CSE306 Software Quality in Practice

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

Recall the rules

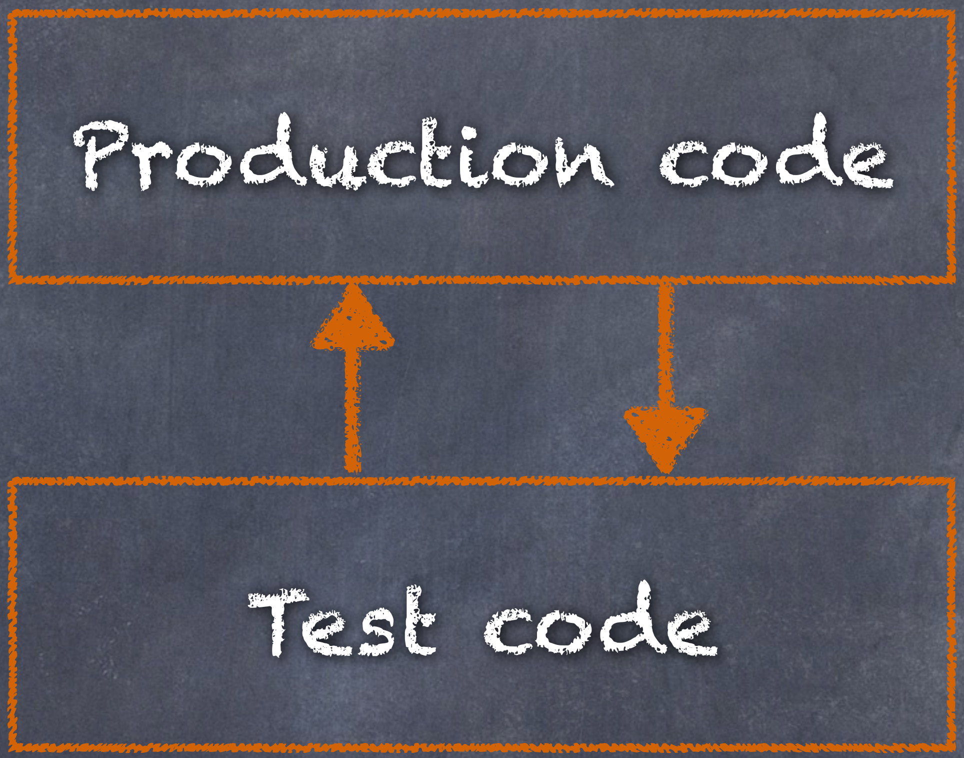
1. Understand the requirements
2. Make it fail
3. Simplify the test case
4. Read the right error message
5. Check the plug
6. Separate fact from fiction
7. Divide and conquer
8. Match the tool to the bug
9. One change at a time
10. Keep an audit trail
11. Get a fresh view
12. If you didnt fix it, it aint fixed
13. Cover your bug fix with a regression test

Recall the rules

1. Understand the requirements
2. Make it fail
3. Simplify the test case
4. Read the right error message
5. Check the plug
6. Separate fact from fiction
7. Divide and conquer
8. Match the tool to the bug
9. One change at a time
10. Keep an audit trail
11. Get a fresh view
12. If you didnt fix it, it aint fixed
13. Cover your bug fix with a regression test

Unit testing frameworks

- uniform way of expressing tests
- manage tests through suites
- automate testing process



Production code

Test code

Test code is separate from production code, but calls production code to verify its functionality.

Unit Testing frameworks

https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

CUnit - C

Criterion - C/C++

JUnit - Java

Mocha - JavaScript

pytest - Python

ScUnit - Scala

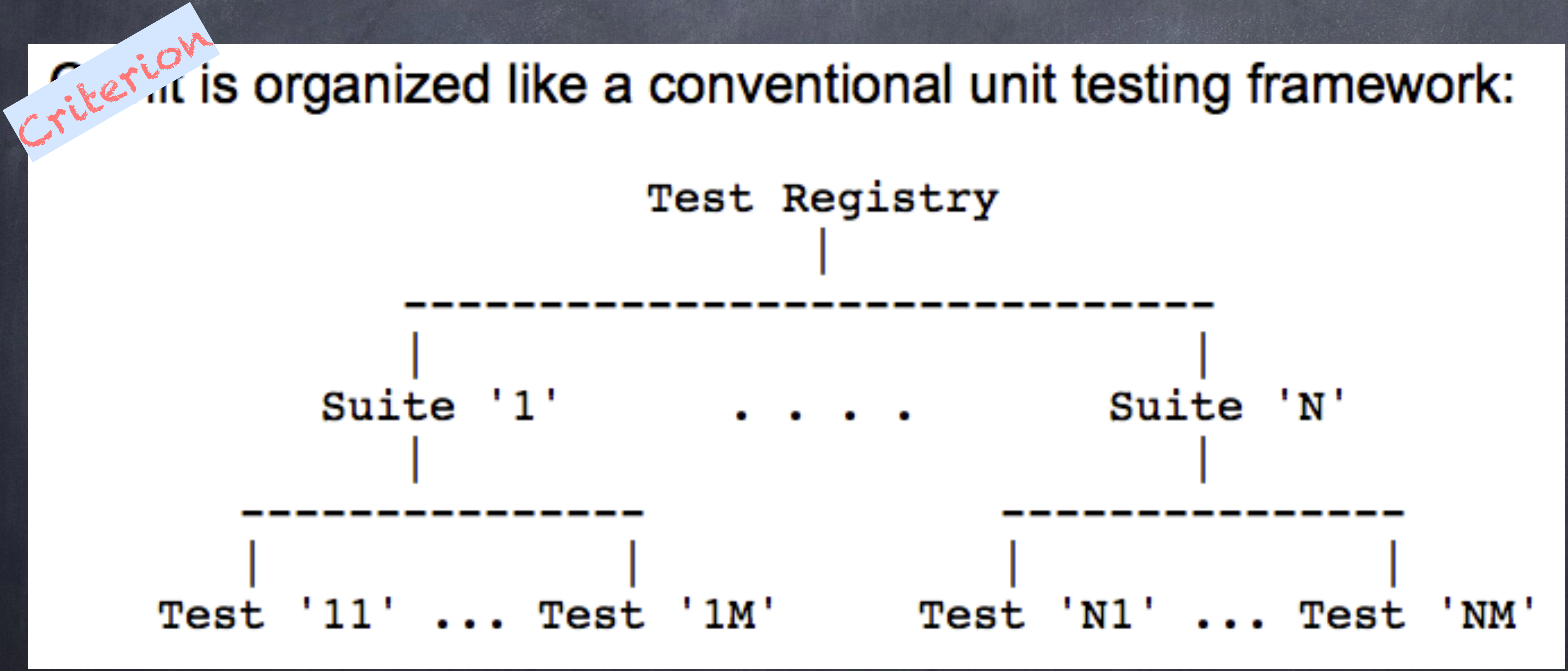
VUnit - Verilog/VHDL

among many, many others

We'll use Criterion

https://criterion.readthedocs.io

https://github.com/Snaipe/Criterion



Criterion

CUnit is organized like a conventional unit testing framework:

```
                      Test Registry
                            |
             ------------------------------
             |                            |
          Suite '1'      . . . .       Suite 'N'
             |                            |
       ---------------             ---------------
       |             |             |             |
    Test '11' ... Test '1M'     Test 'N1' ... Test 'NM'
```

http://cunit.sourceforge.net/doc/introduction.html

- Tests are automatically registered when declared.
- Implements a xUnit framework structure.
- A default entry point is provided, no need to declare a main unless you want to do special handling.
- Test are isolated in their own process, crashes and signals can be reported and tested.

From: https://github.com/Snaipe/Criterion https://github.com/Snaipe/Criterion#readme

Assertions

(the most common ones)

https://criterion.readthedocs.io/en/master/assert_old.html

https://criterion.readthedocs.io/en/master/assert.html

Demo

Write tests for a function named eval which takes two int values (x and y) and returns their sum as an int.