eZ®
Share your information

Test-Driven Development

pfcongrez - Maarsen, the Netherlands

Derick Rethans - dr@ez.no

http://derickrethans.nl/talks.php

eZ
Share your information

About Me

- Dutchman living in Norway
- eZ Systems A.S.
- eZ Components project lead
- PHP development
- mcrypt, input_filter, date/time support, unicode
- QA

It's OK to write code that does not work

It's OK to write code that does not work

About Testing

Web Application Testing

Testable Parts

Front-end

- Acceptance Tests and System Tests that run in the browser
- Testing of Web Services with Unit Tests
- Compatibility Testing for Browser/OS/etc. combinations
- Performance Testing
- Security Testing

Back-end

- Functional Testing of business logic with Unit Tests
- Reusable Components, but they often come with their own tests

Testing Methods

Unit Testing

Tests small parts of an application or library (units) for correctly working code. Tools: PHPUnit, SimpleTest

System Testing

The testing of a whole integrated system against the specified requirements. Tools: Selenium

Non-functional Testing

Testing for performance, load, stress, reliability, availability, security. Tools: ab, siege, httperf, chorizo

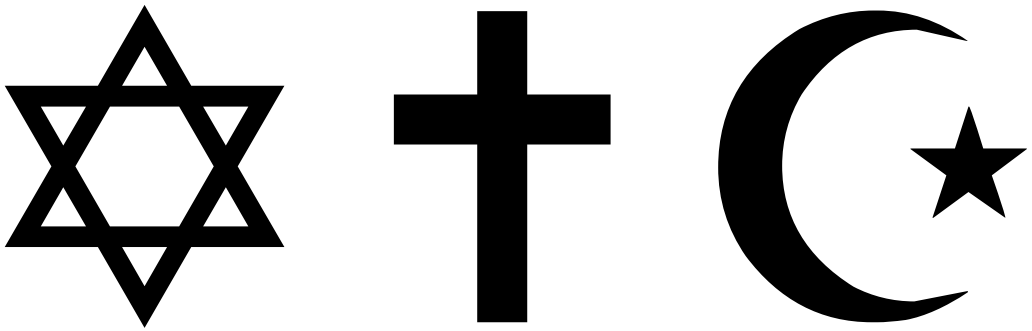
Test-Driven Development

a feature without a test is not a feature

Test-Driven Development

It is not just a method of testing software.

It is a religion.



Back In Time

Traditional Development Phases

Requirements Specification

Define what the software is supposed to do.

Design

Define how the software is supposed to be implemented.

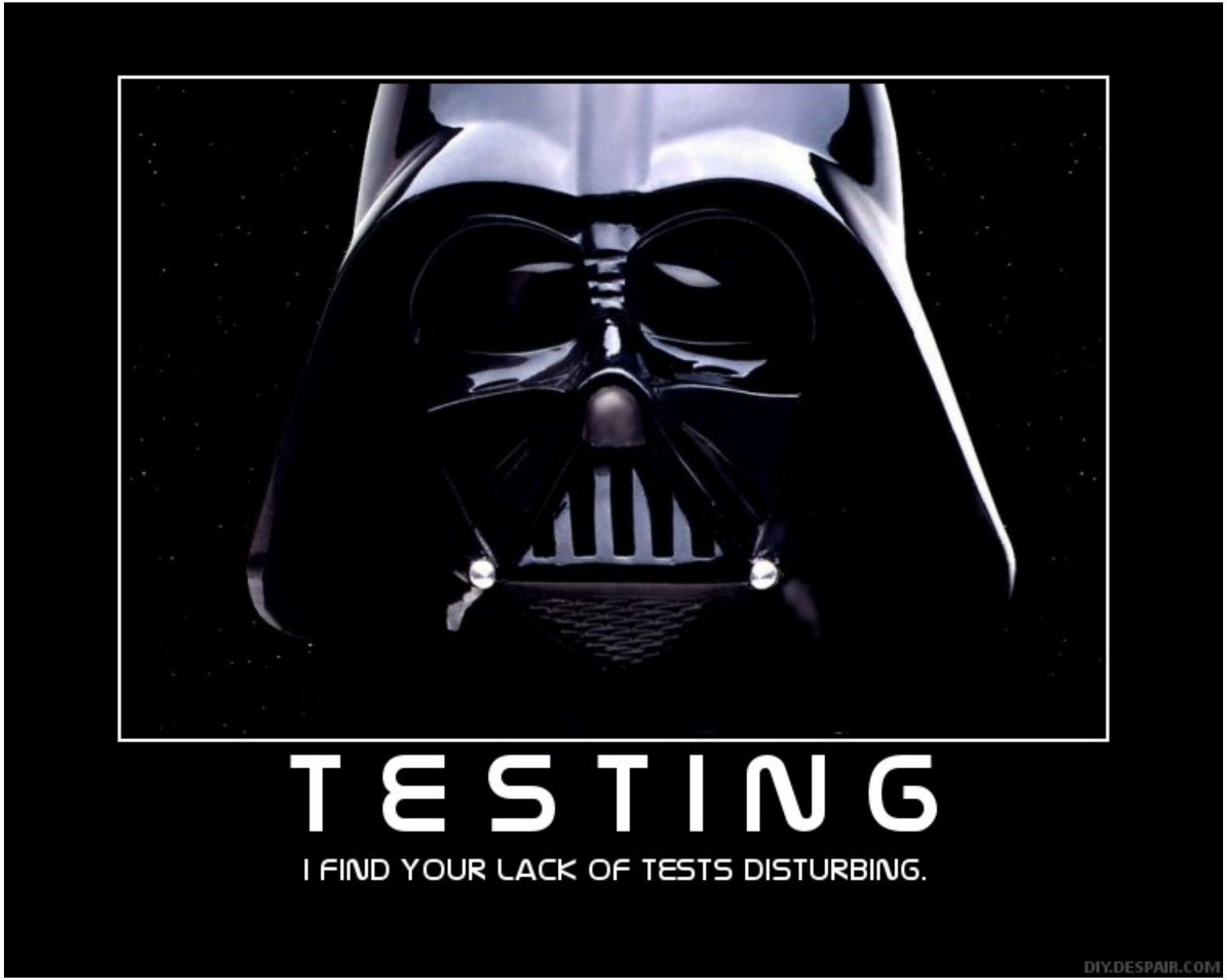
Implementation

The implementation of the software itself.

Testing

The implemented software is tested.

Sometimes.


TESTING
I FIND YOUR LACK OF TESTS DISTURBING.
DIY.DESPAIR.COM

Tests are King

Tests drive the development

- Tests are written before the code
- There is no code without tests

Test Suites

- Contain tests that check whether the code does what it is supposed to do
- Also cover things that should fail

Into the Future

Test-Driven Development Phases

Requirements Specification

Define what the software is supposed to do.

Design

Define how the software is supposed to be implemented.

Implementation ≡ Testing

The implemented software is tested.

The implementation of the software itself.

Present the Idea

Write the Requirements Document

Design the Component

Implementation

- Write API stubs with parameter documentation and descriptions
- Write test cases
- Initial implementation
- Initial implementation review
- Updating implementation according to review
- Implementation review

Pre-release Testing

Test-Driven Development

Addressing Defects

Write Test Case

Write a test case to test the correct behavior of the method, and verify that the test case fails

Fix the Issue

Fix the implementation

Verify Fix with Test Case

Make sure that the test cases pass with the new implementation

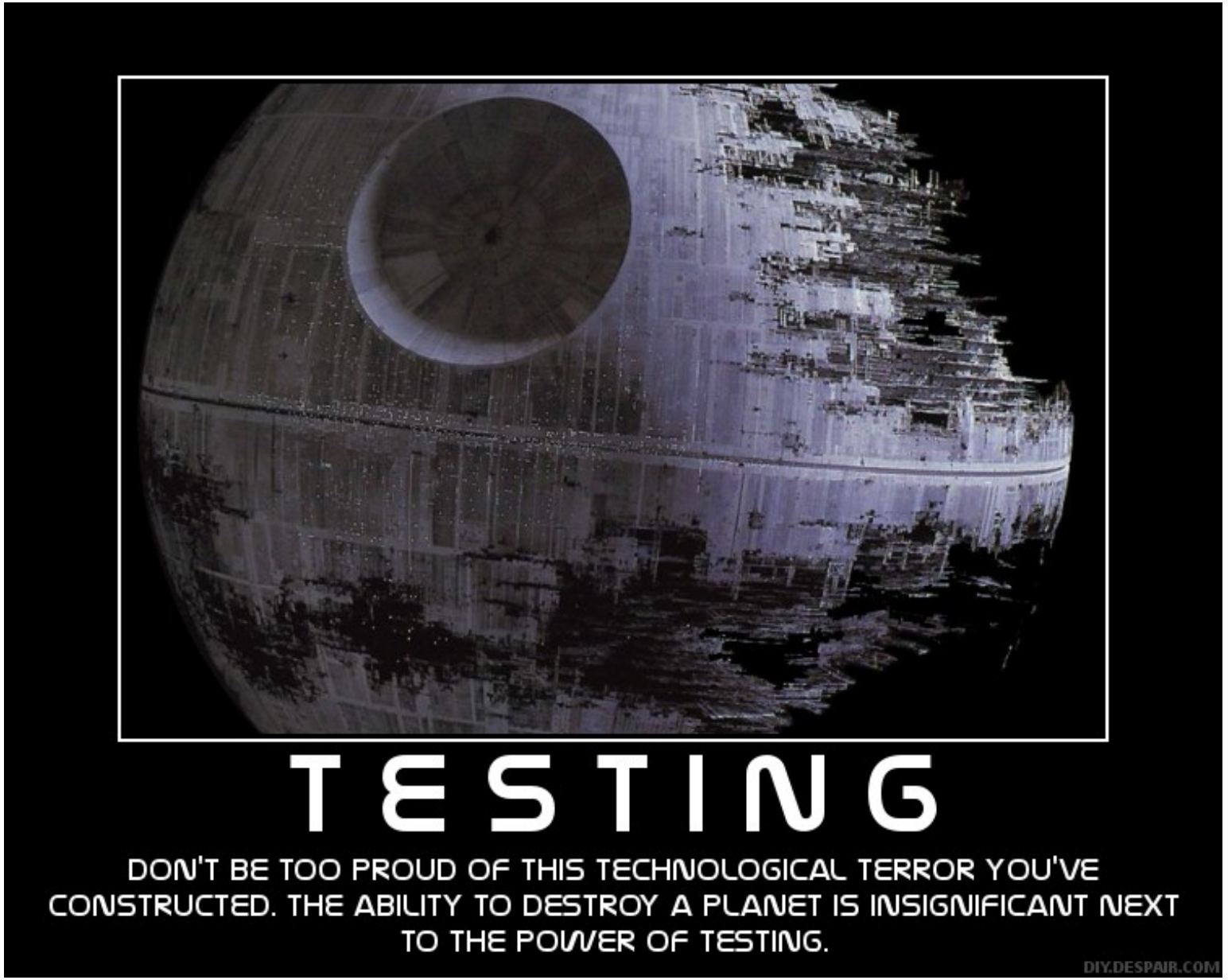
Check Other Test Cases

Verify that the fix for this defect did not break any other test case

The Way of Testivus

- If you write code, write tests.
- Don’t get stuck on unit testing dogma.
- Embrace unit testing karma.
- Think of code and test as one.
- The test is more important than the unit.
- The best time to test is when the code is fresh.
- Tests not run waste away.
- An imperfect test today is better than a perfect test someday.
- An ugly test is better than no test.
- Sometimes, the test justifies the means.
- Only fools use no tools.
- Good tests fail.

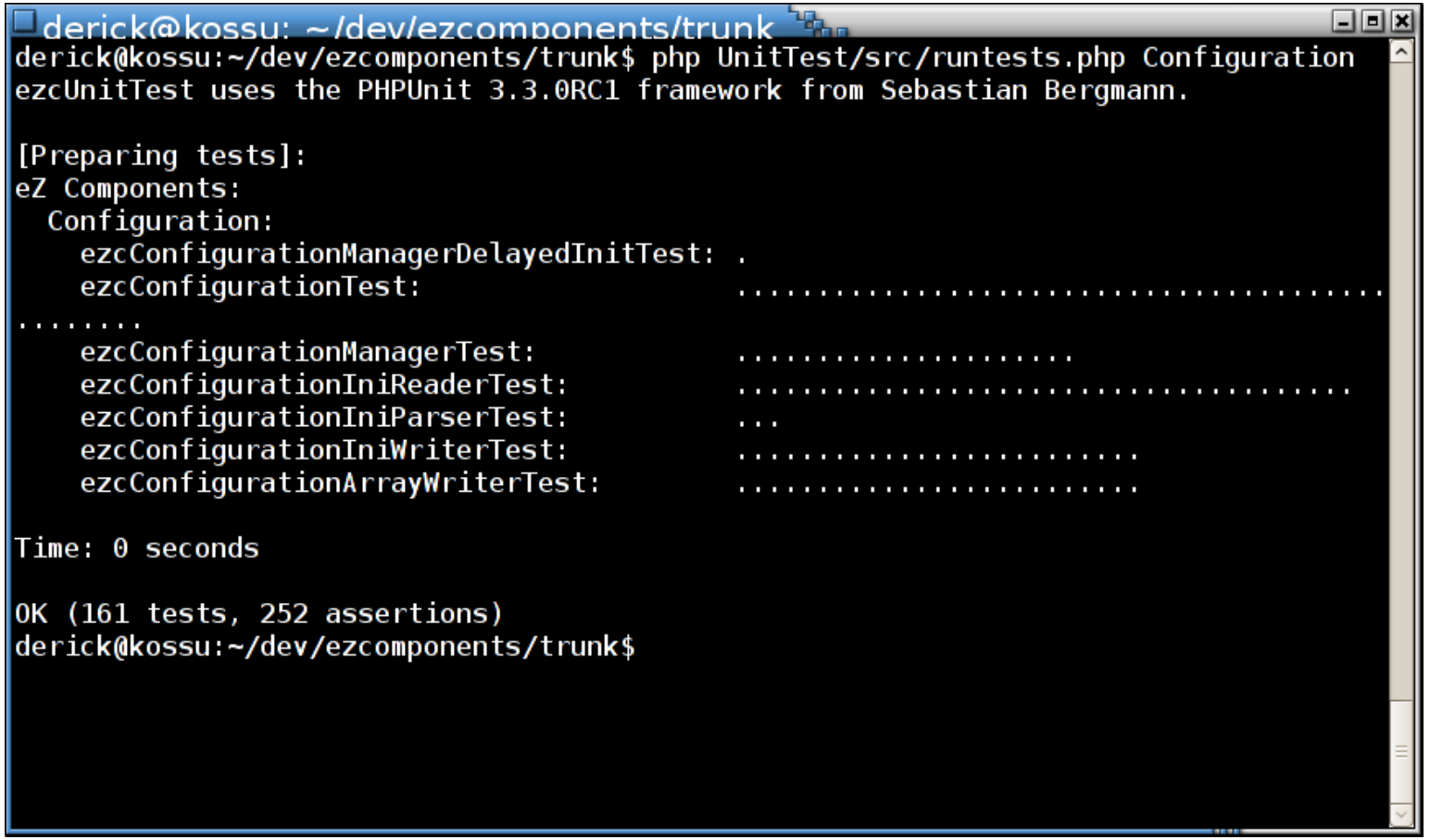
http://www.artima.com/weblogs/viewpost.jsp?thread=203994


TESTING
DON'T BE TOO PROUD OF THIS TECHNOLOGICAL TERROR YOU'VE CONSTRUCTED. THE ABILITY TO DESTROY A PLANET IS INSIGNIFICANT NEXT TO THE POWER OF TESTING.
DIY.DESPAIR.COM

PHP Unit

PHPUnit

Unit Testing Framework for PHP



PHPUnit

Code Coverage: No Code Without Test

eZ Components

Current directory: /home/derick/dev/ezcomponents/trunk/Graph/src

Legend: Low: 0% to 35% | Medium: 35% to 70% | High: 70% to 100%

	Coverage								
	Classes			Methods			Lines		
Total		94.39%	101 / 107		86.89%	444 / 511		86.32%	8990 / 10415
axis		100.00%	5 / 5		85.25%	52 / 61		91.02%	669 / 735
charts		100.00%	5 / 5		100.00%	36 / 36		99.56%	683 / 686
colors		100.00%	3 / 3		100.00%	17 / 17		100.00%	216 / 216
data_container		100.00%	2 / 2		100.00%	13 / 13		100.00%	39 / 39
datasets		100.00%	9 / 9		89.80%	44 / 49		93.63%	235 / 251
driver		50.00%	3 / 6		49.47%	47 / 95		54.97%	1339 / 2436
element		100.00%	4 / 4		100.00%	21 / 21		99.35%	458 / 461
exceptions		96.15%	25 / 26		96.00%	24 / 25		95.50%	106 / 111
interfaces		100.00%	7 / 7		97.73%	43 / 44		93.74%	943 / 1006
math		100.00%	7 / 7		100.00%	43 / 43		99.72%	350 / 351
options		92.86%	13 / 14		90.32%	28 / 31		94.11%	815 / 866
palette		83.33%	5 / 6		100.00%	0 / 0		83.33%	5 / 6
renderer		100.00%	8 / 8		100.00%	67 / 67		96.19%	3005 / 3124
structs		100.00%	3 / 3		100.00%	7 / 7		100.00%	35 / 35
graph.php		100.00%	1 / 1		100.00%	0 / 0		100.00%	1 / 1
tools.php		100.00%	1 / 1		100.00%	2 / 2		100.00%	91 / 91

Generated by PHPUnit 3.3.0RC1 and Xdebug 2.1.0-dev at Mon Sep 8 10:30:06 CEST 2008.

PHPUnit

Code Coverage: No Code Without Test

```
550                 :
551              19 :                 switch ( $this->interval )
552                 :                 {
553              19 :                     case self::MONTH:
554               4 :                         $time = strtotime( '+1 month', $time );
555               4 :                         break;
556              15 :                     case self::YEAR:
557               4 :                         $time = strtotime( '+1 year', $time );
558               4 :                         break;
559              11 :                     case self::DECADE:
560               0 :                         $time = strtotime( '+10 years', $time );
561               0 :                         break;
562              11 :                     default:
563              11 :                         $time += $this->interval;
564              11 :                         break;
565              19 :                 }
566              19 :             }
567                 :
568              19 :             return $steps;
569                 :         }
570                 :
571                 :         /**
```

Green: Covered code.

Red: Not covered code.

Grey: Unreachable code.

White: No code.

PHPUnit

Code Coverage: How important?

< 100% Coverage ≡ Not fully tested code

100% Coverage ≢ Fully tested code

How much coverage should you aim for?

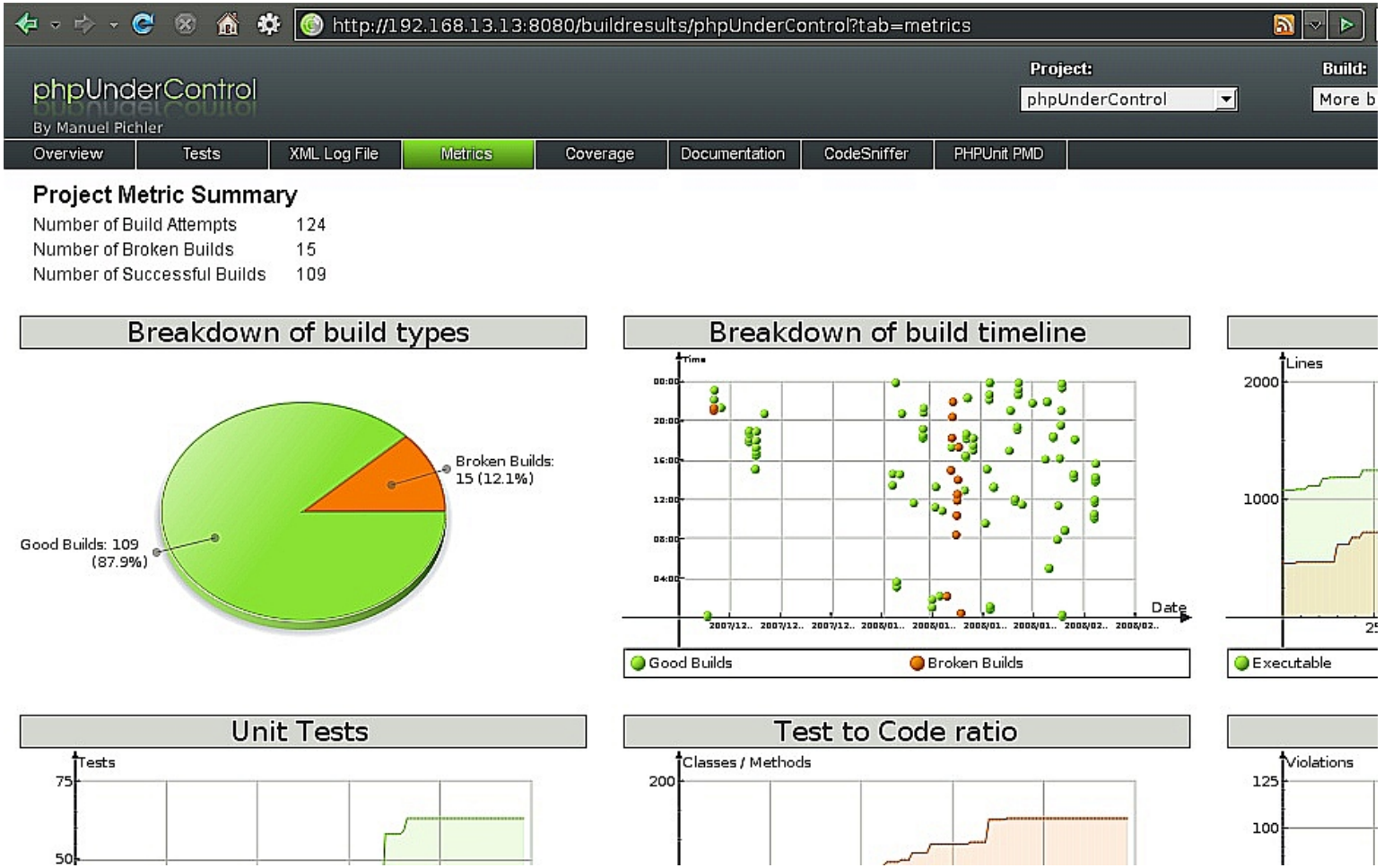
- It doesn't matter, just write some tests.
- How much rice do I need to cook?
- 100%, and no less!

http://www.developertesting.com/archives/month200705/20070504-000425.html

eZ
Share your information

phpUnderControl

Continuous integration with CruiseControl



Un-testable Problems

- Testing singletons more than one time
- System/Operating System dependent tests
- Private methods
- Code that depends on the state of an external resource
- Things that simply should never happen

Politics

resistance to change

Fear - more work to do

- Introduce TDD concepts gently
- Whenever a problem is found, make and retain a test case for futher use
- Start using TDD for new projects

Managers

Ignorance - too much time spent on testing

- Out of date with modern processes
- Belief that testing slows the schedule (only if you follow the ship-and-see process)
- You save time later, because you wouldn't have to re-test or re-debug newly written additions or big fixes.

TESTS
WHERE ARE YOURS ?!

Case Studies

Case Studies

Microsoft Case Study

- TDD project has twice the code quality
- Writing tests requires 15% more time

IBM Case Study

- 40% fewer defects
- No impact on the team's productivity

John Deere / Ericsson Case Study

- TDD produces higher quality code
- Impact of 16% on the team's productivity

Case Studies
Developer Comments

- At first I didn't like that I need to write tests for my code, but now after using it for more than 10 months I can't program without it.
- Helps to come up with better APIs.
- It gives confidence that our software is working well at all times. Even after making major changes and/or changing software we are dependent on.
- Productivity increases - you might loose some when you make the initial tests, but you'll get it back later. The code covered by tests is 'insured' against future changes.
- It works well for libraries, not so well for GUI applications.
- Some things cannot be tested like server errors, unless

Questions

eZ
Share your information

Resources

- http://homepage.mac.com/hey.you/lessons.html
- Testuvius: http://www.artima.com/weblogs/viewpost.jsp?thread=203994

- PHP Unit: http://phpunit.de
- phpUnderControl: http://phpundercontrol.org
- Xdebug: http://xdebug.org
- These Slides: http://derickrethans.nl/talks.php

- PHP: http://www.php.net