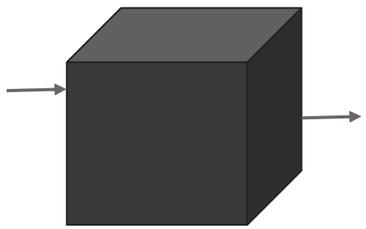
Acceptance Testing



Stress Testing



White Box



Black Box

Not an exhaustive view

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

White Box Testing, Black Box Testing.

White box testing

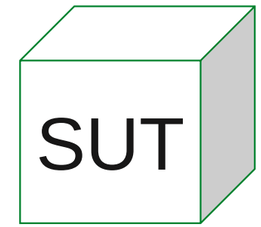
The code is visible to the tester

The internal structure, design, implementation is known to the tester

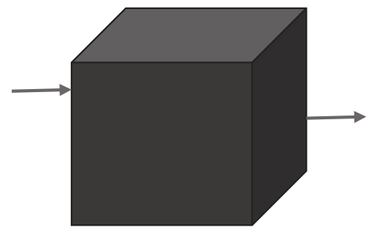
Black box testing

The code is **not** visible to the tester

The internal structure, design, implementation is **not** known to the tester



White Box



Black Box

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

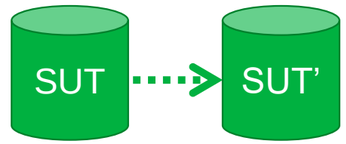
Regression Testing, Stress Testing

Regression Testing

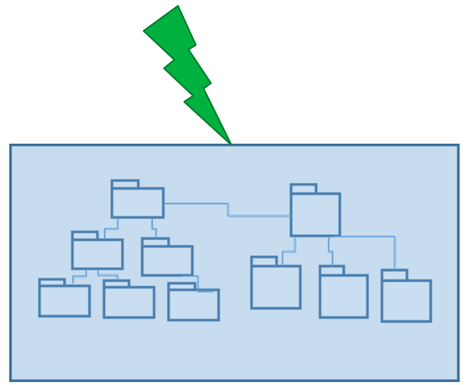
Is done after changes are made to the software, to help ensure that the updated software still possesses the functionality it had before the updates.

Regression testing is a standard part of the maintenance phase of software.

Tests the robustness of a system



Regression Testing



Stress Testing

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

Thank you

UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

University of Limerick,
Limerick, V94 T9PX,
Ireland.

Ollscoil Luimnigh,
Luimneach,
V94 T9PX, Éire.

+353 (0) 61 202020

ul.ie