Prevent a mess with PHP Mess Detector

About me



- Tobias van Beek
- Opensource contributor and maintainer
- Including PHPMD
- https://tjvb.nl/about

(Almost) everybody likes to start a greenfield project



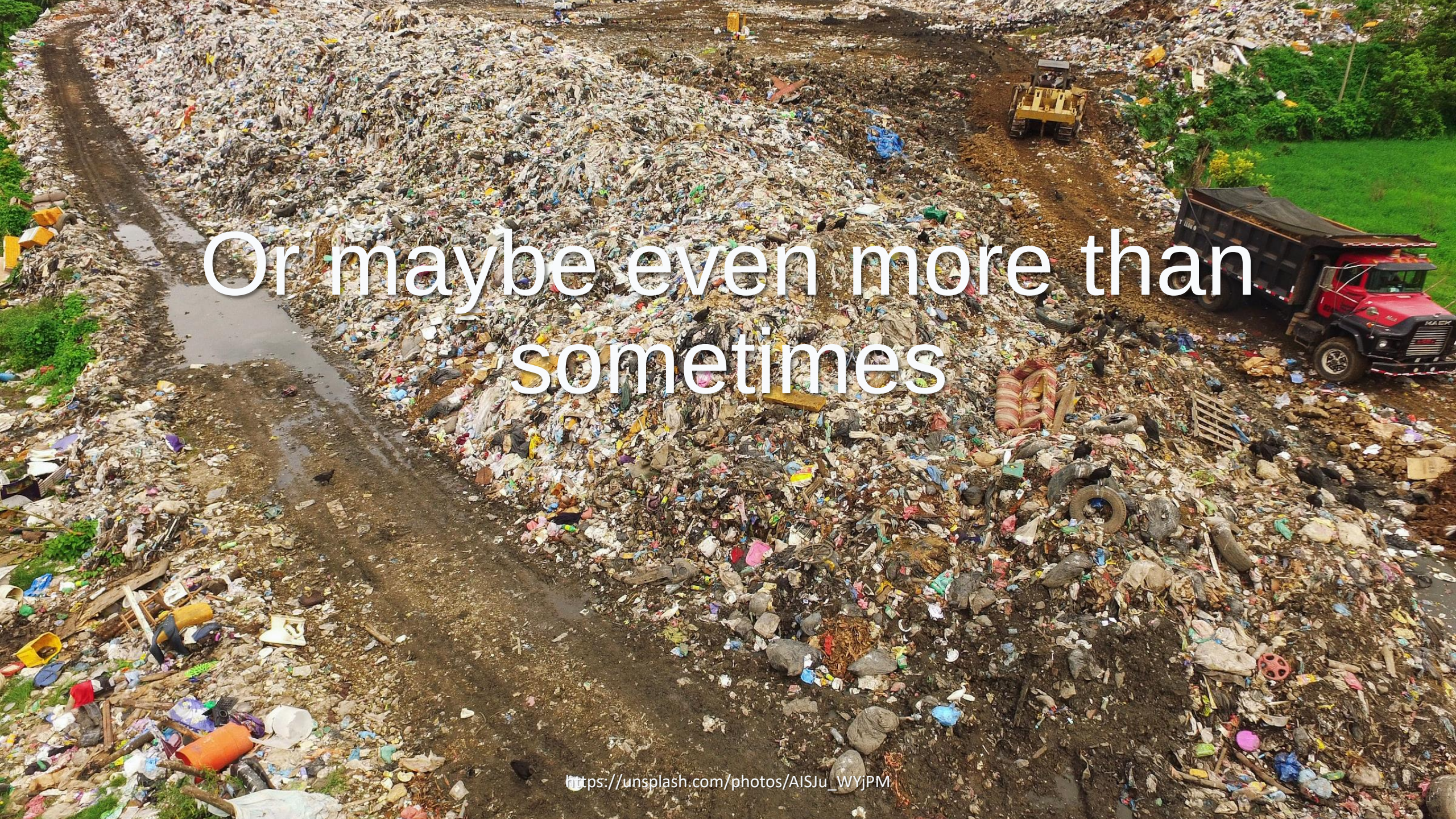
https://unsplash.com/photos/doicC_ATzkQ

To build something nice and beautiful



https://unsplash.com/photos/FanAO4o_g1U


But sometimes we get a dirty project
https://unsplash.com/photos/GmqezLxud8g


Or maybe even more than sometimes
https://unsplash.com/photos/AISJu_WYjPM

And it isn’t always because of:

- The other colleague
- The intern
- An external contractor

How did it happen?

```php
final class ShoppingList
{
    private array $list = [];

    public function add(string $name, int $amount): self
    {
        $this->list[$name] = $amount;
        return $this;
    }

    public function getList(): array
    {
        return $this->list;
    }
}

```

```php
final class ShoppingList
{
    private array $list = [];

    public function add(string $name, int $amount): self {
        if ($name === '') {
            throw new Exception('We can\'t add a product without a name');
        }
        if (isset($this->list[$name])) {
            $amount = $this->list[$name] + $amount;
        }
        if ($amount <0) {
            throw new Exception('We can\'t have a negative amount');
        }
        $this->list[$name] = $amount;
        return $this;
    }

    public function getList(): array {
        return $this->list;
    }
}

```

```php
final class ShoppingList
{
    private array $list = [];

    public function add(string $name, int $amount, bool $update = false): self {
        if ($name === '') {
            throw new Exception('We can\'t add a product without a name');
        }
        if ($amount < 1) {
            throw new Exception('We need to have a positive amount');
        }
        if ($update && isset($this->list[$name])) {
            $amount = $this->list[$name] + $amount;
        }
        $this->list[$name] = $amount;
        return $this;
    }

    public function remove(string $name): self {
        unset($this->list[$name]);
        return $this;
    }

    public function getList(): array {
        return $this->list;
    }
}
```

```php
    public function getList(bool $withTotal = false, string $orderBy = ''): array {
        $list = $this->list;
        if ($orderBy === 'amount')
        {
            asort($list);
        }
        elseif ($orderBy == 'name')
        {
            ksort($list);
        }
        if ($withTotal)
        {
            $total = array_sum($list);
            $list['total articles'] = $total;
        }
        if (ShopConfig::getConfig('withCrates')) {
            $productsInCrates = ShopConfig::getConfig('productsInCrates');
            $newList = [];
            $tempAmount = 0;
            $bag = $crate = 1;
            foreach ($list as $name => $amount) {
                $newList[$name] = $amount;

                if (in_array($name, ShopConfig::getConfig('withoutPackaging')))
                {
                    continue;
                }
                elseif (in_array($name, ShopConfig::getConfig('bagProducts')))
                {
                    $newList['bag ' . $bag] = 1;
                    $bag++;
                }
                else
                {
                    $tempAmount += $amount;
                    if ($tempAmount >= $productsInCrates)
                    {
                        $tempAmount -= $productsInCrates;
                        $newList['crate ' . $crate] = 1;
                        $crate++;
                    }
                }
            }
            if ($tempAmount > 0)
            {
                $newList['crate ' . $crate] = 1;
            }
            $list = $newList;
        }
        return $list;
    }
```

```php
final class ShoppingList
{
    private array $list = [];

    public function add(string $name, int $amount, bool $update = false): self {
        if ($name === '') {
            throw new Exception('We can\'t add a product without a name');
        }
        if ($amount < 1) {
            throw new Exception('We need to have a positive amount');
        }
        if ($update && isset($this->list[$name])) {
            $amount = $this->list[$name] + $amount;
        }
        $this->list[$name] = $amount;
        return $this;
    }

    public function remove(string $name): self {
        unset($this->list[$name]);
        return $this;
    }

    public function getList(bool $withTotal = false, string $orderBy = ''): array {
        $list = $this->list;
        if ($orderBy === 'amount')
        {
            asort($list);
        }
        elseif ($orderBy === 'name')
        {
            ksort($list);
        }
        if ($withTotal)
        {
            $total = array_sum($list);
            $list['total articles'] = $total;
        }
        if (ShopConfig::getConfig('withCrates')) {
            $productsInCrates = ShopConfig::getConfig('productsInCrates');
            $newList = [];
            $tempAmount = 0;
            $bag = $crate = 1;
            foreach ($list as $name => $amount) {
                $newList[$name] = $amount;

                if (in_array($name, ShopConfig::getConfig('withoutPackaging')))
                {
                    continue;
                }
                elseif (in_array($name, ShopConfig::getConfig('bagProducts')))
                {
                    $newList['bag ' . $bag] = 1;
                    $bag++;
                }
                else
                {
                    $tempAmount += $amount;
                    if ($tempAmount >= $productsInCrates)
                    {
                        $tempAmount -= $productsInCrates;
                        $newList['crate ' . $crate] = 1;
                        $crate++;
                    }
                }
            }
            if ($tempAmount > 0)
            {
                $newList['crate ' . $crate] = 1;
            }
            $list = $newList;
        }
        return $list;
    }
}
```

How to prevent the mess?


PHP
Mess Detector

Install

```
composer require --dev phpmd/phpmd¬
vendor/bin/phpmd¬
```

```
wget -c https://phpmd.org/static/latest/phpmd.phar¬
php phpmd.phar¬
```

Run it

```
php phpmd.phar -h
Mandatory arguments:
1) A php source code filename or directory. Can be a comma-separated string
2) A report format
3) A ruleset filename or a comma-separated string of rulesetfilenames

Example: phpmd /path/to/source format ruleset

Available formats: ansi, baseline, checkstyle, github, gitlab, html, json, sarif, text, xml.
Available rulesets: cleancode, codesize, controversial, design, naming, unusedcode.

Optional arguments that may be put after the mandatory arguments:
--minimumpriority: rule priority threshold; rules with lower priority than this will not be used
--reportfile: send report output to a file; default to STDOUT
--suffixes: comma-separated string of valid source code filename extensions, e.g. php,phtml
--exclude: comma-separated string of patterns that are used to ignore directories. Use asterisks to
exclude by pattern. For example *src/foo/*.php or *src/foo/*
--strict: also report those nodes with a @SuppressWarnings annotation
--ignore-errors-on-exit: will exit with a zero code, even on error
--ignore-violations-on-exit: will exit with a zero code, even if any violations are found
--generate-baseline: will generate a phpmd.baseline.xml next to the first ruleset file location
--update-baseline: will remove any non-existing violations from the phpmd.baseline.xml
--baseline-file: a custom location of the baseline file
```

Errors on a messy class

```
php phpmd.phar ShoppingList.php text cleancode,codesize,controversial,design,naming,unusedcode

/path/ShoppingList.php:11  BooleanArgumentFlag   The method add has a boolean flag argument $update, which is a certain
sign of a Single Responsibility Principle violation.

/path/ShoppingList.php:30  BooleanArgumentFlag   The method getList has a boolean flag argument $withTotal, which is a
certain sign of a Single Responsibility Principle violation.

/path/ShoppingList.php:30  CyclomaticComplexity  The method getList() has a Cyclomatic Complexity of 10. The configured
cyclomatic complexity threshold is 10.

/path/ShoppingList.php:45  StaticAccess          Avoid using static access to class 'ShopConfig' in method 'getList'.
/path/ShoppingList.php:46  StaticAccess          Avoid using static access to class 'ShopConfig' in method 'getList'.
/path/ShoppingList.php:53  StaticAccess          Avoid using static access to class 'ShopConfig' in method 'getList'.
/path/ShoppingList.php:57  StaticAccess          Avoid using static access to class 'ShopConfig' in method 'getList'.

/path/ShoppingList.php:63  ElseExpression        The method getList uses an else expression. Else clauses are basically
not necessary and you can simplify the code by not using them.
```

Rules in the documentation

Current Rulesets

List of rulesets and rules contained in each ruleset.

- Clean Code Rules: The Clean Code ruleset contains rules that enforce a clean code base. This includes rules from SOLID and object calisthenics.
- Code Size Rules: The Code Size Ruleset contains a collection of rules that find code size related problems.
- Controversial Rules: This ruleset contains a collection of controversial rules.
- Design Rules: The Design Ruleset contains a collection of rules that find software design related problems.
- Naming Rules: The Naming Ruleset contains a collection of rules about names - too long, too short, and so forth.
- Unused Code Rules: The Unused Code Ruleset contains a collection of rules that find unused code.

Custom ruleset

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ruleset name="GitLab Models for Laravel PHPMD ruleset"
         xmlns="http://pmd.sf.net/ruleset/1.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://pmd.sf.net/ruleset/1.0.0 http://pmd.sf.net
/ruleset_xml_schema.xsd"
         xsi:noNamespaceSchemaLocation="http://pmd.sf.net/ruleset_xml_schema.xsd">
    <description>
        GitLab Models for Laravel PHPMD ruleset
    </description>
    <rule ref="rulesets/design.xml" />
    <rule ref="rulesets/codesize.xml" />
    <rule ref="rulesets/controversial.xml" />
    <rule ref="rulesets/naming.xml">
        <exclude name="LongVariable" />
    </rule>
    <rule ref="rulesets/unusedcode.xml" />
    <rule ref="rulesets/naming.xml/LongVariable">
        <properties>
            <property name="maximum" value="40" />
        </properties>
    </rule>
</ruleset>
```

But my existing project?

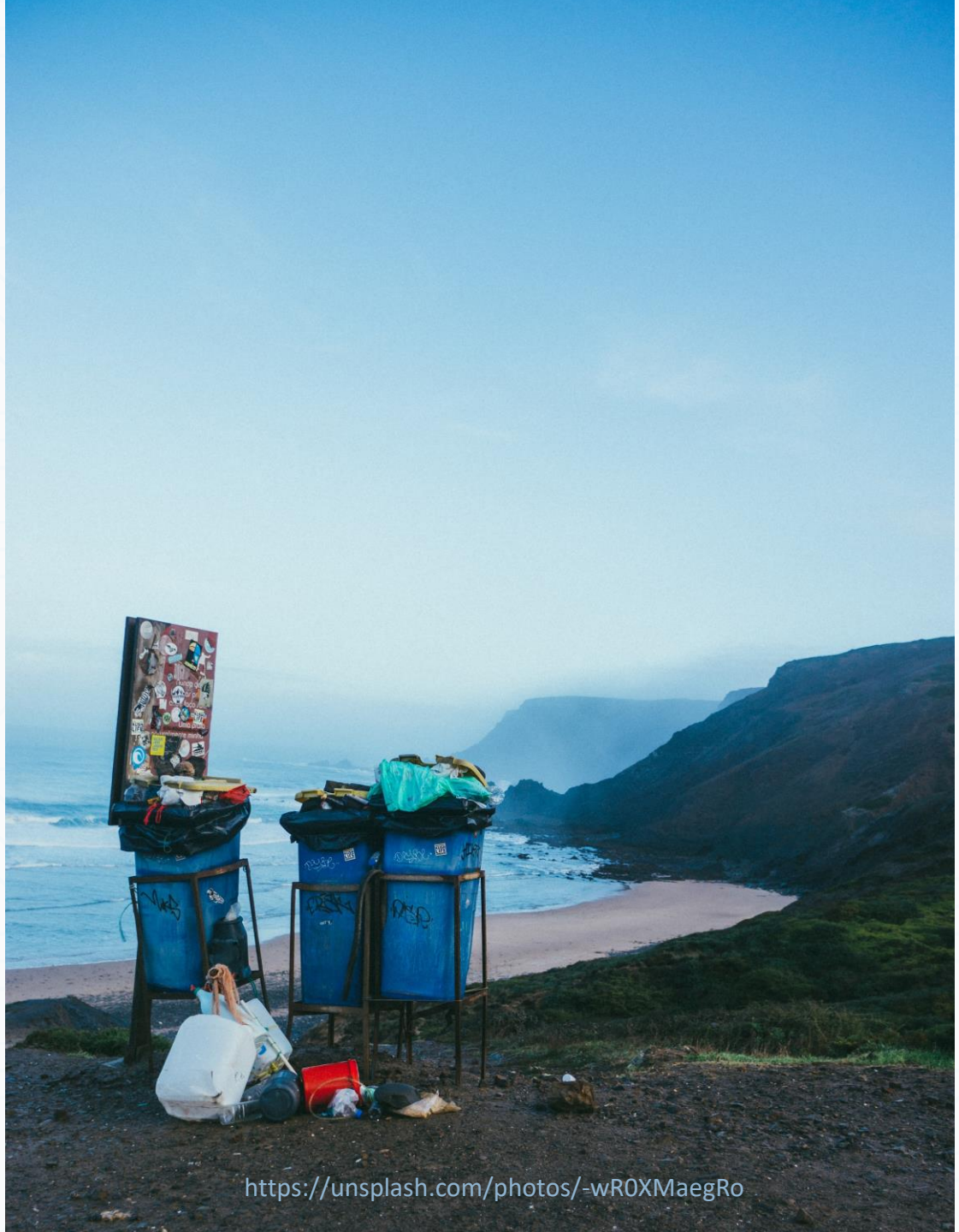
```
php phpmd.phar ShoppingList.php text cleancode,codesize,controversial,design,naming,unusedcode
--generate-baseline --baseline-file phpmd_baseline.xml

php phpmd.phar ShoppingList.php text cleancode,codesize,controversial,design,naming,unusedcode
--baseline-file phpmd_baseline.xml
```





Add it to your CI

Setup and forget.


https://unsplash.com/photos/-wR0XMaegRo

Cleanup the mess!

Questions? -> Contact me

https://tjvb.nl/about

@tvbeek

https://gitter.im/phpmd/community

@php_pmd