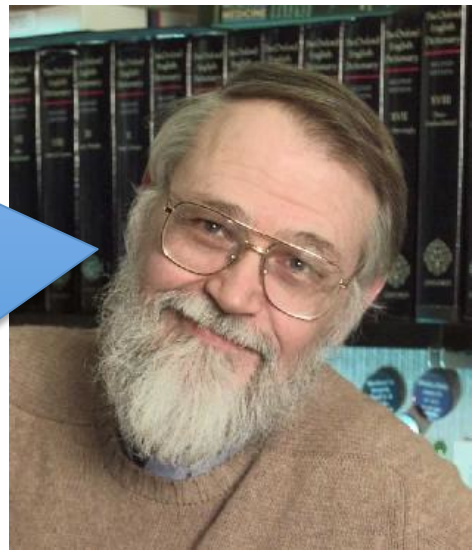
Intro to RSpec & Unit Tests

(Engineering Software as a Service §8.1)

Armando Fox

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.



Testing can never demonstrate the absence of errors in software, only their presence

Survey Finds 58% of Software Bugs Result from Test Infrastructure and Process, Not Design Defects

Developers Prefer Taxes to Dealing with Software Testing

Sunnyvale, Calif. — June 2, 2010 Electric Cloud®, the leading provider of software production management (SPM) solutions, today released the results of a survey conducted in partnership with Osterman Research showing that the majority of software bugs are attributed to poor testing procedures or infrastructure limitations rather than design problems. Additionally, the software test process is generally considered an unpleasant process, with software development professionals rating the use of their companies' test systems more painful than preparing taxes.

Fifty-eight percent of respondents pointed to problems in the testing process or infrastructure as the cause of their last major bug found in delivered or deployed software, not design defects.

Specifically, the survey found:

- Completely automated software testing environments are still rare, with just 12 percent of software development organizations using fully automated test systems. Almost 10 percent reported that all testing was done manually.

Testing Today

- Before
  - developers finish code, some ad-hoc testing
  - "toss over the wall to Quality Assurance [QA]"
  - QA staff manually poke at software
- Today/Agile
  - testing is part of *every* Agile iteration
  - developers test their own code
  - testing tools & processes highly automated
  - QA/testing group improves *testability & tools*

Testing Today

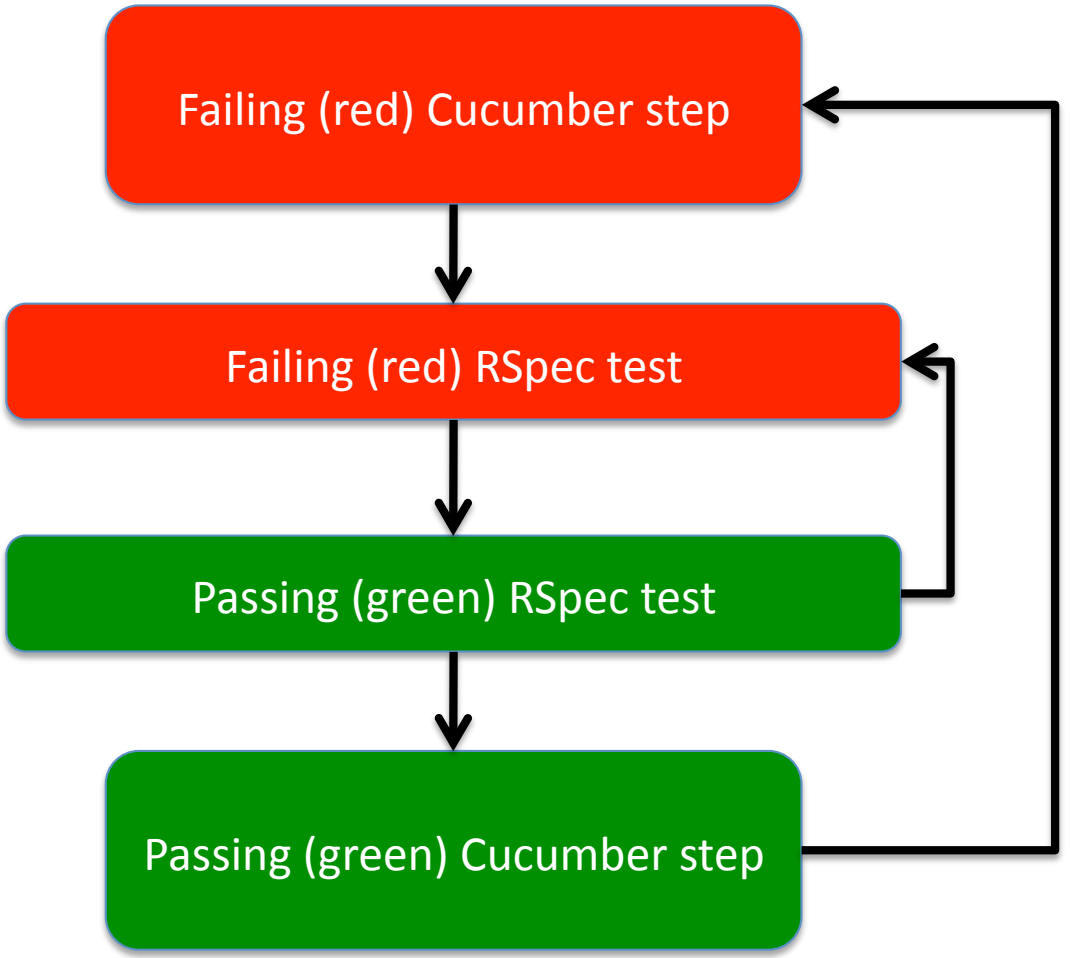
- Before
  - developers finish code, some ad-hoc testing
- *Software Quality is the result of a good **process**, rather than the responsibility of one specific group*
  - testing tools & processes highly automated;
  - QA/testing group improves *testability* & *tools*

BDD+TDD: The Big Picture

- Behavior-driven design (BDD)
  - develop user stories *(the features you wish you had)* to describe how app will work
  - via Cucumber, user stories become *acceptance tests* and *integration tests*
- Test-driven development (TDD)
  - *step definitions* for new story,may require new code to be written
  - TDD says: write unit & functional tests for that code *first, **before*** the code itself
  - that is: write tests for *the code you wish you had*

Cucumber & RSpec

- Cucumber describes *behavior* via features & scenarios (*behavior driven* design)
- RSpec tests individual modules that contribute to those behaviors (*test driven* development)



Which are true about BDD & TDD:
a) requirements drive the implementation
b) they can be used only in Agile development
c) they embrace & deal with change

- ☐ Only (a)
- ☐ Only (a) & (b)
- ☐ Only (a) & (c)
- ☐ (a), (b) and (c)

FIRST, TDD, and Getting Started With RSpec

(Engineering Software as a Service §8.2)

Armando Fox

Unit tests should be FIRST

- **F**ast
- **I**ndependent
- **R**epeatable
- **S**elf-checking
- **T**imely

Unit tests should be FIRST

- **F**ast: run (subset of) tests quickly (since you'll be running them *all the time*)
- **I**ndependent: no tests depend on others, so can run *any subset* in *any order*
- **R**epeatable: run N times, get same result (to help isolate bugs and enable automation)
- **S**elf-checking: test can *automatically* detect if passed (*no human checking* of output)
- **T**imely: written about the same time as code under test (with TDD, written *first!*)

RSpec, a Domain-Specific Language for testing

- DSL: small programming language that simpifies one task at expense of generality
  - examples so far: migrations, regexes, SQL
- RSpec tests are called *specs* or *examples*

- Run the tests in one file: **`rspec`** *fi*
  - Red failing, Green passing, Yellow *pending*
- *Much better: running* **`autotest`**

http://pastebin.com/LKTK36Pb

Which kinds of code can be tested **R**epeatably and **I**ndependently?
a) Code that relies on randomness (e.g. shuffling a deck of cards)
b) Code that relies on time of day (e.g. run backups every Sunday at midnight)

☐ Only (a)

☐ Only (b)

☐ Both (a) and (b)

☐ Neither (a) nor (b)