Coding Dojo World Tour
(of London)

@jasongorman

Software craftsmanship 2012

June 14th, Bletchley Park

www.softwarecraftsmanship.org.uk

#sc2012

BLETCHLEY PARK National Codes Centre

codemanship

www.codemanship.co.uk

codemanship | home

www.codemanship.co.uk

codemanship

"It's not how fast we deliver that decides the winners & losers. It's how fast we *learn* from what we deliver"

home | about | training | blog | contact

"Learning. Caring. Practicing. Sharing"

The Four Pillars Of Software Craftsmanship

We train, coach and consult in:

- Test-driven Development
- Refactoring
- Continuous Integration & Build Automation
- OO, Component-based & Service-Oriented

Software craftsmanship 2012

Proud organisers of the original Software Craftsmanship conference

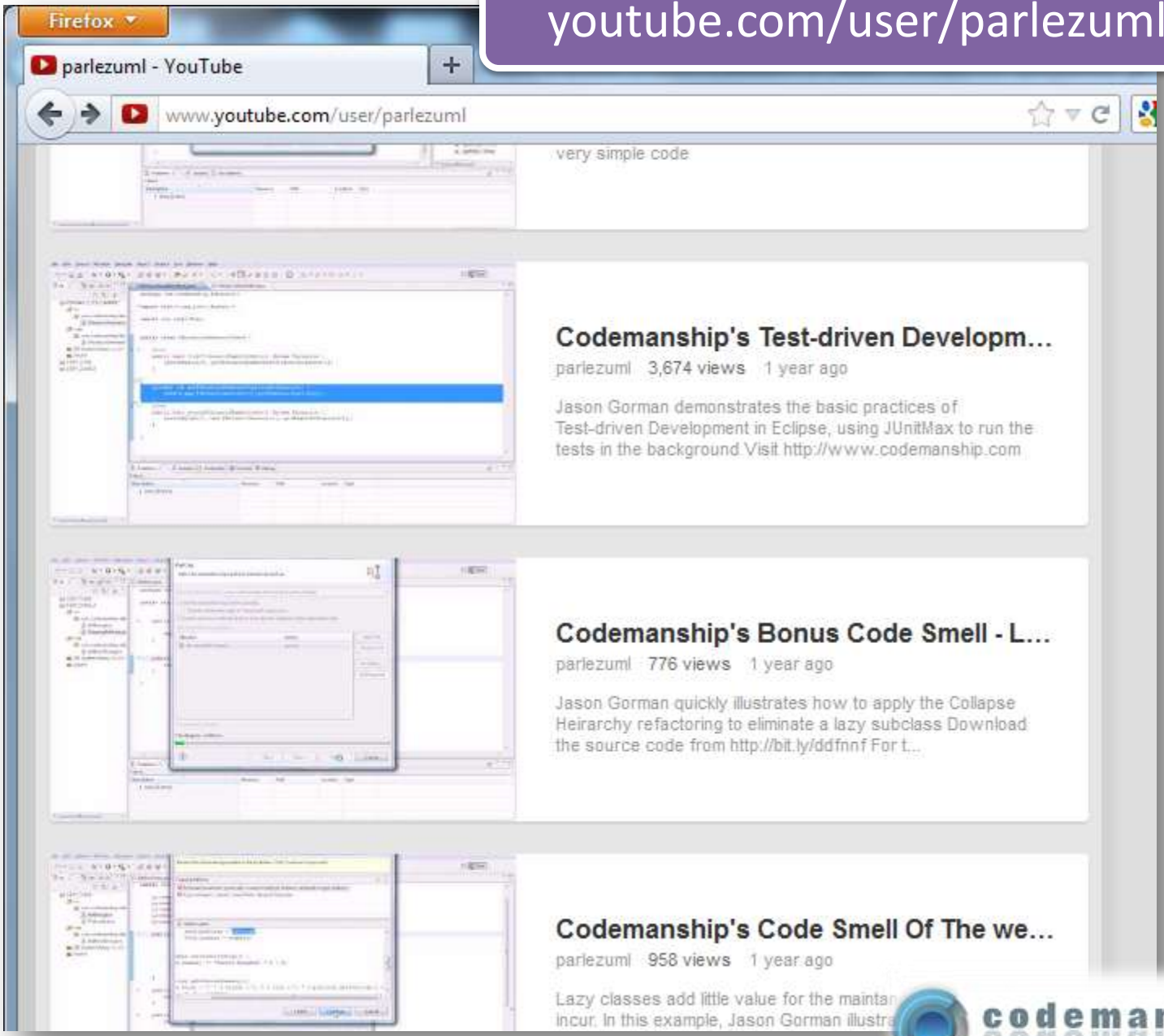
Coding Dojo Tour
Jason Gorman will be visiting organisations throughout the week of March 19-23 to run free coding dojos
More...

Slow & Dirty
Jason Gorman's presentation from SPA2011 makes a strong case for raising quality if teams want to go faster
Download Slides (PDF)

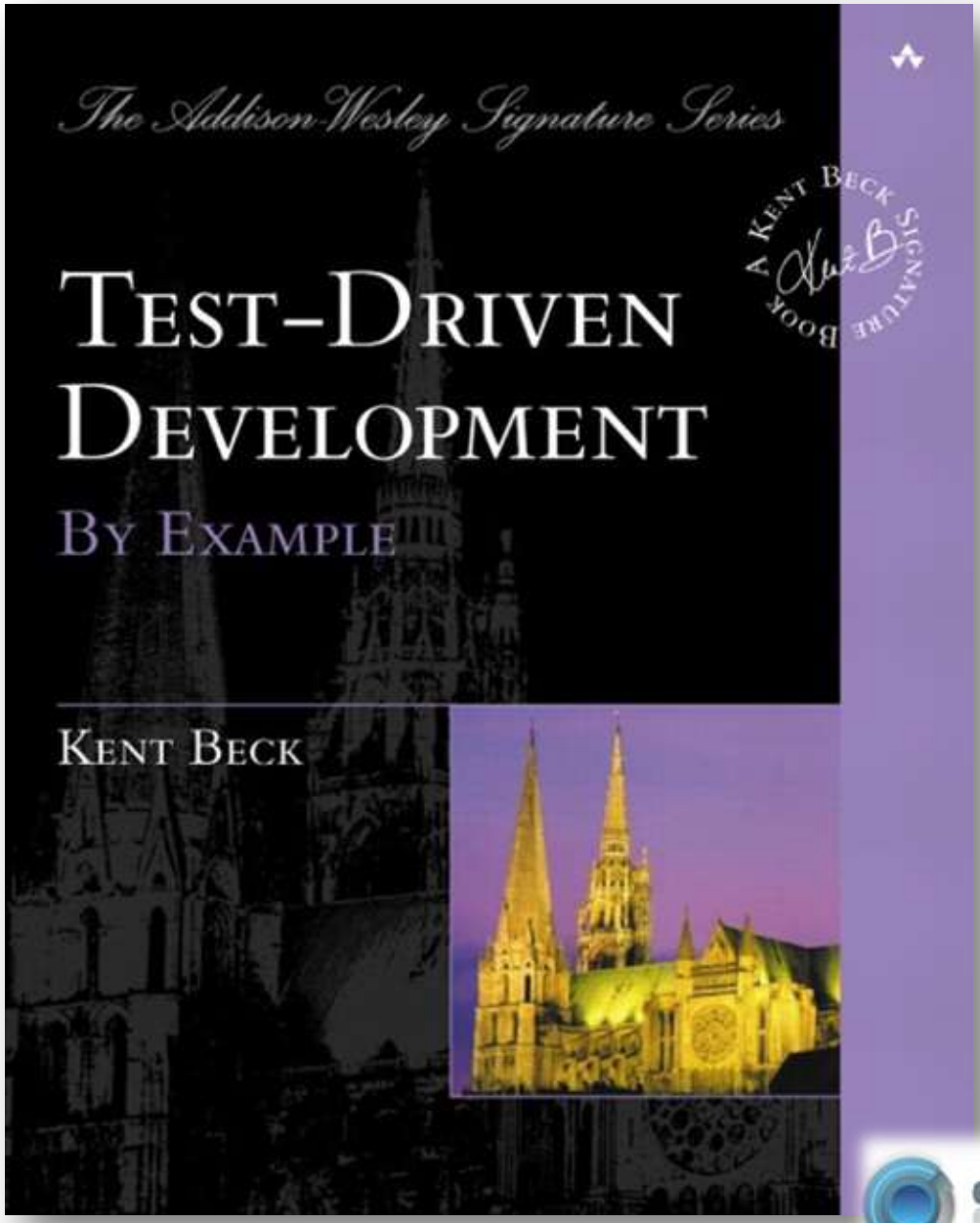
Dependable Dependencies

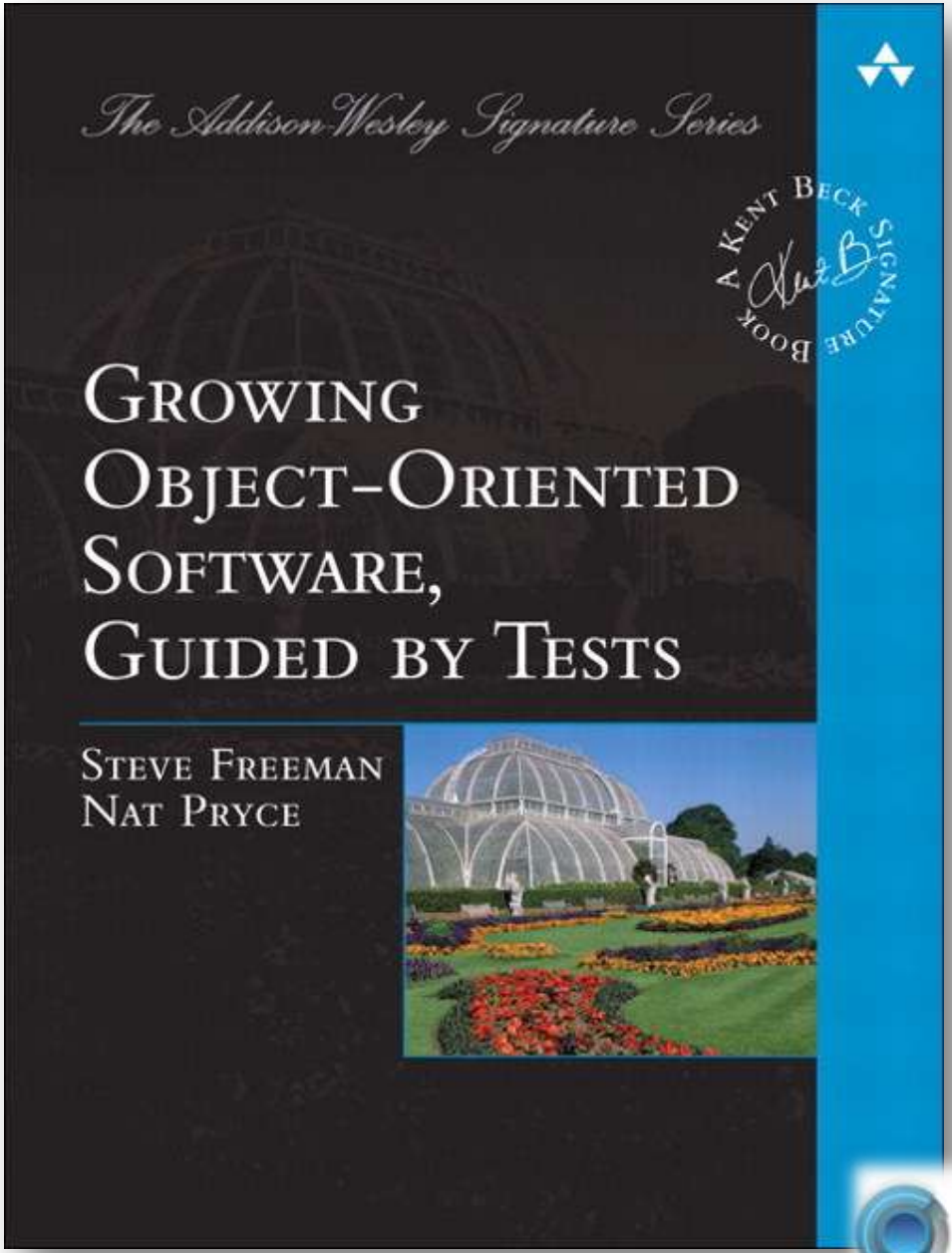
codemanship

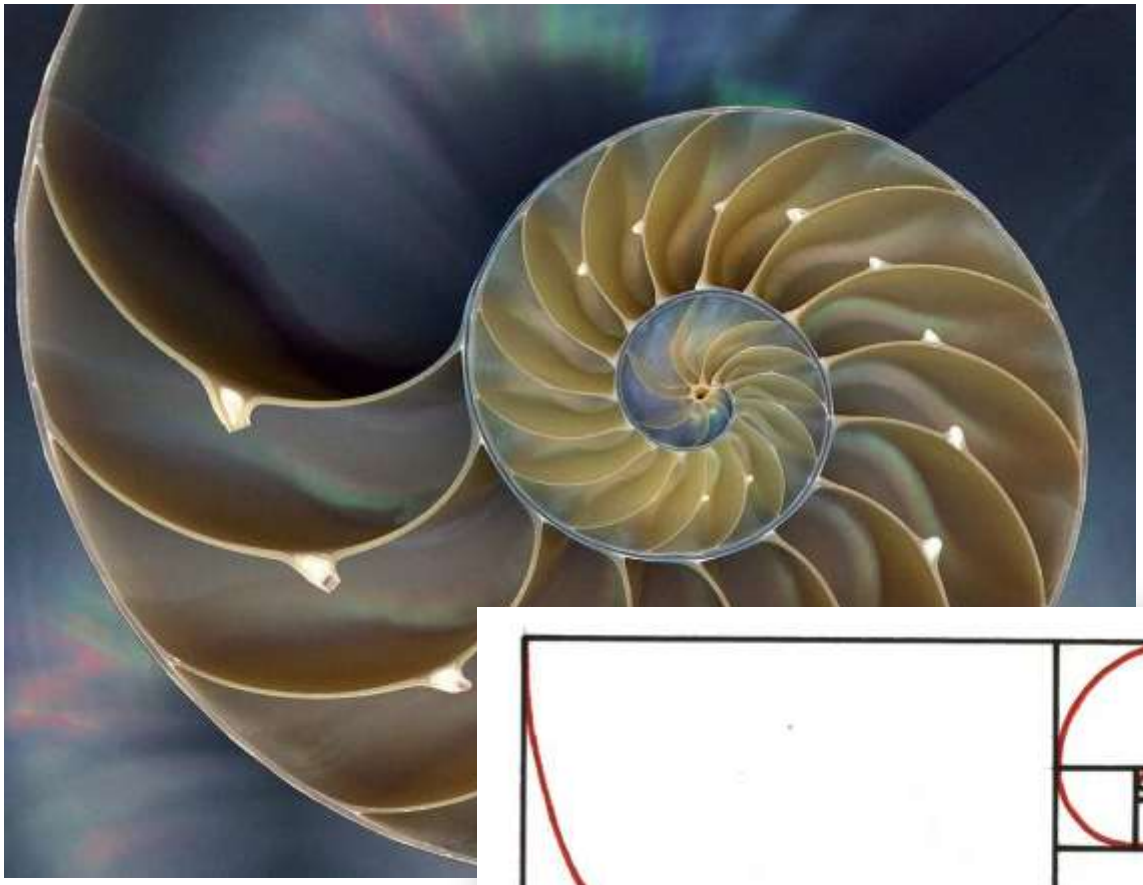
youtube.com/user/parlezuml



codemanship





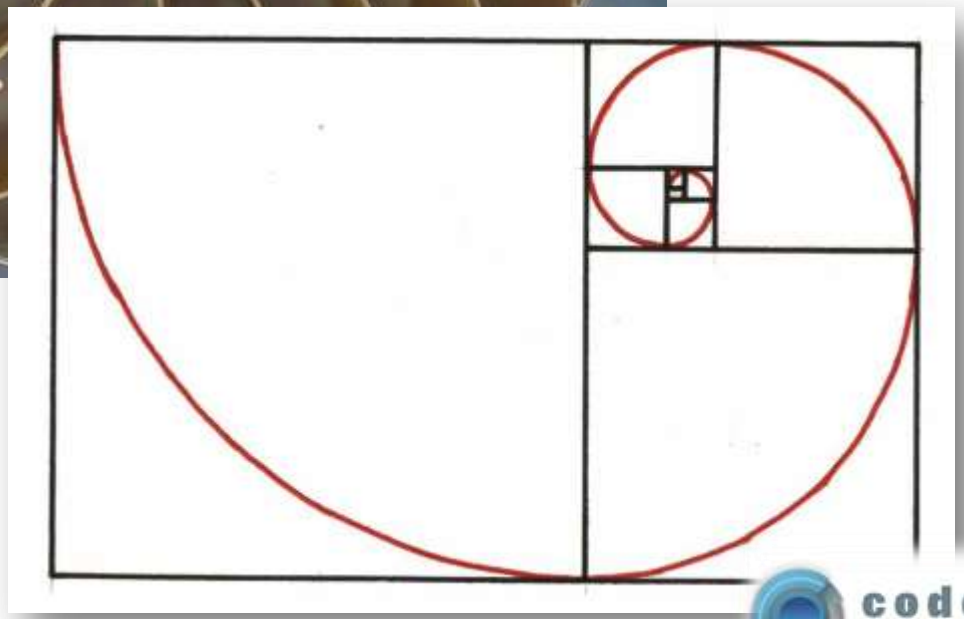


Fibonacci Sequence Generator

$$F_n = F_{n-1} + F_{n-2},$$

seed values

$$F_0 = 0,\ F_1 = 1.$$



codemanship

FizzBuzz

Write some code that will generate a string of integers, starting at 1 and going up to 100, all separated by commas.

Substitute any integer which is divisible by 3 with “Fizz”, and any integer which is divisible by 5 with “Buzz”, and any integer divisible by 3 and 5 with “FizzBuzz”

1,2,Fizz,4,Buzz,Fizz,7,8,Fizz,Buzz,11,Fizz,13,14,FizzBuzz… etc

codemanship

By the Book

Perform Fibonacci Generator or FizzBuzz without breaking ANY of these rules:

1. Start by writing a failing test
2. Write the simplest code to pass the test
3. Refactor to remove any duplication
4. Write the assertion first and work backwards
5. Run the test to see the assertion fail
6. Don’t refactor with failing tests
7. Write meaningful tests
8. Keep your test and model code separate
9. Organise your test code to make it easy to find
10. Isolate your tests so they can pass when run in any order
11. Triangulate interesting behaviour
12. Write tests that only have one reason to fail

codemanship

From Hell

Break ALL these rules in the fewest number of passing tests:

1. Start by writing a failing test
2. Write the simplest code to pass the test
3. Refactor to remove any duplication
4. Write the assertion first and work backwards
5. Run the test to see the assertion fail
6. Don’t refactor with failing tests
7. Write meaningful tests
8. Keep your test and model code separate
9. Organise your test code to make it easy to find
10. Isolate your tests so they can pass when run in any order
11. Triangulate interesting behaviour
12. Write tests that only have one reason to fail

codemanship

As If You Meant It

Perform Fibonacci Generator or FizzBuzz, writing the code to pass the tests INSIDE the tests, and discover the design entirely through refactoring

(first run by Keith Braithwaite at Software Craftsmanship 2009)

codemanship