Static code analysis

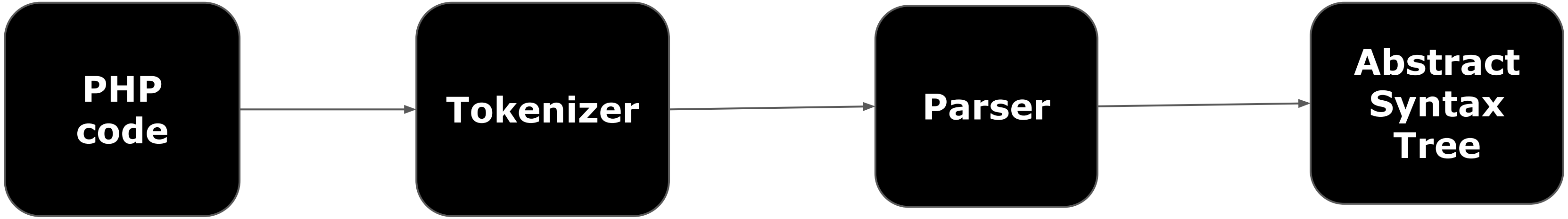
Piotr Osmałek

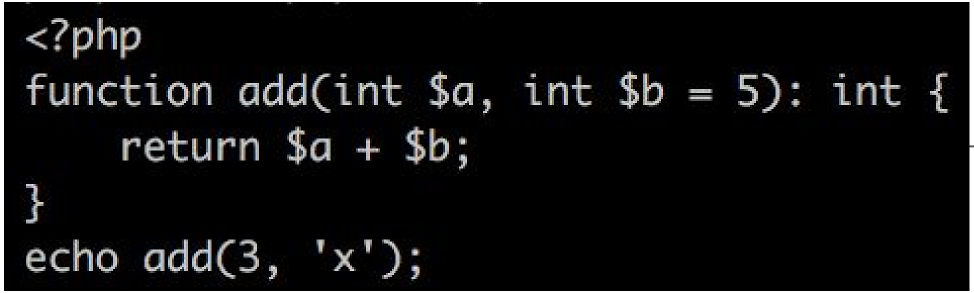
@posmalek

Static program analysis is the analysis of computer software that is performed without actually executing programs.

Wikipedia

How does it work?





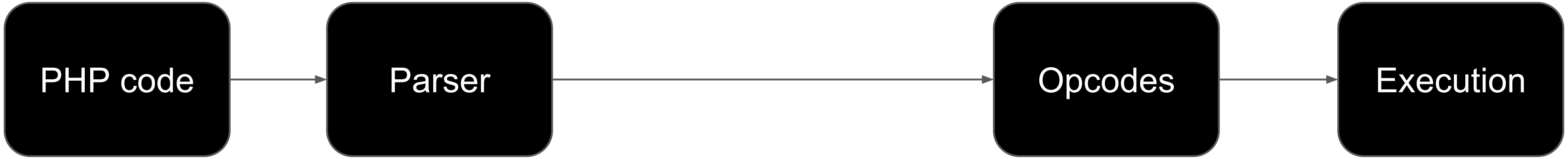
```php
<?php
function add(int $a, int $b = 5): int {
    return $a + $b;
}
echo add(3, 'x');
```

```
array(
    0: Stmt_Function(
        byRef: false
        name: add
        params: array(
            0: Param(
                type: int
                byRef: false
                variadic: false
                name: a
                default: null
            )
            1: Param(
                type: int
                byRef: false
                variadic: false
                name: b
                default: Scalar_LNumber(
                    value: 5
                )
            )
        )
        returnType: int
        stmts: array(
            0: Stmt_Return(
                expr: Expr_BinaryOp_Plus(
                    left: Expr_Variable(
                        name: a
                    )
                    right: Expr_Variable(
                        name: b
                    )
                )
            )
        )
    )
```

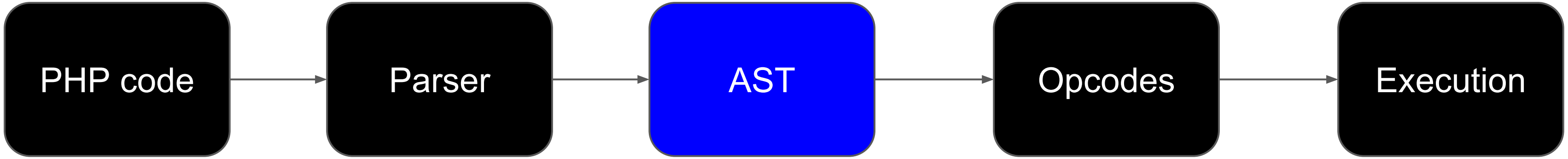
```
1: Stmt_Echo(
    exprs: array(
        0: Expr_FuncCall(
            name: Name(
                parts: array(
                    0: add
                )
            )
            args: array(
                0: Arg(
                    value: Scalar_LNumber(
                        value: 3
                    )
                    byRef: false
                    unpack: false
                )
                1: Arg(
                    value: Scalar_String(
                        value: x
                    )
                    byRef: false
                    unpack: false
                )
            )
        )
    )
)
```

PHP7 internal AST

PHP 5



PHP 7



PHP-AST extension

Better performance than PHP-Parser, but…

- handles only AST construction
- one class for everything
- parse code that is syntactically valid on the version of PHP it runs on

Why static code analysis should be used?

Code quality!

What is it not?

It is not substitute for unit tests!

It is not source of unquestionable truth.

PHP landscape

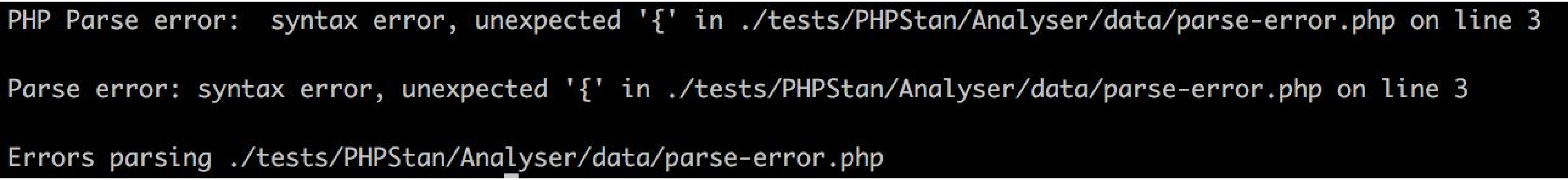
New tools have emerged lately (and more hopefully to come), due to recent advances in PHP.

But there were also some great tools before PHP7 and they are still very useful.

PHP Lint

PHP built-in syntax check

$ php -l <path-to-file>



```
PHP Parse error:  syntax error, unexpected '{' in ./tests/PHPStan/Analyser/data/parse-error.php on line 3

Parse error: syntax error, unexpected '{' in ./tests/PHPStan/Analyser/data/parse-error.php on line 3

Errors parsing ./tests/PHPStan/Analyser/data/parse-error.php
```

PHP Parallel Lint

Wrapper for PHP Lint, it checks files in parallel

```
$ .parallel-lint <path-to-directory>
```

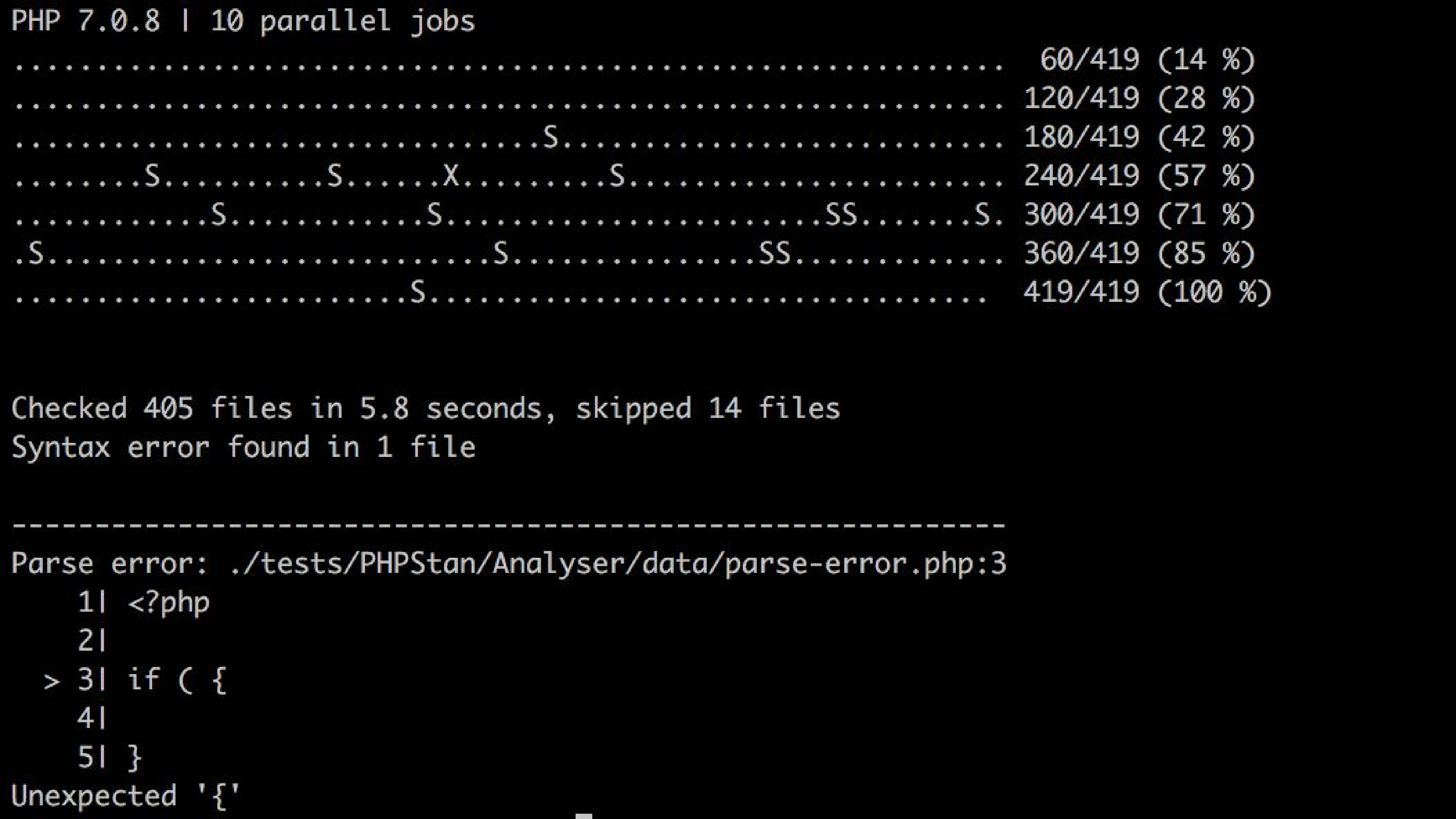
Some flags:

```
-j <num>
```

```
--blame
```

```
--exclude <path-to-directory>
```

https://github.com/JakubOnderka/PHP-Parallel-Lint



```
PHP 7.0.8 | 10 parallel jobs
............................................................  60/419 (14 %)
............................................................ 120/419 (28 %)
................................S........................... 180/419 (42 %)
........S..........S......X.........S....................... 240/419 (57 %)
............S............S.......................SS.......S. 300/419 (71 %)
.S...........................S...............SS............. 360/419 (85 %)
........................S..................................  419/419 (100 %)


Checked 405 files in 5.8 seconds, skipped 14 files
Syntax error found in 1 file


------------------------------------------------------------
Parse error: ./tests/PHPStan/Analyser/data/parse-error.php:3
    1| <?php
    2|
  > 3| if ( {
    4|
    5| }
Unexpected '{'
```

PHP Code Sniffer & PHP CS Fixer

Tools for detecting violations of coding standards

Highly configurable

Both can be easily integrated with PHP Storm

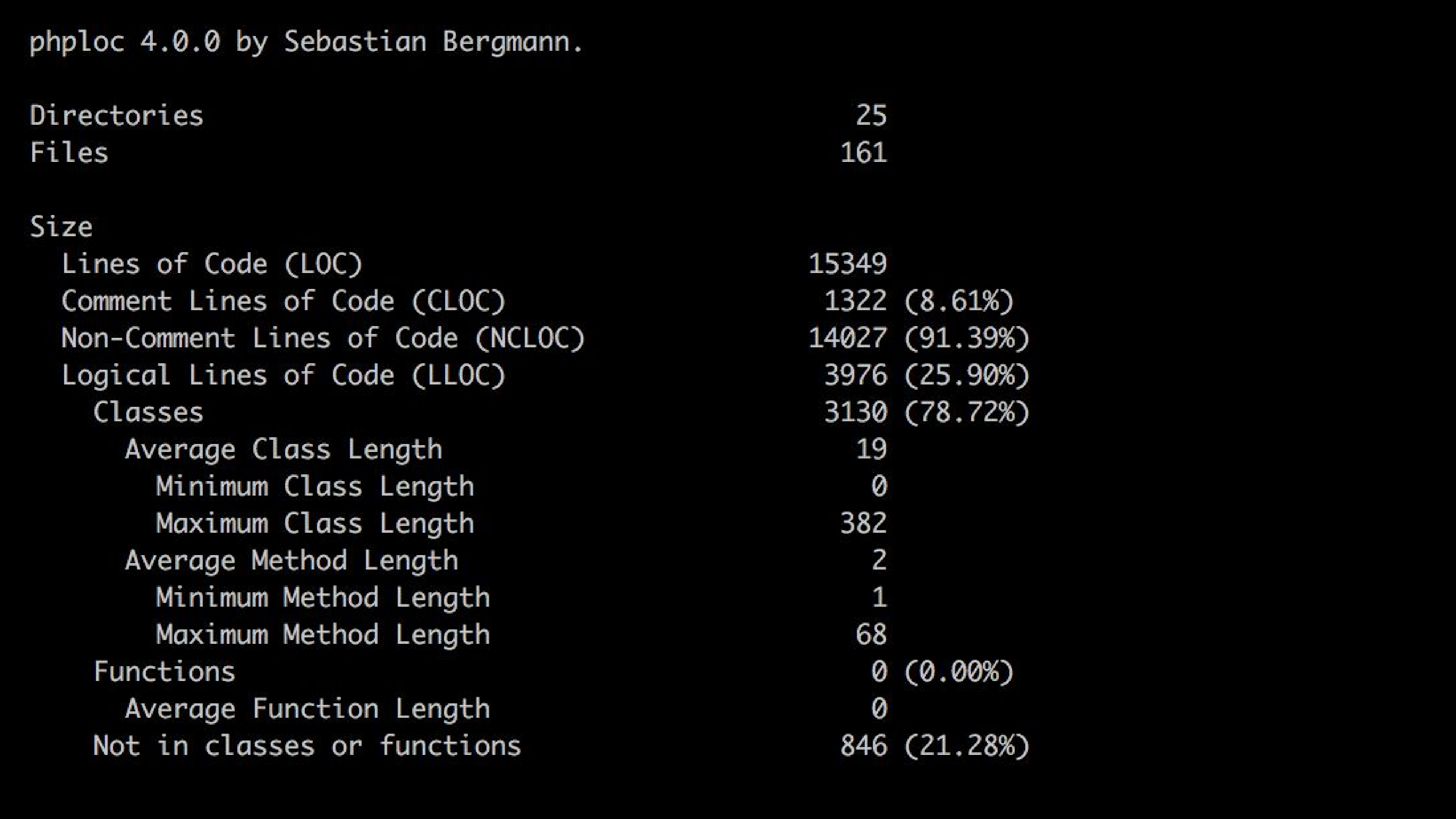
https://github.com/squizlabs/PHP_CodeSniffer
https://github.com/FriendsOfPHP/PHP-CS-Fixer

PHPLOC

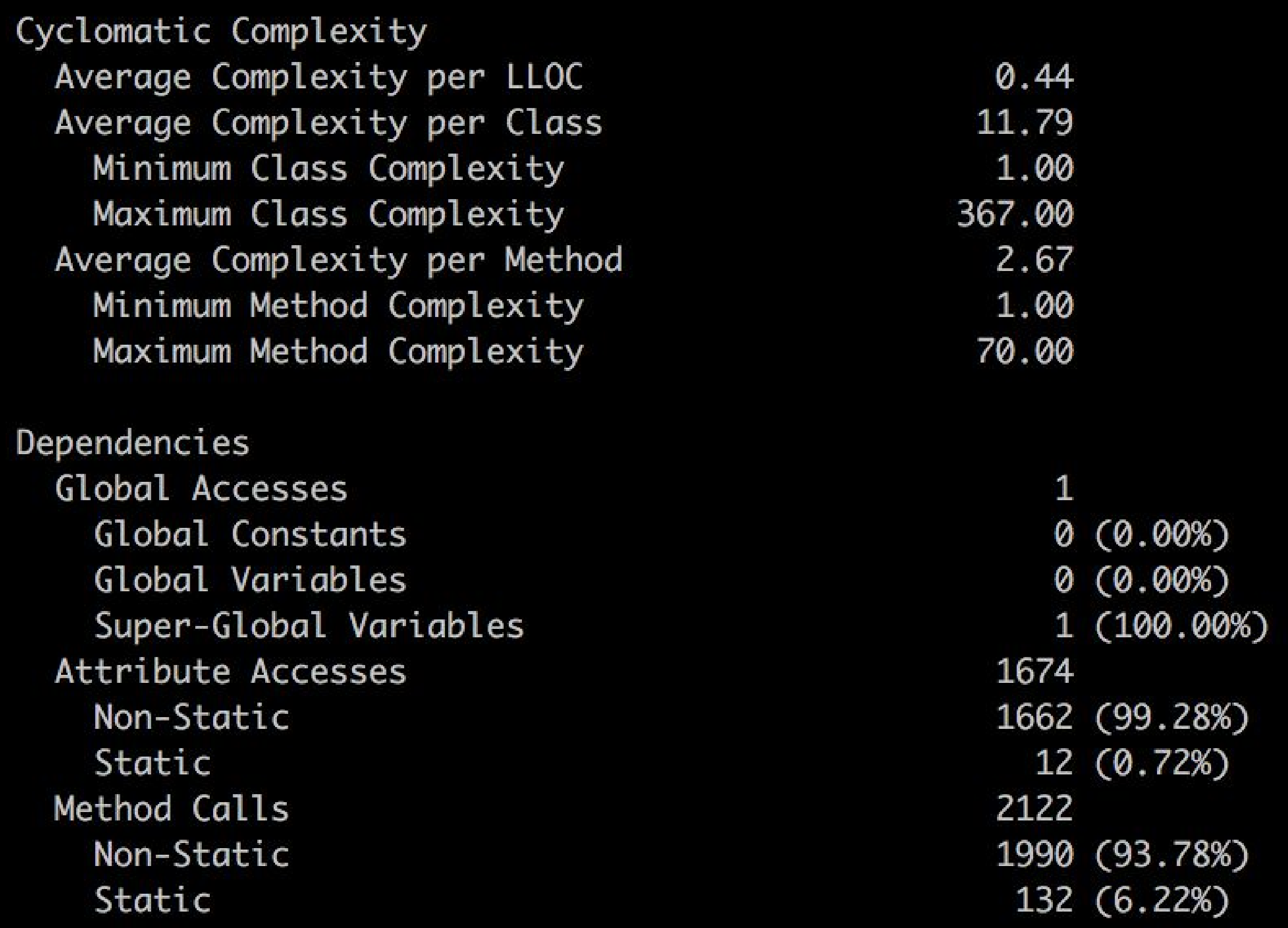
Provide basic metrics

```
$ phploc <path>
```

https://github.com/sebastianbergmann/phploc



```
phploc 4.0.0 by Sebastian Bergmann.

Directories                                         25
Files                                              161

Size
  Lines of Code (LOC)                            15349
  Comment Lines of Code (CLOC)                    1322 (8.61%)
  Non-Comment Lines of Code (NCLOC)              14027 (91.39%)
  Logical Lines of Code (LLOC)                    3976 (25.90%)
    Classes                                       3130 (78.72%)
      Average Class Length                          19
        Minimum Class Length                         0
        Maximum Class Length                       382
      Average Method Length                          2
        Minimum Method Length                        1
        Maximum Method Length                       68
    Functions                                        0 (0.00%)
      Average Function Length                        0
    Not in classes or functions                    846 (21.28%)
```



```
Cyclomatic Complexity
  Average Complexity per LLOC                     0.44
  Average Complexity per Class                   11.79
    Minimum Class Complexity                      1.00
    Maximum Class Complexity                    367.00
  Average Complexity per Method                   2.67
    Minimum Method Complexity                     1.00
    Maximum Method Complexity                    70.00

Dependencies
  Global Accesses                                    1
    Global Constants                                 0 (0.00%)
    Global Variables                                 0 (0.00%)
    Super-Global Variables                           1 (100.00%)
  Attribute Accesses                              1674
    Non-Static                                    1662 (99.28%)
    Static                                          12 (0.72%)
  Method Calls                                    2122
    Non-Static                                    1990 (93.78%)
    Static                                         132 (6.22%)
```

What is Cyclomatic Complexity?

One of the oldest complexity metrics

Complexity is determined by the number of decision points in a method plus one for the method entry.

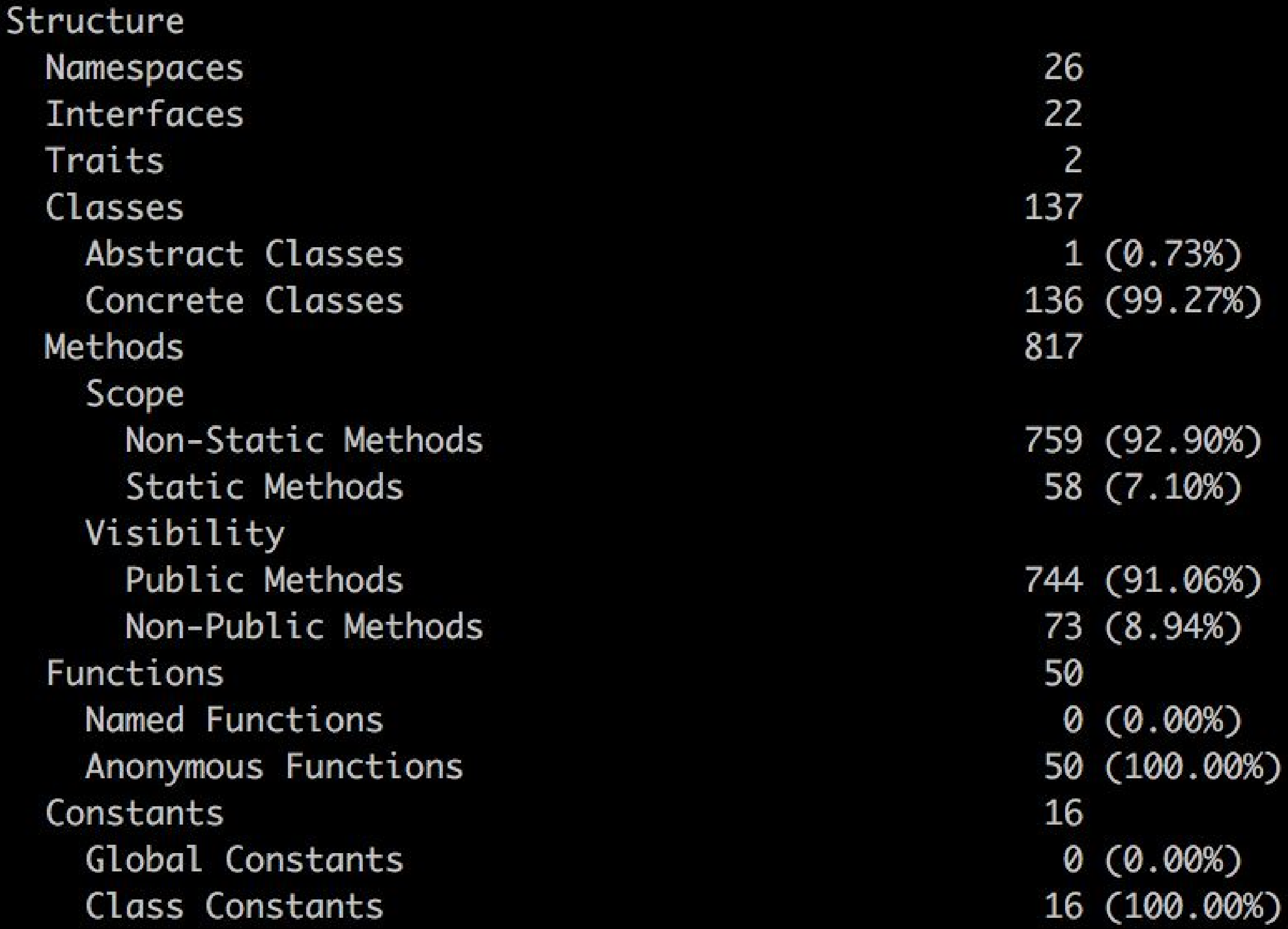
1-4: low complexity

5-7: moderate complexity

8-10: high complexity

11+ very high complexity

```
class Foo {
1   public function example() {
2       if ($a == $b) {
3           if ($a1 == $b1) {
                fiddle();
4           } elseif ($a2 == $b2) {
                fiddle();
            } else {
                fiddle();
            }
5       } elseif ($c == $d) {
6           while ($c == $d) {
                fiddle();
            }
7        } elseif ($e == $f) {
8           for ($n = 0; $n < $h; $n++) {
                fiddle();
            }
        } else {
            switch ($z) {
9               case 1:
                    fiddle();
                    break;
10              case 2:
                    fiddle();
                    break;
11              case 3:
                    fiddle();
                    break;
                default:
                    fiddle();
                    break;
```



```
Structure
  Namespaces                                        26
  Interfaces                                        22
  Traits                                             2
  Classes                                          137
    Abstract Classes                                 1 (0.73%)
    Concrete Classes                               136 (99.27%)
  Methods                                          817
    Scope
      Non-Static Methods                           759 (92.90%)
      Static Methods                                58 (7.10%)
    Visibility
      Public Methods                               744 (91.06%)
      Non-Public Methods                            73 (8.94%)
  Functions                                         50
    Named Functions                                  0 (0.00%)
    Anonymous Functions                             50 (100.00%)
  Constants                                         16
    Global Constants                                 0 (0.00%)
    Class Constants                                 16 (100.00%)
```

PHPMetrics

Provides metrics with readable HTML report

```
$ php ./vendor/bin/phpmetrics --report-html=<reportName> <directory>
```

Generating console report:

```
$ php ./vendor/bin/phpmetrics <directory>
```

https://github.com/phpmetrics/PhpMetrics

Object oriented metrics

Efferent coupling (CE)

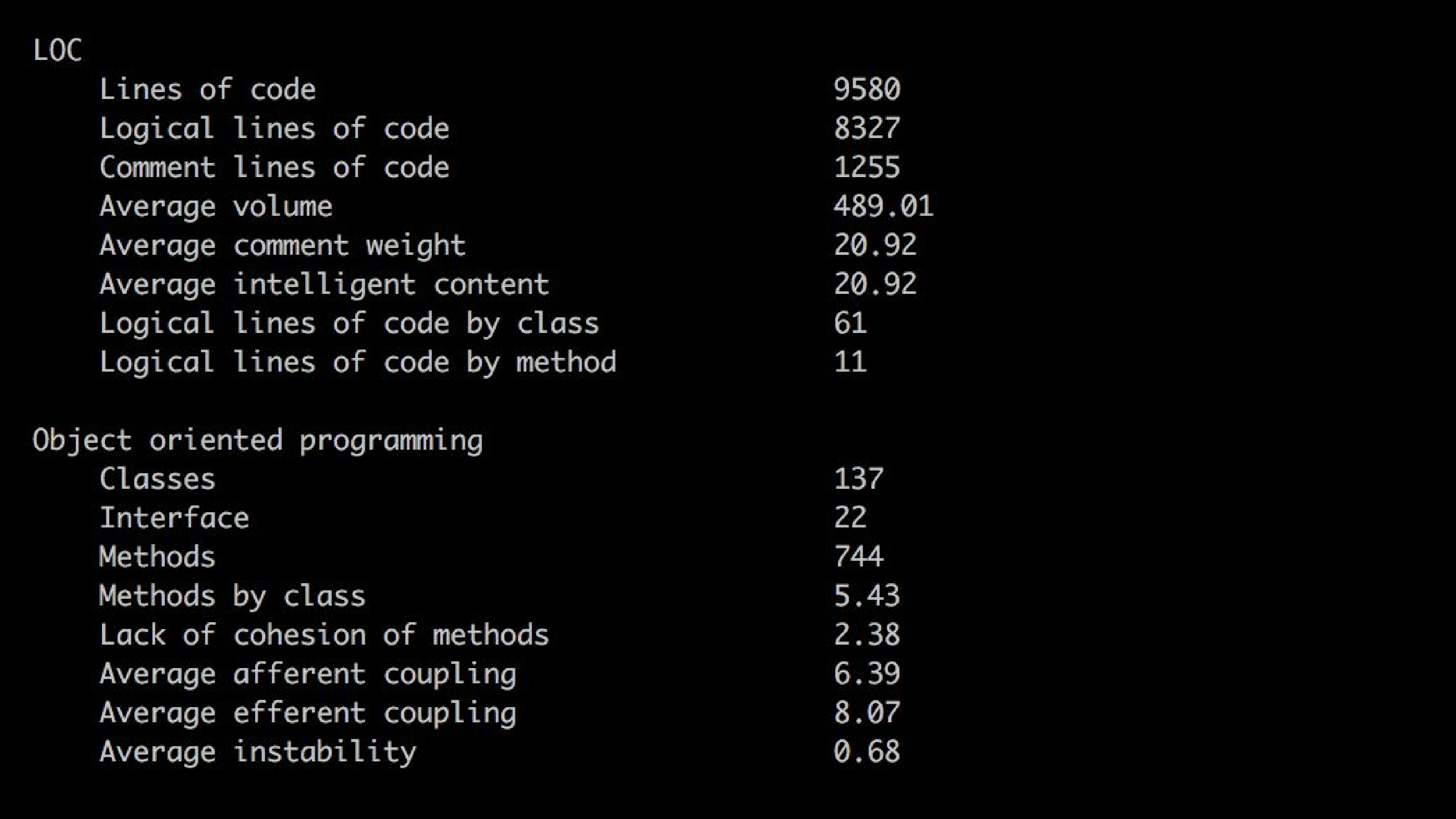
Afferent coupling (CA)

Instability = CE / (CE + CA)

Stable: 0,0 - 0,3

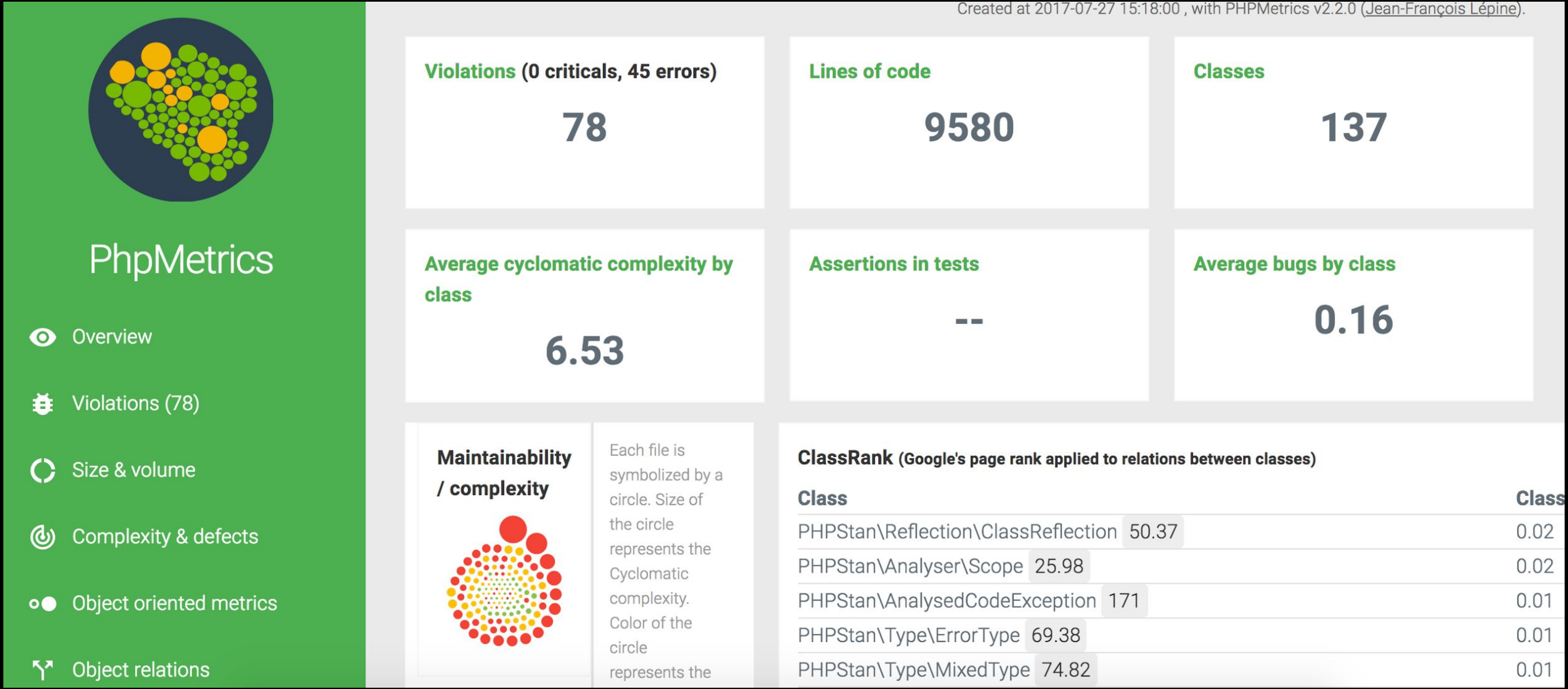
Unstable: 0,7 - 1,0

Abstractness



```
LOC
    Lines of code                               9580
    Logical lines of code                       8327
    Comment lines of code                       1255
    Average volume                              489.01
    Average comment weight                      20.92
    Average intelligent content                 20.92
    Logical lines of code by class              61
    Logical lines of code by method             11

Object oriented programming
    Classes                                     137
    Interface                                   22
    Methods                                     744
    Methods by class                            5.43
    Lack of cohesion of methods                 2.38
    Average afferent coupling                   6.39
    Average efferent coupling                   8.07
    Average instability                         0.68
```

```
Complexity
    Average Cyclomatic complexity by class      6.53
    Average Relative system complexity          135.37
    Average Difficulty                          6.99

Bugs
    Average bugs by class                       0.16
    Average defects by class (Kan)              0.79

Violations
    Critical                                    0
    Error                                       45
    Warning                                     17
    Information                                 16
```



Created at 2017-07-27 15:18:00 , with PHPMetrics v2.2.0 (Jean-François Lépine).

Violations (0 criticals, 45 errors)

78

Lines of code

9580

Classes

137

Average cyclomatic complexity by class

6.53

Assertions in tests

--

Average bugs by class

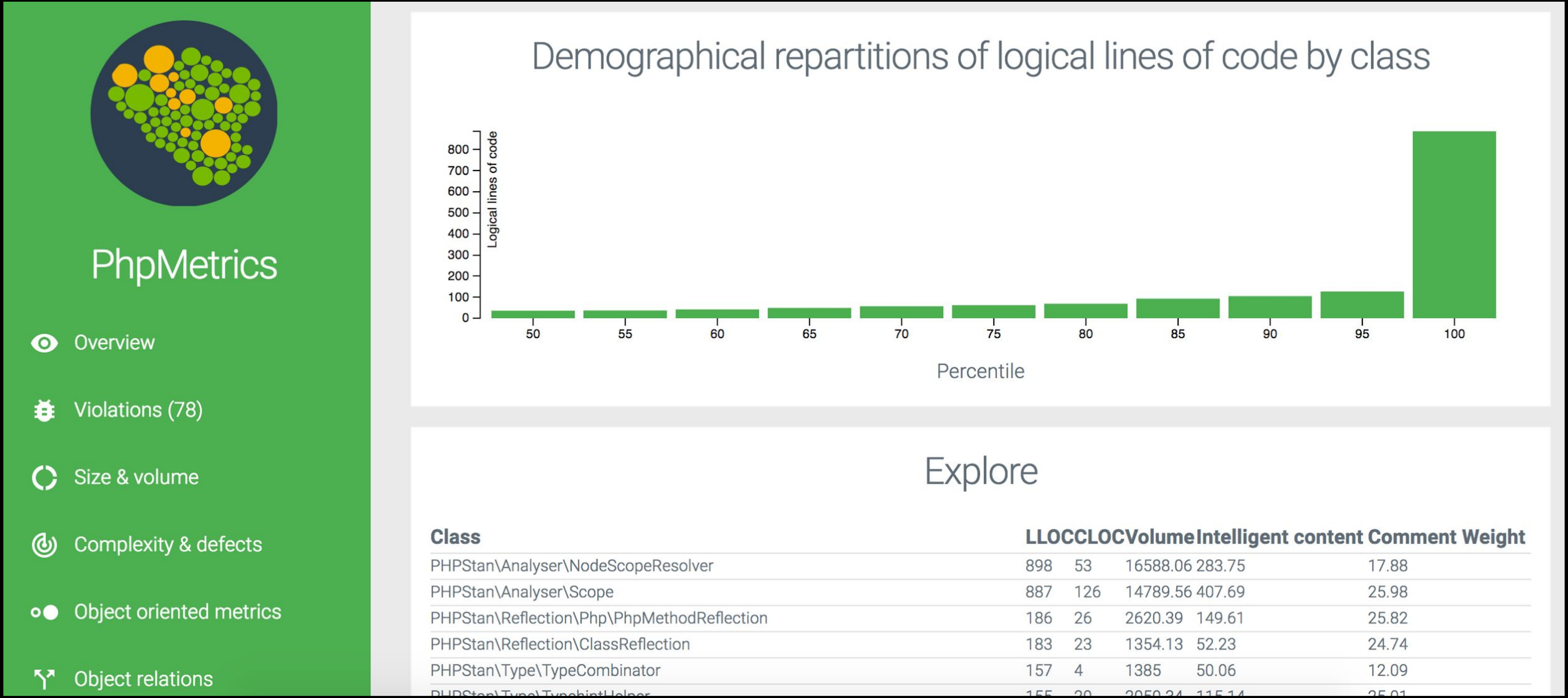
0.16

Maintainability / complexity

Each file is symbolized by a circle. Size of the circle represents the Cyclomatic complexity. Color of the circle represents the

ClassRank (Google's page rank applied to relations between classes)

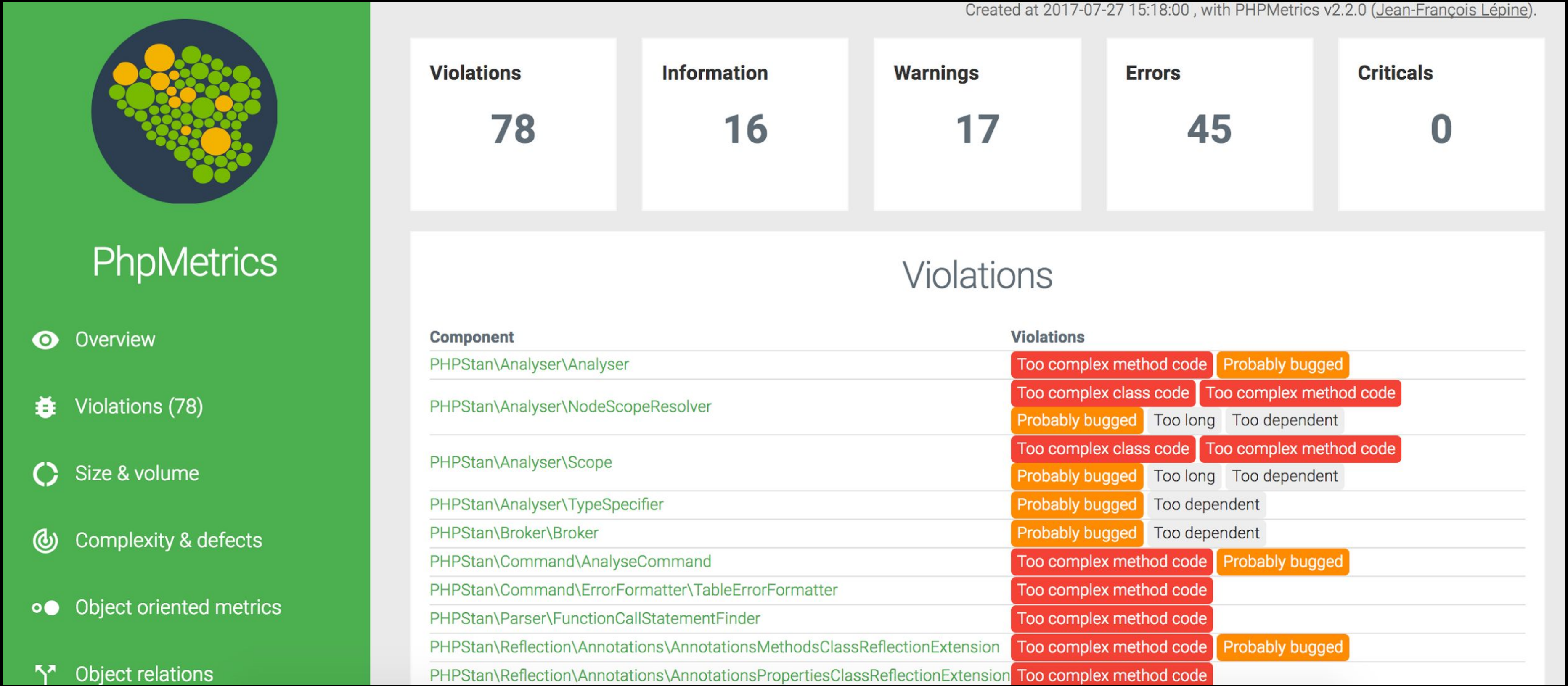
Class		Class
PHPStan\Reflection\ClassReflection	50.37	0.02
PHPStan\Analyser\Scope	25.98	0.02
PHPStan\AnalysedCodeException	171	0.01
PHPStan\Type\ErrorType	69.38	0.01
PHPStan\Type\MixedType	74.82	0.01



Demographical repartitions of logical lines of code by class

Explore

Class	LLOC	CLOC	Volume	Intelligent content	Comment Weight
PHPStan\Analyser\NodeScopeResolver	898	53	16588.06	283.75	17.88
PHPStan\Analyser\Scope	887	126	14789.56	407.69	25.98
PHPStan\Reflection\Php\PhpMethodReflection	186	26	2620.39	149.61	25.82
PHPStan\Reflection\ClassReflection	183	23	1354.13	52.23	24.74
PHPStan\Type\TypeCombinator	157	4	1385	50.06	12.09
PHPStan\Type\TypehintHelper	155	20	2050.24	115.14	25.01

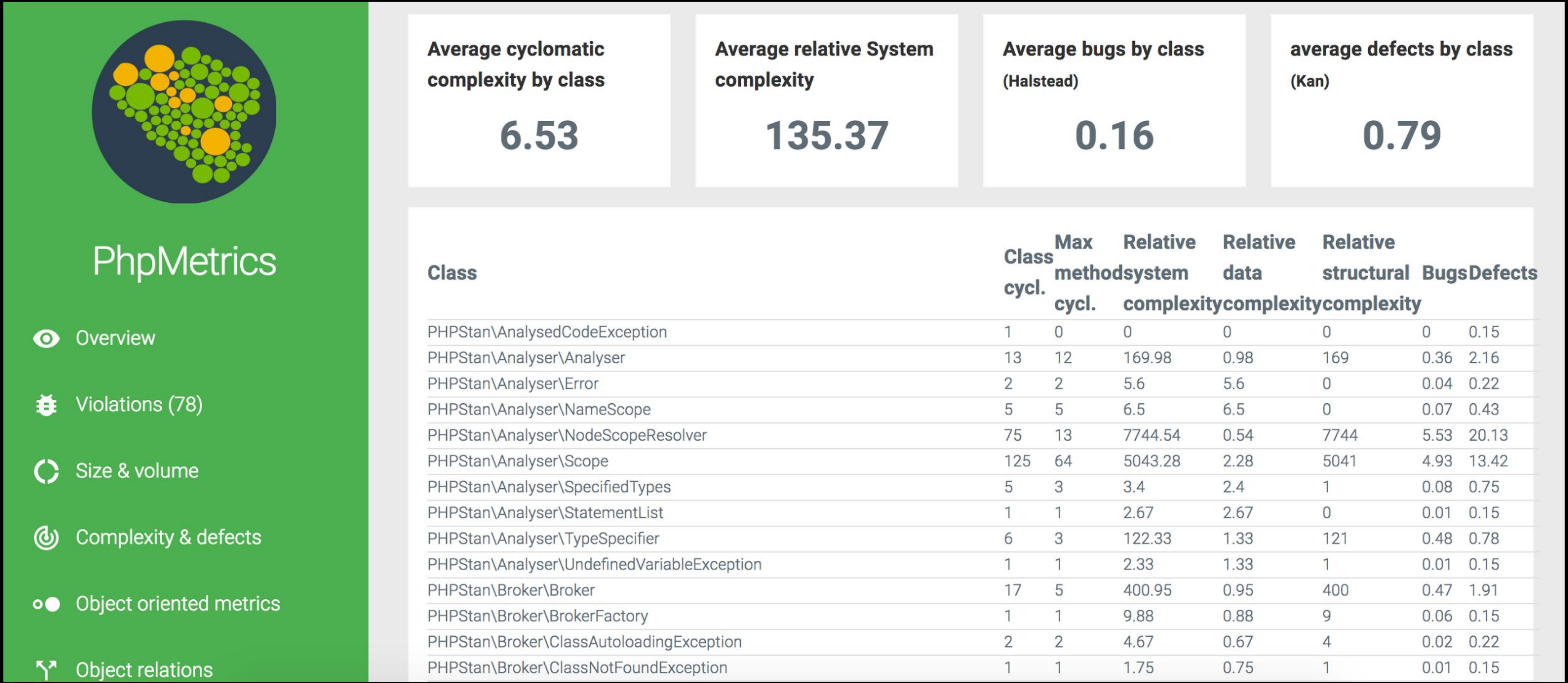


Created at 2017-07-27 15:18:00 , with PHPMetrics v2.2.0 (Jean-François Lépine).

Violations	Information	Warnings	Errors	Criticals
78	16	17	45	0

Violations

Component	Violations
PHPStan\Analyser\Analyser	Too complex method code Probably bugged
PHPStan\Analyser\NodeScopeResolver	Too complex class code Too complex method code Probably bugged Too long Too dependent
PHPStan\Analyser\Scope	Too complex class code Too complex method code Probably bugged Too long Too dependent
PHPStan\Analyser\TypeSpecifier	Probably bugged Too dependent
PHPStan\Broker\Broker	Probably bugged Too dependent
PHPStan\Command\AnalyseCommand	Too complex method code Probably bugged
PHPStan\Command\ErrorFormatter\TableErrorFormatter	Too complex method code
PHPStan\Parser\FunctionCallStatementFinder	Too complex method code
PHPStan\Reflection\Annotations\AnnotationsMethodsClassReflectionExtension	Too complex method code Probably bugged
PHPStan\Reflection\Annotations\AnnotationsPropertiesClassReflectionExtension	Too complex method code



PhpMetrics

- Overview
- Violations (78)
- Size & volume
- Complexity & defects
- Object oriented metrics
- Object relations

Average cyclomatic complexity by class

6.53

Average relative System complexity

135.37

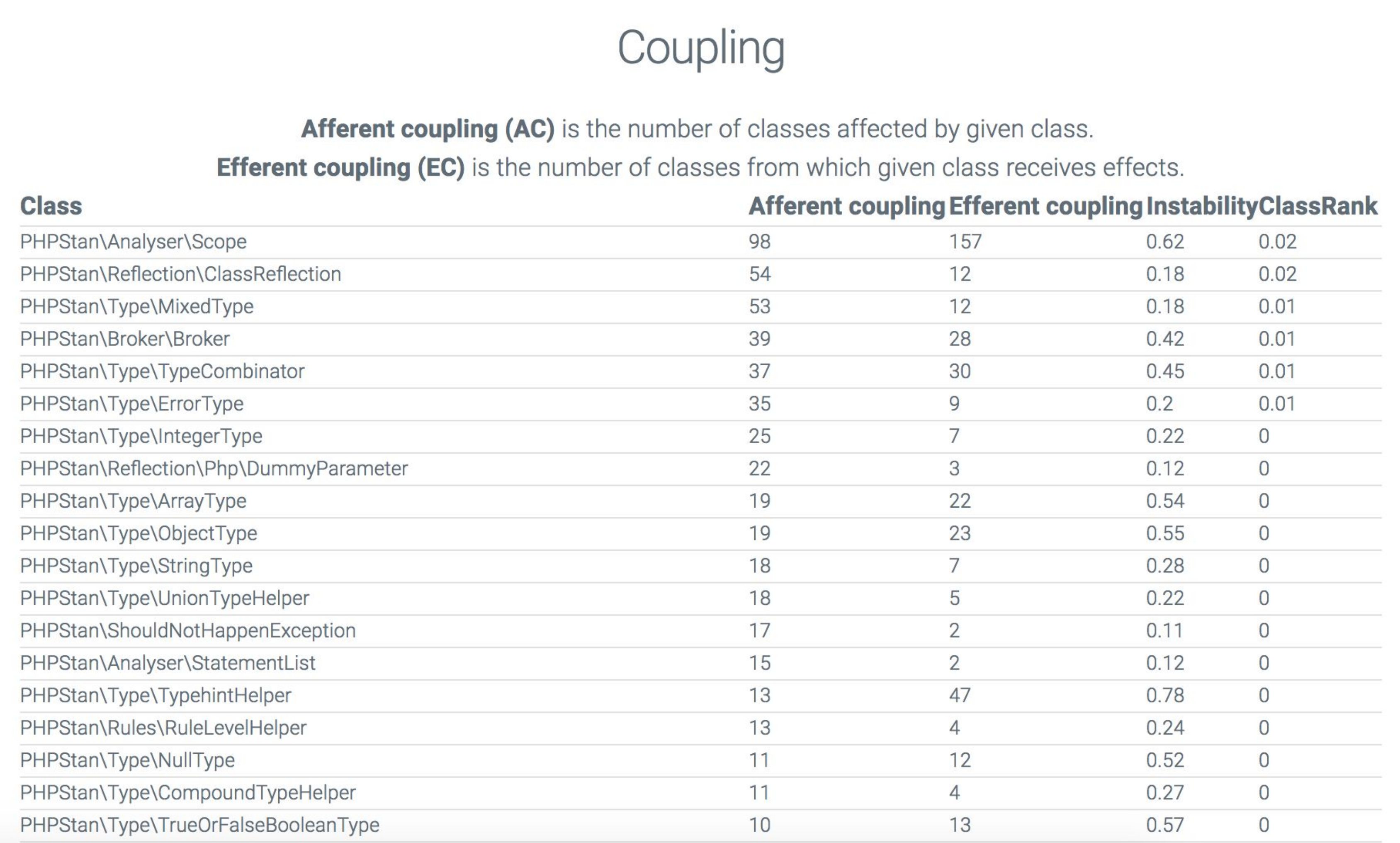
Average bugs by class (Halstead)

0.16

average defects by class (Kan)

0.79

Class	Class cycl.	Max method cycl.	Relative system complexity	Relative data complexity	Relative structural complexity	Bugs	Defects
PHPStan\AnalysedCodeException	1	0	0	0	0	0	0.15
PHPStan\Analyser\Analyser	13	12	169.98	0.98	169	0.36	2.16
PHPStan\Analyser\Error	2	2	5.6	5.6	0	0.04	0.22
PHPStan\Analyser\NameScope	5	5	6.5	6.5	0	0.07	0.43
PHPStan\Analyser\NodeScopeResolver	75	13	7744.54	0.54	7744	5.53	20.13
PHPStan\Analyser\Scope	125	64	5043.28	2.28	5041	4.93	13.42
PHPStan\Analyser\SpecifiedTypes	5	3	3.4	2.4	1	0.08	0.75
PHPStan\Analyser\StatementList	1	1	2.67	2.67	0	0.01	0.15
PHPStan\Analyser\TypeSpecifier	6	3	122.33	1.33	121	0.48	0.78
PHPStan\Analyser\UndefinedVariableException	1	1	2.33	1.33	1	0.01	0.15
PHPStan\Broker\Broker	17	5	400.95	0.95	400	0.47	1.91
PHPStan\Broker\BrokerFactory	1	1	9.88	0.88	9	0.06	0.15
PHPStan\Broker\ClassAutoloadingException	2	2	4.67	0.67	4	0.02	0.22
PHPStan\Broker\ClassNotFoundException	1	1	1.75	0.75	1	0.01	0.15



Coupling

Afferent coupling (AC) is the number of classes affected by given class.

Efferent coupling (EC) is the number of classes from which given class receives effects.

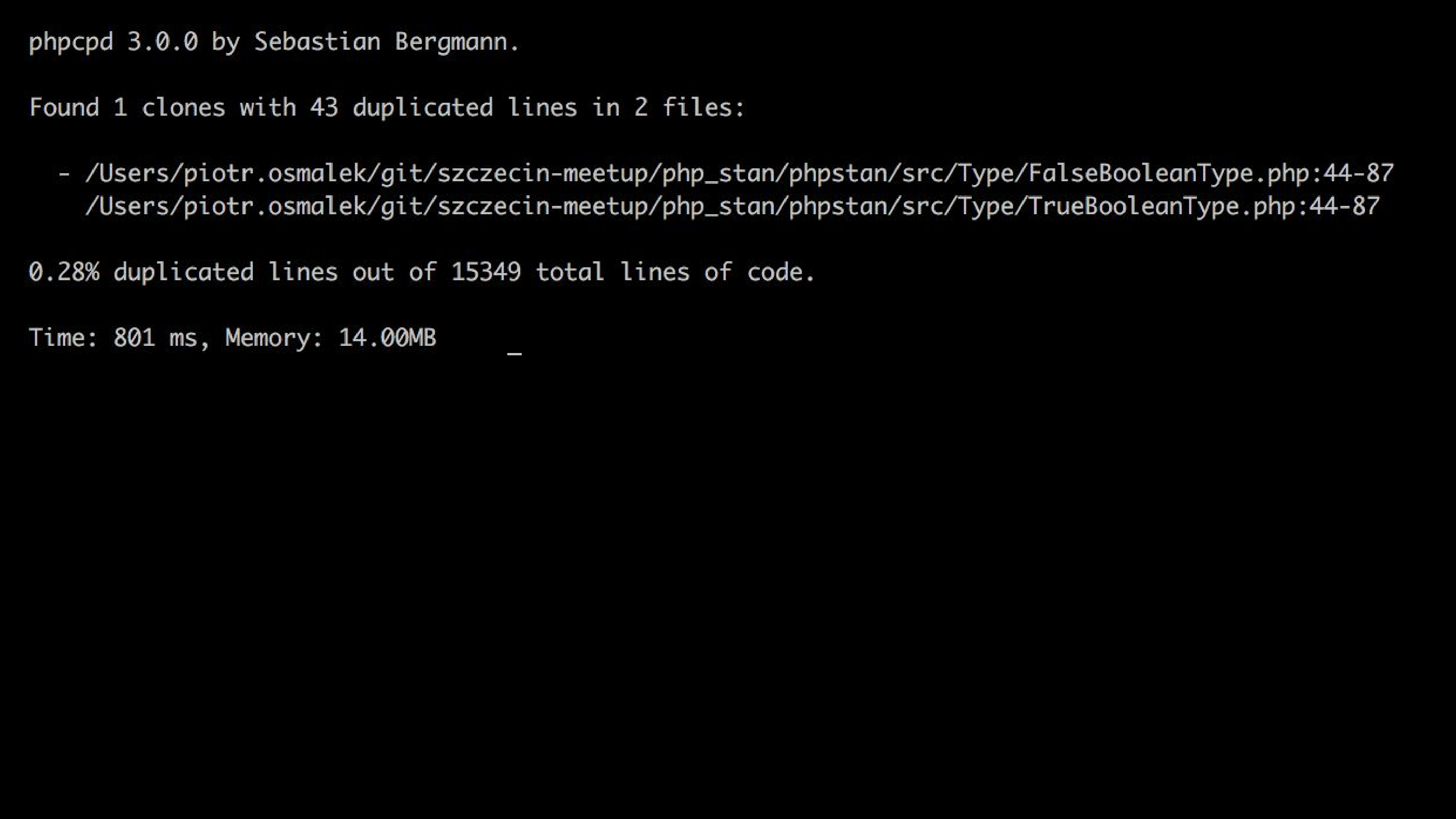
Class	Afferent coupling	Efferent coupling	Instability	ClassRank
PHPStan\Analyser\Scope	98	157	0.62	0.02
PHPStan\Reflection\ClassReflection	54	12	0.18	0.02
PHPStan\Type\MixedType	53	12	0.18	0.01
PHPStan\Broker\Broker	39	28	0.42	0.01
PHPStan\Type\TypeCombinator	37	30	0.45	0.01
PHPStan\Type\ErrorType	35	9	0.2	0.01
PHPStan\Type\IntegerType	25	7	0.22	0
PHPStan\Reflection\Php\DummyParameter	22	3	0.12	0
PHPStan\Type\ArrayType	19	22	0.54	0
PHPStan\Type\ObjectType	19	23	0.55	0
PHPStan\Type\StringType	18	7	0.28	0
PHPStan\Type\UnionTypeHelper	18	5	0.22	0
PHPStan\ShouldNotHappenException	17	2	0.11	0
PHPStan\Analyser\StatementList	15	2	0.12	0
PHPStan\Type\TypehintHelper	13	47	0.78	0
PHPStan\Rules\RuleLevelHelper	13	4	0.24	0
PHPStan\Type\NullType	11	12	0.52	0
PHPStan\Type\CompoundTypeHelper	11	4	0.27	0
PHPStan\Type\TrueOrFalseBooleanType	10	13	0.57	0

PHP Copy/Paste Detector

Detecting duplicated code

```
$ phpcpd <path>
```

https://github.com/sebastianbergmann/phpcpd



```
phpcpd 3.0.0 by Sebastian Bergmann.

Found 1 clones with 43 duplicated lines in 2 files:

  - /Users/piotr.osmalek/git/szczecin-meetup/php_stan/phpstan/src/Type/FalseBooleanType.php:44-87
    /Users/piotr.osmalek/git/szczecin-meetup/php_stan/phpstan/src/Type/TrueBooleanType.php:44-87

0.28% duplicated lines out of 15349 total lines of code.

Time: 801 ms, Memory: 14.00MB
```

PHPStan

Fresh library from 2016

Easy to configure

Eight levels of strictness

Extendable

```
$ ./phpstan analyse <paths>
```

PHPStan - what is checked?

Existence of classes and interfaces

Existence of variables

Existence and visibility of called methods and functions

Existence and visibility of accessed properties and constants

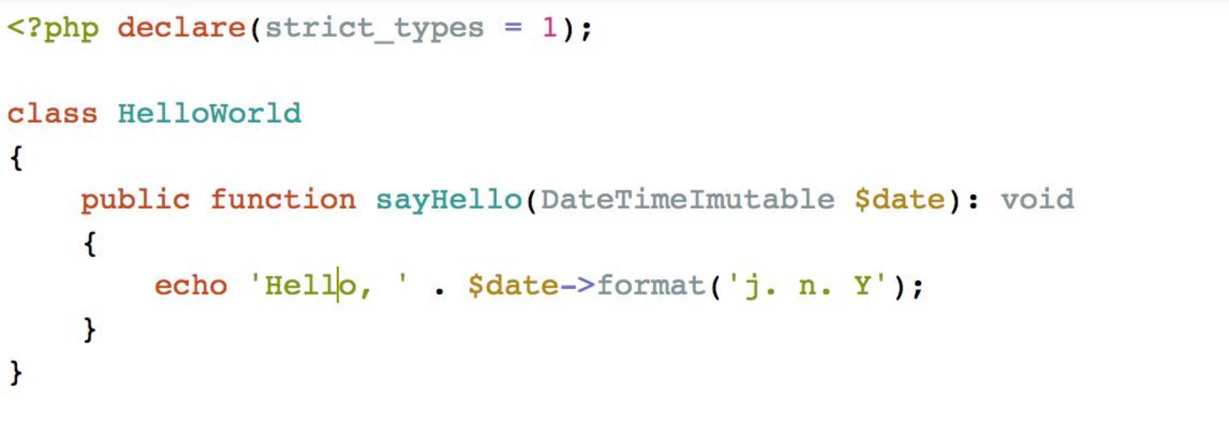
Correct types assigned to properties

Correct number and types of parameters passed

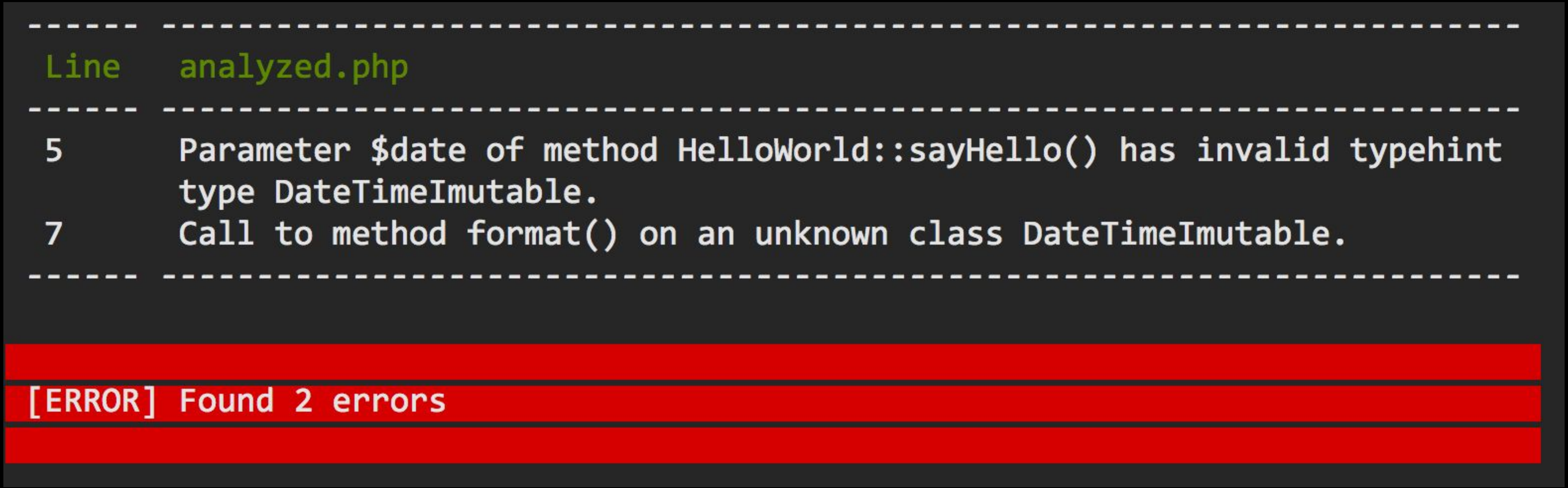
Correct types returned from methods and functions

Useless casts like (string) 'foo'

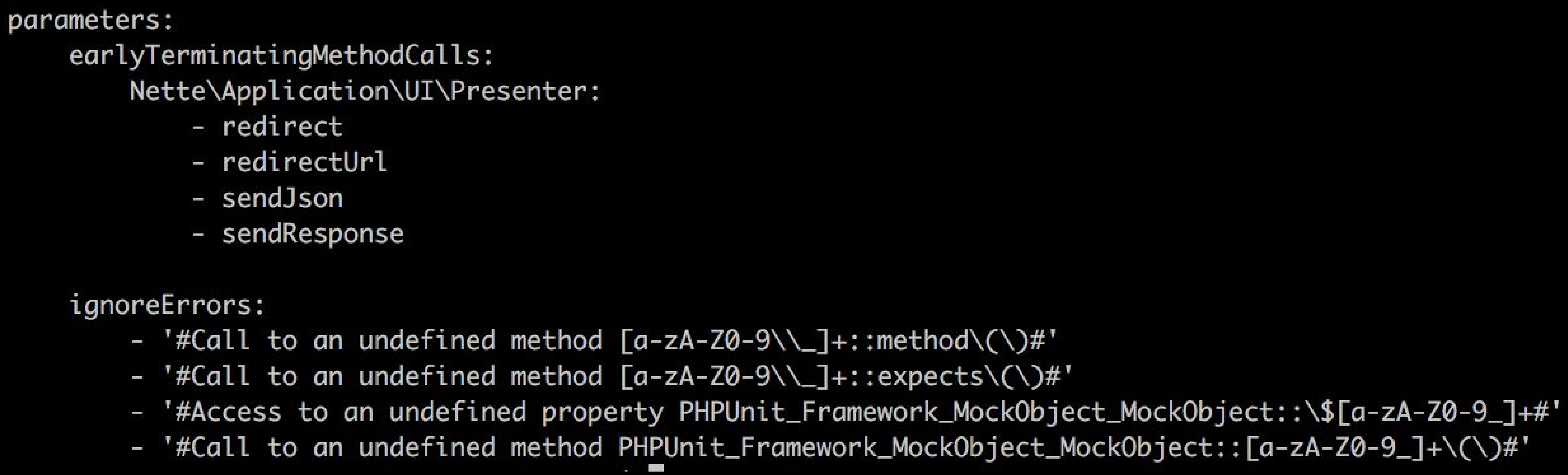
And some others...



```php
<?php declare(strict_types = 1);

class HelloWorld
{
    public function sayHello(DateTimeImutable $date): void
    {
        echo 'Hello, ' . $date->format('j. n. Y');
    }
}
```



```
 ------ ---------------------------------------------------------
  Line   analyzed.php
 ------ ---------------------------------------------------------
  5      Parameter $date of method HelloWorld::sayHello() has invalid typehint
         type DateTimeImutable.
  7      Call to method format() on an unknown class DateTimeImutable.
 ------ ---------------------------------------------------------


 [ERROR] Found 2 errors

```



```
parameters:
    earlyTerminatingMethodCalls:
        Nette\Application\UI\Presenter:
            - redirect
            - redirectUrl
            - sendJson
            - sendResponse

    ignoreErrors:
        - '#Call to an undefined method [a-zA-Z0-9\\_]+::method\(\)#'
        - '#Call to an undefined method [a-zA-Z0-9\\_]+::expects\(\)#'
        - '#Access to an undefined property PHPUnit_Framework_MockObject_MockObject::\$[a-zA-Z0-9_]+#'
        - '#Call to an undefined method PHPUnit_Framework_MockObject_MockObject::[a-zA-Z0-9_]+\(\)#'
```

Cons & possible problems

Forgotten tools

Dynamic nature of PHP

Warnings / errors overwhelming

Tools overwhelming

False positives

Too long running time

Constant refactoring

Final tips

Make static analysis part of your CI process

Limit number of tools

Limit number of false positives

Declare types

Be rational

Questions?