TEST DRIVEN DEVELOPMENT

TDD

TYPES OF TESTING

- Unit
- Integration
- Functional
- Perf/Load
- UAT

HOW MUCH TESTING IS ENOUGH?

- Is there a golden ratio of test to product code?

THE PROCESS

- Add a Test
- Run all tests and see the new one fail
- Make a <u>little</u> change
- Run all tests and see them succeed
- Refactor (the forgotten step)

THE PROCESS TAKE 2

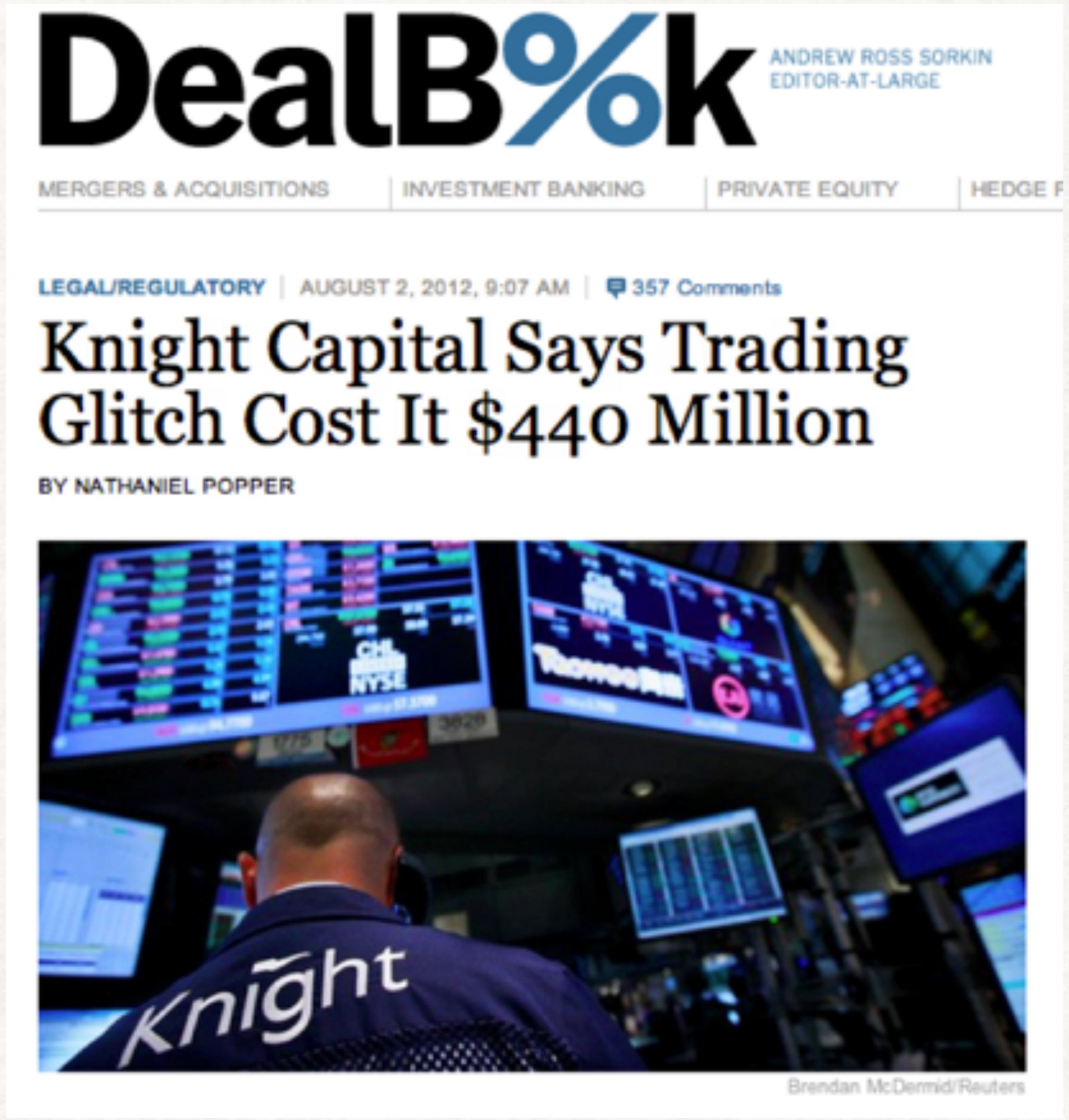
- Write a failing test
- Write code to make it pass
- Repeat the above
- Refactor aggressively
- When you cannot think of any more tests - your done

WHY TDD?

- Move from verification to specification
    - design decisions move up front
- Promotes understanding of code
- Model the API the way you want it
- Communicate the API
- Beach head against Tech Debt
- Can be used with any programming language

COST OF A DEFECT

- TDD reduces production bug density 40% — 80%
  - so says IBM System Sciences Institute
- Cost of a defect
  - Most expensive? (hint: production, as in 100x)



DealBook ANDREW ROSS SORKIN EDITOR-AT-LARGE

MERGERS & ACQUISITIONS | INVESTMENT BANKING | PRIVATE EQUITY | HEDGE F

LEGAL/REGULATORY | AUGUST 2, 2012, 9:07 AM | 357 Comments

Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER

Brendan McDermid/Reuters

CHANGE DEVELOPMENT

- Gamify development
- A failing test is a challenge
- Move away from the debugger

DISADVANTAGES

- Difficult and unintuitive at first
- Requires investment
    - Just how much?
    - Need to change perspective on feature life/cost
- Many do not see advantages until too late
- Requires discipline

ADVANTAGES

- Combat over engineering
  - build only what is necessary
- Produce testable code
- Build your regression suite as you go
- Helps to define done
- One test promotes ideas of others…
- Provide a way to duplicate defects and then fix
- Facilitates CI/CD
- Enables success with Agile based methodologies

UNCLE BOB'S 3 TDD LAWS

- You may not write production code until you have written a failing test
- You may not write more of a unit test than is sufficient to fail (not compiling is failing)
- You may not write more production code than is sufficient to pass the currently failing test.

ADOPTING TDD (INDIVIDUAL)

- Practice. No magic here.
- TDD every new function, module, etc.
- Use it to learn a new language

ADOPTING TDD (TEAM)

- Easier in greenfield projects
  - Measure - must have a solid feedback loop
- Brownfield projects
  - Only touch code when a test has been put in place first
  - We can talk about “testing Thursdays” but much of that is magical thinking.

MEASURING

- Know your code coverage
  - What is your target %?
- Pair programming?
- Sonar
- Code reviews
  - Studies show that code reviews have the same impact as TDD: each hour saves 33

UNIT TESTING

- All TDD is unit testing
- Isolated
  - No calls to FS or RDBMS*
  - Stub/mock dependencies
- Fast
- Use to replicate bugs
- Developer domain
- **Think foundational - wedding cake

INTEGRATION TESTING

- Testing interactions between components there are covered by TDD tests
- Integrating components from multiple teams
- Developer maybe with some QE

ACCEPTANCE TESTING

- Stakeholder/Developer collaboration
- Under the UI layer
    - Think Microservice REST calls
- Cucumber-like frameworks (BDD, DSL)
- Developer, Tech manager, QE

SYSTEM TEST

- Function of the system as a whole
- UI involved
- QE Teams

- UI involvement is problematic

SECURITY TESTING

- System secure?
- DDOS, account breach, etc
    - Several tools available
- Results often in creation of more Unit tests to “plug” holes
- Developers, Tech managers, QE

LOAD/PERFORMANCE

- Manage demand
- Soak Tests
- Using load testing software like JMeter
- Developers, QE

ROBUSTNESS TESTING

- Coping with system errors and outages
- How does the system handle system failures
- Random fault injection
- Chaos monkey

The 2014 AWS Reboot

“WHEN WE GOT THE NEWS ABOUT THE EMERGENCY EC2 REBOOTS, OUR JAWS DROPPED. WHEN WE GOT THE LIST OF HOW MANY CASSANDRA NODES WOULD BE AFFECTED, I FELT ILL. THEN I

CHRISTOS KALANTZIS, NETFLIX CLOUD DB ENGINEERING

KEEP IN MIND…

- The testing levels further up the pyramid (or cake) creates more unit tests!

INFRASTRUCTURE CHANGES NEEDED

- CI/CD Server
- Devops practice (not a 100% requirement but helps)
- You go as a team - the weak link will have an impact on the rest

TESTING LIBRARIES

- Warning: Religious debate territory
- https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks
- Key: pick one
- My Advice: pick the simplest one. You don't want a framework you have to fight

CODE COVERAGE

- What amount of your production code is covered by unit tests
    - Yes, we can debate the quality of those tests..
- Line coverage
    - Conditional paths (if, switch, while, else, etc)
- Cyclomatic Complexity
    - Code complexity
    - Aim for 5. Never more than 10.

COVERAGE

- Code is instrumented
- Code paths are counted

ASSERTION LIBRARIES

- There are many
- IMHO, simple is better
- Do you really need more than equal, pass, fail and maybe a variant of deep equal?
- Be careful of frameworks that:
  - Too much configuration
  - Clutter the global space
  - Make shared state too easy

ISOLATED TESTING

- Mocks - pre-programmed objects for your test
- Fakes - working implementations for your test
- Stubs - canned answer calls. May record info about calls.
- Dummy - Used to fill parameters
- The need for too much of this is a code smell. Some would say that the need to these at all are a code smell. Single responsibility principle.

BEST PRACTICES

- Code review - find a way to do it if you do not
- Anti Pattern: mock returning mock (mock graphs)
    - As discussed.. mocks often are code smells
- Test a single concept in one test function. Avoid the rambling test.
- Trust your VCS
- Old code should be refactored out
- Fail Fast and early

THE NAYSAYERS

IT TAKES A LOT OF TIME

SEEMS I WILL SPEND MY WHOLE LIFE TESTING!

TESTING SUCKS WHEN ASKED TO MAKE A CHANGE.

WASN'T TAUGHT THAT WAY!

WHAT THE PROPONENTS SAY

- As a programmer, do you deserve to feel confident? (Sleep at night knowing your code works?) Kent Beck
- Primary benefit of TDD is self testing code. Kent Beck
- Testing extends what the interpreter/compiler does: checks against your domain to ensure what you are doing is accurate.
- Just like surgeons should not have to defend hand washing, programmers should not have to defend TDD. Bob Martin
- Legacy code is simple code with no tests. Michael Feathers

RECAP

- Test First - Always
- Learn your IDE - speed at the keyboard
- Learn your VCS
- Turn it into a game