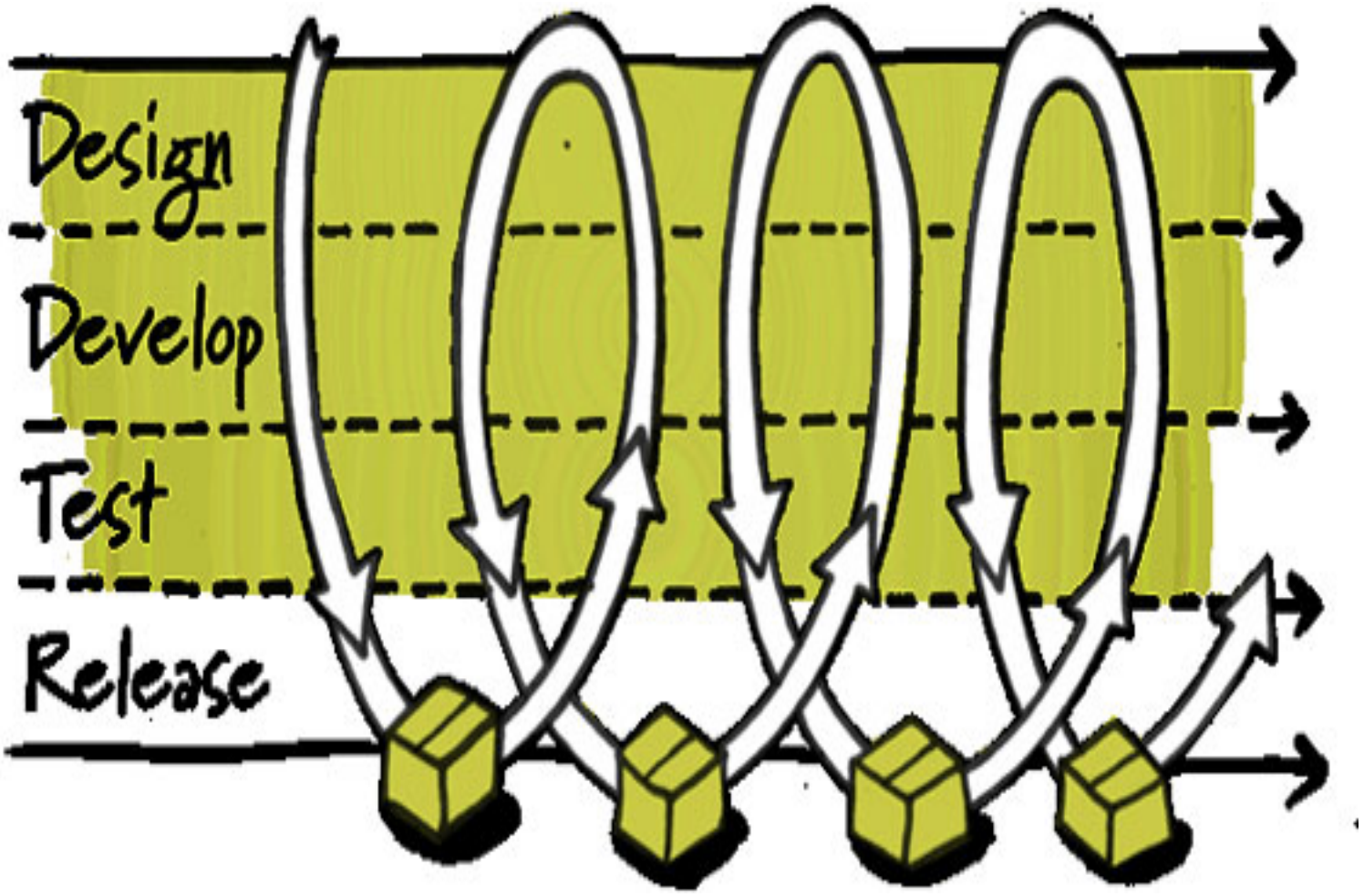
Introduction to Software Testing

CS480 Software Engineering

Yu Sun, Ph.D.
http://yusun.io
yusun@cpp.edu

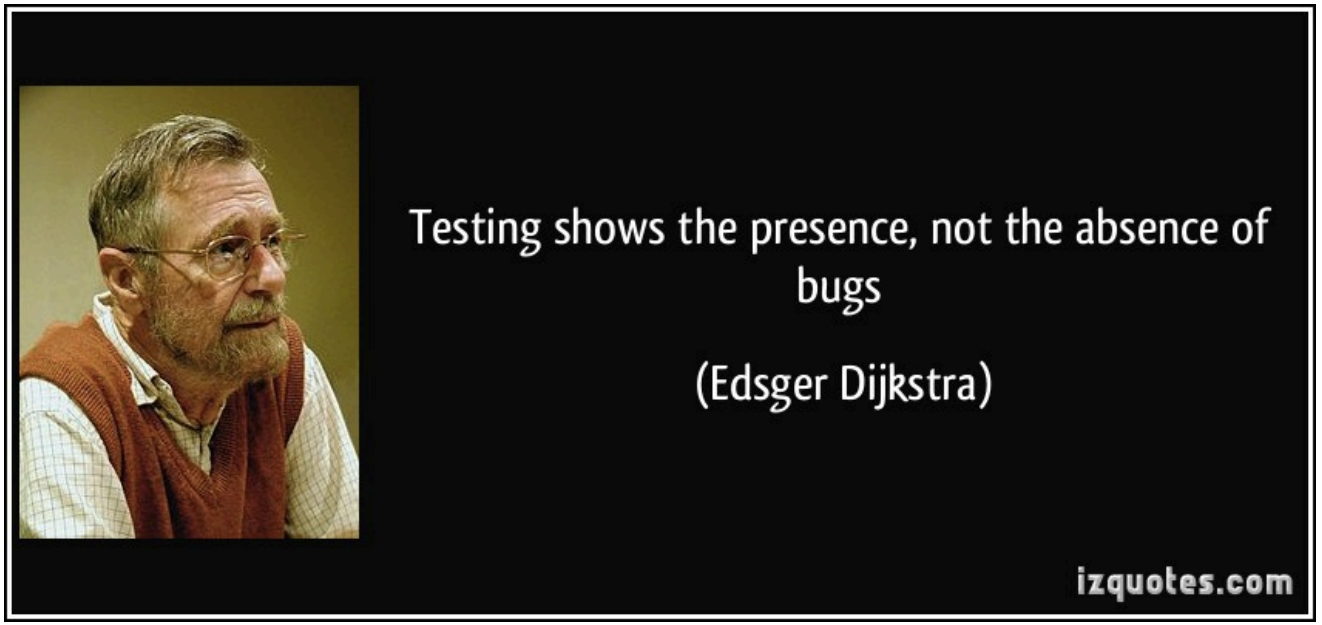
CAL POLY POMONA

Test in Agile Development



Program Testing

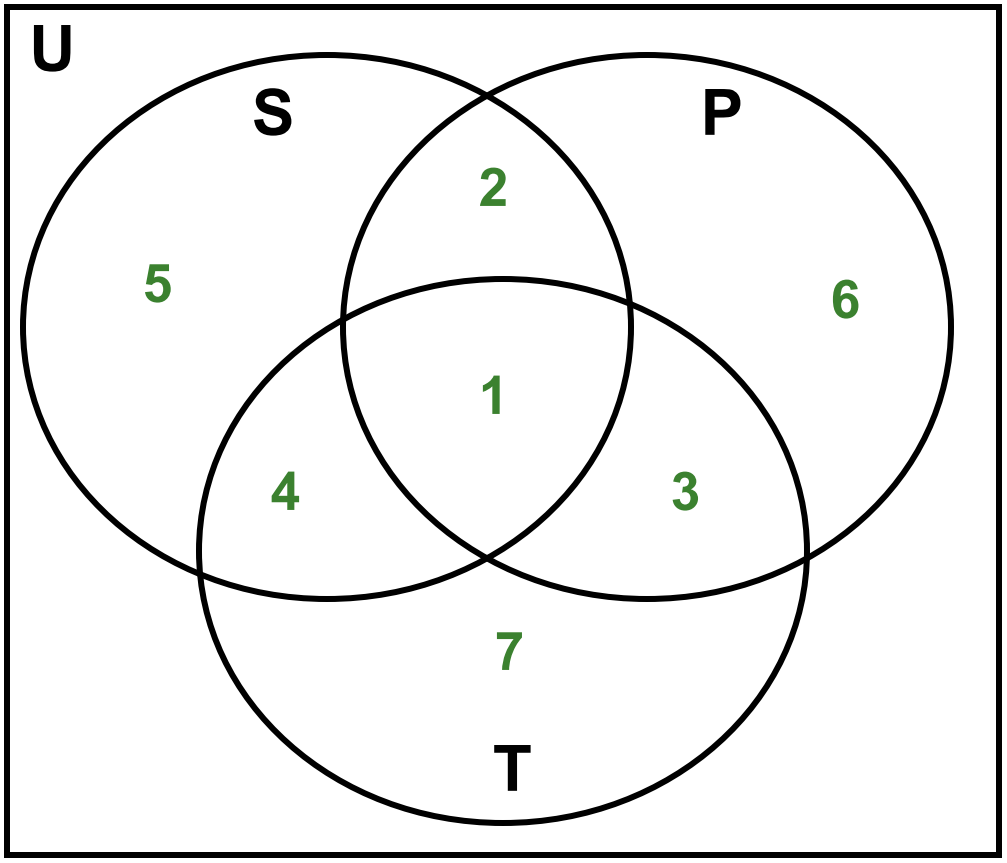
- Can reveal the presence of errors NOT their absence
  - Only exhaustive testing can show a program is free from defects
  - Exhaustive testing for anything but trivial programs is impossible



Program Testing

- Can reveal the presence of errors NOT their absence
  - Only exhaustive testing can show a program is free from defects
  - Exhaustive testing for anything but trivial programs is impossible
- A successful test discovers one or more errors

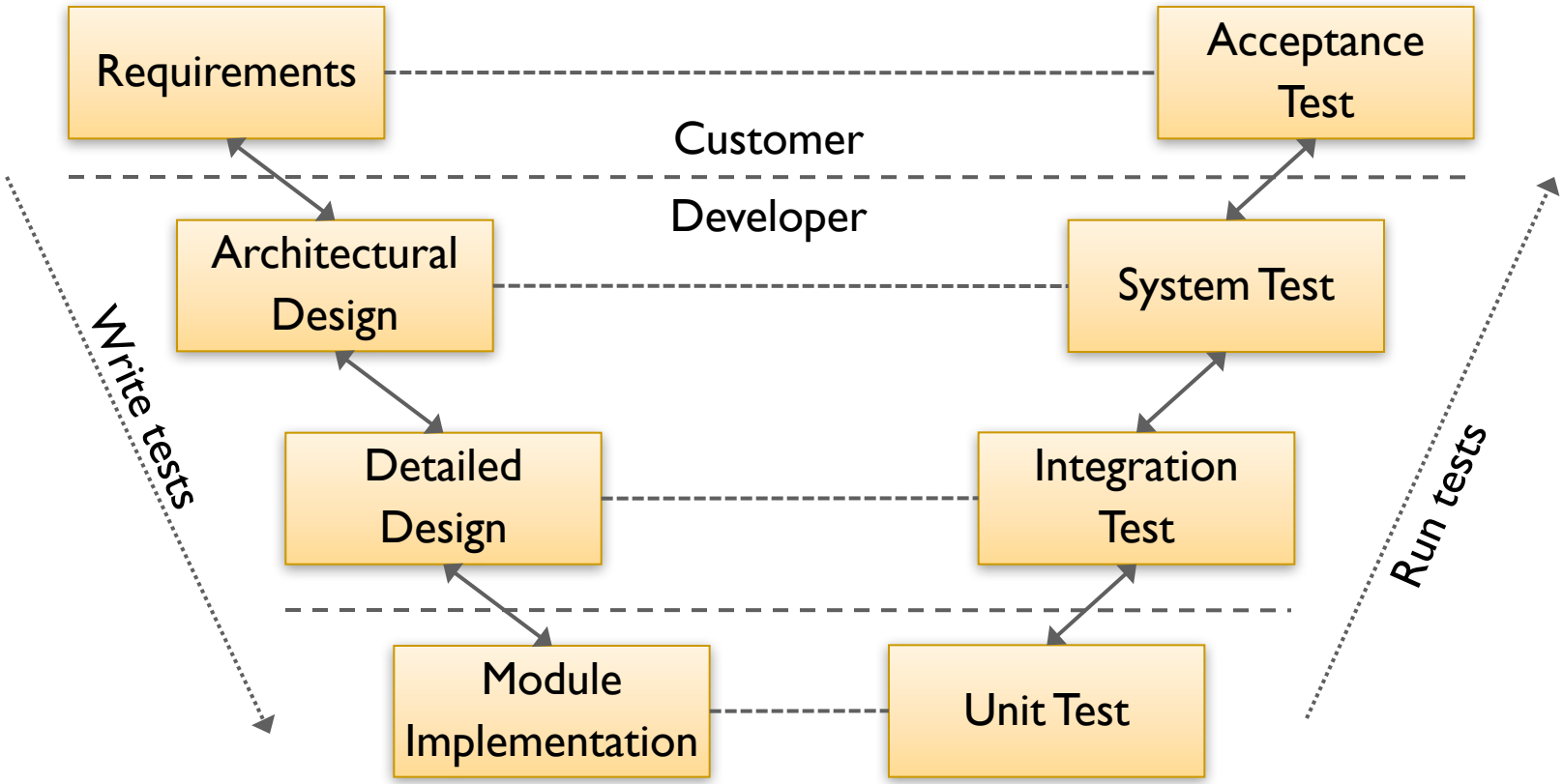
Specified, Programmed, and Tested Behaviors



S = Specified behaviors
P = Programmed behaviors
T = Tested behavior
U = All possible behaviors

We want to make region 1 as large as possible

The V-model of Testing



Testing Stages

- Unit testing
  - Testing of individual components
- Integration testing
  - Testing to expose problems arising from the combination of components
- System testing
  - Testing the complete system prior to delivery
- Acceptance testing
  - Testing by users to check that the system satisfies requirements

Testing Stages

- Alpha testing
  - When a product is used by many users, an alpha test is conducted in a controlled environment at the development site with end-user participation
- Beta testing
  - An extension to alpha testing where the users test the software in a "live" environment; developers are typically not present

Distinction Between Debugging and Testing

- Defect testing and debugging are distinct processes
- Defect testing is concerned with confirming the presence of errors
- Debugging is concerned with locating and repairing these errors
- Debugging involves formulating a hypothesis about program behavior then exploring these hypotheses to find the system error

Historical Views: Thinking About Testing

- Phase 0
  - Testing = Debugging
- Phase 1
  - Testing is an act whose purpose is to show that the software works
- Phase 2
  - Testing is an act whose purpose is to show that the software does not work

Historical Views: Thinking About Testing

- Phase 3
  - Testing is an act whose purpose is not to prove anything, but to reduce the perceived risk of failure to an acceptable level

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

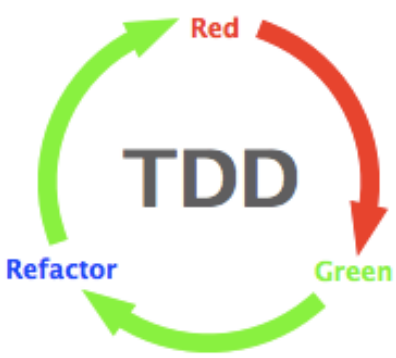
All Prime Numbers (1-100)

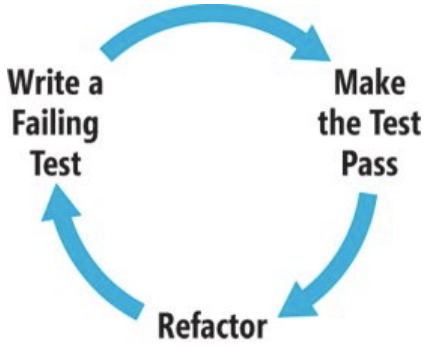
Availability	Downtime Per Year (24X7X365)
99.000%	3 Days 15 Hours 36 Minutes
99.500%	1 Day 8 Hours 48 Minutes
99.900%	8 Hours 46 Minutes
99.950%	4 Hours 23 Minutes
99.990%	53 Minutes
99.999%	5 Minutes
99.9999%	30 Seconds

Service Level Agreement (SLA)

Historical Views: Thinking About Testing

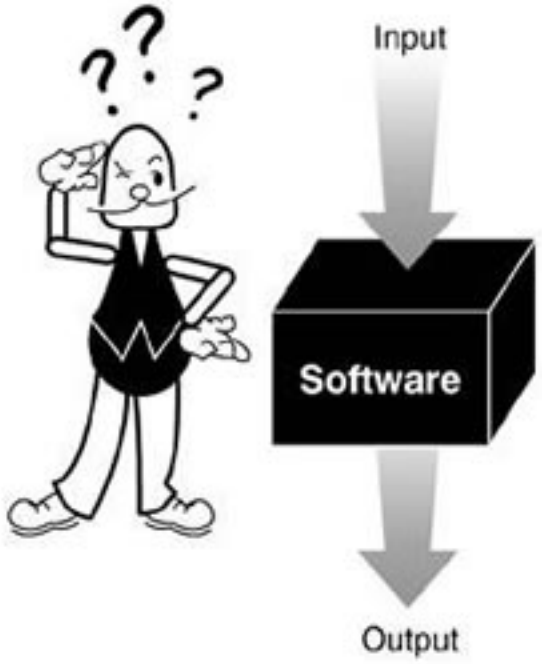
- Phase 3
  - Testing is an act whose purpose is not to prove anything, but to reduce the perceived risk of failure to an acceptable level
- Phase 4
  - Testing is not an act; rather, it is a mindset that involves development and coding practices along with a systematic approach to exercising the software





Testing Methods

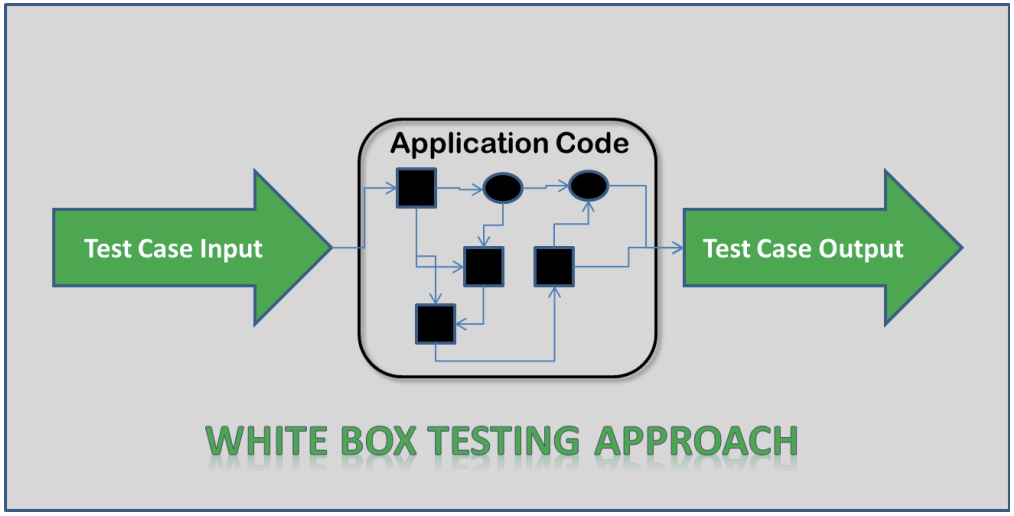
- Functional (Black Box) Testing
  - Knowing the specified functions that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational
  - Test cases are based on external behavior
  - Aka: specification-based, data-driven, or input/output driven testing



Black-Box Testing

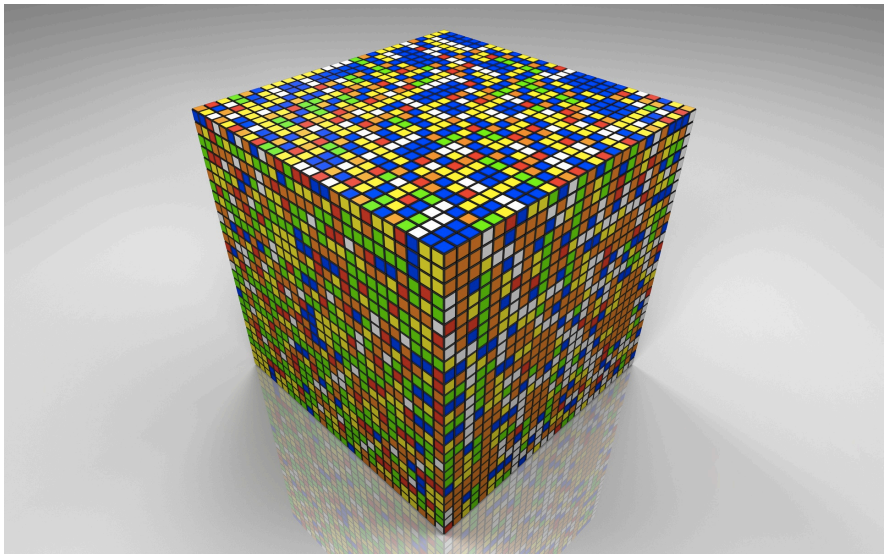
Testing Methods

- Structural (White Box) Testing
  - Knowing the internal workings of a program, tests can be conducted to assure that the internal operation performs according to specification, and all internal components have been exercised
  - Test cases are based on internal structure of the program and a specific level of coverage.



Feasibility of Black-Box Testing

- Suppose specs include 20 factors, each taking on 4 values
  - 4^{20} or 1.1×10^{12} test cases
  - If each takes 30 seconds to run, running all test cases takes > 1 million years
- Combinatorial explosion makes exhaustive testing to specifications impossible



Feasibility of White-Box Testing

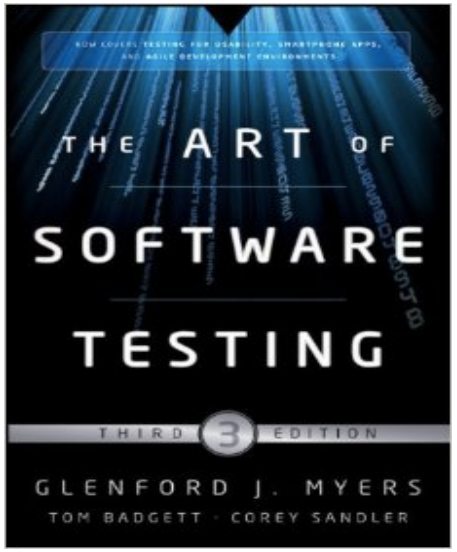
- Can exercise every path without detecting every fault (what if x=2, y=1, z=3?)

```
if ((x + y + z)/3 == x)
   print "x, y, z are equal in value";
else
   print "x, y, z are unequal";
```

Test case 1: $x = 1, y = 2, z = 3$
Test case 2: $x = y = z = 2$

Coping with the Combinatorial Explosion

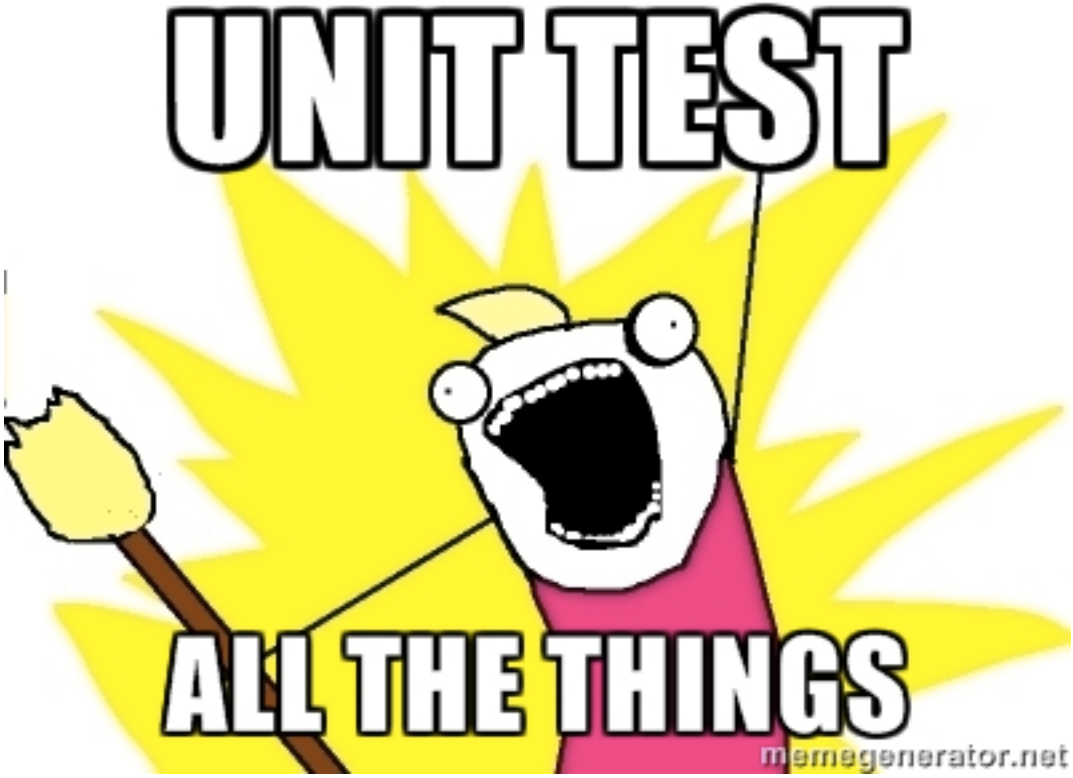
- Neither testing to specifications nor testing to code is feasible toward ensuring complete correctness
- The art of testing
  - Select a small, manageable set of test cases to
    - Maximize chances of detecting fault, while
    - Minimizing chances of wasting test case
  - Every test case must detect a previously undetected fault



Coping with the Combinatorial Explosion

- We need a method that will highlight as many faults as possible
  - First black-box test cases (testing to specifications)
  - Then white-box methods (testing to code)

Unit Test - jUnit

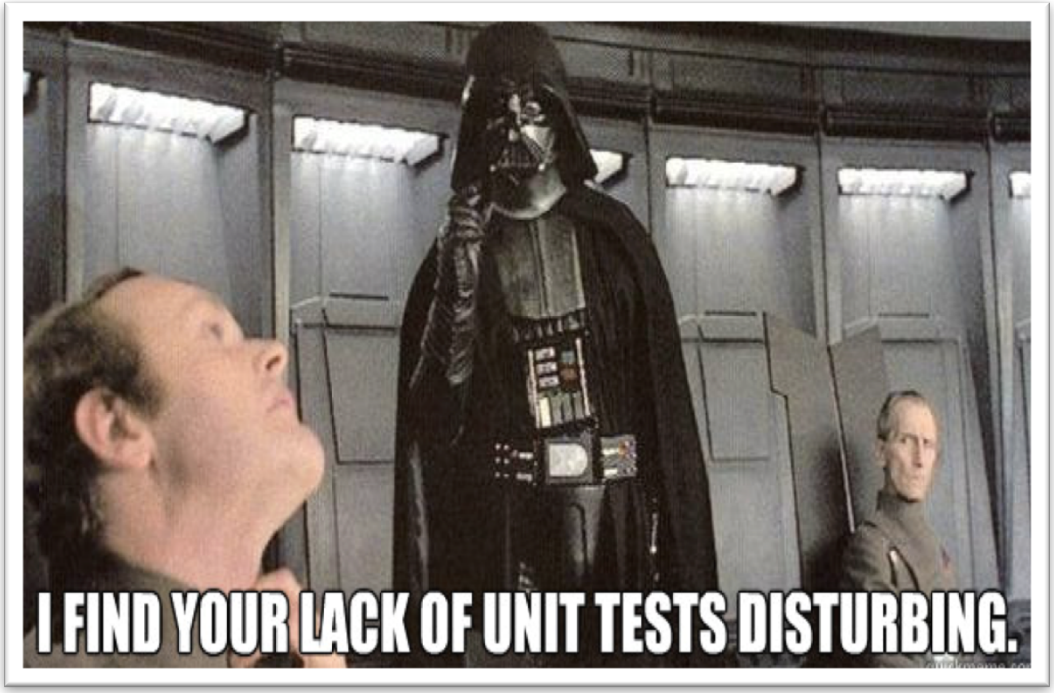


The Most Fundamental Testing Step

- The smallest test unit
- Test every single function

A MUST-HAVE Skill for Developers

- You are required to write unit test for every change you made
- There is no way that you can skip the process, because your code will be reviewed by your peer developers



From JUnit to xUnit

- JUnit is a modern and mature testing framework
- Learning JUnit helps learning xUnit
  - ASP, C++, C#, Eiffel, Delphi, Perl, PHP, Python, REBOL, Smalltalk, and Visual Basic

JUnit

JUnit Resources

- http://junit.org
- Petar Tahchiev, Felipe Leme, Vincent Massol, and Gary Gregory. *JUnit in Action*. Manning Publications Co., 2010.
- Latest version: JUnit 4.12
- Source code: https://github.com/junit-team/junit