Living on the Bleeding Edge In The Financial Industry

Using Clojure, AMQP, Chef, Cucumber and JRuby in the Financial Industry

Philadelphia ETE, April 2010

Algorithmics, Inc.

Risk Management

Collateral Management

Aaron Feng

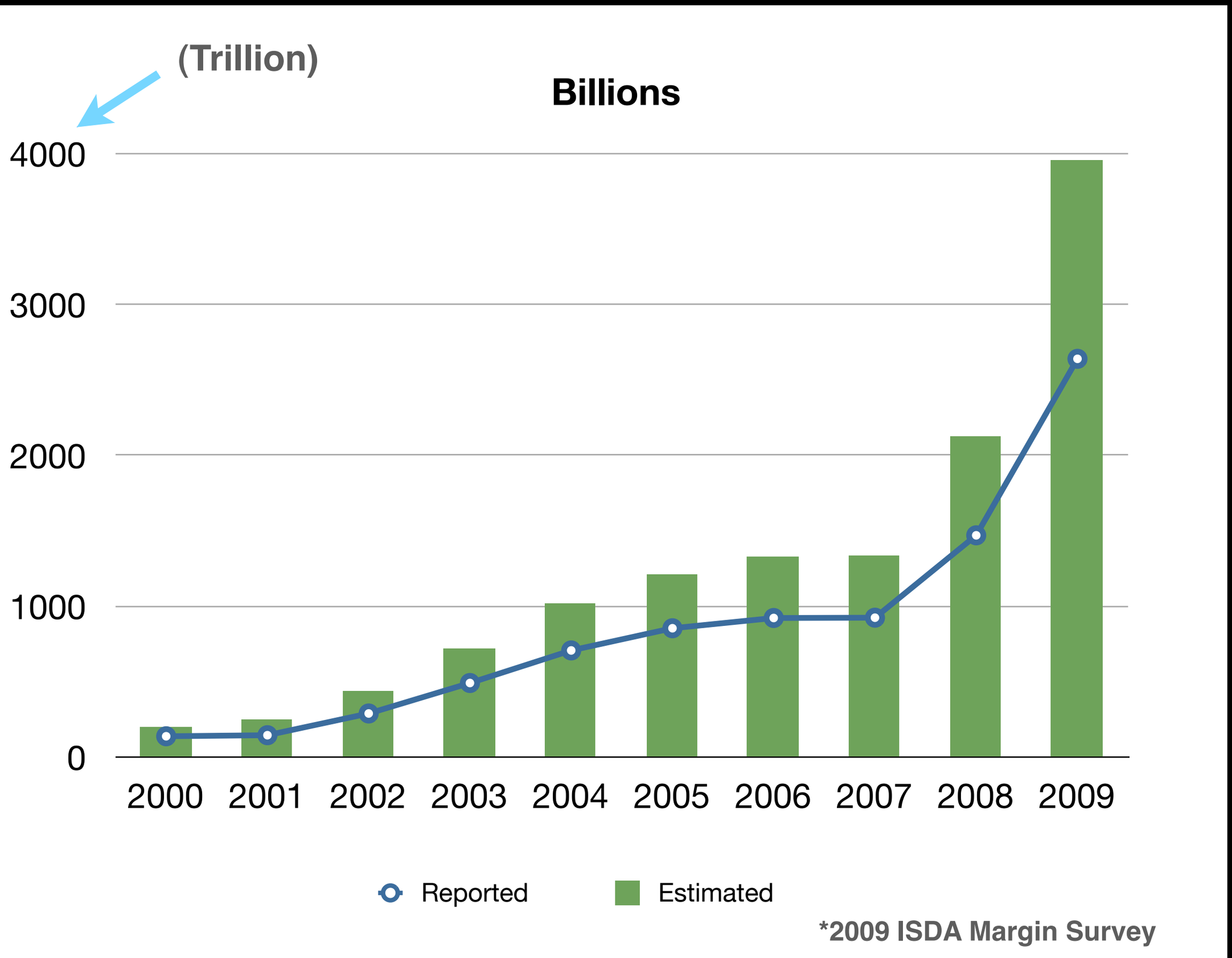
Kyle Burton

Ok, WTF is Collateral Management?

Reduces Credit Risks for
Unsecured Financial Transactions

Credit Risk == Debtor Fails to Pay

Unsecured Financial Transactions are
OTC (over the counter)


(Trillion)
Billions
4000
3000
2000
1000
0
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
Reported
Estimated
*2009 ISDA Margin Survey

Talk about the product here

- Today
  - Manual Processing
  - Email, Phone, Fax
- Tomorrow
  - Secure Messaging
  - Formalized Workflow
  - Standards based Protocol

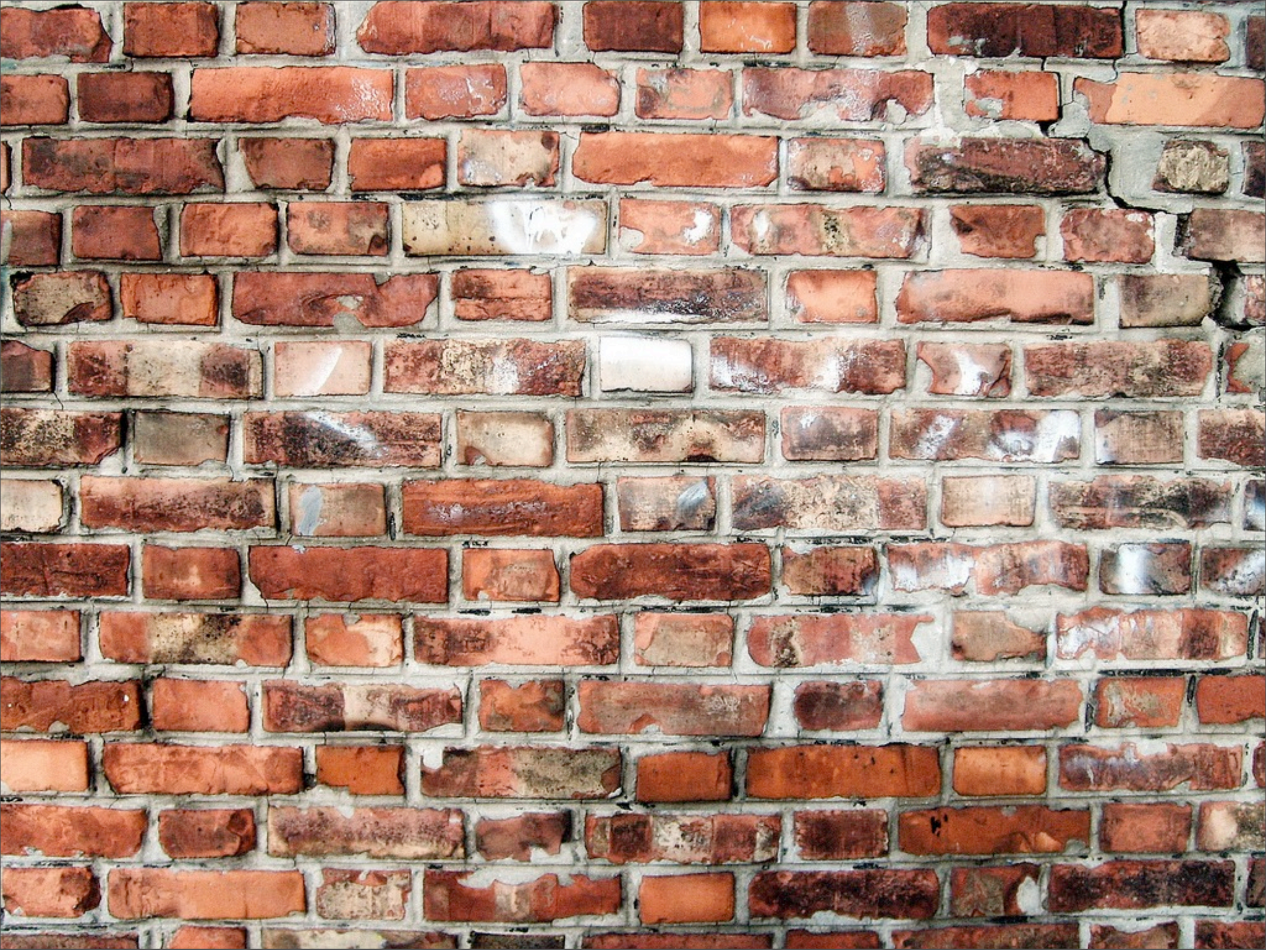
Overview

- What do we all want?
  - What Challenges Did we face
- How did we do/get it?
- What did we do?
  - What challenges did we face
- What has it done for us?

What do we Want?

- Best People
- Best Technologies
- To Create: new code, not legacy
- Interesting Problems
- Best Processes

Monday, March 8, 2010



Monday, March 8, 2010

Challenges

- Financial Industry: Conservative
- Management: Conservative, Skeptical
- Programmers: Skeptical
- Technology Choices: Clojure, Ruby, AMQP
- Gasp: Where do we go for support?

Monday, March 8, 2010

How we did it: People

- Product Visionaries Sold Higher Ups on Idea
- Time To Market Critical

- Aaron Built a Prototype *very* Rapidly
  - All by His Lonesome
  - Demonstrated Value of Technologies

Project Given Green Light June 2009

How we did it: People

- Aaron Invested in Philly Lambda
- Networked with Members
- Organized, Brought in Speakers
- Bought Lots of Pizza

First Two Developer Hires July 2009

How we did it: Support

- LShift: unlike many open source techs, a company sits behind RabbitMQ
- We knew RabbitMQ was great, it was built on Erlang
- Erlang devs don't grow on trees in Philly
- Contracted with LShift to extend and accelerate product roadmap

What did we do?

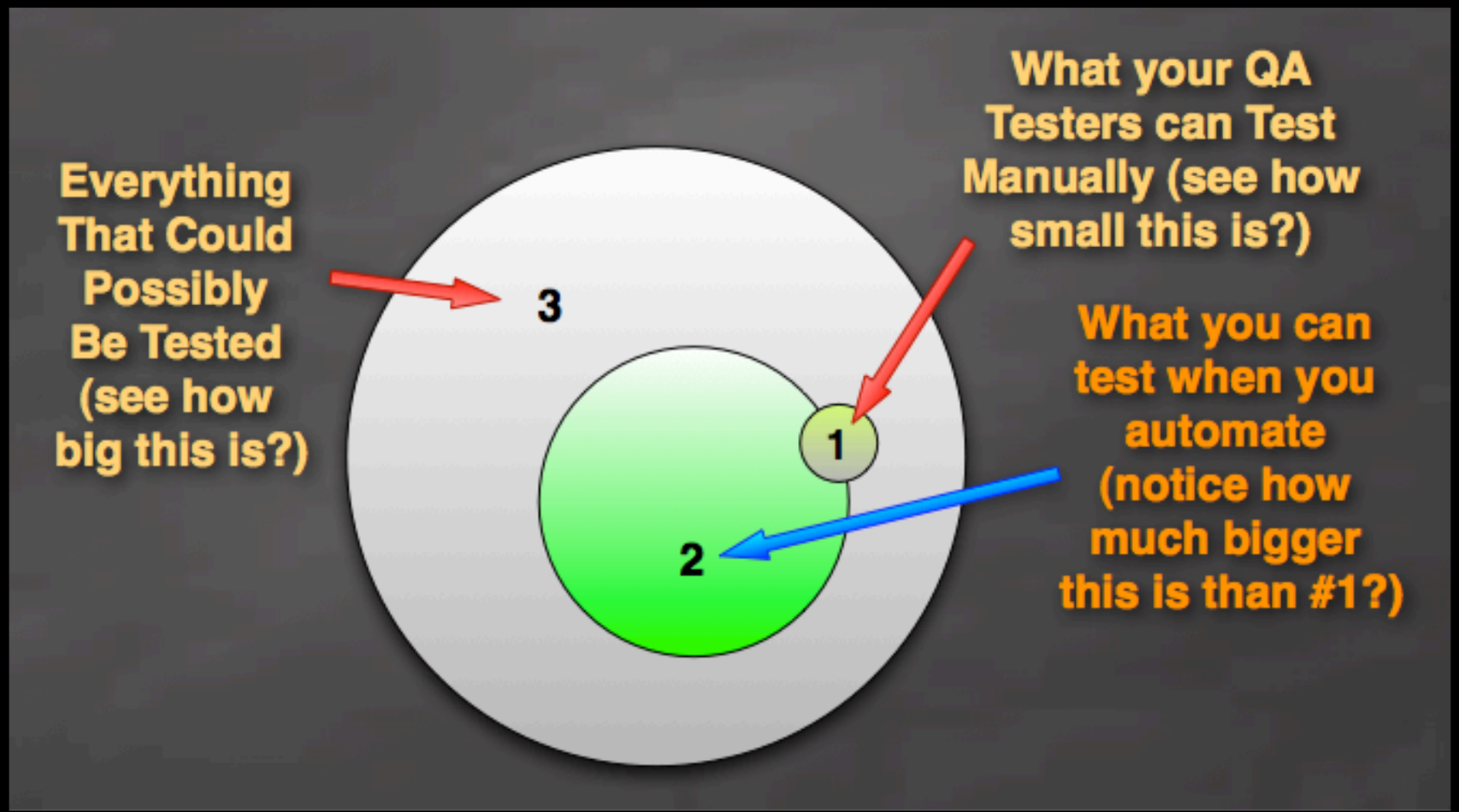
- Small Team
- Agile Methodologies
- Pair Programming
- Continuous Improvement
- Automation

What did we do: Small Team

- Focus on Automation
  - Frequent Releases
    - Chef for Provisioning
    - Cucumber for Acceptance Tests
    - Lots of Other Small Automations
      - A Mindset, Core Value of Team

Focus On Automation: QA

Reduced QA Resource Needs Initially (zero)
Suite Provided Regression Testing: Reduced Errors
Sped Up Development

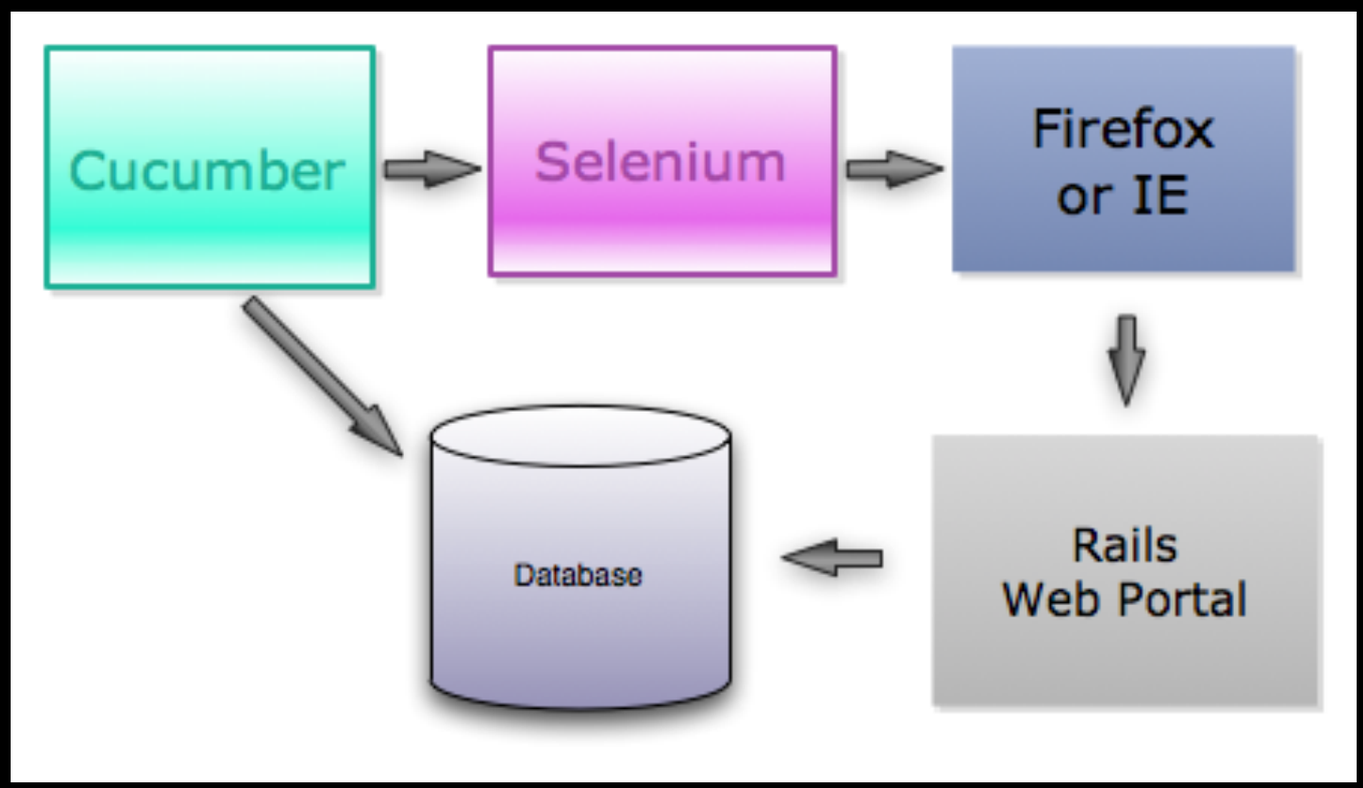


Focus On Testing: TDD

Core Value: Test Driven Development

Tools:

- Cucumber: Front End Integration Testing
- Rspec: Behavior Driven Unit Testing



Testing: BDD With Cucumber

Default

```
1  Feature: Send Margin Calls
2    In order to request money
3    As an analyst
4    I want to be able to send a margin call demand request
5
6    Background:
7      Given I reset the data
8      And I am logged in as "guest"
9      And I am on the margin calls page
10     And I am on the "Unsent" tab
11
12   Scenario: Login and see margin call
13     Then "Guestbank vs Algobank" should be visible
14     And "Send Calls" should be visible
15
16   Scenario: Send Margin Call Demand with status
17     When I check the checkbox to the left of "Guestbank vs Algobank"
18     And I click "Send Calls"
19     Then "No records found" should be visible
20     When I go to the "Awaiting Demand Response" tab
21     Then "Guestbank vs Algobank" should be visible
22     And "sent" should be visible
23
                                                    1,1           Top
```

What did we do: Small Team

- Pair Programming
  - Skill Transfer Happens Quickly
  - Bus Factor == Team Size
  - Lost one of our best members in Feb: OMM*

* OMM: Oh! My!....meh.

What did we do?

- Web Portal for the low end of the market
- Ruby on Rails (JRuby), jQuery, YUI
- Lots of Plugins, Gems, Libraries
- Cucumber for automated testing

What did we do?

- Clojure for the ‘router’
  - Concurrency is fantastically easy
  - FP, Immutability by default => reduction of bugs caused by common errors
  - JVM: It’s the libraries(!)
  - Live Image: easier introspection into running system (and sometimes modifications)

Challenges

- Acceptance of the stack
- Finding Developers
- Learning These Technologies
- Integration

What has it done for us?

- Automated Testing and why it is win
- Automated Provisioning and why it is win
- Functional Programming and the reduction of errors
- Tiny components and ability to quickly adapt to changes in the product

NOT LOC Metrics Again!

- Widely Believed that LOC/Dev is Roughly Constant
- Two Devs of Equal Skill/Experience, the one using a higher level language will be more productive
- See also: Mythical Man Month

NOT LOC Metrics Again!

- sloccount
- For the first 9mo of the project:
  - 24kloc, 687loc per dev per month
  - 5.81 developer years
  - Our People & Process put us roughly 45% ahead of that estimate

Conclusion

- We Came
- We Learned
- We Conquered
- We ‘re hiring