

**International PHP Conference - Mörfelden, Germany**

**Derick Rethans - dr@ez.no**

**<http://derickrethans.nl/talks.php>**

- 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



Just Don't Ship It

# About Testing

### 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

## 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, httpperf, chorizo

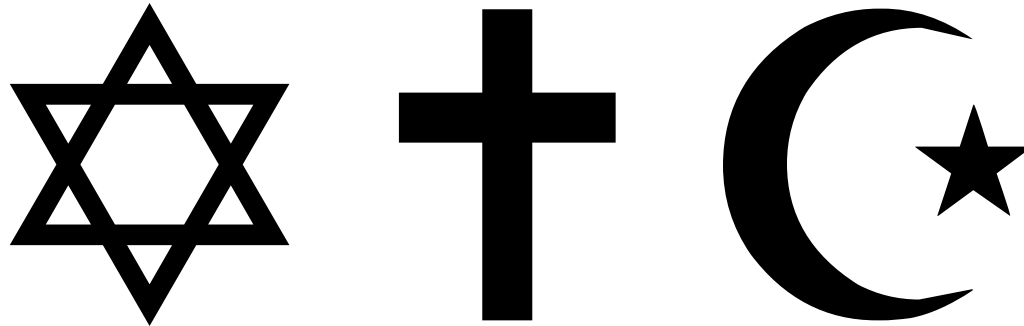
# Test-Driven Development

a feature without a test is not a feature



**It is not just a method of testing  
software.**

**It is a religion.**



### 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.

## 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

### Requirements Specification

Define what the software is supposed to do.

### Design

Define how the software is supposed to be implemented.

### Implementation $\equiv$ 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**

### 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

# PHP Unit



# PHPUnit

## Unit Testing Framework for PHP

```
derick@kossu: ~/dev/ezcomponents/trunk
derick@kossu:~/dev/ezcomponents/trunk$ php UnitTest/src/runtests.php Workflow
ezcUnitTest uses the PHPUnit 3.0.6 framework from Sebastian Bergmann.

[Preparing tests]:
eZ components:
  Workflow:
    ezcWorkflowDefinitionStorageXmlTest: .....
    ezcWorkflowExecutionTest: .....
    ezcWorkflowTest: .....
    ezcWorkflowNodeTest: .....
    ezcWorkflowConditionTest: .....
    ezcWorkflowVisitorVisualizationTest: .....

Time: 00:00

OK (132 tests)
derick@kossu:~/dev/ezcomponents/trunk$
```

Current view: [/home/derick/dev/ezcomponents/trunk/Graph/src](#)

Date: Thu Jun 14 11:04:17  
CEST 2007

Executable lines: 8756

Code covered: 96.04%

Executed lines: 8409

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

	Coverage (show details)		
<a href="#">axis</a>	<div style="width: 90.67%; background-color: green; height: 10px;"></div>	90.67%	612 / 675 lines
<a href="#">charts</a>	<div style="width: 99.42%; background-color: green; height: 10px;"></div>	99.42%	518 / 521 lines
<a href="#">colors</a>	<div style="width: 100.00%; background-color: green; height: 10px;"></div>	100.00%	222 / 222 lines
<a href="#">data_container</a>	<div style="width: 100.00%; background-color: green; height: 10px;"></div>	100.00%	41 / 41 lines
<a href="#">datasets</a>	<div style="width: 93.88%; background-color: green; height: 10px;"></div>	93.88%	230 / 245 lines
<a href="#">driver</a>	<div style="width: 96.51%; background-color: green; height: 10px;"></div>	96.51%	1217 / 1261 lines
<a href="#">element</a>	<div style="width: 98.26%; background-color: green; height: 10px;"></div>	98.26%	451 / 459 lines
<a href="#">exceptions</a>	<div style="width: 98.89%; background-color: green; height: 10px;"></div>	98.89%	89 / 90 lines
<a href="#">interfaces</a>	<div style="width: 91.58%; background-color: green; height: 10px;"></div>	91.58%	859 / 938 lines
<a href="#">math</a>	<div style="width: 99.72%; background-color: green; height: 10px;"></div>	99.72%	359 / 360 lines
<a href="#">options</a>	<div style="width: 99.09%; background-color: green; height: 10px;"></div>	99.09%	762 / 769 lines
<a href="#">palette</a>	<div style="width: 100.00%; background-color: green; height: 10px;"></div>	100.00%	8 / 8 lines
<a href="#">renderer</a>	<div style="width: 95.71%; background-color: green; height: 10px;"></div>	95.71%	2811 / 2937 lines
<a href="#">structs</a>	<div style="width: 100.00%; background-color: green; height: 10px;"></div>	100.00%	39 / 39 lines
<a href="#">graph.php</a>	<div style="width: 100.00%; background-color: green; height: 10px;"></div>	100.00%	2 / 2 lines
<a href="#">graph_autoload.php</a>	<div style="width: 100.00%; background-color: green; height: 10px;"></div>	100.00%	103 / 103 lines
<a href="#">tools.php</a>	<div style="width: 100.00%; background-color: green; height: 10px;"></div>	100.00%	86 / 86 lines

```
151 : /**
152 :  * Ensure proper timestamp
153 :  *
154 :  * Takes a mixed value from datasets, like timestamps, or strings
155 :  * describing some time and converts it to a timestamp.
156 :  *
157 :  * @param mixed $value
158 :  * @return int
159 :  */
160 : protected static function ensureTimestamp( $value )
161 : {
162 :     19 :     if ( is_numeric( $value ) )
163 :     19 :     {
164 :         16 :         $timestamp = (int) $value;
165 :         16 :     }
166 :     3 :     elseif ( ( $timestamp = strtotime( $value ) ) === false )
167 :     :     {
168 :     0 :         throw new ezcGraphErrorParsingDateException( $value );
169 :     :     }
170 :     :
171 :     19 :     return $timestamp;
172 :     : }
```

< 100% Coverage  $\equiv$  Not fully tested code

100% Coverage  $\neq$  Fully tested code

- 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 further use
- Start using TDD for new projects

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)

# 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

- 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



These Slides: <http://derickrethans.nl/talks.php>

PHP: <http://www.php.net>