Extracting Complexity

Oxid Partner Day

Kore Nordmann (@koredn)

October 29th, 2013

Reasoning

Complexity Matters

- High complexity leads to . . .
  - . . . higher Time To Bugfix
  - . . . higher Time To Feature
  - . . . **Lower Business Value / Hour**

Reasoning

High complexity usually indicates mixed application and business logic.

We are

Helping people to create high quality web applications.

`http://qafoo.com`

- Trainings, Workshops and Consulting
- Twitter `@qafoo`

Measure Complexity

What makes code complex?

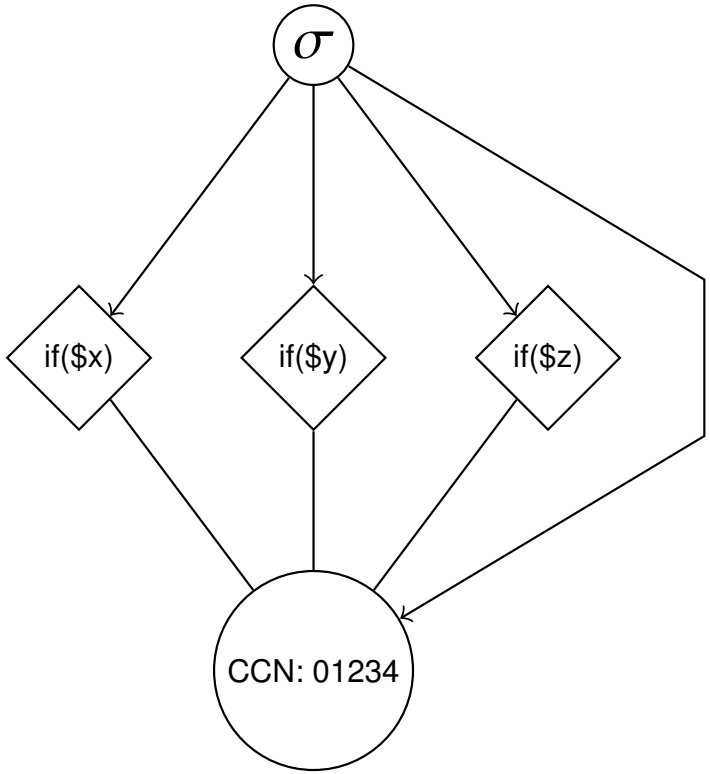
- Control structures are the key point to complexity:
  - if, elseif, for, while, foreach, catch, case, xor, and, or, &&, ||, ?:

Two Ways of Counting

- Cyclomatic Complexity (CCN)
  - Number of *branches*
  - Extended Cycomatic Complexity (CCN2) actually minds all those control structures
- NPath Complexity
  - Number of *execution paths*
  - Depends on the structure of code

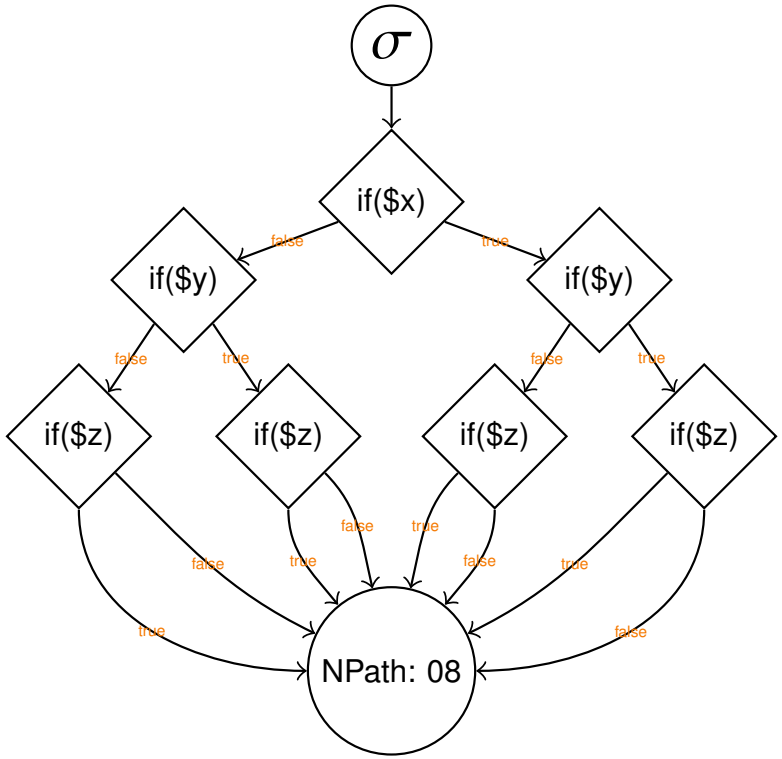
Cyclomatic Complexity

```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



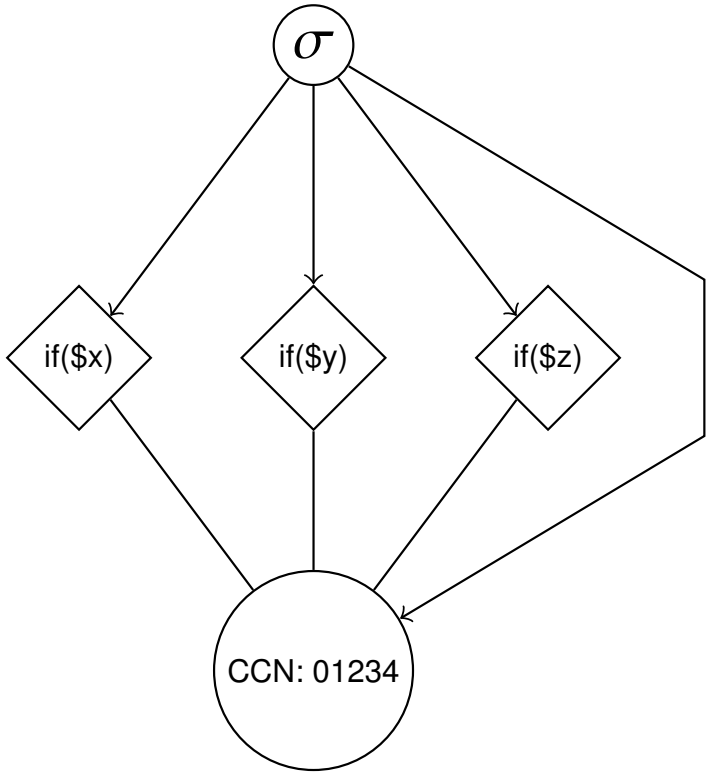
NPath Complexity

```
1  <?php
2  class Foo {
3      public function foo() {
4          if ($x) { }
5          if ($y) { }
6          if ($z) { }
7          return $x;
8      }
9  }
```



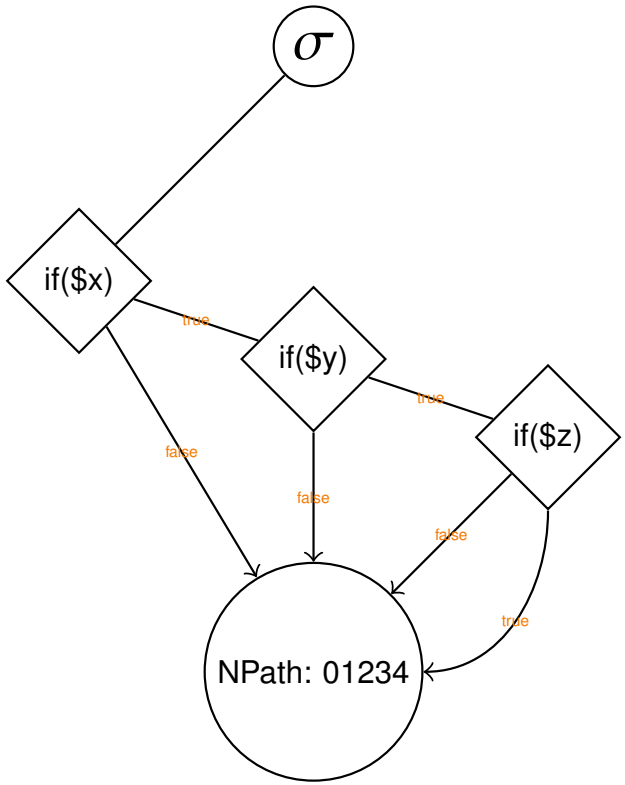
Cyclomatic Complexity

```php
<?php
class Foo {
    public function foo() {
        if ($x) {
            if ($y) {
                if ($z) { }
            }
        }
        return $x;
    }
}
```



NPath Complexity

```
1   <?php
2   class Foo {
3       public function foo() {
4           if ($x) {
5               if ($y) {
6                   if ($z) { }
7               }
8           }
9           return $x;
10      }
11  }
```



Comparing Structure

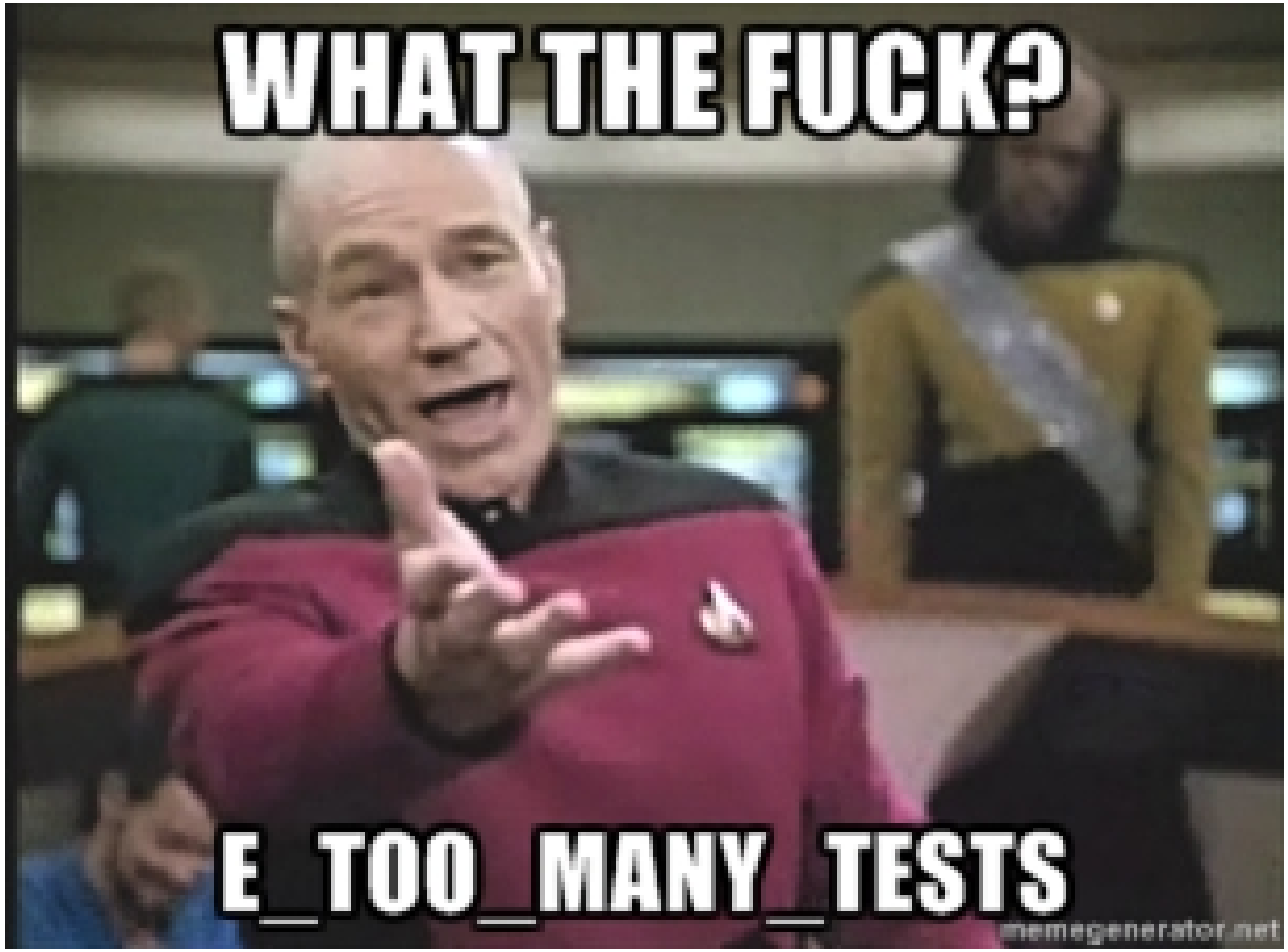
```php
<?php
class Foo {
    public function foo() {
        if ($x) {
            if ($y) {
                if ($z) { }
            }
        }
        return $x;
    }
}
```

```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```

How many tests do I need?

- Cover every line of code
  - Code Coverage: Shows which lines have been executed (by tests)
- Path Converage (been worked on)
  - Shows which execution paths have been covered
  - **You should write at least $nPath tests for every method!**
- Parameter Value Coverage
  - Test all execution paths with sane boundary values for every parameter
  - Common integer boundaries: $-2^{63}, -2^{31}, -1, 0, 1, 2^{31}, 2^{63}$
  - **You should write at least** $\$nPath * \$parameterCount * \$boundaries$ **tests per method!**

Are you kidding me?



Sensible limits

- Numbers do not tell anything by themselves
- To judge you need limiting values
  - Cyclomatic Complexity
    - 1-4: low, 5-7: medium, 8-10: high, 11+: hell
  - NPath Complexity
    - 200: critical mass
- Limiting values are at your discretion

Run yourself

```
$ pear install --alldeps phpmd/PHP_PMD
$ phpmd src/main/ text codesize

Review/Analyzer/PDepend/Model.php:165
    The method getAnnotations() has a Cyclomatic Complexity of 12. The configured
        cyclomatic complexity threshold is 10.
Review/Controller/Source.php:225
    The method toArray() has an NPath complexity of 251. The configured NPath complexity
        threshold is 200.
Review/DIC/Base.php:55
    The method initialize() has 142 lines of code. Current threshold is set to 100. Avoid
         really long methods.
Review/MySQLi.php:34
    The method __construct() has an NPath complexity of 31250. The configured NPath
        complexity threshold is 200.
```

Refactoring

- Code refactoring is the “disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior”
  - “Change code, but do not break it”
  - (Functional) tests are *really, really* useful during refactoring.
- Goals
  - Increase maintainability (reduce complexity)
  - Increase testability
  - Increase re-usability

Common techniques

- Rename method
  - Readability / maintainability
- Extract method
  - Move reused code into its own methods
  - Reduces complexity
- Extract class
  - Move code segments into its own class / implementation
  - See: *Separation of Concerns*, *Interface Segregation Principle*
- Extract module / component
  - Make code reusable across projects
  - See: *Separation of Concerns*, *Interface Segregation Principle*, *Open Closed Principle*

Extract Method 101

- Context
  - Used variables
  - Modified variables
  - Exit points (`return`)
- Good tool support:
  - PHP Storm
  - PHP Refactoring Browser (by Qafoo)
  - ...

Extract Method

```php
<?php
class SearchController
{
    public function searchAction(Request $req)
    {
        if ($req->has('q')) {
            $solarium = new SolariumClient('localhost:8080');
            $select = $solarium->createSelect();

            // configure dismax
            $dismax = $select->getDisMax();
            $dismax->setQueryFields(array('name^2', 'description'));
            $dismax->setPhraseFields(array('description'));
            $dismax->setMinimumMatch(1);
            $dismax->setQueryParser('edismax');

            if ($req->has('q')) {
                $escapedQuery = $select->getHelper()->escapeTerm($req->get('q'));
                $select->setQuery($escapedQuery);
            }

            $paginator = new Pagerfanta(new SolariumAdapter($solarium, $select));
            $paginator->setMaxPerPage(15);
            $paginator->setCurrentPage($req->get('page', 1), false, true);

            // ... here be 100 lines
        }
    }
```

Extract Method

```php
<?php

class SearchController
{
    public function searchAction(Request $req)
    {
        if ($req->has('q')) {
            $query= $req->get('q');
            $products = $this->findProducts($query);
        }
    }

    protected function findProducts($query)
    {
        // ... here be 100 lines
    }
```

Extract Class

```php
<?php

class SearchController
{
    public function searchAction(Request $req)
    {
        if ($req->has('q')) {
            $query= $req->get('q');
            $finder = new ProductFinder();
            $products = $finder->search($query);
        }
    }
}
```

Extract Class

```
1  <?php
2
3  class ProductFinder
4  {
5      public function search($query)
6      {
7          // ... here be 100 lines
8      }
9  }
```

Summary

- Complexity metrics highlight potential problems
- Extracting classes and methods lead to better readable code
- . . . and you'll pass the Oxid module certification easily.

THANK YOU

Rent a quality expert
qafoo.com