Stop Testing.

Start Storytelling.

Mike Schutte

self.conference

August 18, 2018

@tmikeschu

TIME	TOPIC
1	Who Is This Person?
2	Goals
7	Testing Paradigms
20	Storytelling
10	Mad Libs
5	Wrap-up

About Me

@tmikeschu

Haught Codeworks (Ruby, Elixir, JavaScript)

@tmikeschu

DU -> IU -> Turing -> Quikly -> HCW





Here's to the crazy ones.
The misfits. The rebels.
The troublemakers. The
round pegs in the
holes.

@tmikeschu

Goals

PERSPECTIVE

@tmikeschu

CONFIDENCE

@tmikeschu

Jason Lengstorf
@jlengstorf

Following

"Speaking isn't about killing the butterflies in your stomach, it's about making those fuckers fly in formation."

This advice from @GantLaborde may be my new favorite quote on dealing with nervousness before going on stage.

```
describe("Stop Testing, Start Storytelling", () => {
  before(() => {
    tmikeschu.prepareForTalk()
    tmikeschu.getButterfliesInFormation()
    meetup.garnerInterest()
    audience.map(showUp)
  })

  after(() => {
    audience.map(goHome).map(thinkDifferently)
  })

  it("gives people perspective on and confidence in testing", () => {
    const actual = audience
      .map(stopTestingStartStorytelling)
      .flat_map(takeAways).uniq
    const expected = ["perspective", "confidence"]
    expect(actual).toEqual(expected)
  })
})
```

```ruby
describe "testing paradigms" do
  it "covers different testing paradigms" do
    actual   = stop_testing_start_storytelling.testing_paradigms
    expected = ["test last", "test first", "test driven", "behavior driven"]
    expect(actual).to eq(expected)
  end
end
```

Testing Paradigms

@tmikeschu

bonus word: para-dig-MAT-ic (adj)

a framework containing the basic assumptions, ways of thinking, and methodolog[ies] [for discovering new ideas] that are commonly accepted by members of a ~~scientific~~ community.[1]

[1] http://www.dictionary.com/browse/paradigm

Test last

@tmikeschu

The thing that's supposed to happen is the the thing that happens.

@tmikeschu


sheepfilms.co.uk

@tmikeschu

Test first

@tmikeschu

Writing a test to fit an implementation

@tmikeschu

Test driven && behavior driven

@tmikeschu

Implementing code to fit a test

@tmikeschu

"Program to an interface, not an implementation"

- GOF

@tmikeschu

@tmikeschu

```
describe "testing paradigms" do
  it "covers different testing paradigms" do
    actual   = stop_testing_start_storytelling.testing_paradigms
    expected = ["test last", "test first", "test driven", "behavior driven"]
    expect(actual).to eq(expected)
  end
end
```

Storytelling and Software

@tmikeschu

Software is always means to an end

From function to feature, the end is to help users

```
func TestStoryTellingAndSoftware(t *testing.T) {
  var (
    actual   []string
    expected []string
  )

  actual = stopTestingStartStorytelling.StorytellingAndSoftware()
  expected = []string{"communication", "context", "abstraction", "encapsulation"}

  if actual != expected {
    t.Error("Expected talk to cover communication, context, abstraction, and encapsulation, got: ", actual)
  }
}
```

communication

context

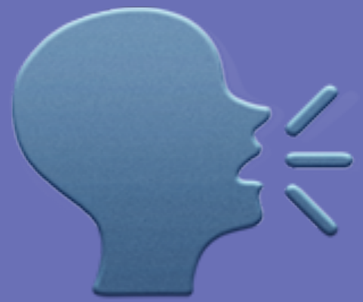
abstraction

encapsulation

@tmikeschu

Communication

Users communicate with software.



@tmikeschu

Let's make good listeners.

@tmikeschu

Context

```
cause && effect
```

more context ->

@tmikeschu

more understanding ->

@tmikeschu

more predictive power ->

more confident decision-making ->

more 😊 ->

@tmikeschu

better teaching and advocacy

When we tell stories about software, we write it better.

@tmikeschu

Abstraction

@tmikeschu

Touching a stove

Breathing

Muscle memory

Language

Encapsulation

We encapsulate our code, why not our thoughts?

(Too many jobs makes you bad at all of them.)

Code constrains clarity and imagination.

Tests can be your compass home

Meta alert!

- Programs solve problems
- Programming is a problem-solving process
- The problems we solve solve problems

Maybe the principles that make better programs can also result in better programming?

```
func TestStoryTellingAndSoftware(t *testing.T) {
  var (
    actual   []string
    expected []string
  )

  actual = stopTestingStartStorytelling.StorytellingAndSoftware()
  expected = []string{"communication", "context", "abstraction", "encapsulation"}

  if actual != expected {
    t.Error("Expected talk to cover communication, context, abstraction, and encapsulation, got: ", actual)
  }
}
```

Still learning the domain specific language (DSL)?

```
/*
description: developers
assert:      should be mindful of context

actual:      perspective while implementing
expected:    developer
  actual should be the same as expected

actual:      perspective while testing
expected:    user
  actual should be the same as expected
*/
```

Duality

57 — 74

@tmikeschu

User <----------X----------> Programmer

Implementing

<----------X> Programmer

@tmikeschu

Storytelling/testing

@tmikeschu

User <X---------->

Crowdsource Demo!

63 — 74
@tmikeschu

Mad lib

3 Nouns

1 Verb

1 Datatype

@tmikeschu

Wrap-up

Testing is a paradigmatic process, not a product.

Communication: build good listeners.

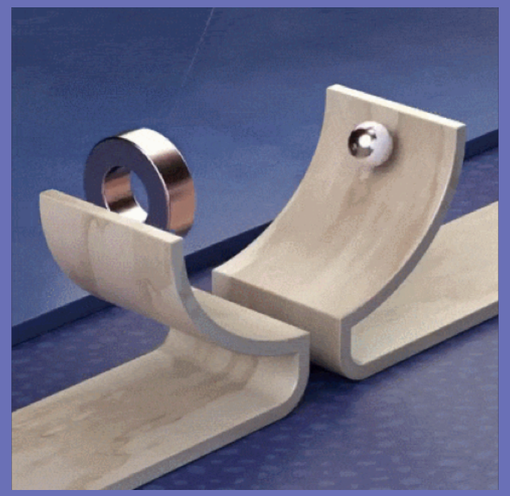
Context: enable confident decision making.

Abstraction: you're good at it, it opens up doors.

Encapsulation: apply the same discipline to your process as you do to your code.

Storytelling keeps you closer to users.

Action Plan



Develop a pendulum workflow between storyteller and developer.

@tmikeschu

```
describe("Stop Testing, Start Storytelling", () => {
  before(() => {
    tmikeschu.prepareForTalk()
    tmikeschu.getButterfliesInFormation()
    meetup.garnerInterest()
    audience.map(showUp)
  })

  after(() => {
    audience.map(goHome).map(thinkDifferently)
  })

  it("gives people perspective on and confidence in testing", () => {
    const actual = audience
      .map(stopTestingStartStorytelling)
      .flat_map(takeAways).uniq
    const expected = ["perspective", "confidence"]
    expect(actual).toEqual(expected)
  })
})
```

Thank you!

@tmikeschu