TESTING

minimizing mistaqes

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."

TESTS

- Ensure code is working
- Ensure code will continue to work after someone changes it
- Document what the code actually does
- Precision/accuracy/certainty of behavior

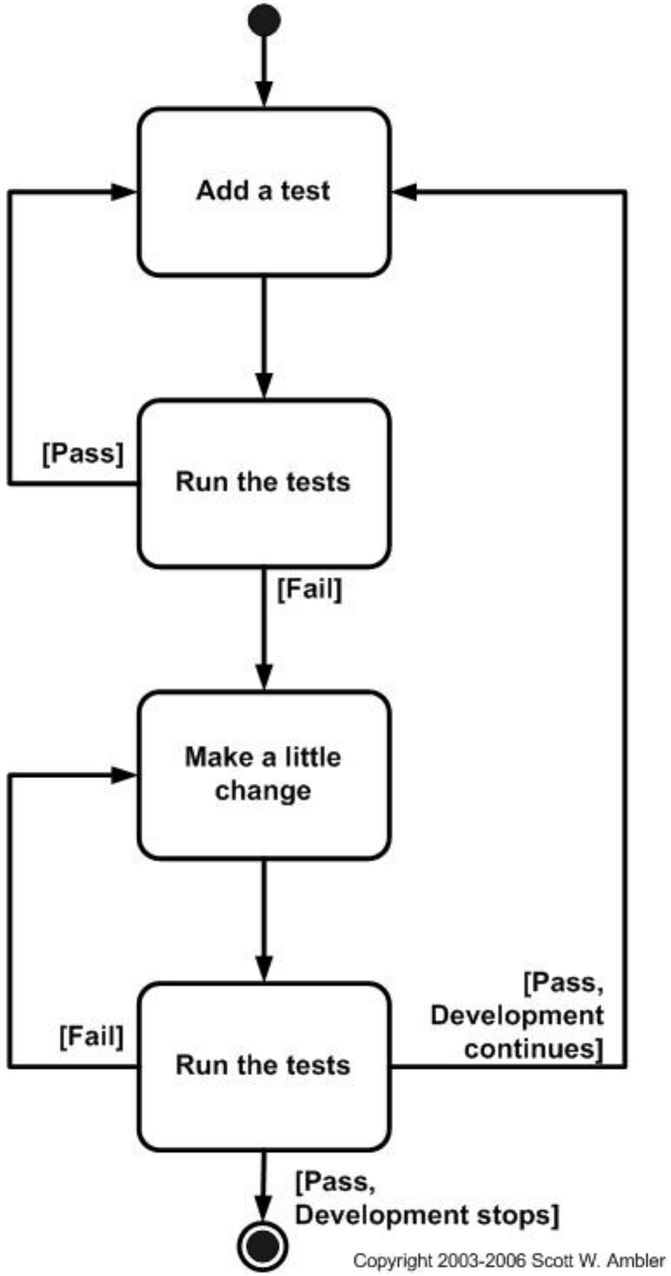
HISTORY OF TESTING

- **< 1970s:** Developers tested their own code
- **1970s:** Dedicated testers following written scripts
- **1980s:** Capture/replay testing
- **1990s:** Scriptable unit tests
- **2000s:** Test pyramid, test driven development, continuous integration

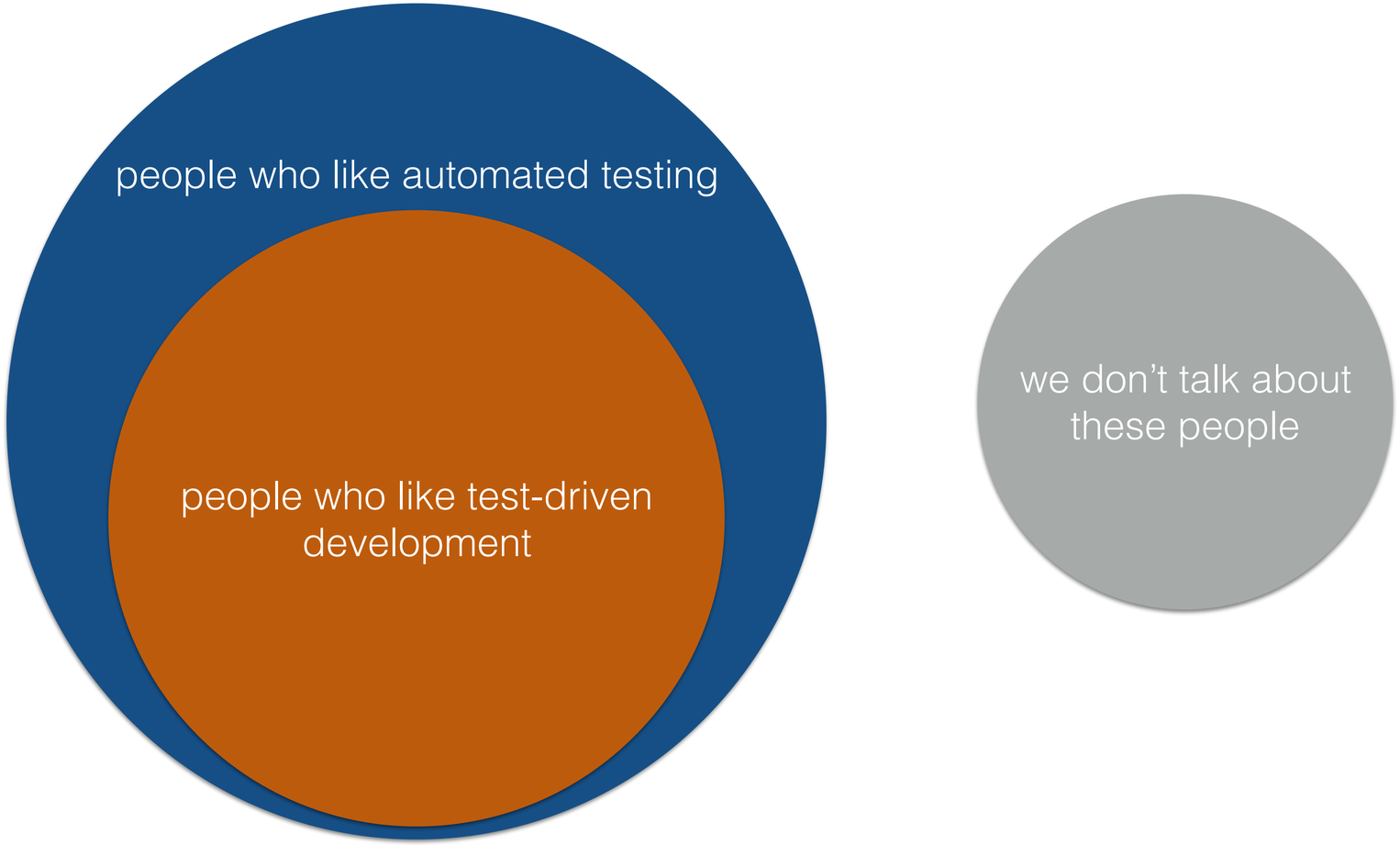
TEST-DRIVEN DEVELOPMENT

- 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
Run the tests
[Pass]
[Fail]
Make a little change
Run the tests
[Fail]
[Pass, Development continues]
[Pass, Development stops]
Copyright 2003-2006 Scott W. Ambler

AUTOMATED TESTING ≠ TEST-DRIVEN DEVELOPMENT



GETTING STARTED

- Less complicated than you might think
- Labels + functions + assertions = test specs

GETTING STARTED

- Less complicated than you might think
- Labels + functions + assertions = test specs

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

GETTING STARTED

- Less complicated than you might think
- Labels + functions + assertions = test specs

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

GETTING STARTED

- Less complicated than you might think
- Labels + functions + assertions = test specs

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

GETTING STARTED

- Less complicated than you might think
- Labels + functions + assertions = test specs

```
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…

ASSERTIONS

things that throw errors…

```
/* Our testing library */
function assert (result) {
  if (!result) {
    throw new Error("A test failed");
  }
}
/* end of testing library */
```

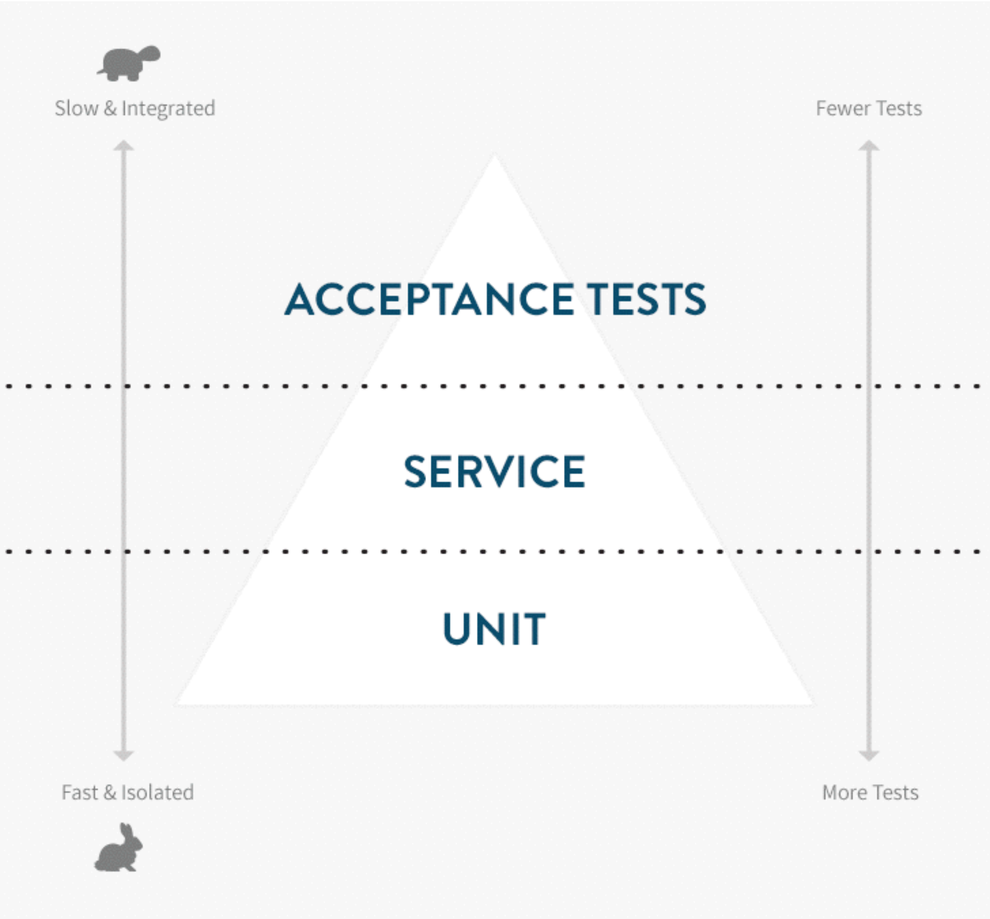
ASSERTIONS

things that throw errors…

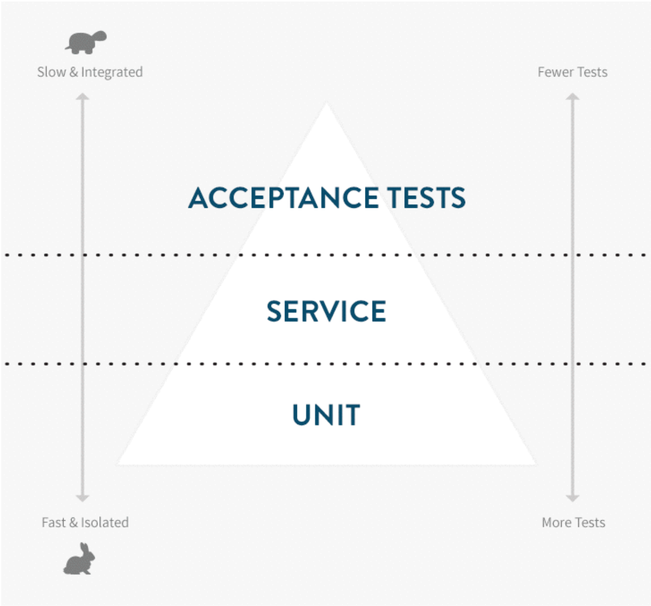
```
/* Our testing library */
function assert (result) {
  if (!result) {
    throw new Error("A test failed");
  }
}
/* end of testing library */
```

```
/* tests */
result = MyMathLibrary.add(1, 2);
assert(result === 3)
```

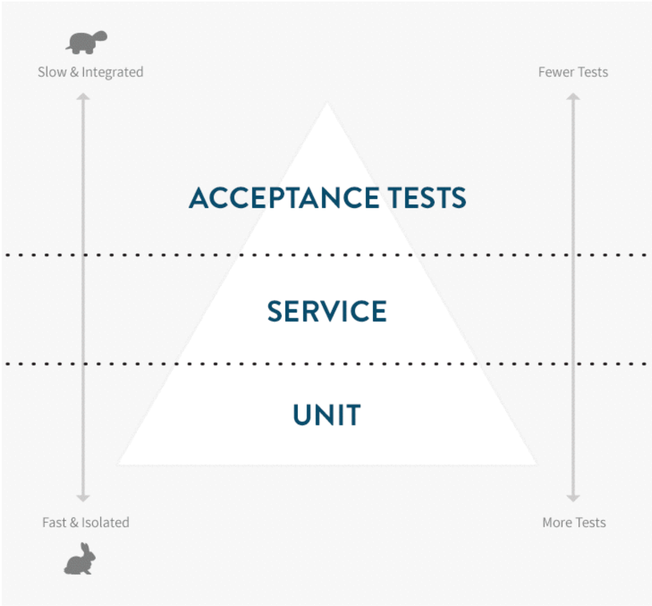
TEST PYRAMID



TEST PYRAMID

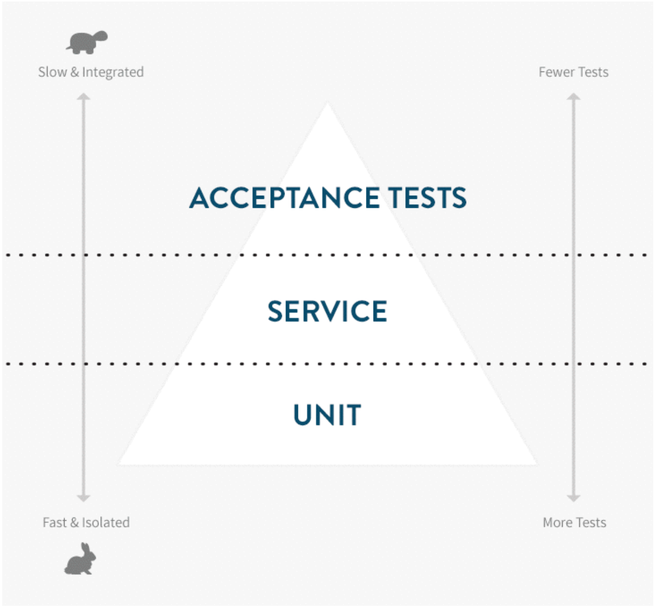


FRONTEND

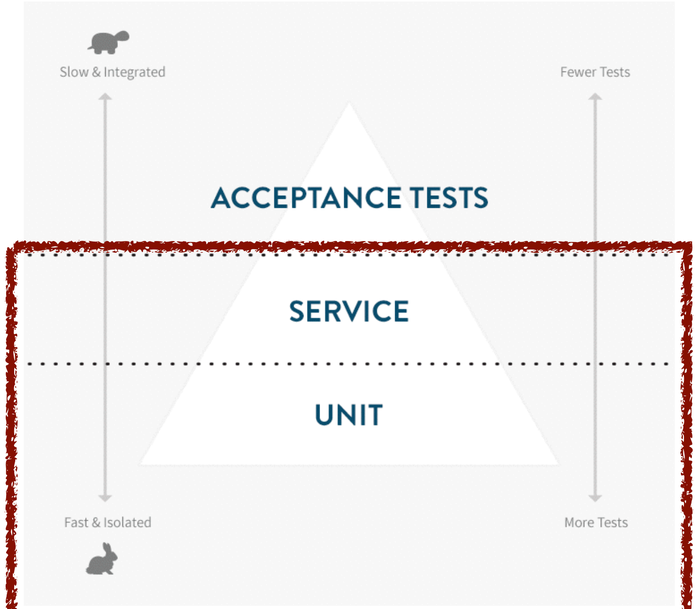


BACKEND

TEST PYRAMID



FRONTEND



BACKEND

today's workshop

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

TOOLS

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

Jasmine

mocha

simple, flexible, fun

+

Chai Assertion Library

MODEL TESTING EXAMPLE

```
describe('Kitten Model', function() {
  describe('methods', function() {
    describe('scratch', function() {
      Math.random = function() {
        return 0.5
      }
      it('returns ouch if scratch probability is greater than random number', function() {
        k.scratchProbability = 0.6
        expect(k.scratch()).to.equal('ouch')
      })

      it('returns prrrrrrr if scratch probabilty is less than random number', function() {
        k.scratchProbability = 0.4
        expect(k.scratch()).to.equal('prrrrrrr')
      })
    })
  })
})
```

ROUTE TESTING EXAMPLE

with supertest

```
var supertest = require('supertest')
var app = require('./path/to/your/express/app')
var agent = supertest(app)
describe('server', function() {
  it('responds with 404 for routes that do not exist', function (done) {
    agent
    .get('/blablabla')
    .expect(404, done)
  })
})
```

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