How We Analyze Your Code

PHP Unconference Europe 2016

Kore Nordmann (@koredn)

29th May 2016

Qafoo
passion for software quality

php unconference

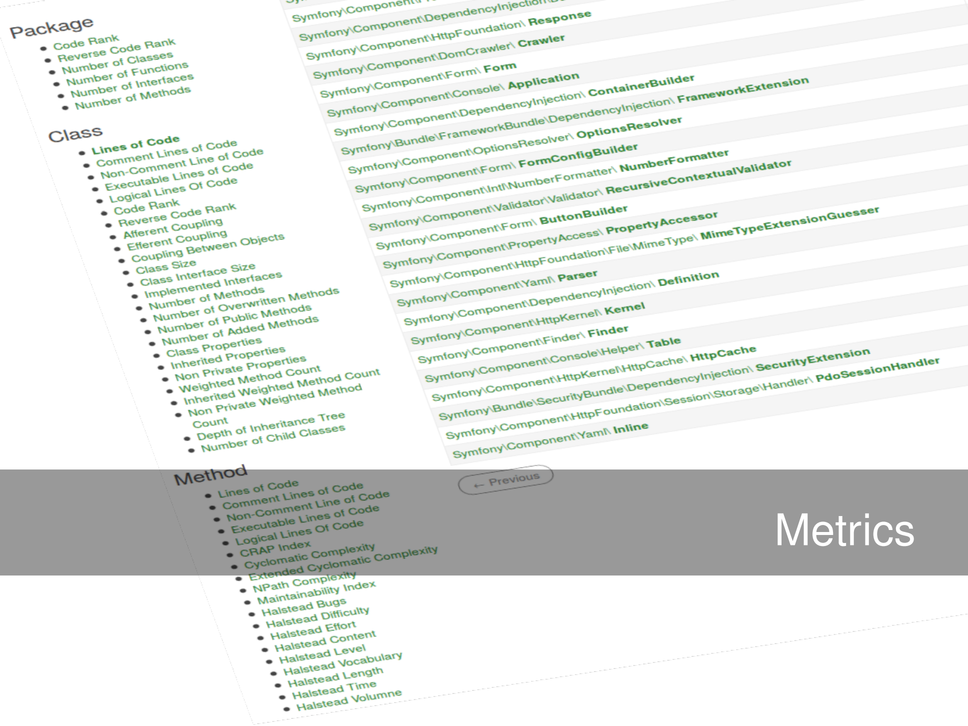
Hi, I’m Kore (@koredn)

R.P.M.
MANIFOLD PRESS
OIL PRESSURE
FUEL PRESSURE
OIL PRESS
FUEL PRESS
OIL TEMP

Metrics

Analyze Progress

Analyze Legacy Code



Package

- Code Rank
- Reverse Code Rank
- Number of Classes
- Number of Functions
- Number of Interfaces
- Number of Methods

Class

- **Lines of Code**
- Comment Lines of Code
- Non-Comment Line of Code
- Executable Lines of Code
- Logical Lines Of Code
- Code Rank
- Reverse Code Rank
- Afferent Coupling
- Efferent Coupling
- Coupling Between Objects
- Class Size
- Class Interface Size
- Implemented Interfaces
- Number of Methods
- Number of Overwritten Methods
- Number of Public Methods
- Number of Added Methods
- Class Properties
- Inherited Properties
- Non Private Properties
- Weighted Method Count
- Inherited Weighted Method Count
- Non Private Weighted Method Count
- Depth of Inheritance Tree
- Number of Child Classes

Method

- Lines of Code
- Comment Lines of Code
- Non-Comment Line of Code
- Executable Lines of Code
- Logical Lines Of Code
- CRAP Index
- Cyclomatic Complexity
- Extended Cyclomatic Complexity
- NPath Complexity
- Maintainability Index
- Halstead Bugs
- Halstead Difficulty
- Halstead Effort
- Halstead Content
- Halstead Level
- Halstead Vocabulary
- Halstead Length
- Halstead Time
- Halstead Volumne

Symfony\Component\
Symfony\Component\DependencyInjection\
Symfony\Component\HttpFoundation\ **Response**
Symfony\Component\DomCrawler\ **Crawler**
Symfony\Component\Form\ **Form**
Symfony\Component\Console\ **Application**
Symfony\Component\DependencyInjection\ **ContainerBuilder**
Symfony\Bundle\FrameworkBundle\DependencyInjection\ **FrameworkExtension**
Symfony\Component\OptionsResolver\ **OptionsResolver**
Symfony\Component\Form\ **FormConfigBuilder**
Symfony\Component\Intl\NumberFormatter\ **NumberFormatter**
Symfony\Component\Validator\Validator\ **RecursiveContextualValidator**
Symfony\Component\Form\ **ButtonBuilder**
Symfony\Component\PropertyAccess\ **PropertyAccessor**
Symfony\Component\HttpFoundation\File\MimeType\ **MimeTypeExtensionGuesser**
Symfony\Component\Yaml\ **Parser**
Symfony\Component\DependencyInjection\ **Definition**
Symfony\Component\HttpKernel\ **Kernel**
Symfony\Component\Finder\ **Finder**
Symfony\Component\Console\Helper\ **Table**
Symfony\Component\HttpKernel\HttpCache\ **HttpCache**
Symfony\Bundle\SecurityBundle\DependencyInjection\ **SecurityExtension**
Symfony\Component\HttpFoundation\Session\Storage\Handler\ **PdoSessionHandler**
Symfony\Component\Yaml\ **Inline**

Metrics

What to find?

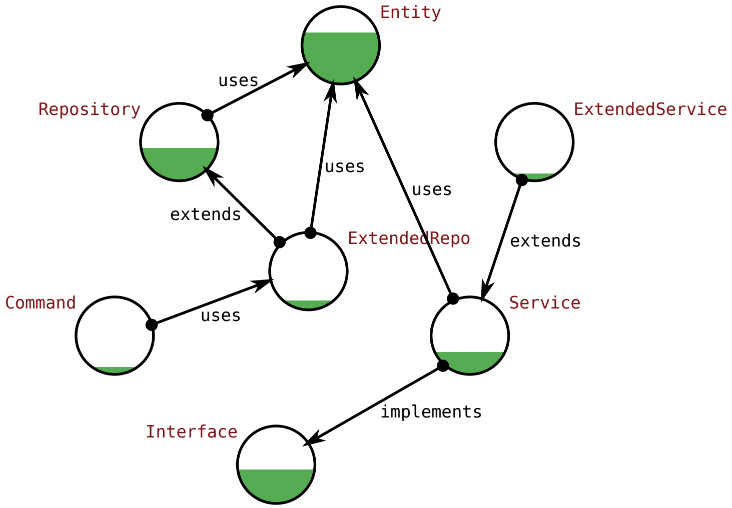
What do I want to find?

- Important Code
- Potentially buggy code
- Badly tested code
- Design violations

Important, buggy & untested code!

Finding The Core

Code Rank



Entity
uses
Repository
uses
uses
ExtendedService
extends
ExtendedRepo
extends
Command
uses
Service
implements
Interface

Code Rank

- Googles PageRank^{TM}for classes!
- Maps software to a graph
  - A node (π) for each software artifact
    - Package or Class
  - An edge (ρ) for each relation
    - Inheritance, Call, Parameter, Exceptions, Construction
- CodeRank:

$$CR(\pi_i) = \sum_r r((1 - d) + d \sum_r r(CR(\pi_r)/\rho_r))$$

Code Rank

Demo time

`http://stuff.qafoo.com/symfony`

passion for software quality

Reverse Code Rank

Shows fragile code

(Just reverse all edges)

Reverse Code Rank

Demo time

`http://stuff.qafoo.com/symfony`

Qafoo Quality Analyzer

Qafoo Quality Analyzer

- “Just” visualizes metrics
- Get it: `https://github.com/Qafoo/QualityAnalyzer`

```
$ ./phpunit --log-junit junit.xml --coverage
     -clover clover.xml
$ analyze [--coverage=clover.xml --tests=
     junit.xml] --exclude=Tests analyze src/
Analyze source code in /path/to/symfony/src/
* Running source
* Running coverage
* Running pdepend
* Running dependencies
* Running phpmd
* Running checkstyle
* Running tests
* Running cpd
* Running phploc
Done
$ analyze serve
Starting webserver on http://localhost:8080/
$ analyze bundle symfony
$ scp -r symfony/ qafoo-web:stuff/htdocs/
```

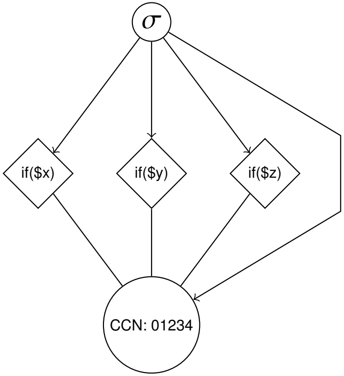
Where Will Be The Bugs?

Complexity metrics

- Bugs are often introduced where code is hard to understand
  - Control structures introduce complexity
    - `if`, `elseif`, `for`, `while`, `foreach`, `catch`, `case`, `xor`, `and`, `or`, &&, ||, `?:`
- Cyclomatic Complexity (CCN)
  - Number of *branches*
- NPath Complexity
  - Number of *execution paths*
  - Minds the structure of blocks

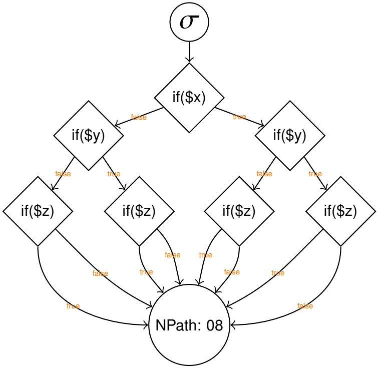
Cyclomatic Complexity

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



NPath Complexity

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



Sensible limits

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

Complexity

Demo time

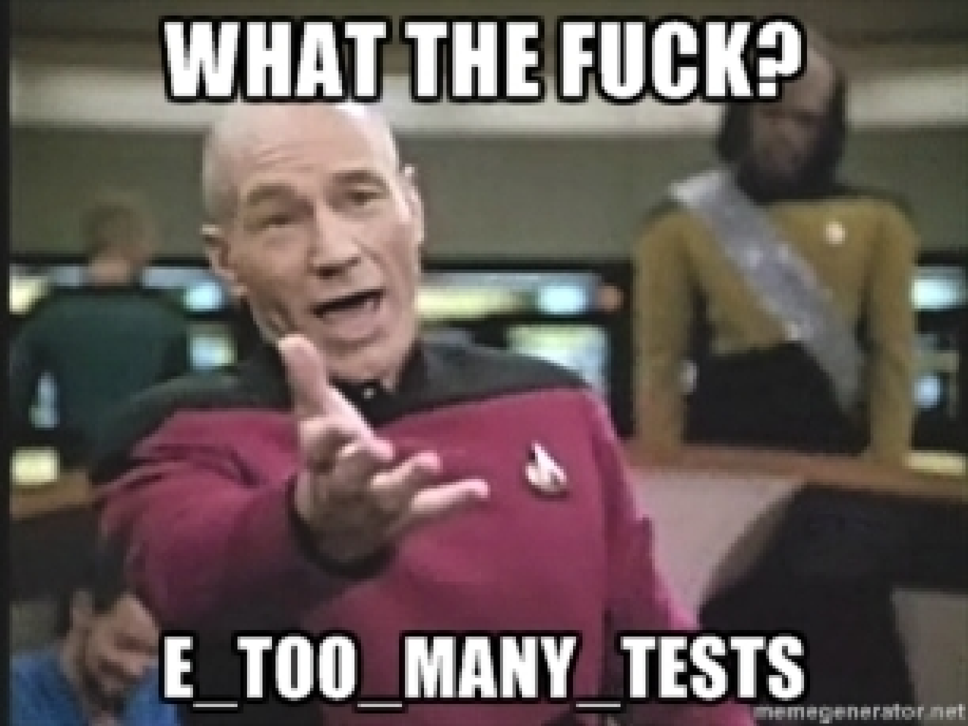
`http://stuff.qafoo.com/symfony`

What Should Be Tested?

How many tests do I need?

- 100% Line Converage?
  - Shows which lines have *not* been executed (by tests)
- Path Converage (been worked on)
  - Shows which execution paths have been covered
  - Write $nPath tests for every method?
- Parameter Value Coverage
  - Test all execution paths with sane boundary values for every parameter
  - Common integer boundaries: $-2^{63}, -2^{31}, -1, 0, 1, 2^{31}, 2^{63}$
- Write at least $\$nPath * \$parameterCount * \$boundaries$ tests per method!

passion for software quality


WHAT THE FUCK?
E_TOO_MANY_TESTS
memegenerator.net

Better combine metrics: CRAP

Is your code CRAP?

$$CRAP(m) = \begin{cases} ccn(m)^2 + ccn(m), & \text{if } cov(m) = 0 \\ ccn(m), & \text{if } cov(m) \geq .95 \\ ccn(m)^2 * (1 - cov(m))^3 + ccn(m), & \text{else} \end{cases}$$

- Change Risk Anti Patterns
  - $ccn(m)$ – Cyclomatic complexity of a method
  - $cov(m)$ – Line coverage of a method

Qafoo
passion for software quality

Tests

Demo time

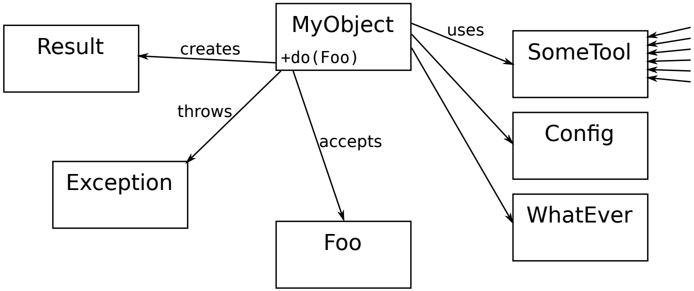
`http://stuff.qafoo.com/symfony`

What is Coupled?

Composition

Are there any misbheaving entities?

Object Oriented Systems



Artifact

- Package (Namespace)
- *Class*
- Method

Coupling

- Excessive coupling is one of the key problems
  - Dependencies between artifacts are established by:
    - Object instantiations
    - Static method calls
    - Method parameters
    - Thrown and catched exceptions
- (High) Efferent Coupling C_E (outgoing dependencies)
  - Artifact relies on a lot of code
  - Artifact tends to be unstable
  - Also called “Coupling Between Objects” (CBO)
- (High) Afferent Coupling C_A (incoming dependencies)
  - A lot of code relies on artifact
  - Artifact should be really stable

Code Rank

Code Rank

- Direct and indirect C_A (incoming dependencies)

Reverse Code Rank

- Direct and indirect C_E (outgoing dependencies)

Coupling

Demo time

`http://stuff.qafoo.com/symfony`

Final Notes

There are valid reasons behind every line of code

- You might not know or understand the reasons
- Code should be easy to understand – but not every line you do not understand is bad
- Be empathic
- Be gentle

Summary

- The Bad
  - It is not hard to trick metrics
  - It is easy to get dogmatic about metrics
- The Good
  - Metrics allow us to locate problematic code
  - Metrics allow for objective discussions about code – intepretetions are still subjective.
  - Finding this code is the base for refactorings, discussions & even rewrites

Qafoo

passion for software quality

THANK YOU

Rent a quality expert

qafoo.com