

Mastering automated refactoring with custom Rector rules



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.

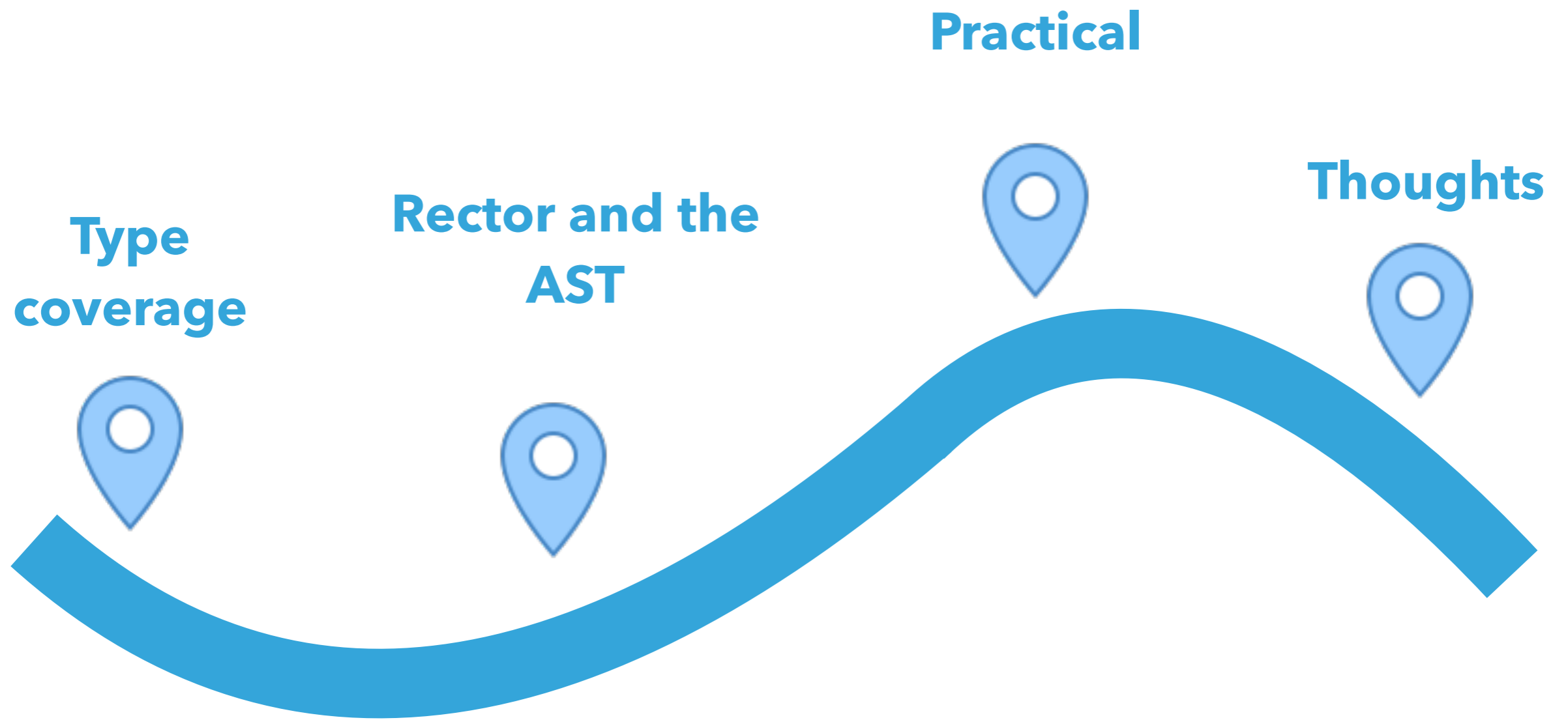
I must write a blog/talk about it.

+50,509 -69,952 ■■■■■■

ALT

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

@daveliddament



Type coverage

```
function getNames($users)
{

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

    return join(', ', $names);
}
```

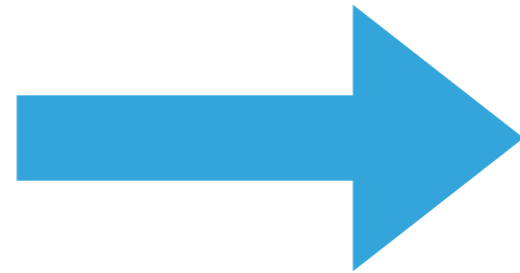
0% type coverage

```
/** @param array<int,User> $users */  
function getNames(array $users): string  
{  
  
    $names = [];  
    foreach($users as $user) {  
        $names[] = $user->getUser();  
    }  
    return join(', ', $names);  
  
}
```

100% type coverage

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!



**Type
coverage**

**Rector and the
AST**

Practical

Thoughts



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();
```



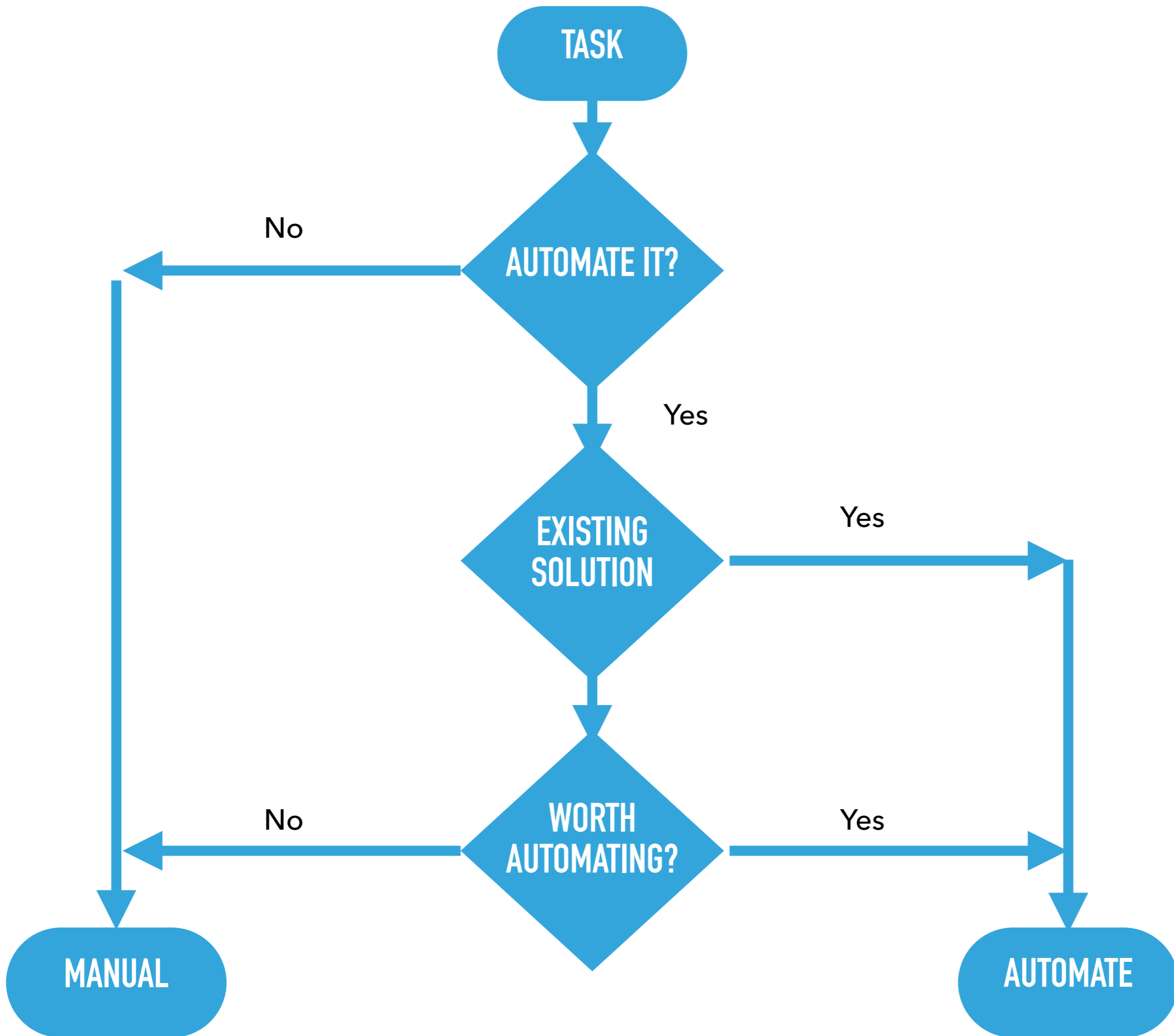


Existing rules

<https://getrector.com/>

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

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



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
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');
    }
}
```

```
<?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);
    }
}
```

New

**No
change**



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



```
class Car
```

```
public function users() {...}
```

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

Name

Statements

IDENTIFIER
Name: Car

CLASS
Flags: 0

CLASS METHOD
Flags: 1

CLASS METHOD
Flags: 1

```
public function user ()
```

```
{
```

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

```
}
```

Name

CLASS METHOD
Flags: 1

Statements

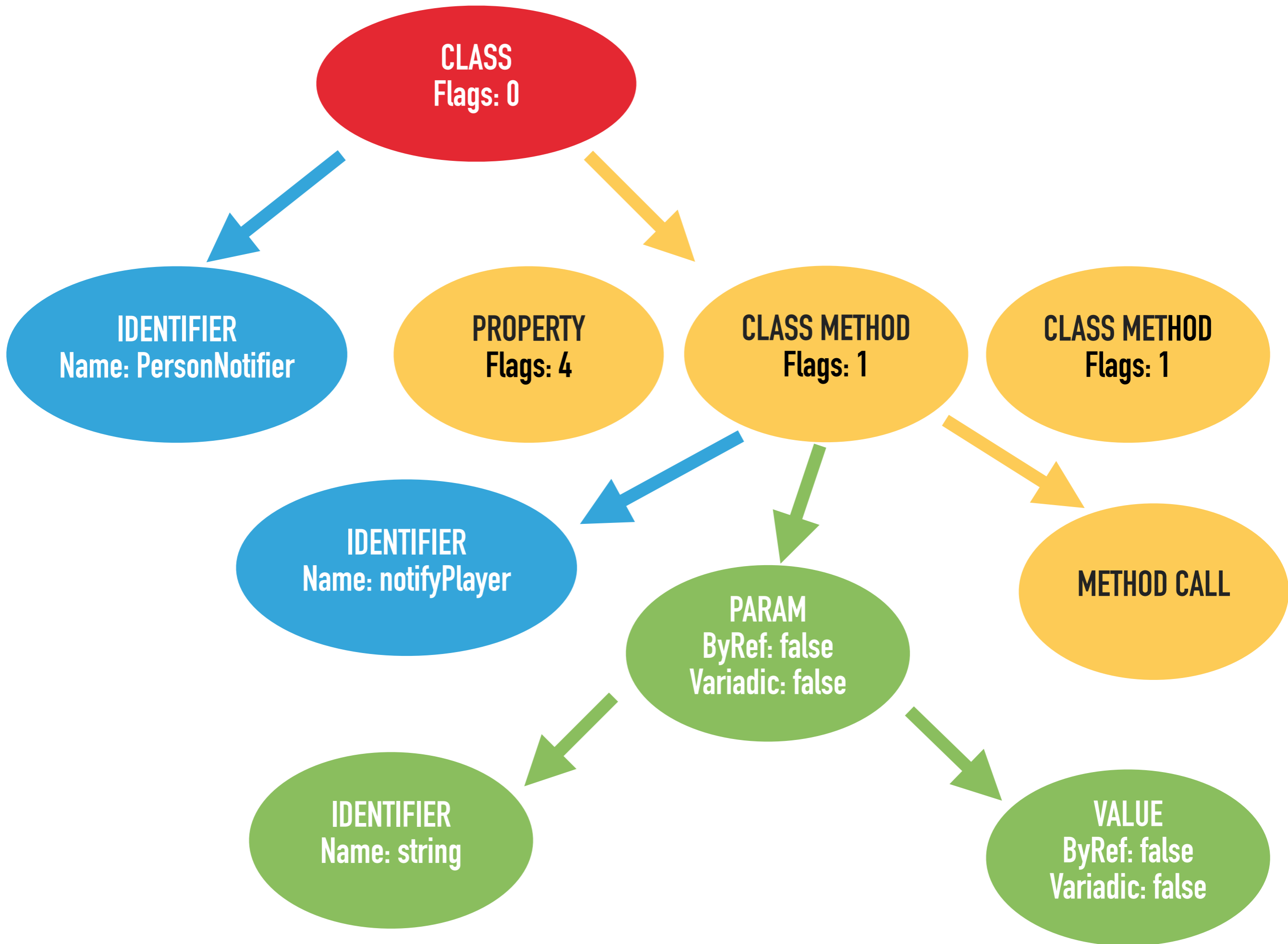
IDENTIFIER
Name: user

METHOD CALL

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

The screenshot shows the GitHub repository page for 'nikic/PHP-Parser'. The repository is public and has 232 watchers and 891 forks. The main navigation bar includes links for Code, Issues (44), Pull requests (9), Actions, Wiki, Security, and Insights. The repository is currently on the 'master' branch, with 9 branches and 80 tags. A recent commit by 'nikic' is shown with the message 'Bail out on PHP tags in removed code ...', commit hash 'b0edd4c', and a timestamp of '2 hours ago'. There are 1,526 commits in total. A workflow file is listed: '.github/workflows/Test PHP 8.2 in CI', updated '3 days ago'. The 'About' section describes it as 'A PHP parser written in PHP' and lists tags: 'php', 'parser', 'static-analysis', and 'ast'. A 'Readme' link is also visible.

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



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

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



METHOD CALL

```
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;
```

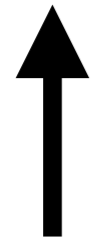
```
// Rest of class ...
```

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

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

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

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



var



name



args

- ▶ **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

```
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;
}
```

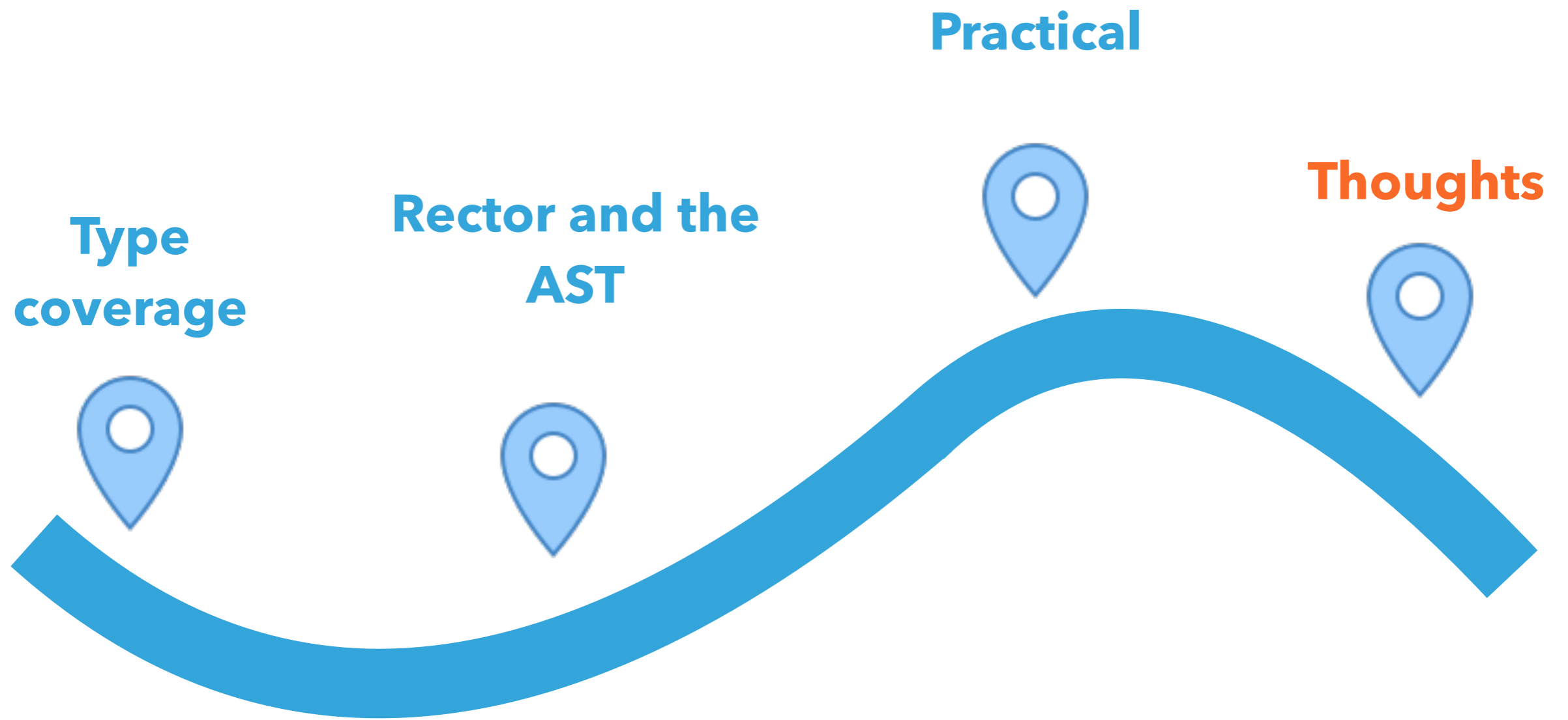


**Type
coverage**

**Rector and the
AST**

Practical

Thoughts



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

RECTOR

The Power of
Automated Refactoring



2024 Edition
Rector 1.0

Matthias Noback
Tomas Votruba

Dave Liddament

@daveliddament

@daveliddament.bsky.social

github.com/DaveLiddament

www.linkedin.com/in/daveliddament/



Me when I'm not coding!