TESTING

minimizing mistaqes

TRAJECTORY

◎ **Why test**

◎ **Tools & terminology I**

◎ **What makes a good test**

◎ **Testing Sequelize models**

◎ **Testing Express routes**

◎ **Tools & terminology II**

◎ **Test Driven Development**

◎ **Cake!!**

SOFTWARE PRODUCTION WAR STORIES

Legacy Code — "Someone dumber, sloppier, and less good looking than me wrote that code."

Emergency Push — "It needs to go out right now because the CMO said so."

Rush to Finish — "We'll do our testing in the three months before launch."

Production Destruction — "It's just a small fix to the database update code."

Spray/Pray — "Don't worry, QA will find it."

Maintenance Nightmare — "Only Roy in the basement knows how that module works."

WHY TEST?

TESTS

◉ **Ensure/prove code is working**

◉ **Ensure code will continue to work after someone changes it**

◉ **Document what the code actually does**

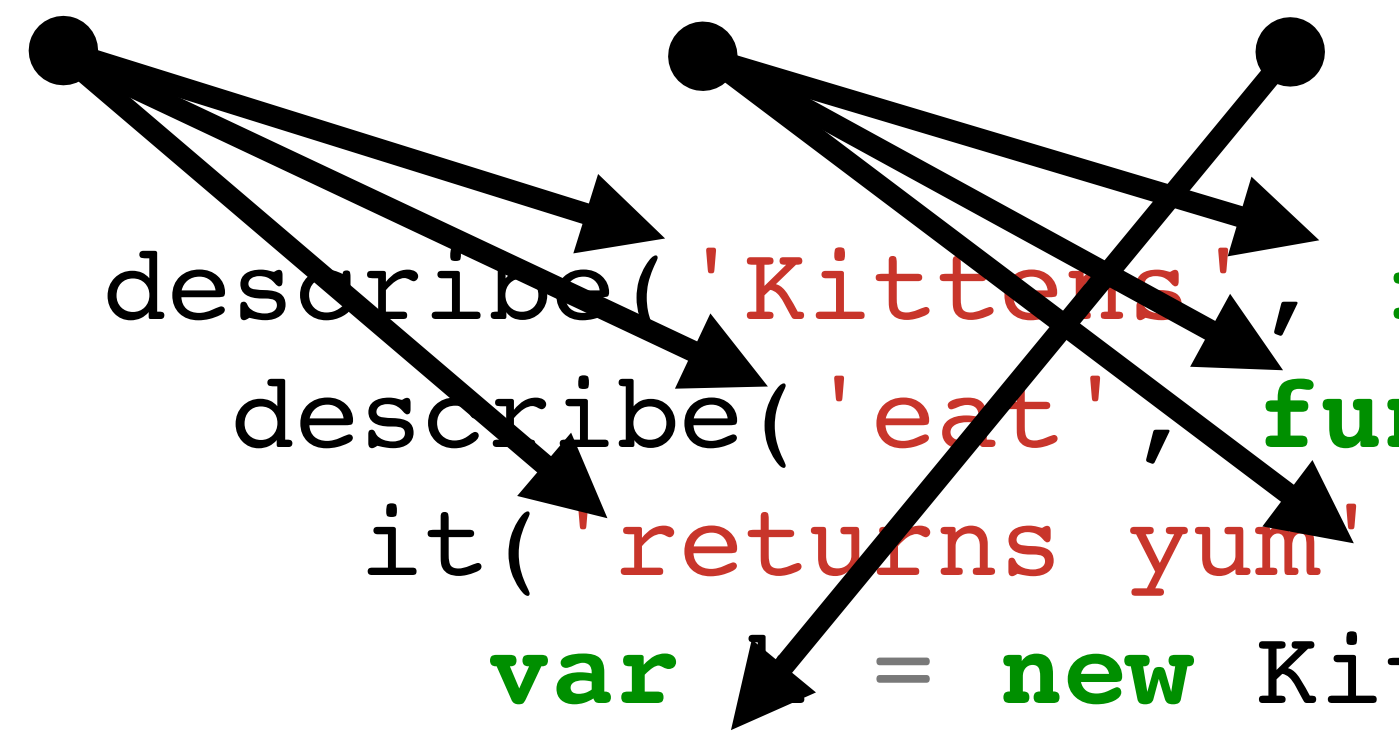◉ **Precision/accuracy/certainty of behavior**

TOOLS & TERMS I

TESTING FRAMEWORKS

In JavaScript – the two contenders for most popular testing framework are Jasmine by Pivotal and Mocha/Chai by TJ Holowaychuk

GETTING STARTED

◉ **Less complicated than you might think**

◉ **Labels + functions + assertions = test specs**

```javascript
describe('Kittens', function() {
  describe('eat', function() {
    it('returns yum', function() {
      var k = new Kitten()
      expect(k.eat()).to.equal('yum')
    })
  })
})
```
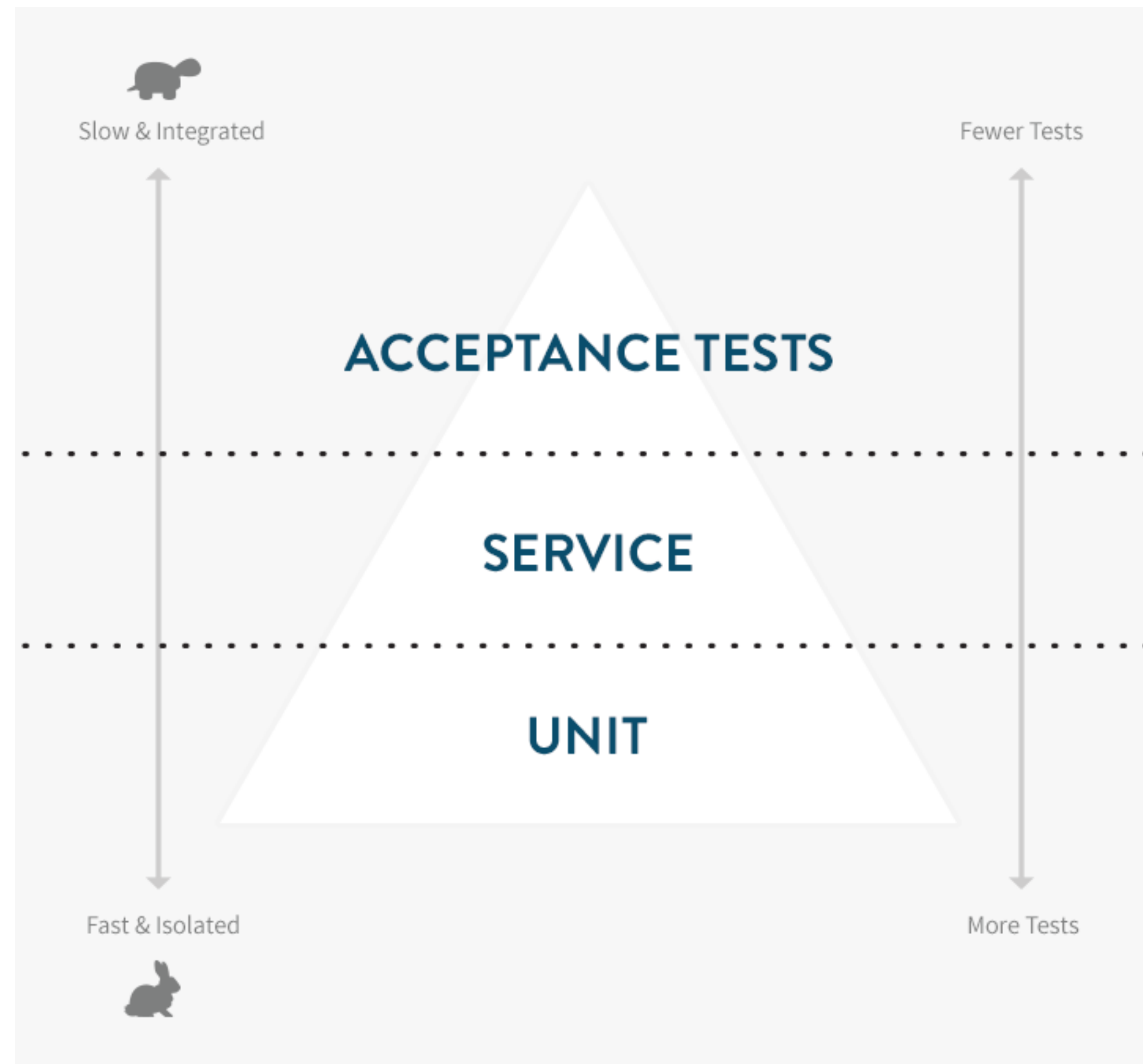
ASSERTIONS

things that throw errors…

```
/* Our testing library */
function assert (result) {
  if (!result) {

    throw new Error("A test failed")

  }
}
/* end of testing library */
```
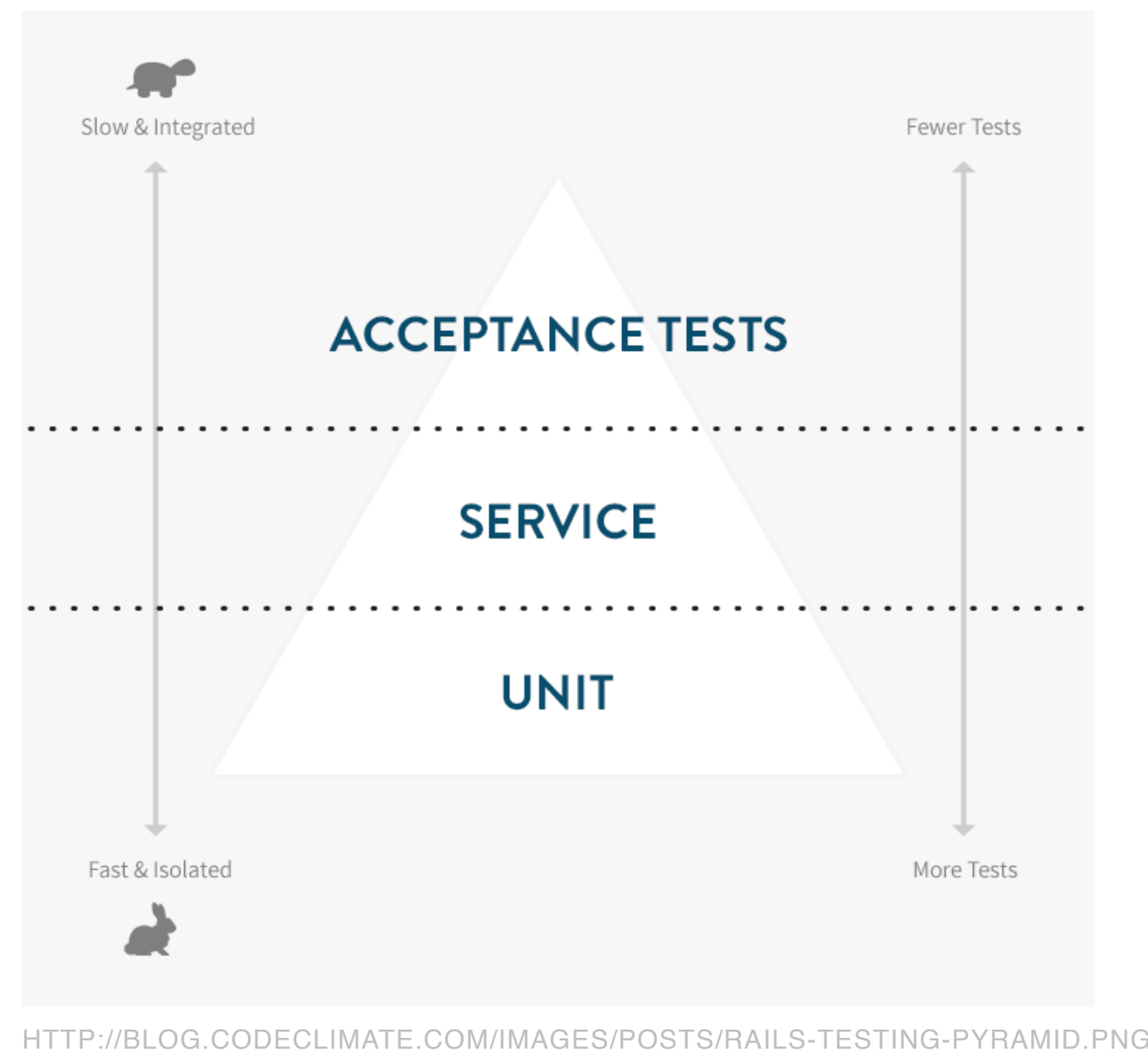
```
/* tests */
result = MyMathLibrary.add(1, 2)
assert(result === 3)
```
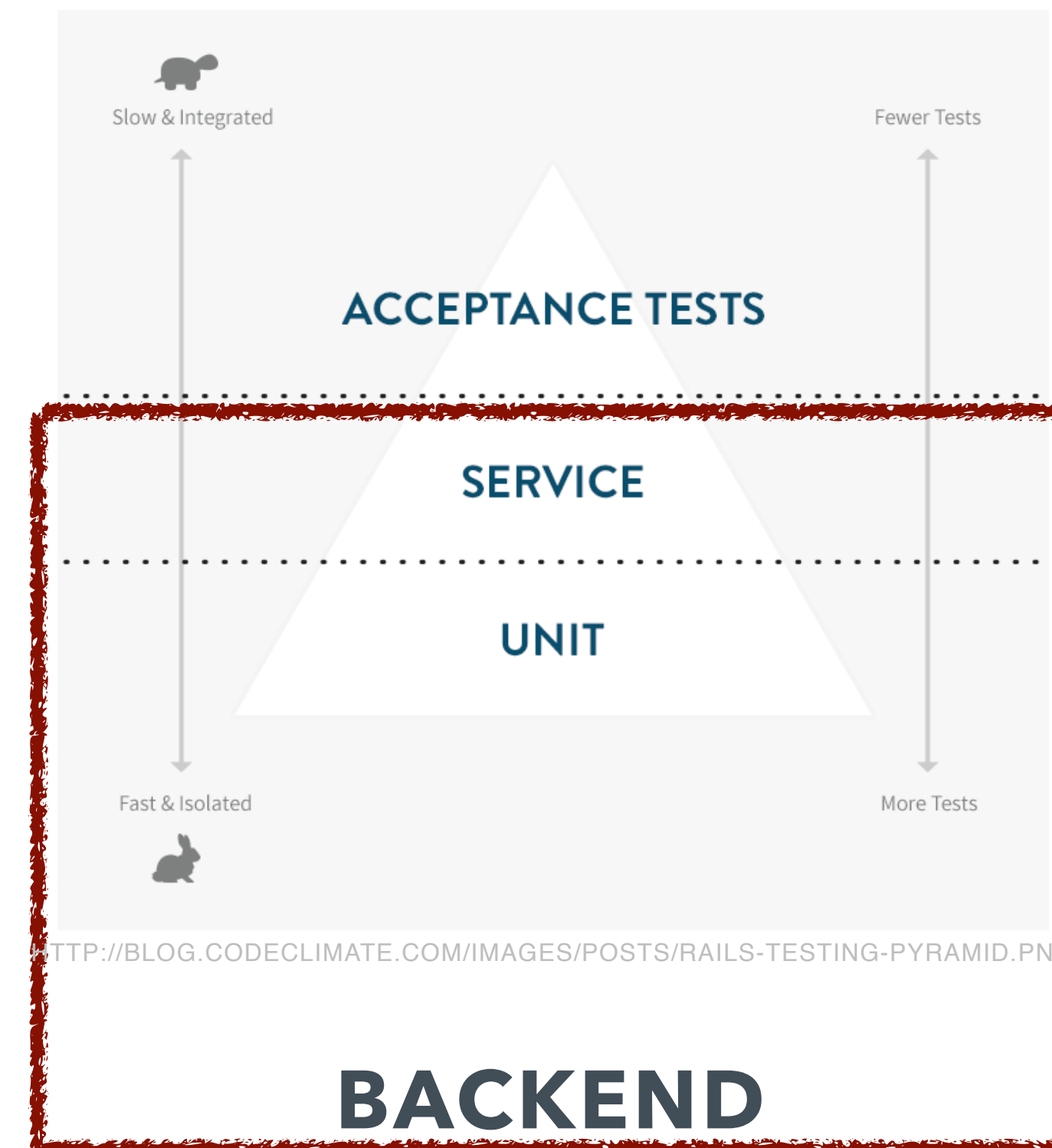
TEST PYRAMID

HTTP://BLOG.CODECLIMATE.COM/IMAGES/POSTS/RAILS-TESTING-PYRAMID.PNG

TEST PYRAMID

FRONTEND

BACKEND

today's workshop

WHAT MAKES A GOOD TEST?

FEATURES OF GOOD UNIT TESTING

- **Specific/Accurate**
- **Concise, but Thorough**
- **Isolated**

```javascript
function testMe (result) {
  if (result < 0) {
    throw new Error('too low!')
  } else if (result === 100) {
    return 'Keep it 100!'
  } else {
    return `You said ${result}`
  }
}
```

ERM....

```
describe('the testMe function', () => {
  it('returns a string', () => {
    const result = testMe(200)
    expect(result).to.be.a('string')
  })
})
```

OH DEAR...

```
describe('the testMe function', () => {
  it('returns 'You said 1' when the input is 1', () => {
    const result = testMe(1)
    expect(result).to.be.a('You said 1')
  })

  it('returns 'You said 2' when the input is 2', () => {
    const result = testMe(2)
    expect(result).to.be.a('You said 2')
  })

  it('returns 'You said 3' when the input is 3', () => {
    const result = testMe(2)
    expect(result).to.be.a('You said 3')
  })
})
```

BEING THOROUGH…WITHOUT GOING OVERBOARD

- "Equivalence classes"
- Reduce redundancy
- Input => Output

ISOLATE TESTS

- **Highly intertwined tests are brittle – change one thing and the whole thing will break**

- **Reduce state**

- **Reduce moving pieces / things running**

- **Reduce dependence on other components**

REDUCING STATE

```
/**
 * appendToFile is a function that adds some content to
 * the end of a text file
 */
describe('appendToFile', () => {

  it('adds to the end of the file', () => {
    appendToFile('./testDoc.txt', 'Hello world')
    const fileContent = fs.readFileSync('./testDoc.txt', 'utf-8')
    expect(fileContent).to.be.equal('Hello world')
  })
})
```

```javascript
/**
 * appendToFile is a function that adds some content to
 * the end of a text file
 */
describe('appendToFile', () => {

  beforeEach(() => { // set things up!
    fs.writeFileSync('./testDoc.txt', 'utf-8', '')
  })

  afterEach(() => { // clean up when we're done
    fs.writeFileSync('./testDoc.txt', 'utf-8', '')
  })

  it('adds to the end of the file', () => {
    appendToFile('./testDoc.txt', 'Hello world')
    const fileContent = fs.readFileSync('./testDoc.txt', 'utf-8')
    expect(fileContent).to.be.equal('Hello world')
  })

})
```

TESTING SEQUELIZE

TESTING EXPRESS

TOOLS & TERMS II

SPIES, STUBS, MOCKS

◎ **Spies:** Simply functions that record info about how and when they were called

◎ **Stubs:** Spies + ability to return preset "canned" values

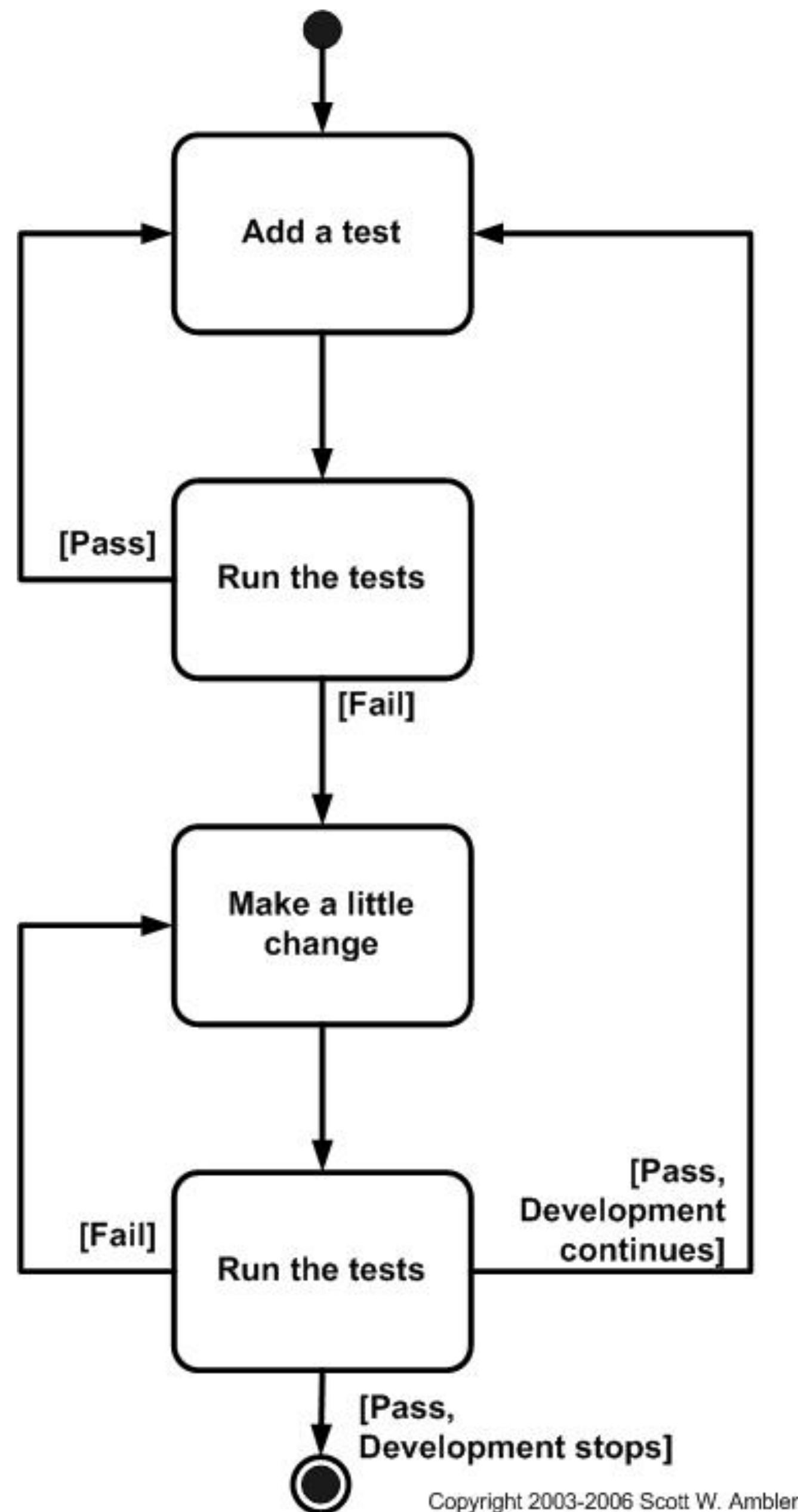◎ **Mocks:** Stubs + expectations about how it will be used
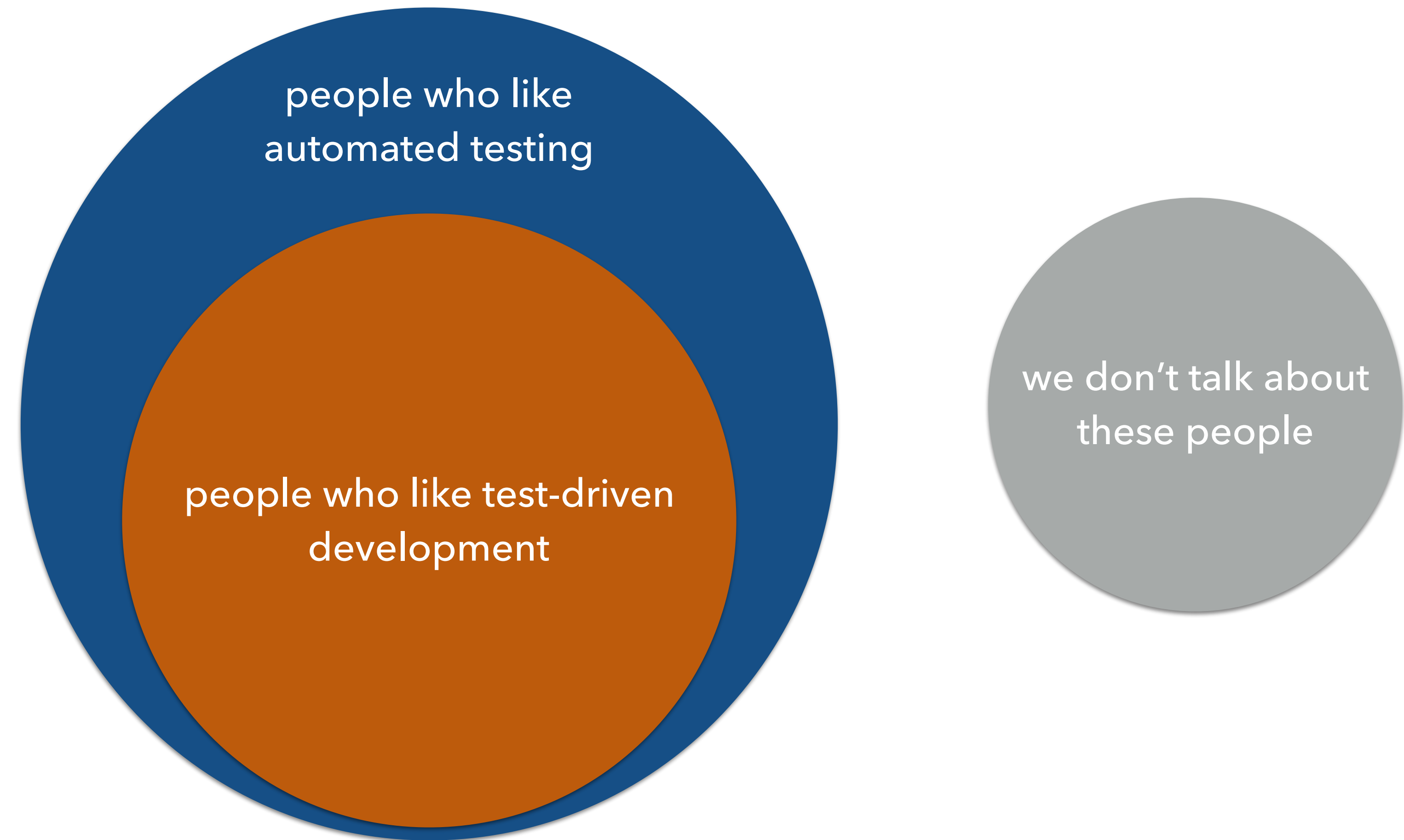

Sinon.JS

TDD

TEST-DRIVEN DEVELOPMENT

◉ **A practice where you write your automated unit tests BEFORE you write your implementation code**

◉ **Focus on what code is supposed to do**

◉ **Have a goal**

◉ **Ensure you don't blow off automated testing**

◉ **Improves design and modularity of code**

◉ **"Refactorability"**

TDD

Add a test

[Pass]

Run the tests

[Fail]

Make a little change

[Fail]

Run the tests

[Pass, Development continues]

[Pass, Development stops]

Copyright 2003-2006 Scott W. Ambler

TEST DRIVEN DEVELOPMENT

people who like
automated testing

people who like test-driven
development

we don't talk about
these people

WORKSHOP