Developer Testing

welcome to the beta test

Jay Fields

Assumptions

Assumptions

your tests are unmaintainable

Assumptions

your tests are unmaintainable

my tests are unmaintainable

Assumptions

your tests are unmaintainable

my tests are unmaintainable

in context, our ideas are solid

Assumptions

your tests are unmaintainable

my tests are unmaintainable

in context, our ideas are solid

out of context, our ideas don't seem to fit or make sense

Assumptions

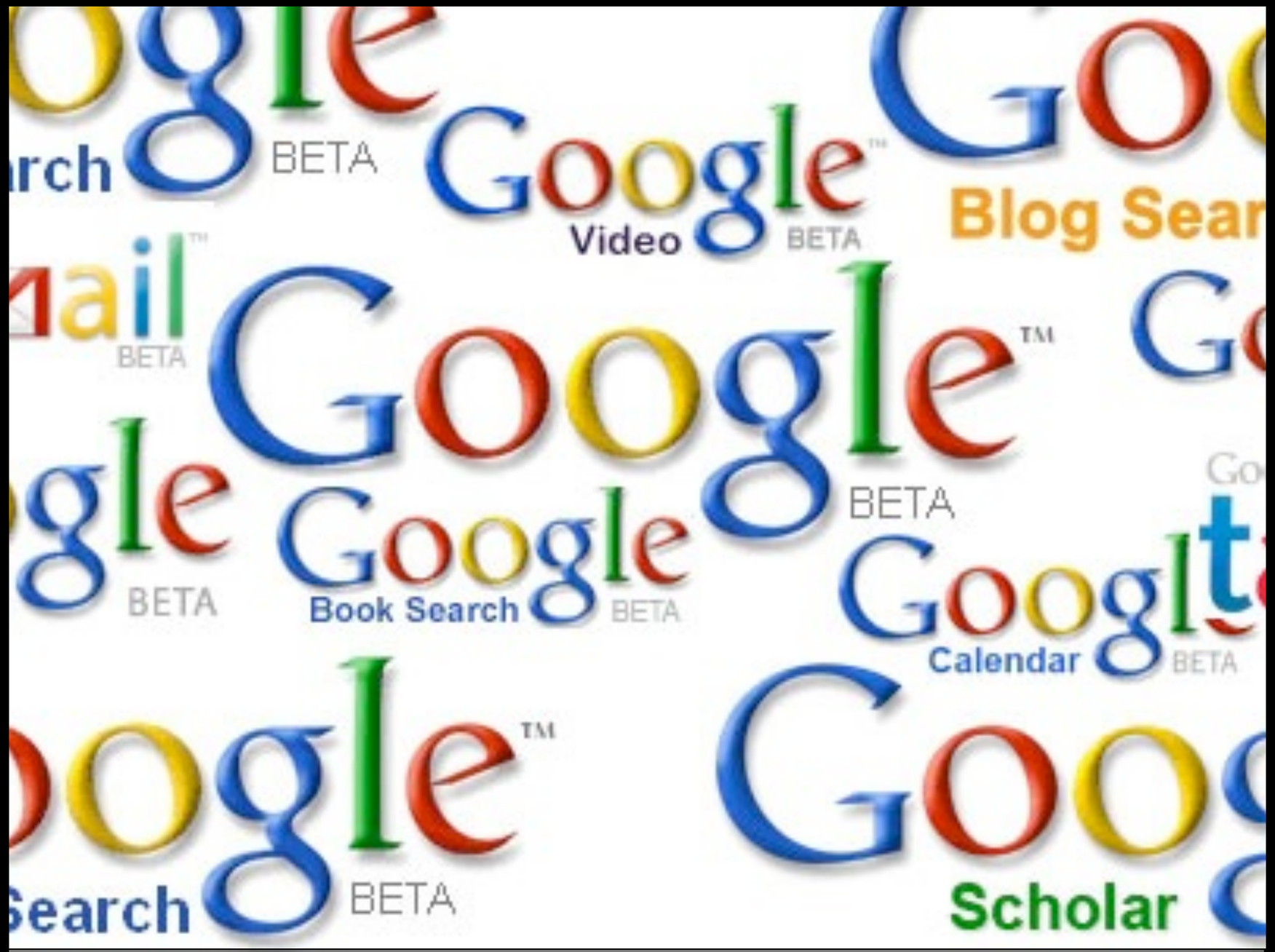
your tests are unmaintainable

my tests are unmaintainable

in context, our ideas are solid

out of context, our ideas don't seem to fit or make sense

we care about this significantly more than most people, and don't have it figured out


BETA
Google
Video
BETA
Blog Sear
BETA
Google
BETA
BETA
Google
Book Search
BETA
Calendar
BETA
Google
Scholar
BETA

- Design
- Protect against regression
- Achieve sign-off
- Increase customer interaction
- Document the system
- Refactor confidently
- Ensure the system works correctly


WWW.HETEMEEL.COM
HA!
HA!
I'M PETE AXELROD,
AND I APPROVED THIS MESSAGE!

unit testing suggestions I follow

setup methods are evil

```
test "when attribute is not nil or empty, then valid is true" do
  validation = Validatable::ValidatesPresenceOf.new stub, :name
  assert_equal true, validation.valid?(stub(:name=>"book"))
end

test "when attribute is not nil or empty, then valid is true" do
  assert_equal true, @validation.valid?(@stub)
end
```

unit testing suggestions I follow

setup methods are evil

test one thing at a time

```
class PhoneNumberTest < Test::Unit::TestCase
  def test_initialize
    number = PhoneNumber.new "212", "555", "1212"
    assert_equal "212", number.area_code
    assert_equal "555", number.exchange
    assert_equal "1212", number.station
  end
end

# >> Loaded suite -
# >> Started
# >> F
# >> Finished in 0.006025 seconds.
# >>
# >>   1) Failure:
# >> test_initialize(PhoneNumberTest) [-:14]:
# >> <"212"> expected but was
# >> <nil>.
# >>
# >> 1 tests, 1 assertions, 1 failures, 0 errors
```

unit testing suggestions I follow

setup methods are evil

test one thing at a time

maintainability > no duplication

```
class PhoneNumberTest < Test::Unit::TestCase
  def test_area_code_is_initialized_correctly
    number = PhoneNumber.new "212", "555", "1212"
    assert_equal "212", number.area_code
  end

  def test_exchage_is_initialized_correctly
    number = PhoneNumber.new "212", "555", "1212"
    assert_equal "555", number.exchange
  end

  def test_station_is_initialized_correctly
    number = PhoneNumber.new "212", "555", "1212"
    assert_equal "1212", number.station
  end
end
```

unit testing suggestions I follow

setup methods are evil

test one thing at a time

maintainability > no duplication

test names are comments

```
expect Person.new.to.delegate(:save).to(:record)
  person.save(1)
end

expect Associate.new.to.have.id.nil?

expect Process.new.to.have.finished do |process|
  process.finished = true
end

expect Process.new.not.to.have.finished
```

unit testing suggestions I follow

setup methods are evil

test one thing at a time

maintainability > no duplication

test names are comments

zero or one mock per test

```
def test_instance_name_is_used_as_table_name
  instance = mock
  instance.expects(:name).returns('Foo')
  where = mock
  where.expects(:to_sql).returns("")
  builder = Statement.new(instance, where)
  assert_equal "select * from Foo", builder.buil
end
```

unit testing suggestions I follow

setup methods are evil

test one thing at a time

maintainability > no duplication

test names are comments

zero or one mock per test

expect literals

```
def test_popularity_with_variables
  votes = 2
  assert_equal WEIGHT * votes * votes,
    Topic.new(votes).popularity
end

def test_popularity_with_literals
  assert_equal 4.56,
    Topic.new(2).popularity
end
```

unit testing suggestions I follow

setup methods are evil

test one thing at a time

maintainability > no duplication

test names are comments

zero or one mock per test

expect literals

create test data builders

```
aNew().car().
  with(mock(Engine.class)).
  with(fake().radio().fm_only()).
  with(aNew().powerWindows()).
  build();
```

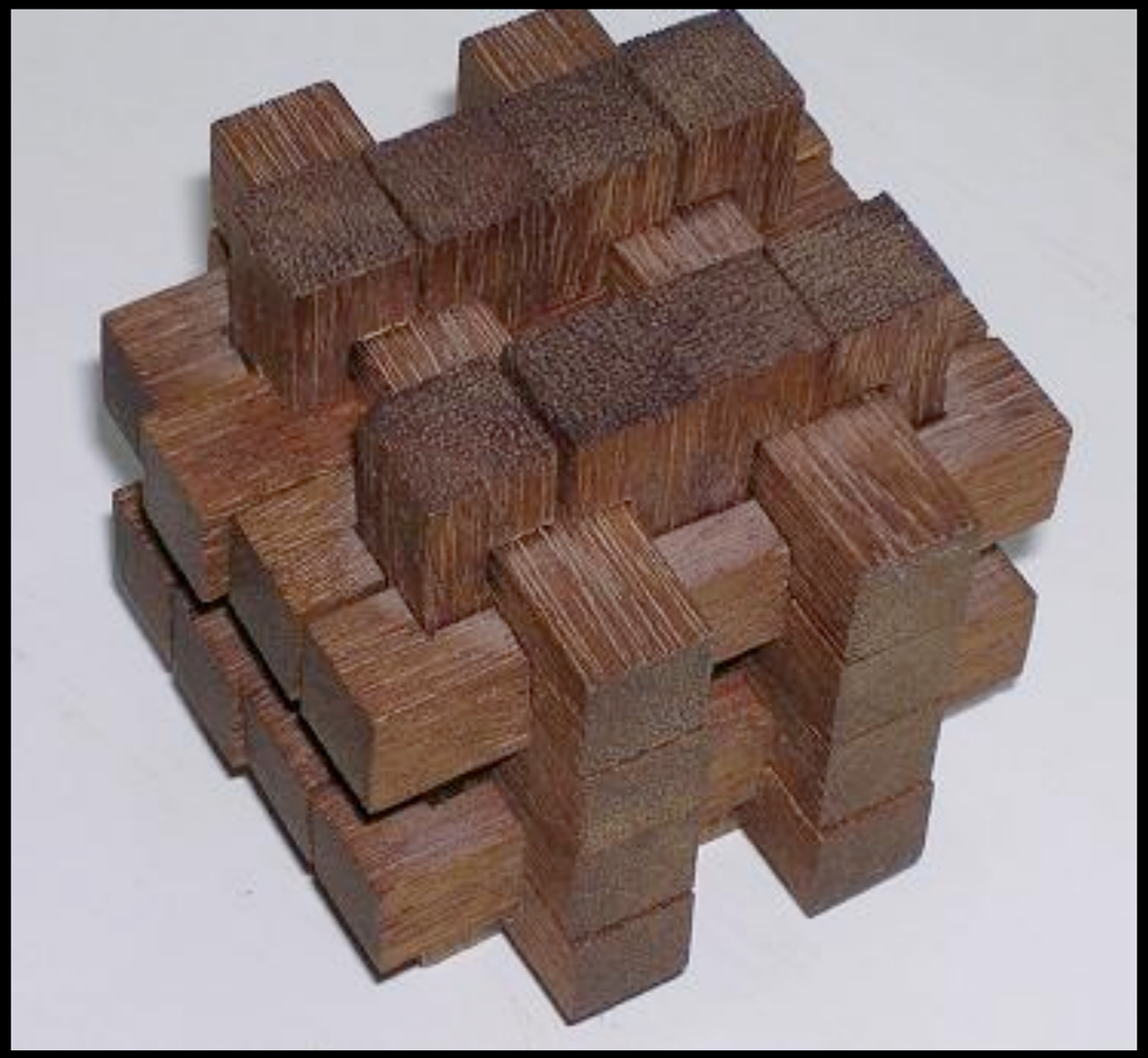

(gratuitous car porn)

readable, reliable, & performant unit tests

scoring tests

low scores are better

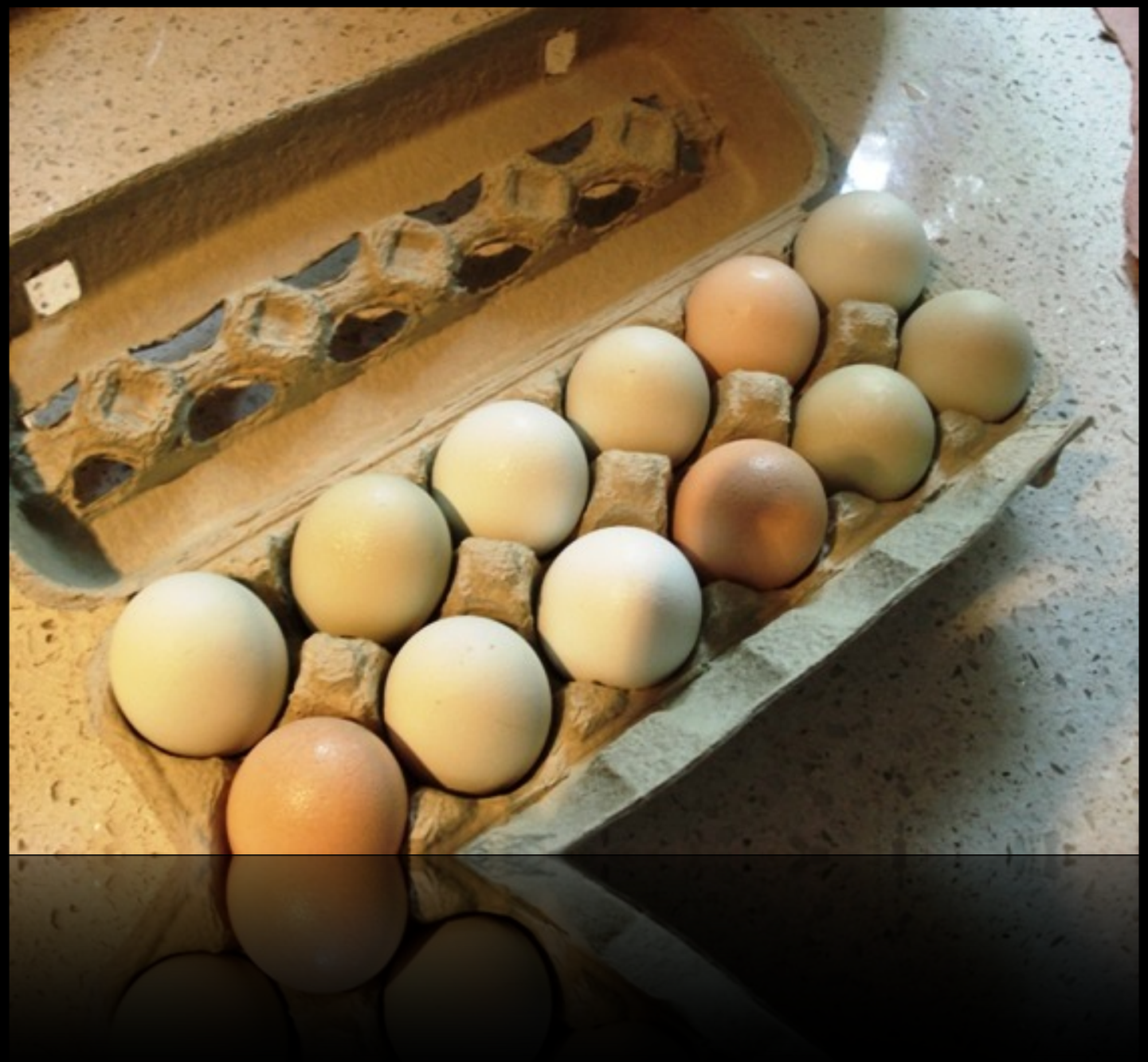
cycolmactic complexity
number of assertions
number of mock expectations
number of expected exceptions
number of local variables
number of helper methods



integration testing
suggestions I follow

integration testing suggestions I follow

a dozen or less tests



integration testing suggestions I follow

a dozen or less tests

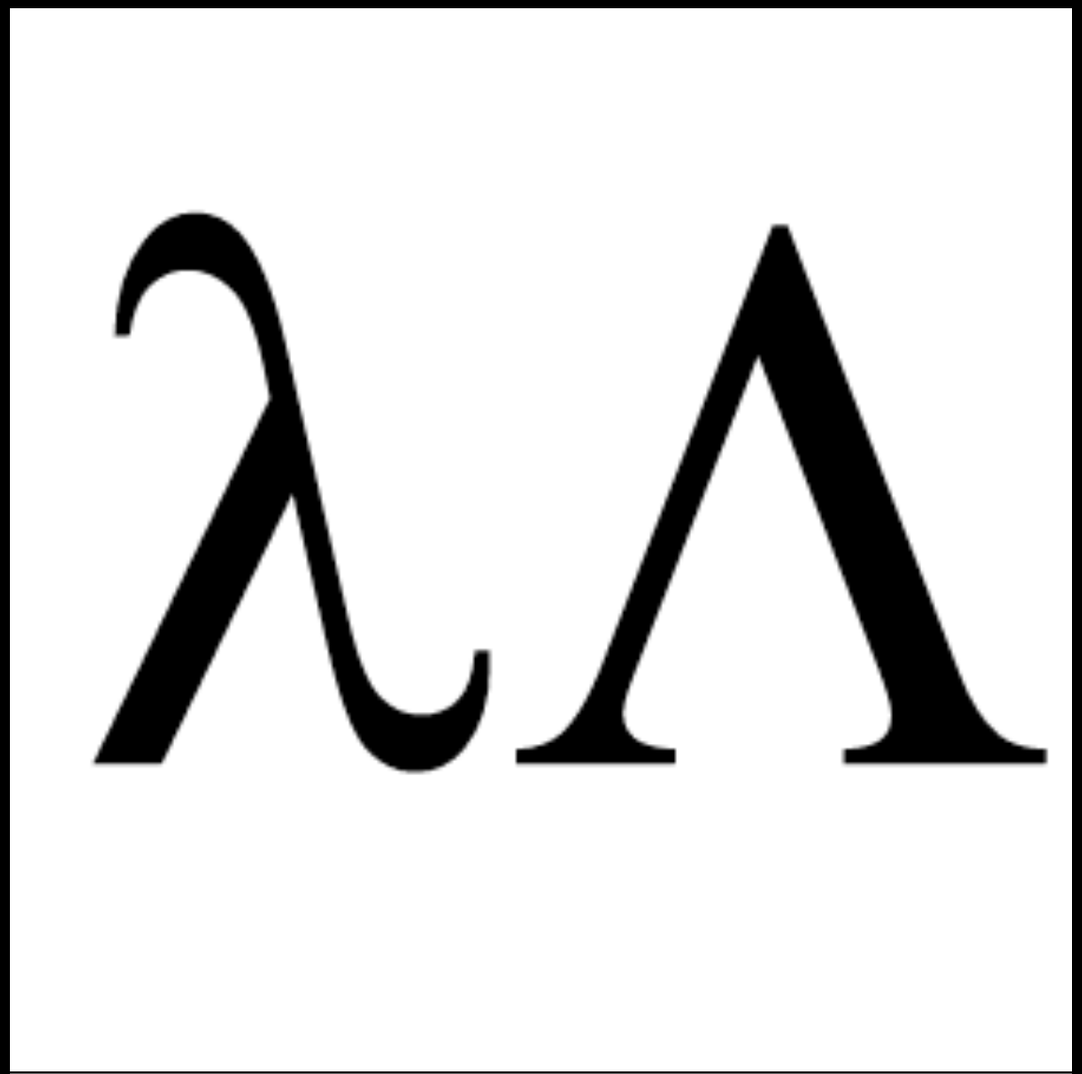
run as part of the build

integration testing suggestions I follow

a dozen or less tests

run as part of the build

use a powerful, high level language



integration testing suggestions I follow

a dozen or less tests

run as part of the build

use a powerful, high level language

stub external dependencies

integration testing suggestions I follow

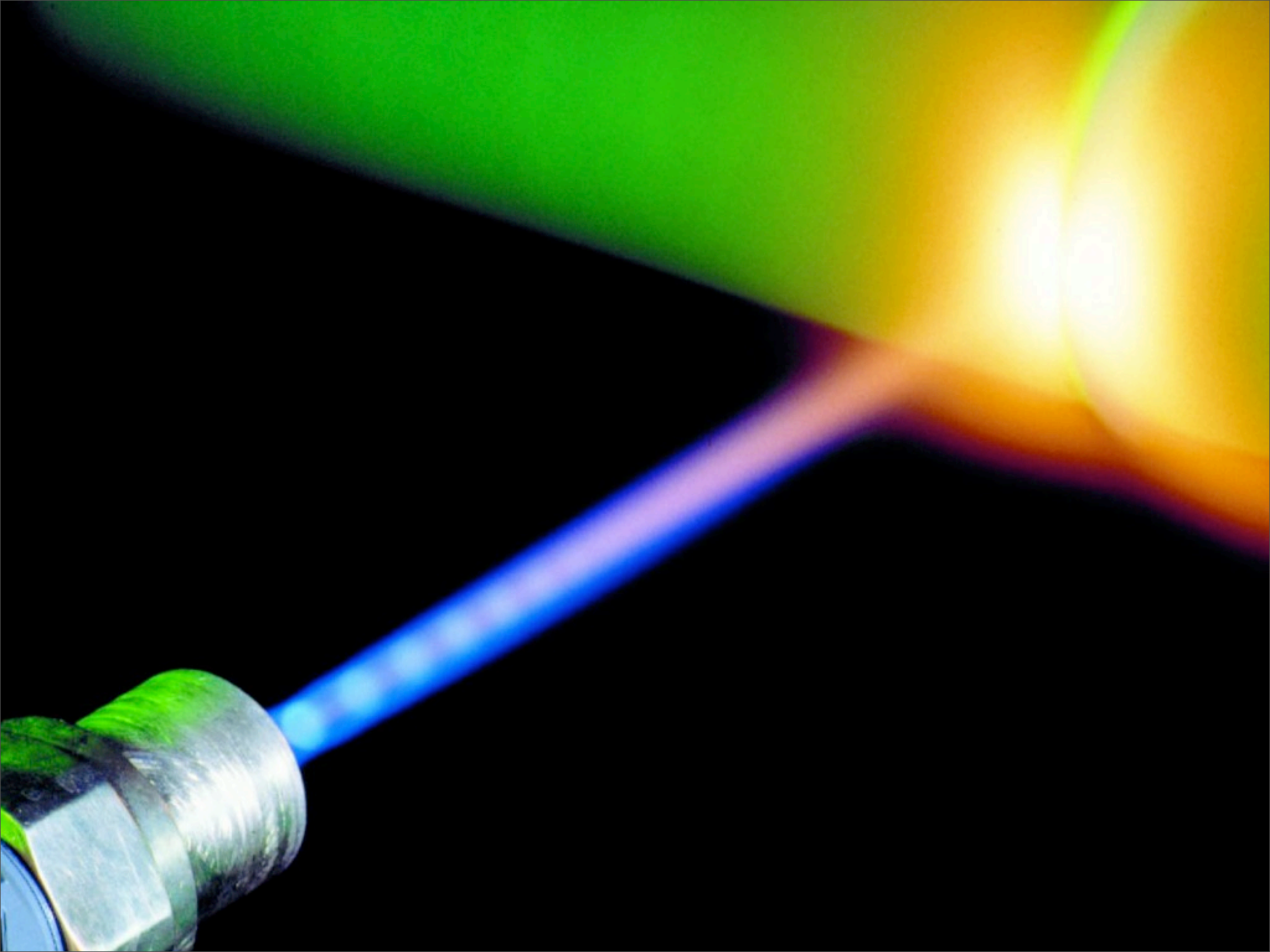
a dozen or less tests

run as part of the build

use a powerful, high level language

stub external dependencies

happy path testing



tomorrows tools today

xUnit.net embraces testing evolution by providing no setup or teardown methods

mockito evolved mocking by allowing you to interact with your mocks freely, and verify interactions at the end

related topics I'm happy to discuss

expectations - a unit testing framework that encourages you to follow the suggestions previously mentioned

integration testing Java applications using JRuby or Clojure


Questions?