Mastering automated refactoring with custom Rector rules

PHP[TEK] 2025

May 20th - 22nd 2025

Sheraton Suites Chicago O'Hare

github.com/DaveLiddament/rector-custom-rules-tutorial

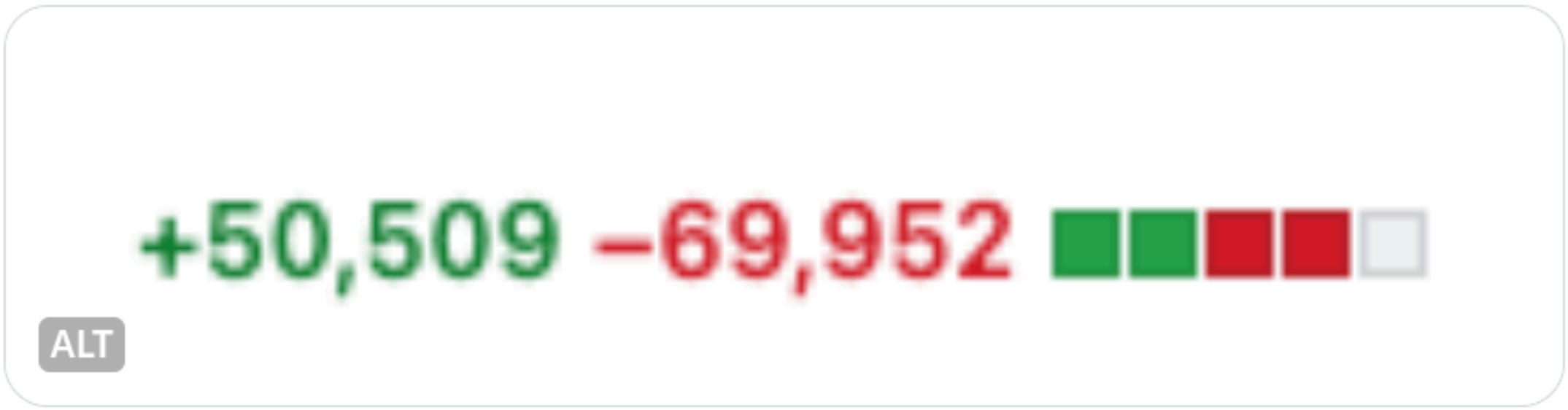
Dave Liddament
@DaveLiddament

Finally completed an upgrade from #laravel 4.2 to 9.

To celebrate Laravel have released version 10!

Remember your tests folks. This upgrade wouldn't have been possible, or as smooth, without tests.
Also @rectorphp is pretty handy for this.

I must write a blog/talk about it.



5:25 PM · Feb 24, 2023 · **3,813** Views

Practical

Thoughts

Rector and the AST

Type coverage

@daveliddament

Type coverage

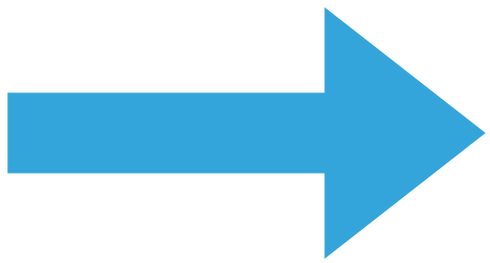
```
function getNames($users)
{

   $names = [];
   foreach($users as $user) {
      $names[] = $user->getUser();
   }

   return join(‘,’, $names);
}
```

0% type coverage

```php
/** @param array<int,User> $users */
function getNames(array $users): string
{

  $names = [];
  foreach($users as $user) {
    $names[] = $user->getUser();
  }
  return join(‘,’, $names);

}
```

100% type coverage

User

getUser  getUsername

0% type coverage

```
function getNames($users)
{

   $names = [];
   foreach($users as $user) {
      $names[] = $user->getUser();
   }

   return join(‘,’, $names);
}
```

100% type coverage

```
/** @param array<int,User> $users */
function getNames(array $users): string
{

   $names = [];
   foreach($users as $user) {
     $names[] = $user->getUser();
   }
   return join(‘,’, $names);

}
```

100% type coverage

```
/** @param array<int,User> $users */
function getNames(array $users): string
{

    $names = [];
    foreach($users as $user) {
      $names[] = $user->getUsername();
    }
    return join(‘,’, $names);

}
```

100% automated!

Practical

Thoughts

Type coverage

Rector and the AST

@daveliddament

359 Rules (and counting)

RenameMethodRector

Turns method names to new ones.

🔧 configure it!

- class: `Rector\Renaming\Rector\MethodCall\RenameMethodRector`

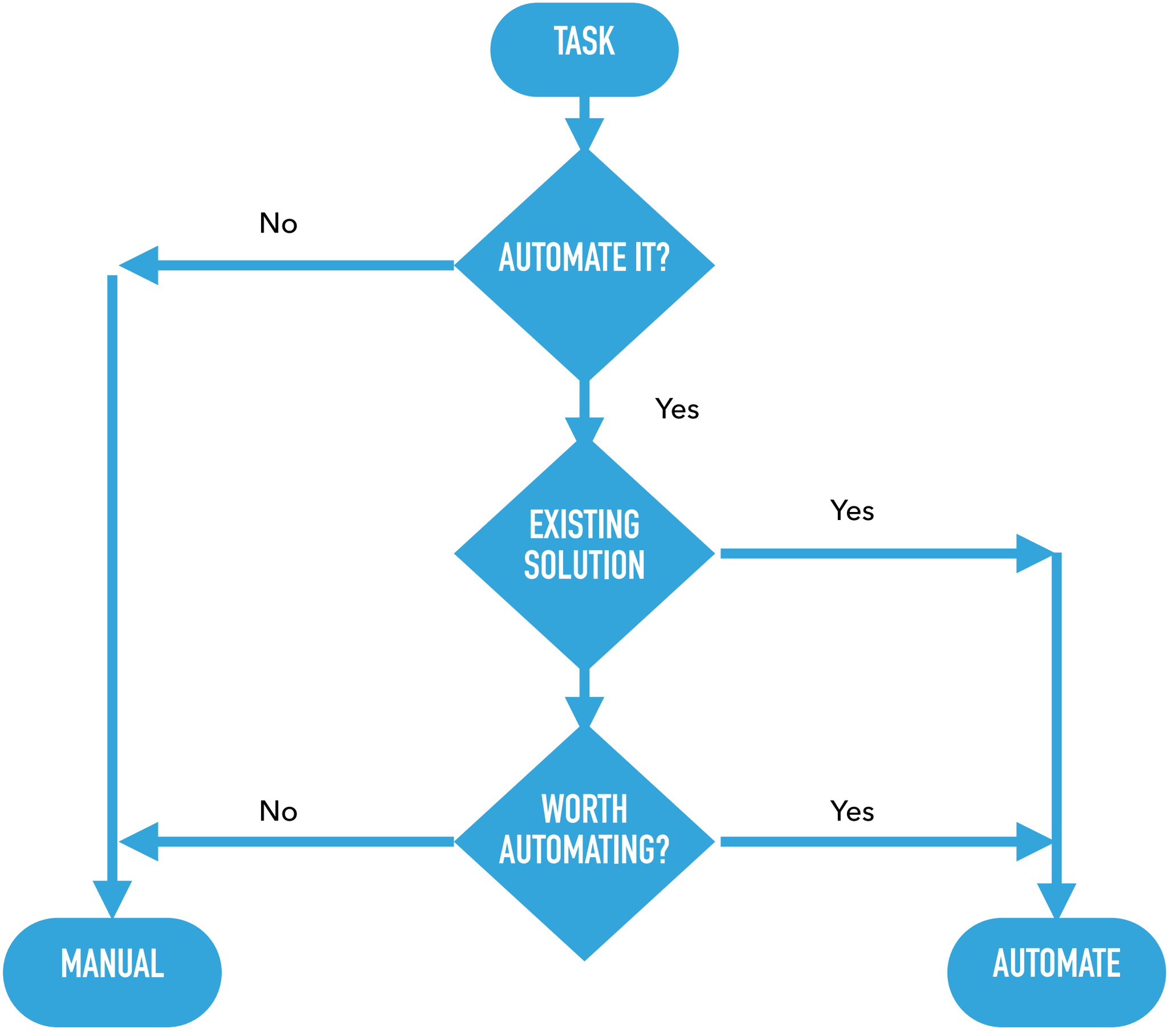
```
 $someObject = new SomeExampleClass;
-$someObject->oldMethod();
+$someObject->newMethod();
```

Existing rules

https://getrector.com/

https://github.com/rectorphp/rector-symfony

https://github.com/driftingly/rector-laravel


TASK
AUTOMATE IT?
No
Yes
EXISTING SOLUTION
Yes
WORTH AUTOMATING?
No
Yes
MANUAL
AUTOMATE

Old

```
class Car extends Model
{
  public function user()
  {
    return $this->belongsTo('User');
  }
}
```

New

```
class Car extends Model
{
  public function user()
  {
    return $this->belongsTo(App\Models\User::class);
  }
}
```

No Change 1

```
class Car extends Model
{
  public function user()
  {
    return $this->belongsTo(User::class);
  }
}
```

No Change 2

```
class CarService
{
  public function user()
  {
    return $this->belongsTo('User');
  }
}
```

```
$ vendor/bin/rector custom-rule
```

```
 What is the name of the rule class (e.g. "LegacyCallToDbalMethodCall")?:
 > FixModelMappingsRector

Generated files
===============

 * utils/rector/src/Rector/FixModelMappingsRector.php
 * utils/rector/tests/Rector/FixModelMappingsRector/Fixture/some_class.php.inc
 * utils/rector/tests/Rector/FixModelMappingsRector/config/configured_rule.php
 * utils/rector/tests/Rector/FixModelMappingsRector/FixModelMappingsRectorTest.php

 [OK] Base for the "FixModelMappingsRector" rule was created.
Now you can fill the missing parts

 [OK] We also update composer.json autoload-dev, to load Rector
rules. Now run "composer dump-autoload" to update paths
```

Old

```php
<?php
namespace Utils\Rector\Tests\Rector\FixModelMappingsRector\Fixture;

use App\Framework\BelongsTo;
use App\Framework\Model;

final class Car extends Model
{
    public function user(): BelongsTo
    {
        return $this->belongsTo('User');
    }
}
```

New

```php
<?php
namespace Utils\Rector\Tests\Rector\FixModelMappingsRector\Fixture;

use App\Framework\BelongsTo;
use App\Framework\Model;

final class Car extends Model
{
    public function user(): BelongsTo
    {
        return $this->belongsTo(App\Models\User::class);
    }
}
```

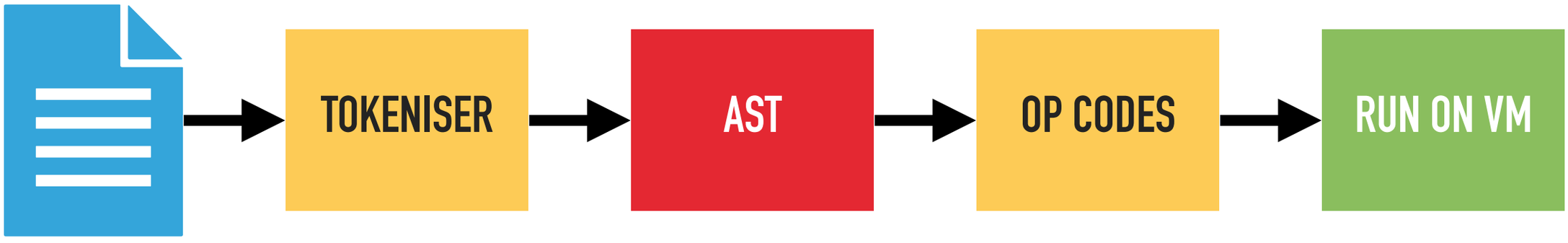
No change

```php
<?php
namespace Utils\Rector\Tests\Rector\FixModelMappingsRector\Fixture;

use App\Framework\BelongsTo;
use App\Framework\Model;

final class Car extends Model
{
    public function user(): BelongsTo
    {
        return $this->belongsTo(App\Models\User::class);
    }
}
```

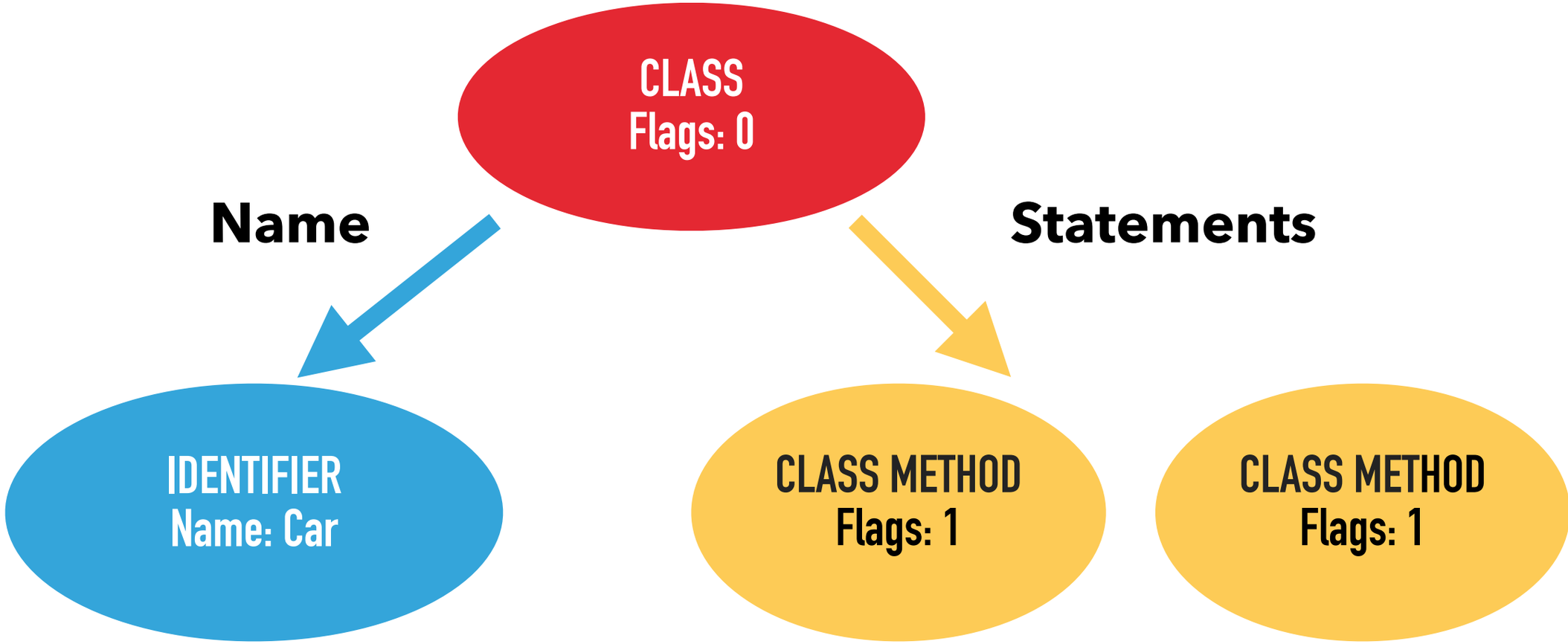
```
FixModelMappingsRector extends AbstractRector
{
    public function getRuleDefinition(): RuleDefinition

    public function getNodeTypes(): array

    public function refactor(Node $node): ?Node
}
```

```php
public function getRuleDefinition(): RuleDefinition
{
    return new RuleDefinition(
        "Fix Model mappings",
        [
            new CodeSample(
                '$this->belongsTo("User");',
                '$this->belongsTo(User::class);',
            ),
        ]
    );
}
```

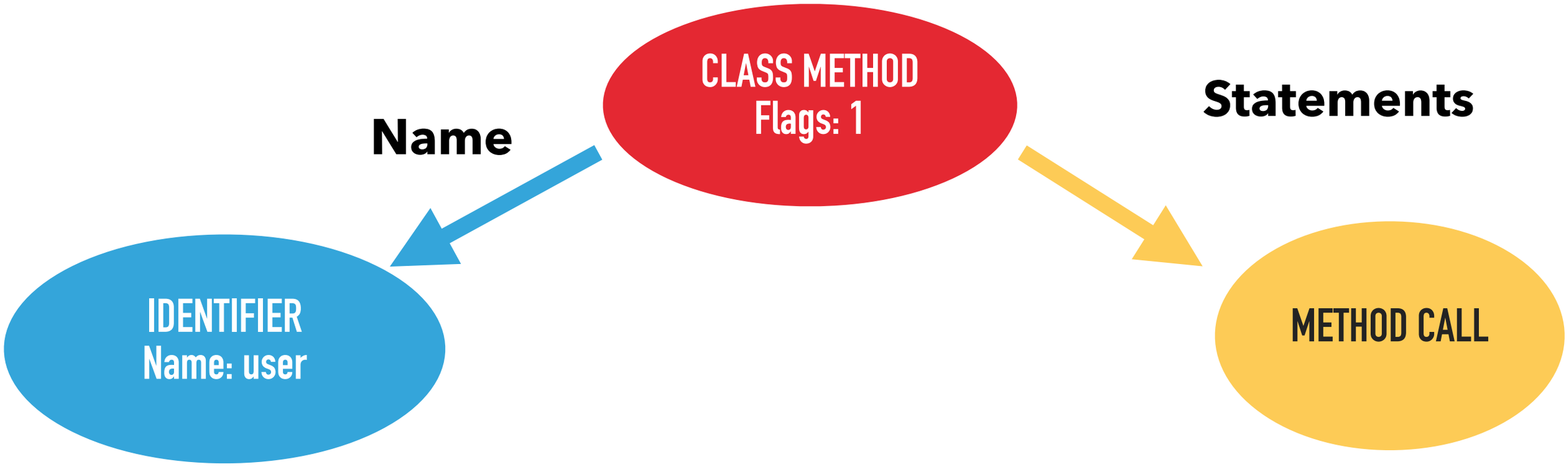
```
FixModelMappingsRector extends AbstractRector
{
    public function getRuleDefinition(): RuleDefinition

    public function getNodeTypes(): array

    public function refactor(Node $node): ?Node
}
```


TOKENISER
AST
OP CODES
RUN ON VM

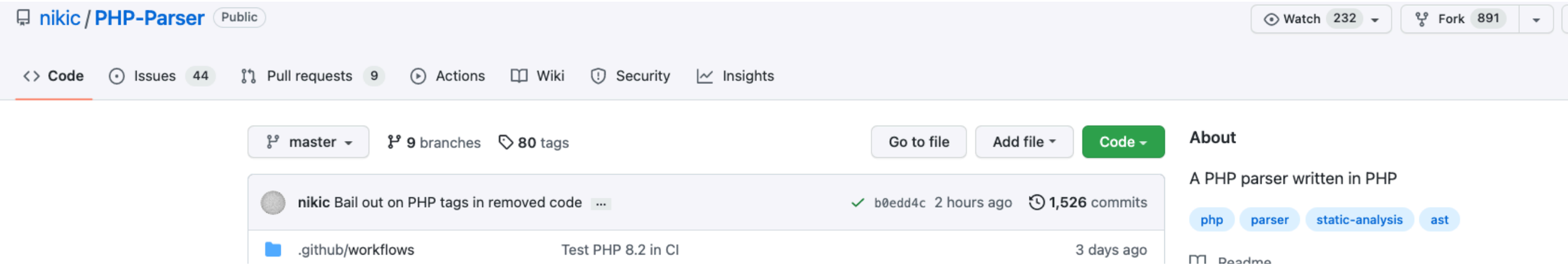
```
class Car
{
    public function users() {…}

    public function anotherMethod() {…}
}
```



```
public function user ()
{
    return $this->belongsTo('User');
}
```



https://github.com/nikic/PHP-Parser



- PHP code can be represented by an AST
- Different types of Node
- Nodes contain information
- Each type of node has different information

CLASS
Flags: 0

IDENTIFIER
Name: PersonNotifier

PROPERTY
Flags: 4

CLASS METHOD
Flags: 1

CLASS METHOD
Flags: 1

IDENTIFIER
Name: notifyPlayer

PARAM
ByRef: false
Variadic: false

METHOD CALL

IDENTIFIER
Name: string

VALUE
ByRef: false
Variadic: false

```
$this->belongsTo('User');

$this->belongsTo(App\Models\User::class);
```

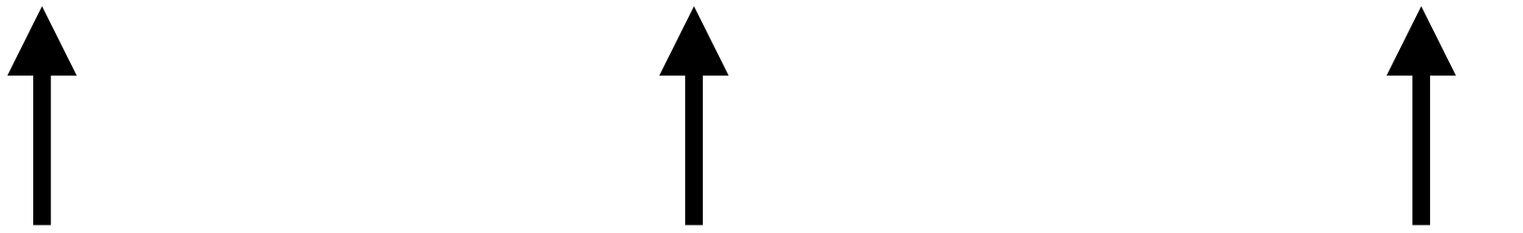
```
class MethodCall extends \PhpParser\Node\Expr\CallLike
{
    /** @var Expr Variable holding object */
    public $var;

    /** @var Identifier|Expr Method name */
    public $name;

    /** @var array<Arg|VariadicPlaceholder> Arguments */
    public $args;

    // Rest of class …
```

```
$this -> belongsTo ('User');
```

```
public function getNodes(): array
{
    return [MethodCall::class];
}
```

```
public function refactor(Node $node): ?Node
{

}
```

```
$this->belongsTo('User');
```



- var is an object that extends Model
- name is belongsTo
- 1st argument is a string

```
public function refactor(Node $node): ?Node
{
    // Is var an object that extends Model?

    $modelType = new ObjectType(Model::class);
    if (!$this->isObjectType($node->var, $modelType)) {
        return null;
    }
```

```
// Is name belongsTo

if (!$this->isName($node->name, 'belongsTo')) {
    return null;
}
```

```
// Is 1st argument a string?

$arg = $node->args[0] ?? null;
if (! $arg instanceof Arg) {
    return null;
}

if (! $arg->value instanceof String_) {
    return null;
}
```

```
// Return updated node

$fqcn = 'App\\Models\\' . $arg->value->value;

$arg->value = new ClassConstFetch(
                        new Name($fqcn),
                        new Identifier("class"));

    return $node;
}
```

User::class is a ClassConstFetch Node in the AST

```php
public function refactor(Node $node): ?Node
{
    // Is var an object that extends Model?
    $modelType = new ObjectType(Model::class);
    if (!$this->isObjectType($node->var, $modelType)) {
        return null;
    }

    // Is name belongsTo
    if (!$this->isName($node->name, 'belongsTo')) {
        return null;
    }

    // Is 1st argument a string?
    $arg = $node->args[0] ?? null;
    if (! $arg instanceof Arg) {
        return null;
    }
    if (! $arg->value instanceof String_) {
        return null;
    }

    // Return updated node
    $fqcn = 'App\\Models\\' . $arg->value->value;
    $arg->value = new ClassConstFetch(new Name($fqcn),new Identifier("class"));

    return $node;
}
```

Practical

Type coverage

Rector and the AST

Thoughts

@daveliddament

Practical

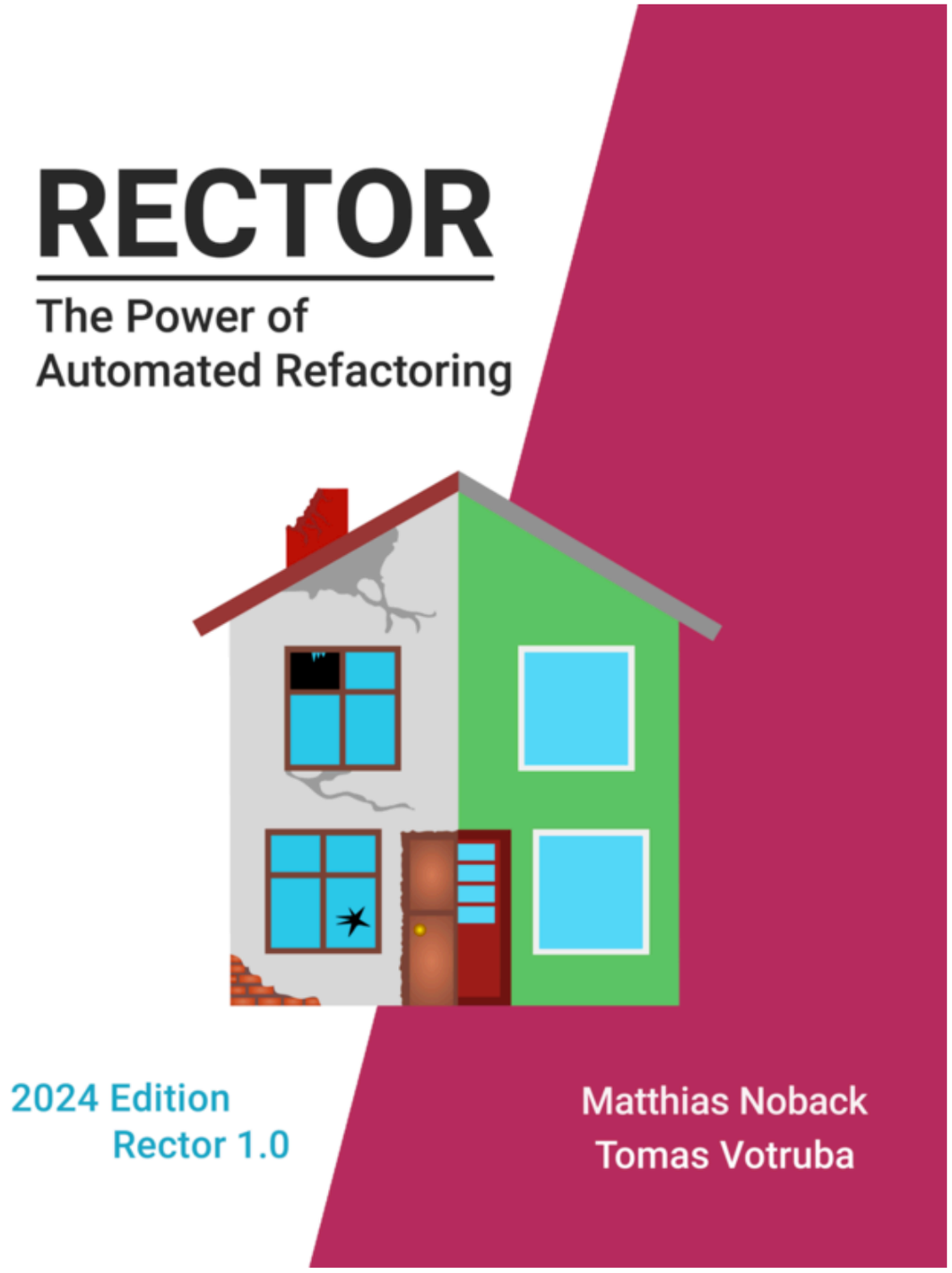
Thoughts

Type coverage

Rector and the AST

@daveliddament

https://github.com/DaveLiddament/rector-rule-demo-update-belongs-to



@daveliddament

Dave Liddament

@daveliddament

@daveliddament.bsky.social

github.com/DaveLiddament

www.linkedin.com/in/daveliddament/

Me when I'm not coding!