Tests Automation

Let’s write a test

```
class CreditCardSanitizer {

  boolean luhnCheck(String text) {

  }

}
```

```
class CreditCardSanitizerTest {

  private CreditCardSanitizer sanitizer;

  @Before public void init() {
    sanitizer = new CreditCardSanitizer();
  }

  @Test public void validCard() {
    assertTrue(sanitizer
        .luhnCheck("378282246310005"));
  }

}
```

*JUnit 4 test

Why?..

Why?..

do we waste our time…

Because you don’t!

You need to test anyway!!!

You avoid manual testing

Dot not let machines laugh at you! Automate!

More tests

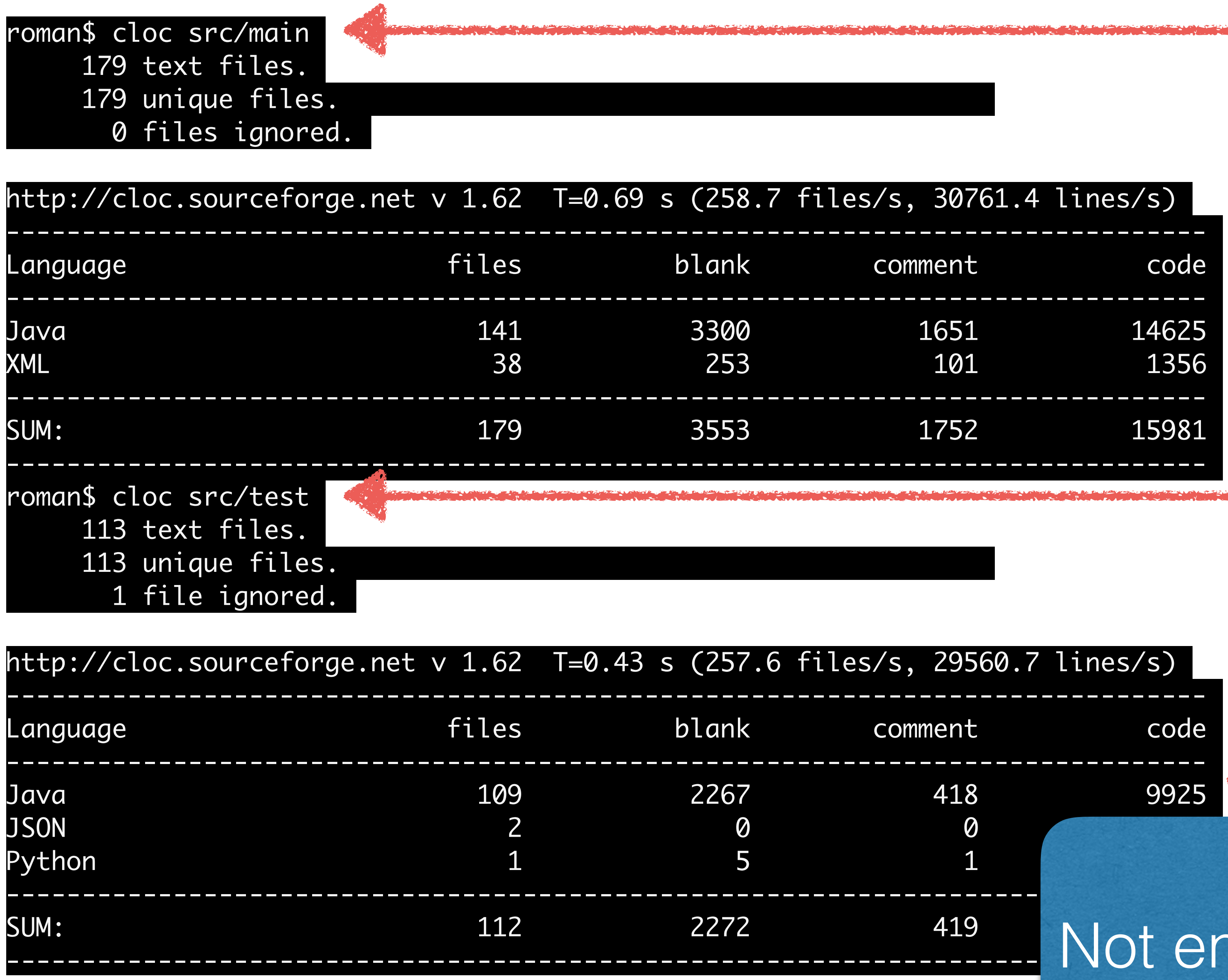
```
class CreditCardSanitizerTest {

  // ...

  @Test public void validCard() {
    assertTrue(sanitizer
        .luhnCheck("378282246310005"));
  }

  @Test public void invalidCard() {
    assertFalse(sanitizer
        .luhnCheck("378282246315"));
  }

}
```

*JUnit 4 test

Make it easy to run

- mvn test
- gradle test
- python -m pytest

Some statistics



```
roman$ cloc src/main
     179 text files.
     179 unique files.
       0 files ignored.

http://cloc.sourceforge.net v 1.62  T=0.69 s (258.7 files/s, 30761.4 lines/s)
--------------------------------------------------------------------------------
Language                     files          blank        comment           code
--------------------------------------------------------------------------------
Java                           141           3300           1651          14625
XML                             38            253            101           1356
--------------------------------------------------------------------------------
SUM:                           179           3553           1752          15981
--------------------------------------------------------------------------------
roman$ cloc src/test
     113 text files.
     113 unique files.
       1 file ignored.

http://cloc.sourceforge.net v 1.62  T=0.43 s (257.6 files/s, 29560.7 lines/s)
--------------------------------------------------------------------------------
Language                     files          blank        comment           code
--------------------------------------------------------------------------------
Java                           109           2267            418           9925
JSON                             2              0              0
Python                           1              5              1
---------------------------------------------------------------------------
SUM:                           112           2272            419
---------------------------------------------------------------------------
```

Main code

14625

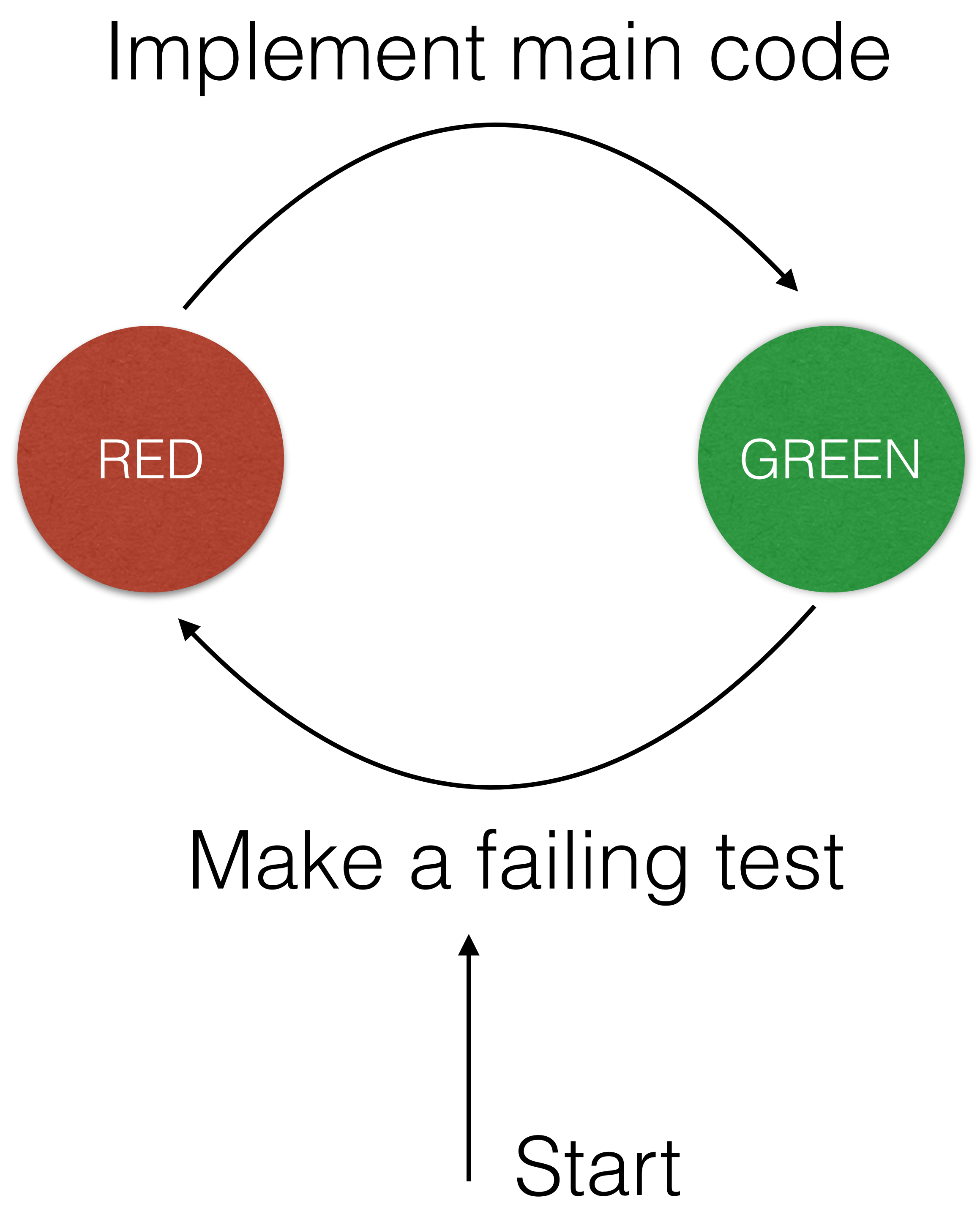
Test code

9925

Not enough!

40%

TDD



“In the beginning there was nothing. And a test was written.”

Dev performance: fluent API

```
assertThat(sanitizer.luhnCheck(“123”)).isFalse();

assertThat(sanitizer.process(“card 378282246310005”))
            .contains(“XXXXXXXXXXX0005”)
```

*AssertJ

See how your code will be used

```
File in = new File(“test-in”);
File out = new File(“test-out”);
CreditCardSanitizer sanitizer = new CreditCardSanitizer(
    new FileReader(in)
)
sanitizer.process(out);
assertThat(out).doesNotContainText(“378282246310005”);
```

Test is documentation

```
RSpec.describe Order do
  it "sums the prices of its line items" do
    order = Order.new

    order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(1.11, :USD)
    )))
    order.add_entry(LineItem.new(:item => Item.new(
      :price => Money.new(2.22, :USD),
      :quantity => 2
    )))

    expect(order.total).to eq(Money.new(5.55, :USD))
  end
end
```

*AssertJ

Test is documentation

```
def "offered PC matches preferred configuration"() {
  when:
  def pc = shop.buyPc()

  then:
  pc.vendor == "Sunny"
  pc.clockRate >= 2333
  pc.ram >= 4096
  pc.os == "Linux"
}
```

*AssertJ

BDD with mocks

```
Server server = mock(Server.class);
doReturn(500).when(server).getTodoList();

Todo todo = new Todo(server);

assertThat(todo.sync()).isFalse();
verify(server).getTodoList();
```

*Mockito

Tests benefits

- Automation
- Documentation/specification
- Better encapsulation

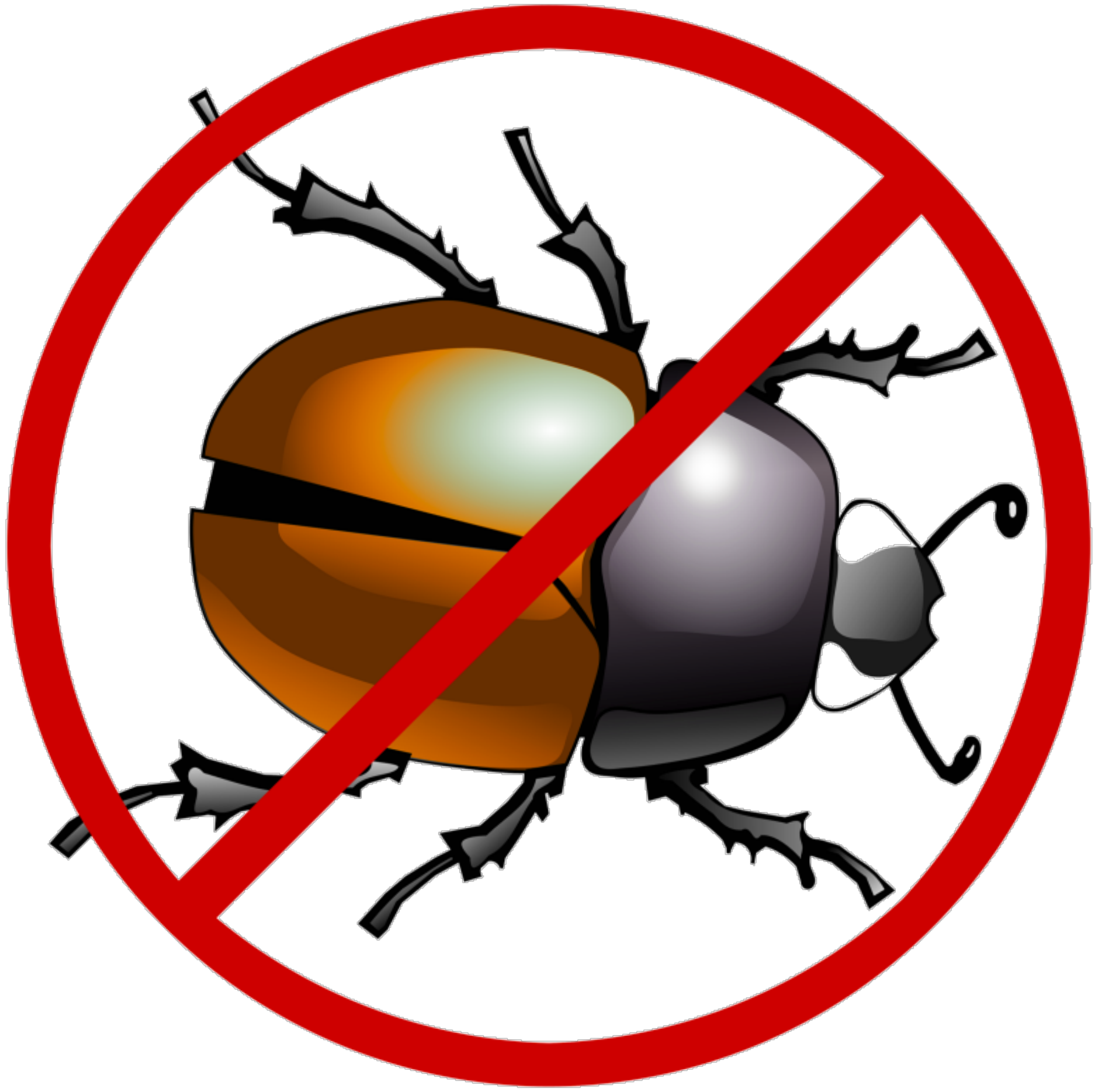
“Knock knock.
Race condition.
Who's there?”

Tests must be deterministic

Tests must be deterministic

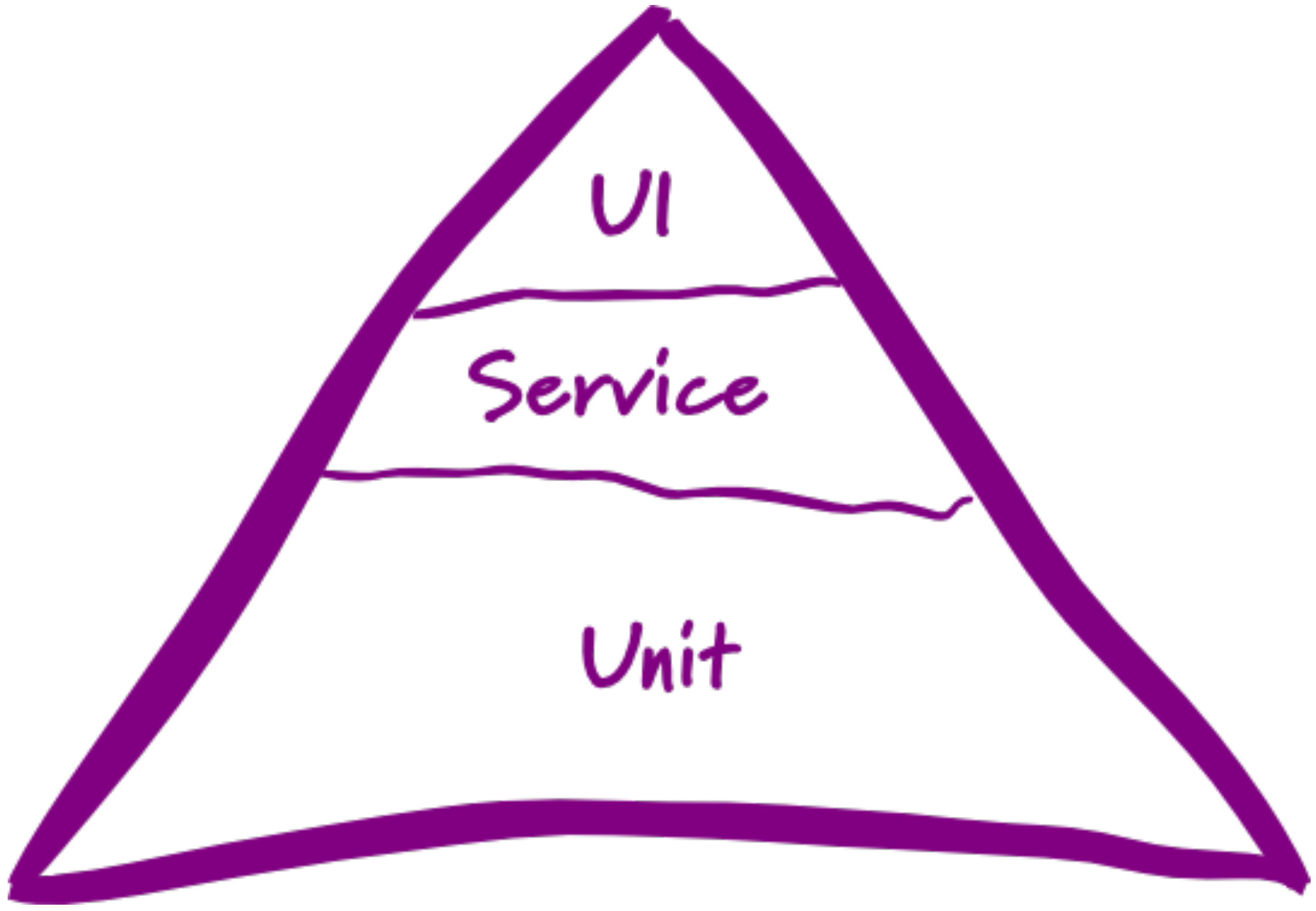
- Prefer single thread
- Mock workers/schedulers, control execution in tests

Start bug fix with a test

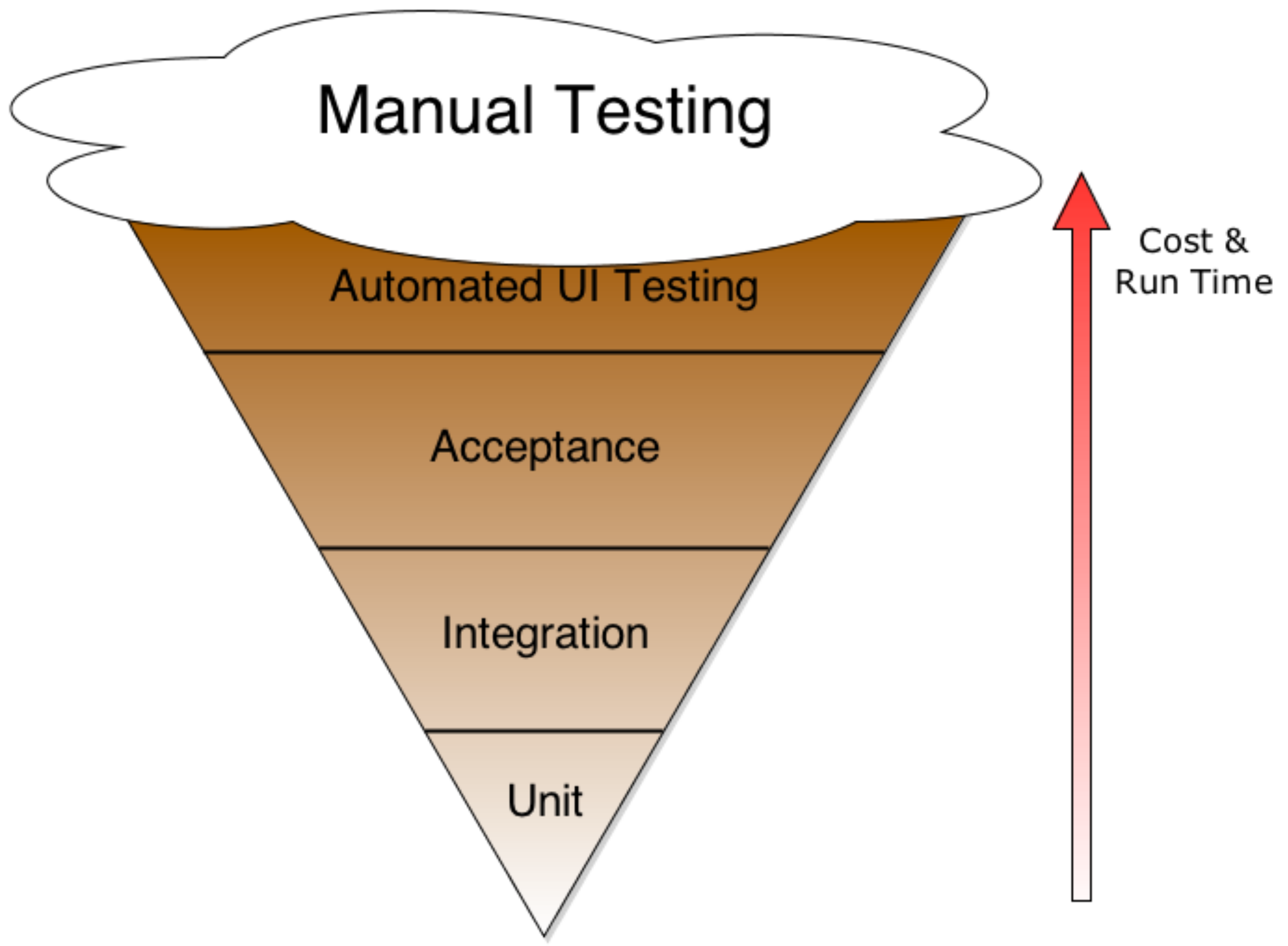


Regression tests

DO



DON'T


Manual Testing
Automated UI Testing
Acceptance
Integration
Unit
Cost &
Run Time

Test