Software Testing:
Why?
How?

Mohammad Mousavi

KING'S College LONDON

What?
Faults, Errors, Failures

- Fault: incorrect implementation:
  - commission: wrong implementation
  - omission: forgotten implementation (the more difficult one)
- Error: incorrect system state
- Failure (anomaly, incident) : visible error in the behavior

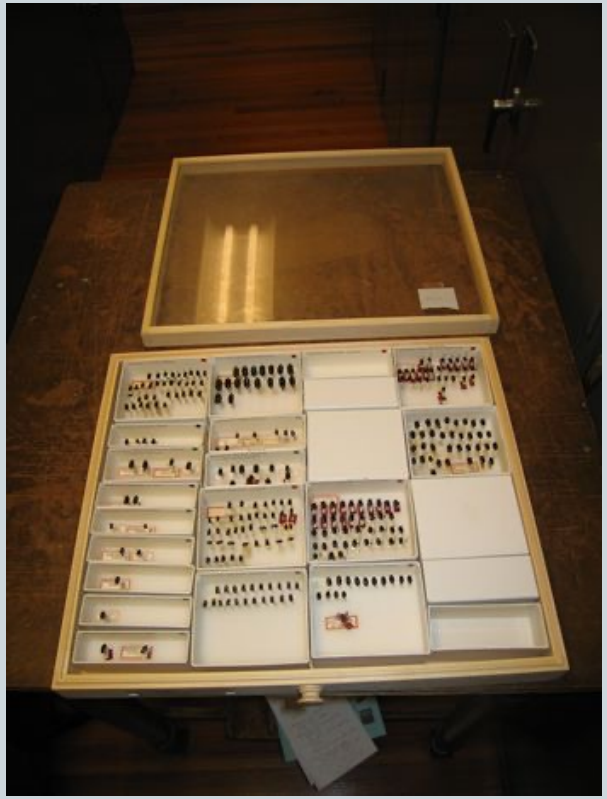


Photo from Wikipedia

Example

Spec: inputs an integer, and outputs $2*i^3$

Implementation:

```
#include <iostream>
#include  <math.h>

int main() {
    int i;
    cin >> i;
    i = 2 * i;
    i = pow(i, 3);
    cout << i;
   return 0;
}
```

Example

1. cin >> i;
2. i = 2 * i;
3. i = pow(i, 3);
4. cout << i;

- Conceptual mistake: confusing the binding power of operators
- Fault: Statements 2 and 3 are in the wrong order
- Error: State of the program after line 3 may have the wrong value for i.
- Failure:
- Test-case: input 1, expected output 2.
- Actual execution: input 1 ... output 8!

What?
Testing

Planned experiments to:

1. reveal bugs (turn faults into failures, test to fail), “Testing can show the presence of bugs, but not their absence.” [Dijkstra]

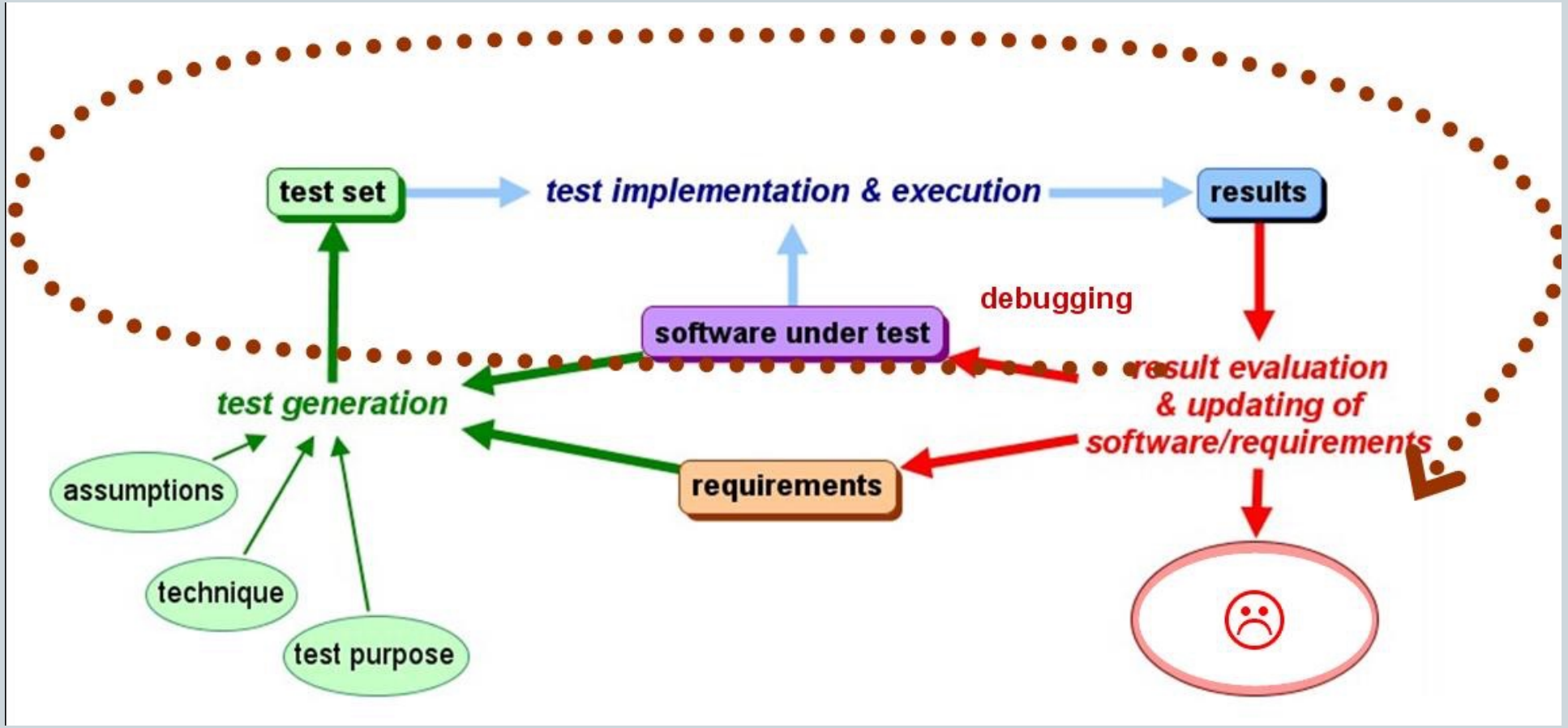
2. gain confidence in software quality (test to pass) (lots of interesting literature on reliability)

How?
Test-Case, Test-Suite

- **Test-Case:** a pair of
  - inputs (e.g., running environment, input values or pre-conditions, timing of events) and
  - expected outputs (e.g., concrete output values or symbolic properties input and output)

- **Test-Suite:** a set or list of test-cases

What?
Testing

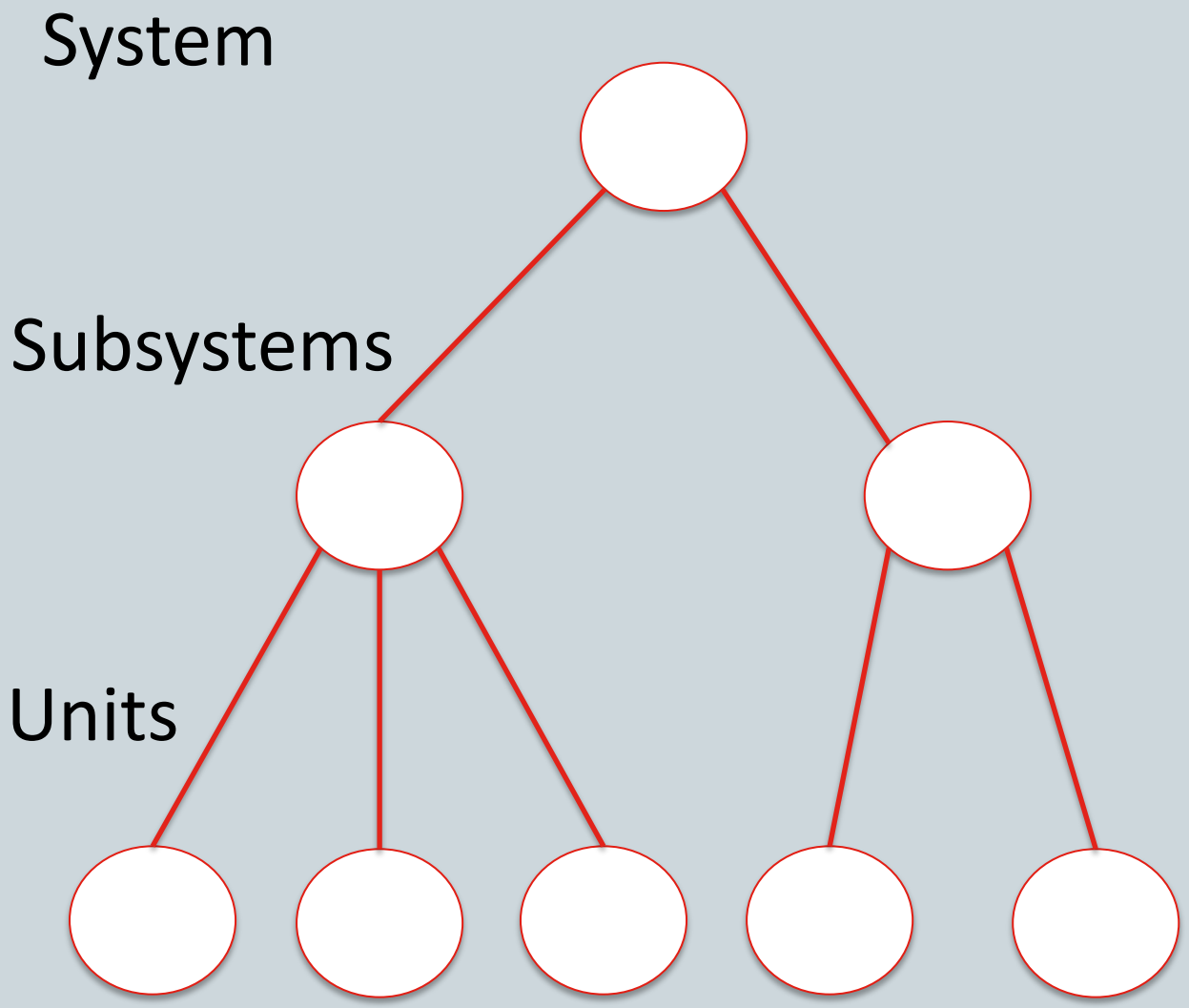


What?
Testing: Validation and Verification

- **Validation:** Have we made the right product; compliance with the intended usage (often: user-centered, manual process, on the end product)
- **Verification:** Have we made the product right; compliance between artifacts of different phases (often: artifact-driven, formalizable and mechanizable process among all phases)

Our Focus

What?
Levels of Testing

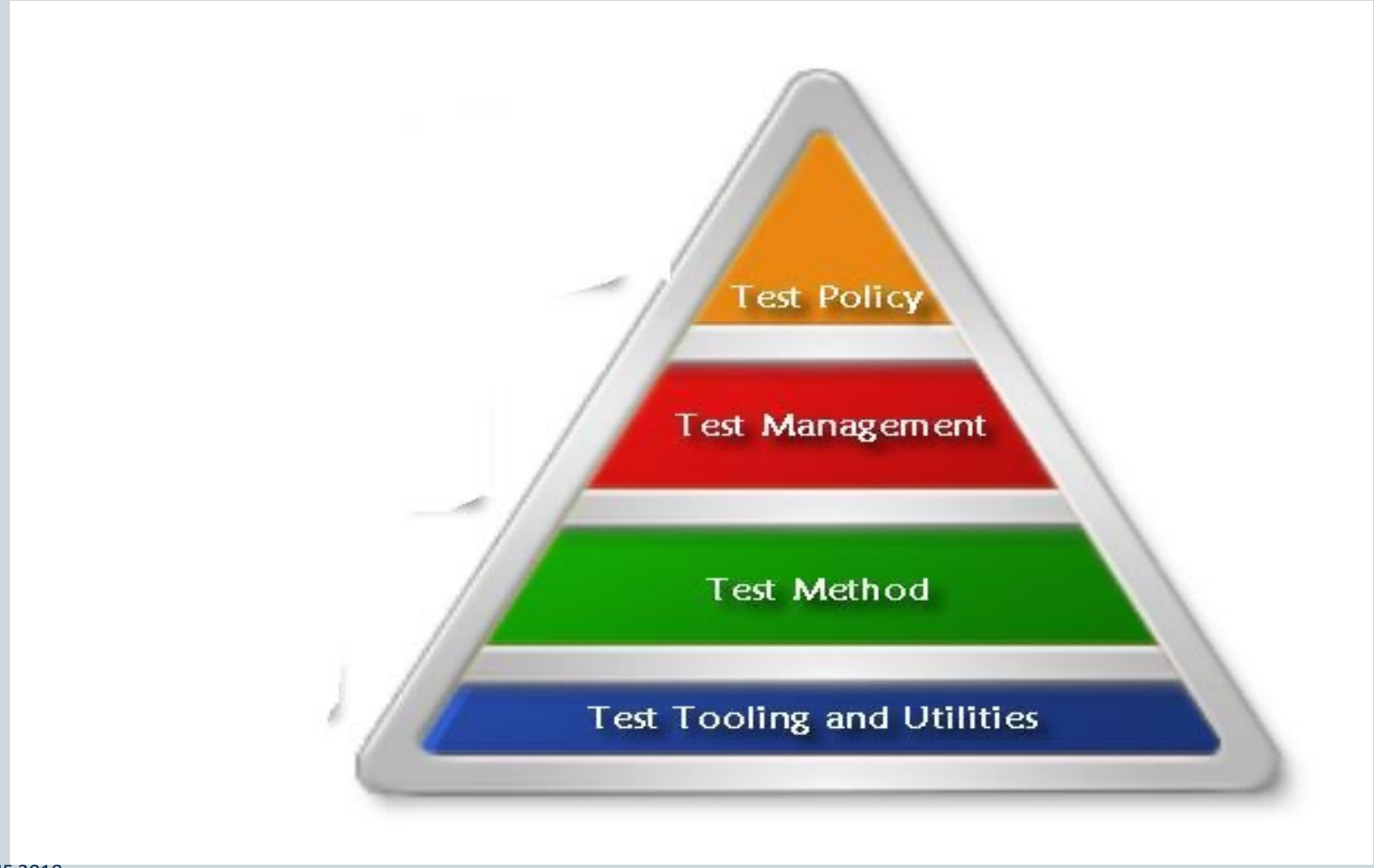


System Testing (Acceptance, User)
Selenium
GUI Model Testing,
Automated GUI Testing

Integration Testing
jUnit, Mockito
Dependency Injection
Mocking

Unit Testing
jUnit, QuickCheck
Test-Driven Dev.
Equivalence Partitioning,
Decision Tables,
Classification Trees

What We Do Not Cover:
Test Management and Policy



Alternatives to Testing

- **Model Checking:** test the state-space (all executions) for formally specified properties

  \+ rigorous analysis, push-button technology

  \- not (yet) scalable to very large systems (state-space explosion)

Alternatives to Testing

- **Static Analysis:** test abstract properties without running the program, e.g., division by zero and empty/unspecified cases

  \+ automatic and scalable for generic and abstract properties;
  \+ existing powerful tools;

  \- involves approximation (true negatives and false positives);
  \- complicated (may involve theorem proving) for concrete and specific properties (proving the abstraction function to be “correct")

Thank you

Mohammad Mousavi
mohammad.mousavi@kcl.ac.uk