tretton37

Testing the Essential

with AutoFixture

Enrico Campidoglio

@ecampidoglio

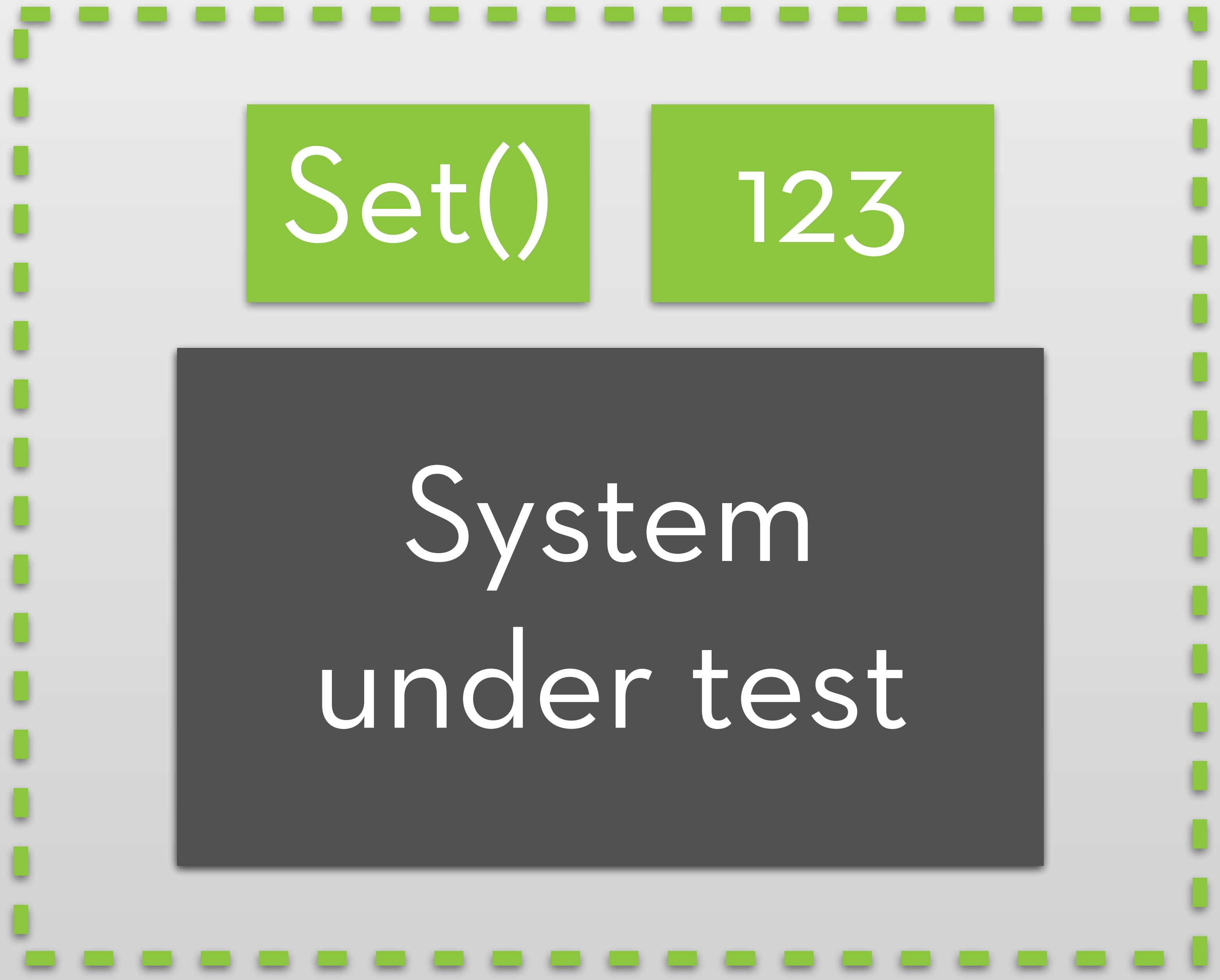
Premise:

Small + Expressive = 👍

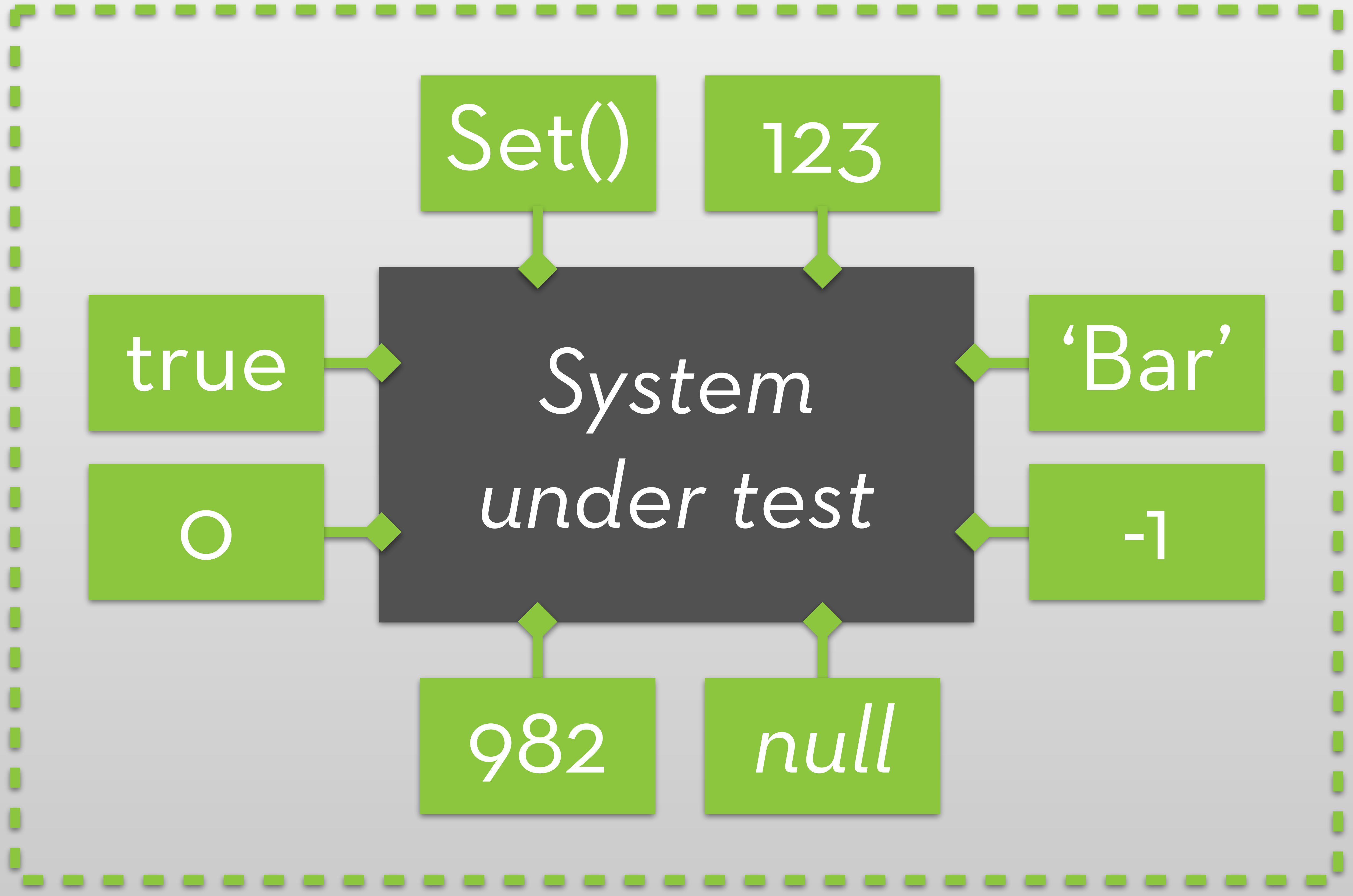
Essential

3 unit testing patterns → AutoFixture

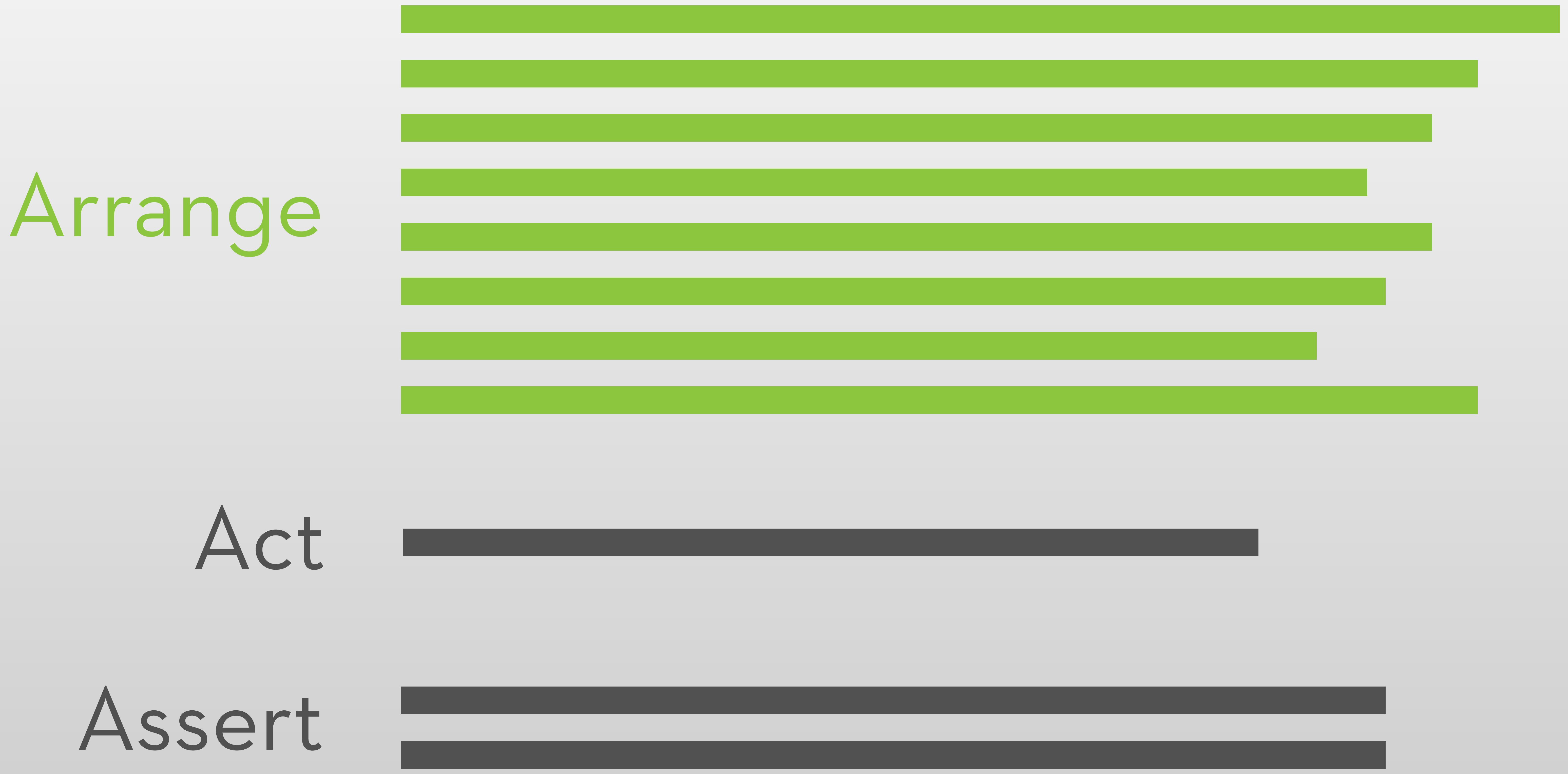
Context

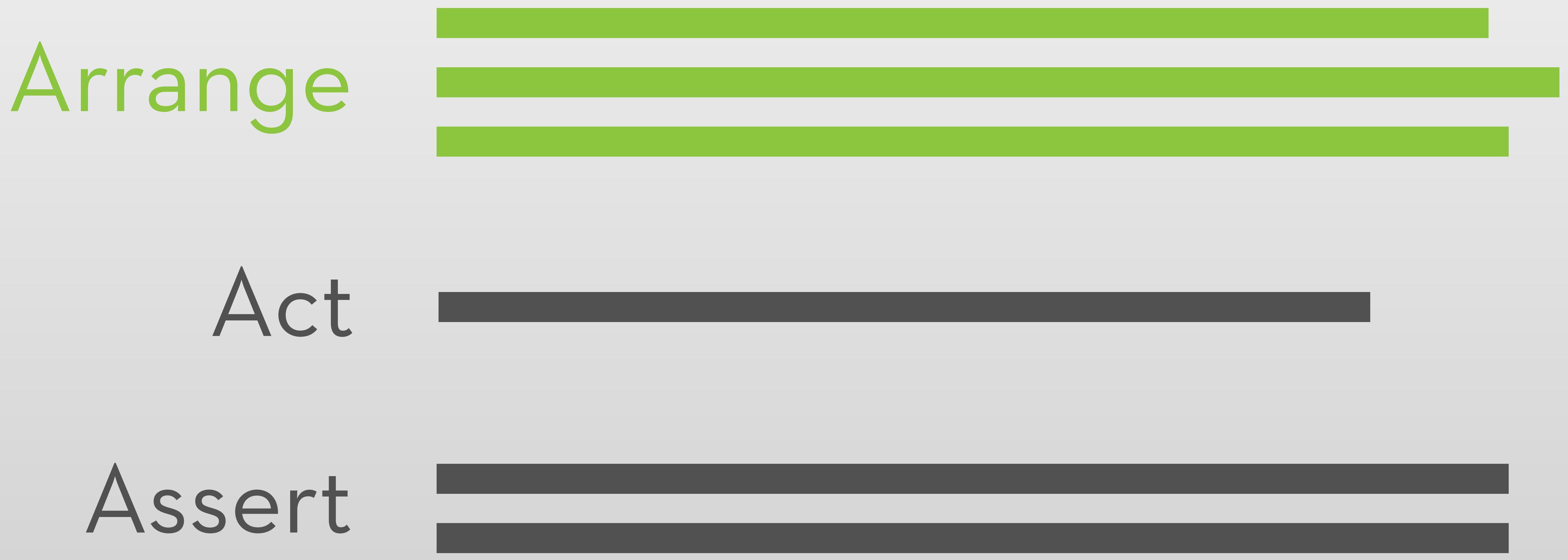


Fixture

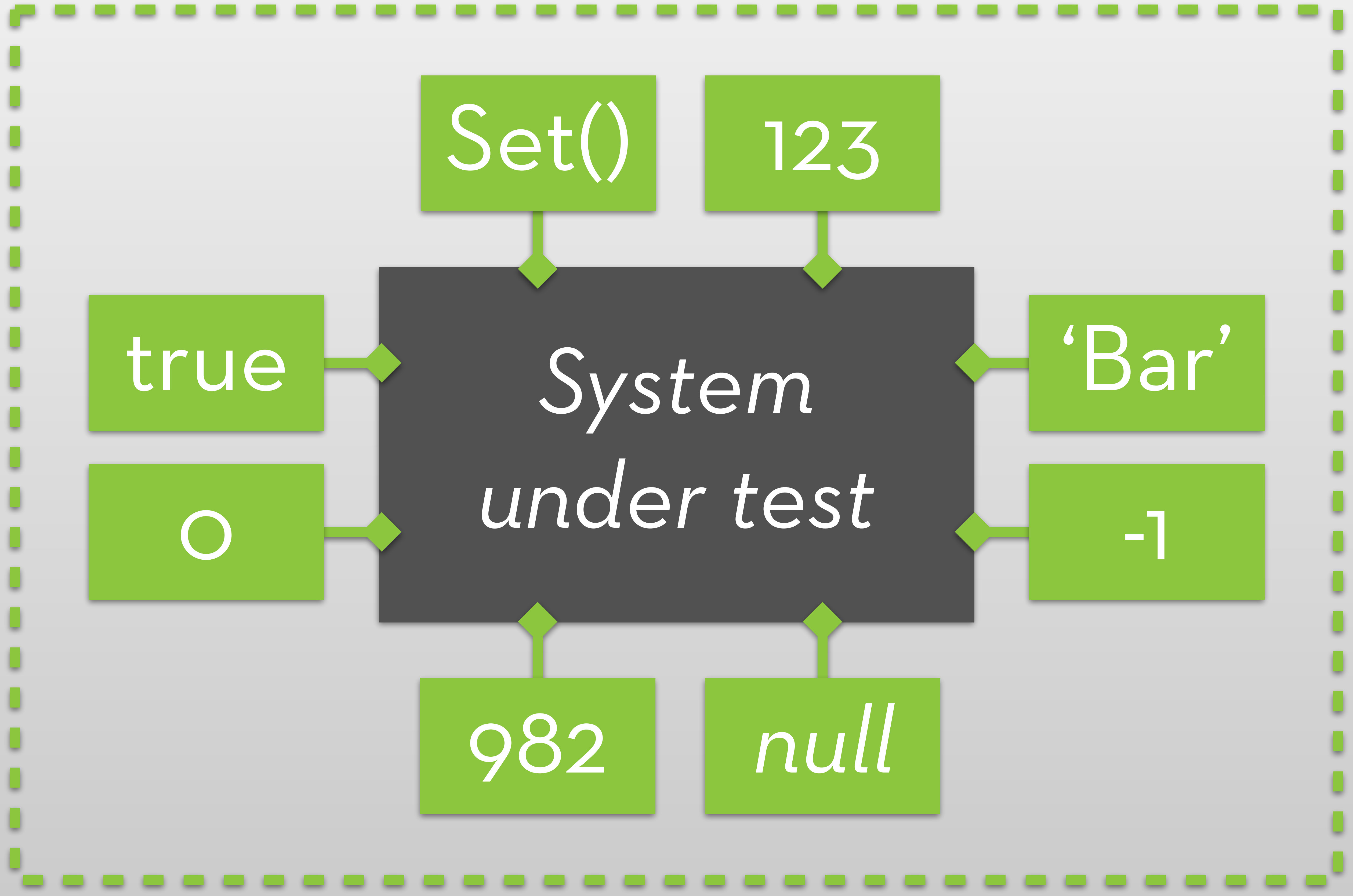


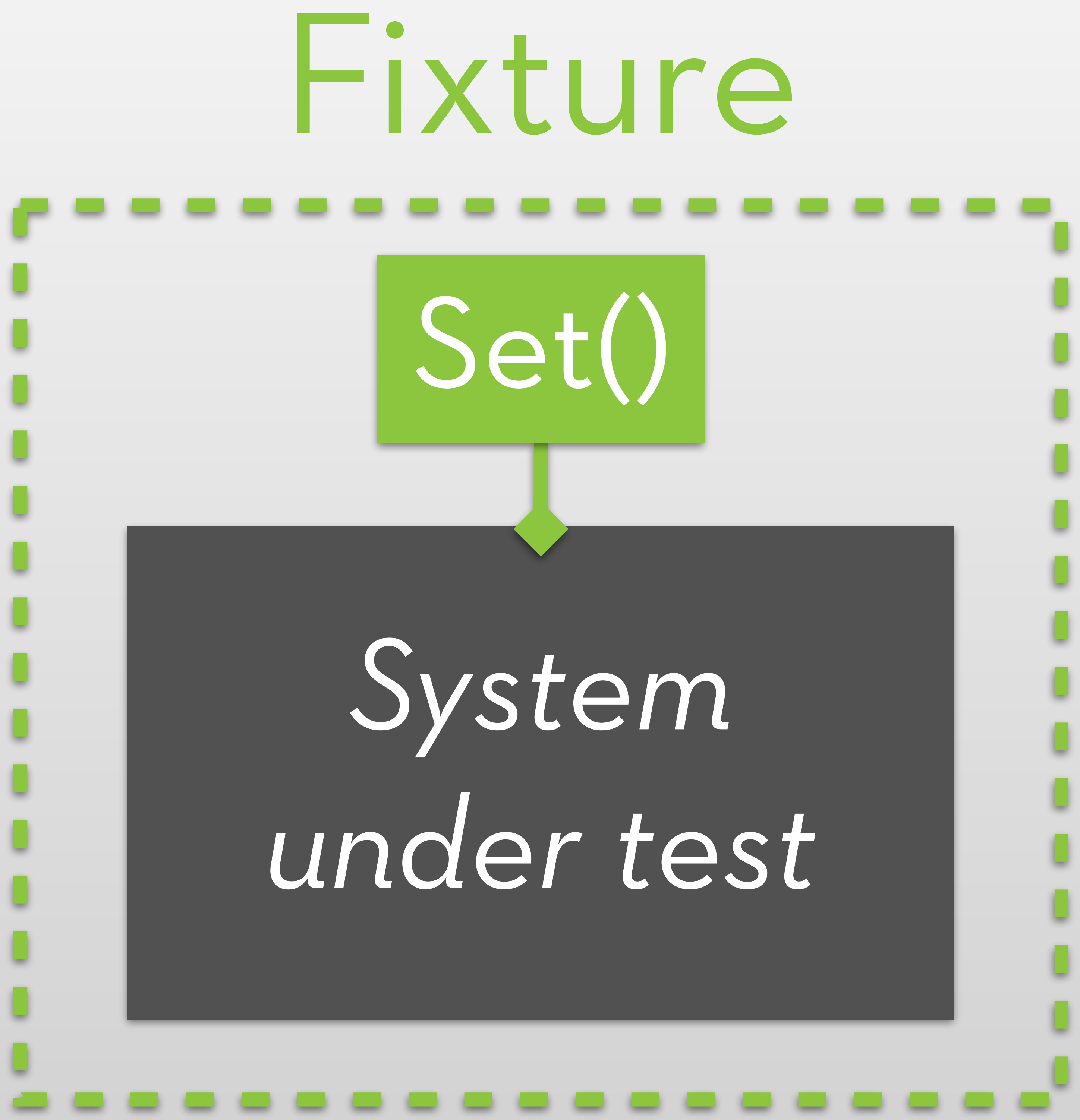
3 parts


Arrange
Act
Assert

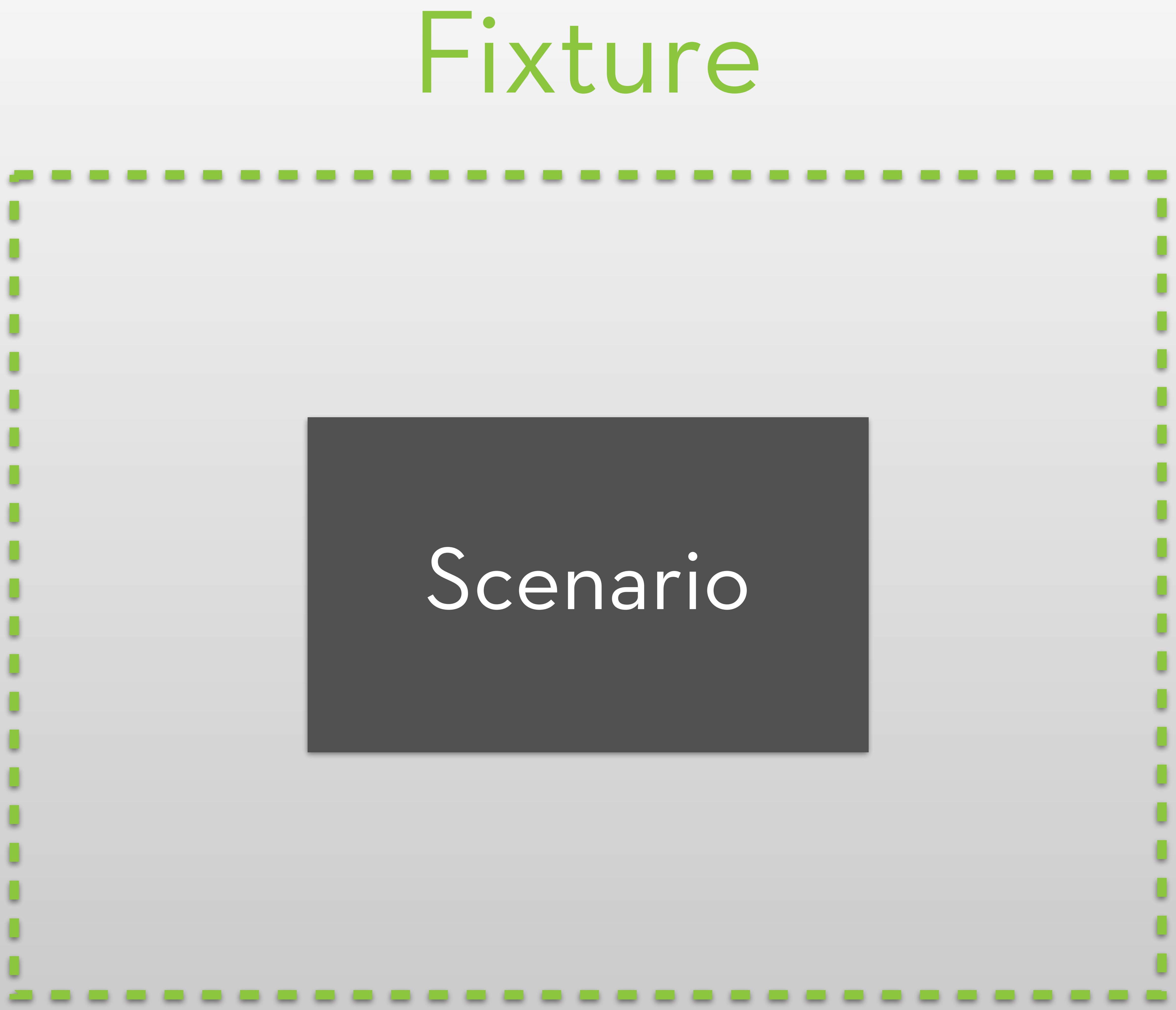

Arrange
Act
Assert

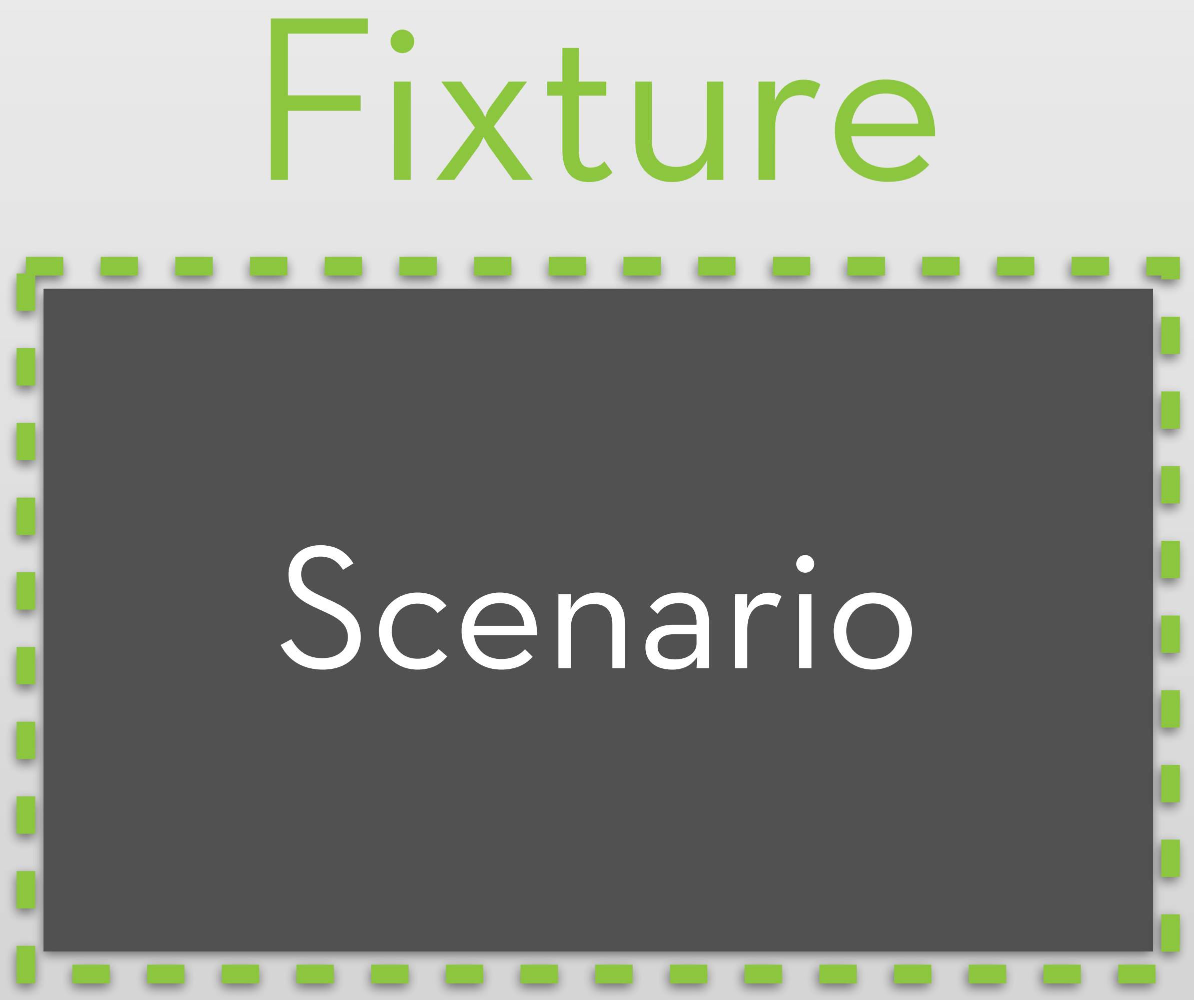
Fixture




Fixture
Set()
System under test

Fewer explicit calls = Easier to change


Fixture
Scenario


Fixture
Scenario

3 unit testing patterns

1 Anonymous Data

Any input value
that exercises the code path
under test

Anonymous

```
public bool IsPositive(int value)
{
   return value > 0;
}
```

2 Equivalence Classes

The group of input values
that exercise the same path
through the code

Equivalence classes

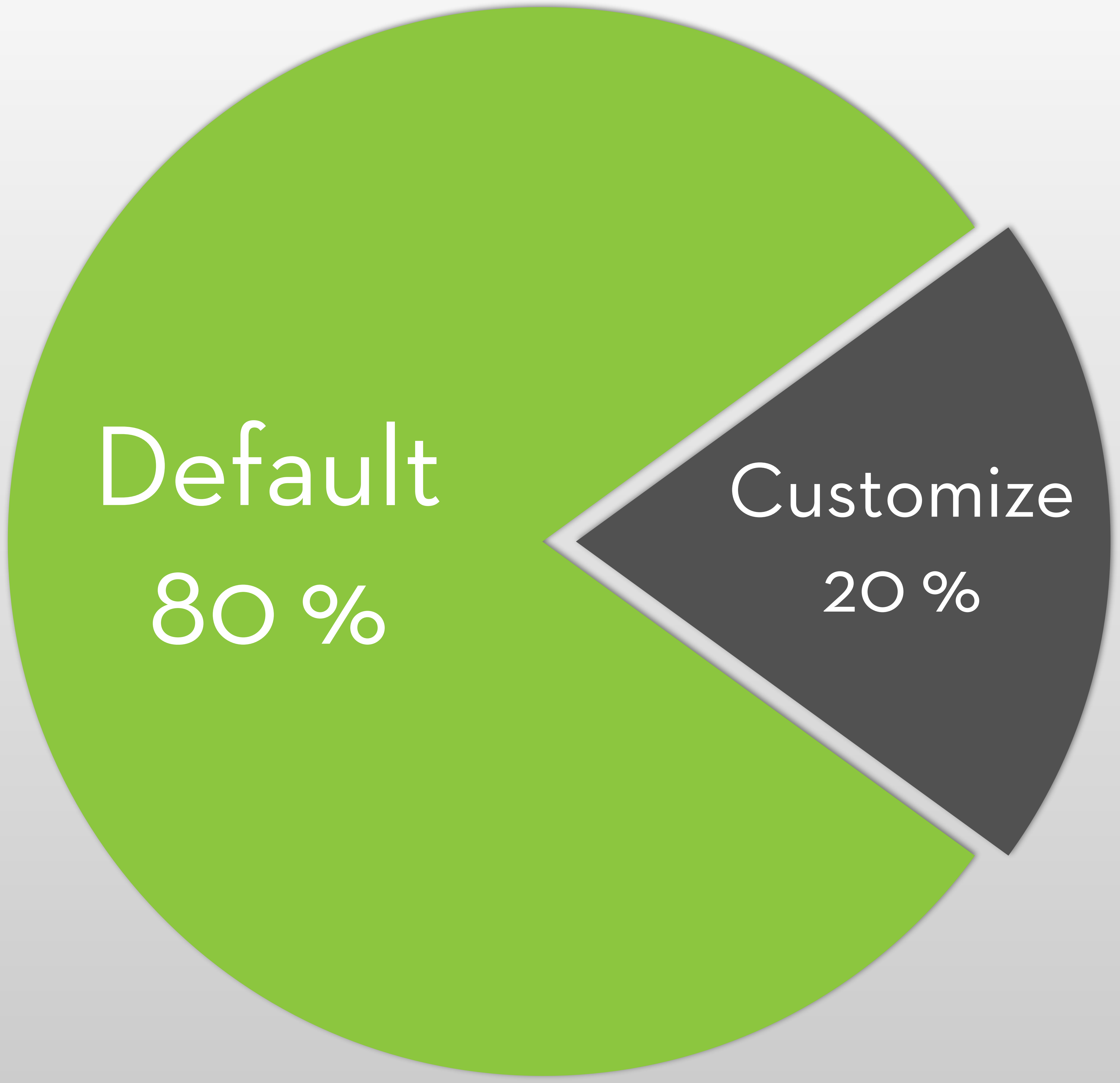
```
Fork(4);
Fork(9);
```

```
Fork(0);
Fork(2);
```

```
public void Fork(int value)
{
    if (value > 3)
    {
        // Do this
    }
    else
    {
        // Do that
    }
}
```

3 Test Data Builder

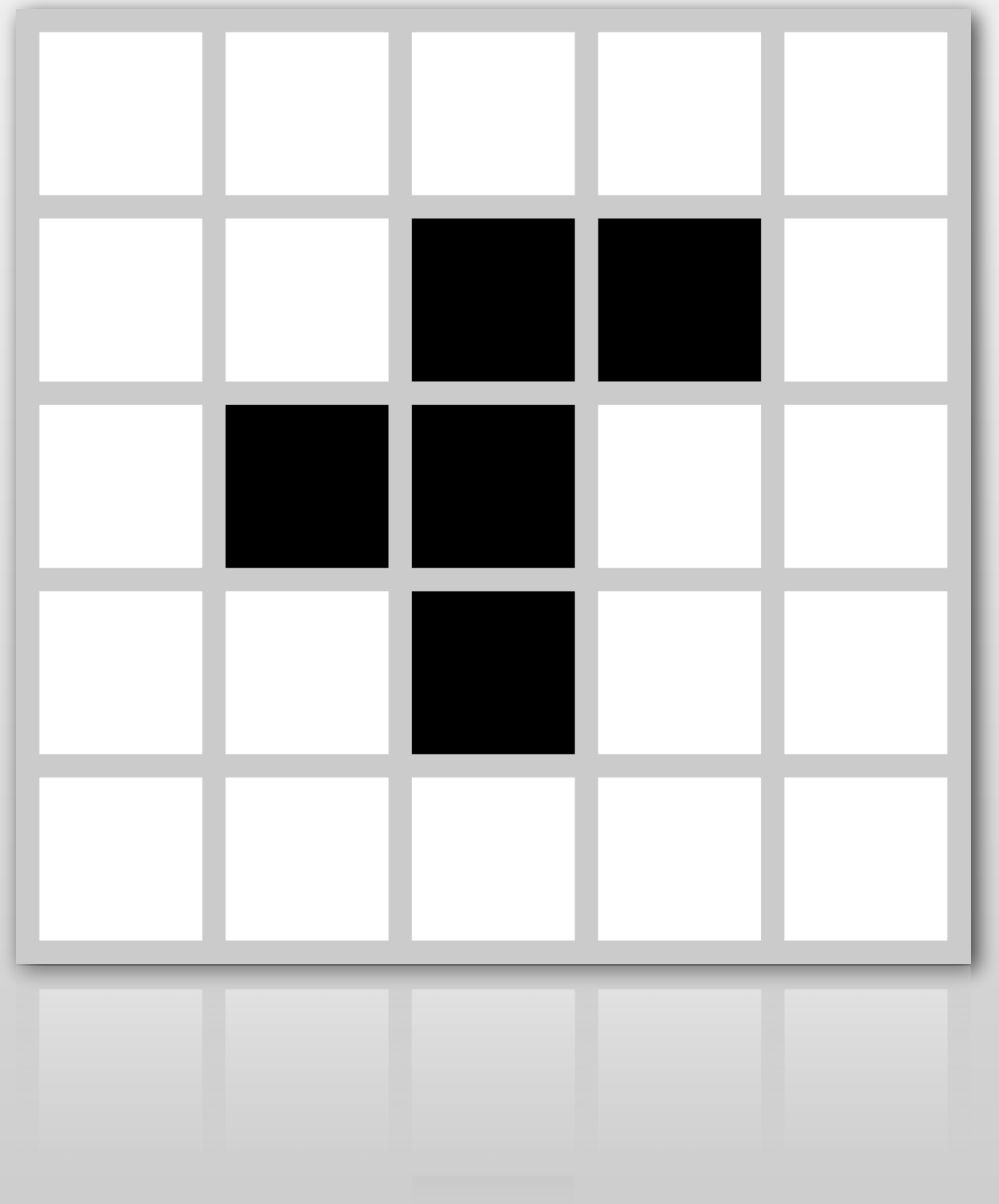
A factory that creates values
used to test
specific code paths


Default
80 %
Customize
20 %

```
public void Fork(int value)
{
    if (value > 9)
    {
        // Do this
    }
    else
    {
        // Do that
    }
}
```

A small positive number is good enough

To cover most possible code paths



An implementation of
Conway's Game of Life
written in C#

3 takeaways

No magic values

Less coupling

Easier maintenance

stackoverflow.com/tags/autofixture

github.com/autofixture

Thank you.

@ecampidoglio