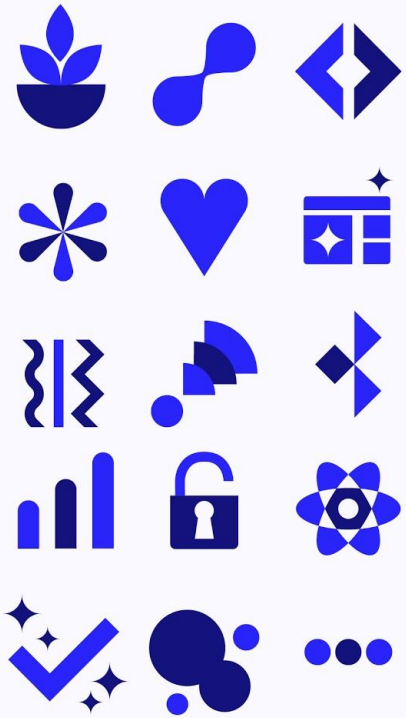


Polidea

Testowanie jednostkowe

Jak robić to dobrze?

Szymon Przedwojski

Senior Software Engineer @ Polidea

- umniedziala.com
- pythonowiec.pl
- facebook
- twitter
- instagram
- tiktok


Story Time


BFMP

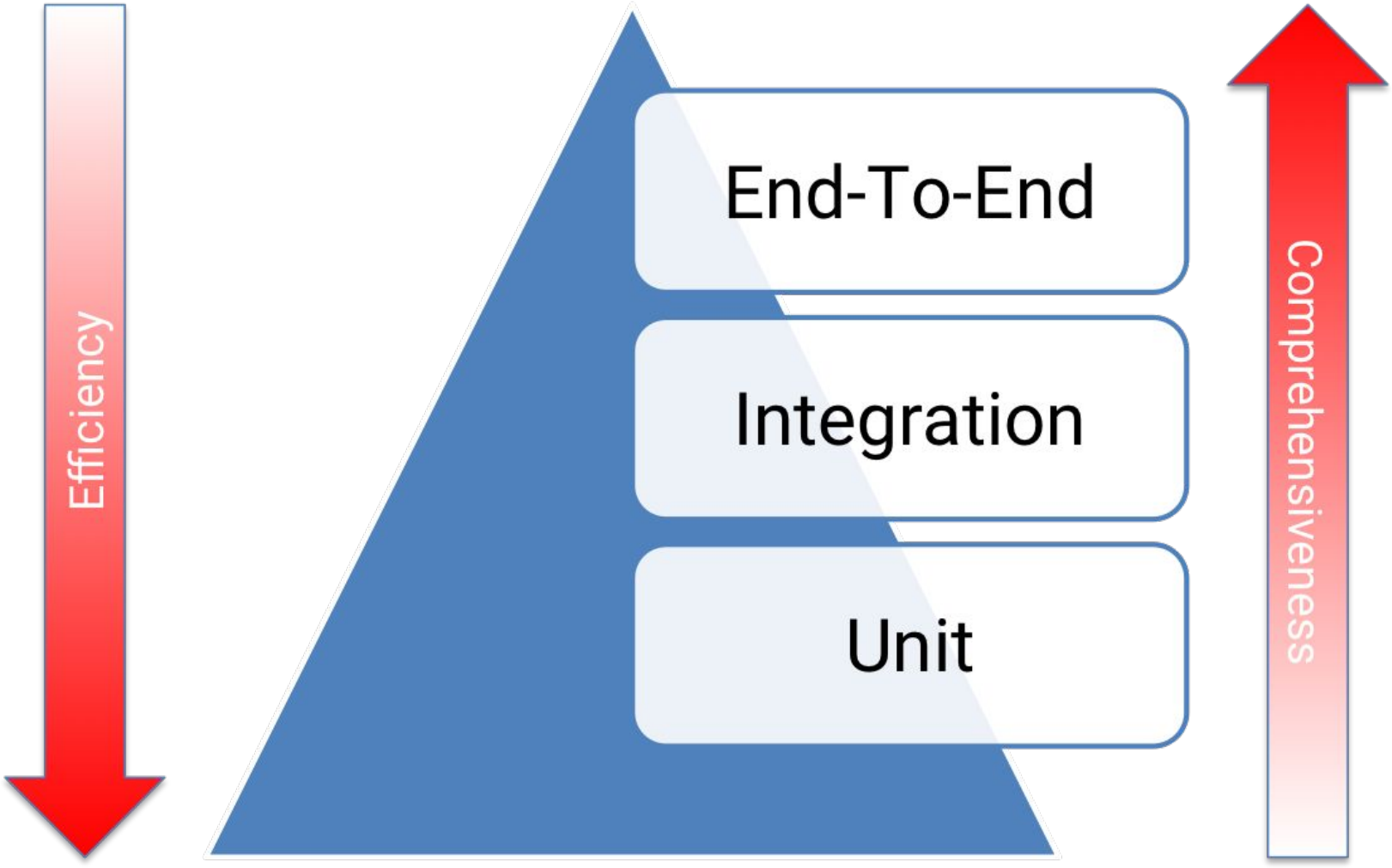

Look how they massacred my boy.

Testowanie jednostkowe

👉 **Dlaczego warto?**

👉 **Jak robić to dobrze?**

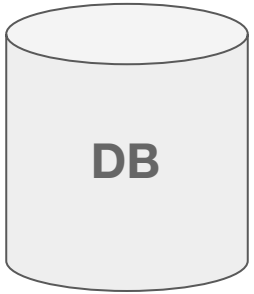
👉 **Jak przekonać innych?**

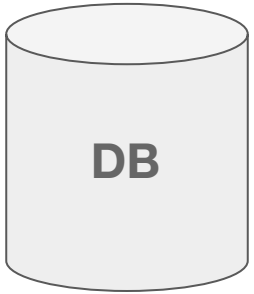

Efficiency
End-To-End
Integration
Unit
Comprehensiveness

```python
class TaskStatus(Enum):
    TODO = 1
    IN_PROGRESS = 2
    CODE_REVIEW = 3
    QA = 4
    DONE = 5


class Task:
    def __init__(self, name: str, status: TaskStatus):
        self._name: str = name
        self._status: TaskStatus = status
```

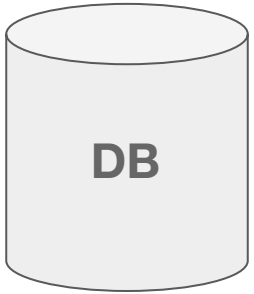
```python
class Task:
    def __init__(self, name: str, status: TaskStatus):
        self._name: str = name
        self._status: TaskStatus = status

    def change_status(self, new_status: TaskStatus):
        self._status = new_status

    @property
    def status(self):
        return self._status
```

```
def test_status_change():
    # Given
    task = Task("Prepare presentation", TaskStatus.TODO)
    # When
    task.change_status(TaskStatus.CODE_REVIEW)
    # Then
    assert TaskStatus.CODE_REVIEW == task.status
```

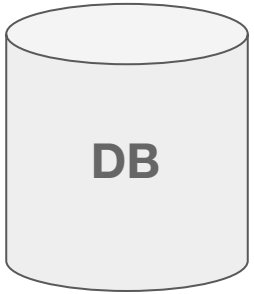

DB



→ **Integracyjne**

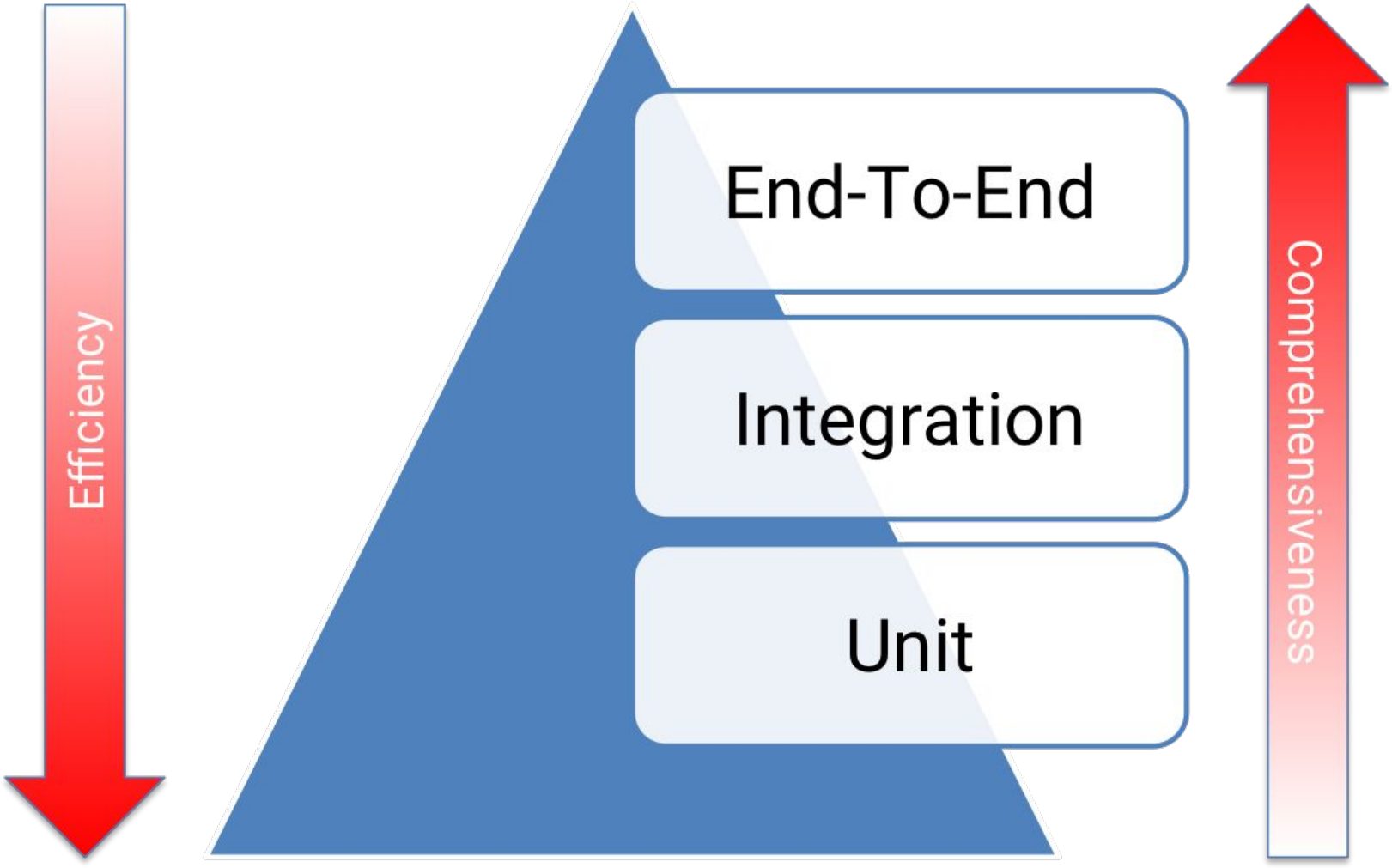


→ **Integracyjne**

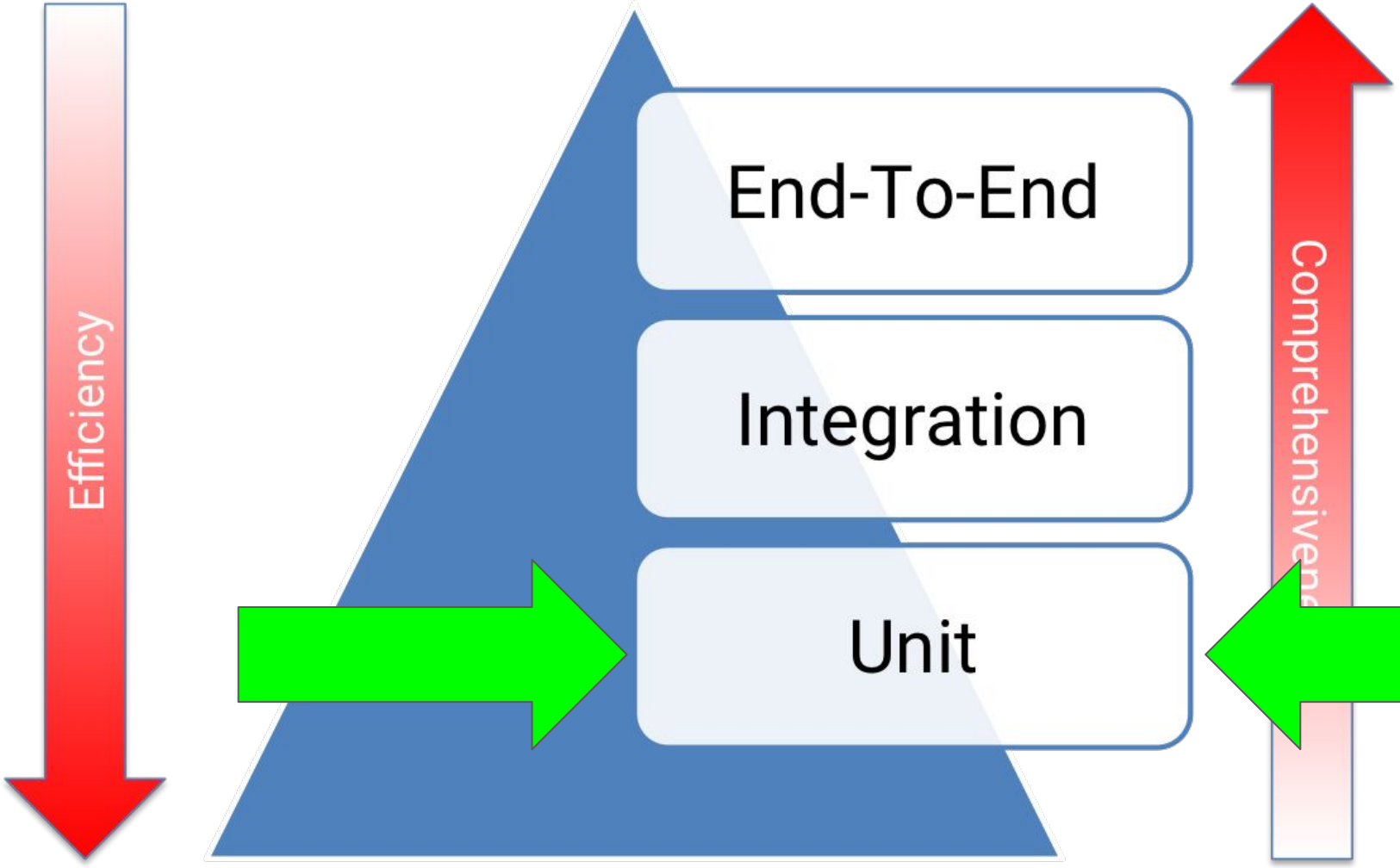


→ **Integracyjne**

→ **End-to-end**


Efficiency
End-To-End
Integration
Unit
Comprehensiveness

Jednostkowe: 100%, integracyjne: 0%


Efficiency
End-To-End
Integration
Unit
Comprehensiven

Dlaczego warto?

1. Poprawność kodu

Żeby sprawdzić, czy nasz kod działa tak, jak nam się wydaje

Żeby sprawdzać, czy nasz kod działa tak, jak nam się wydaje

Żeby sprawdzać, czy nasz kod działa tak, jak nam się wydaje

⇒ Regresja ⇐

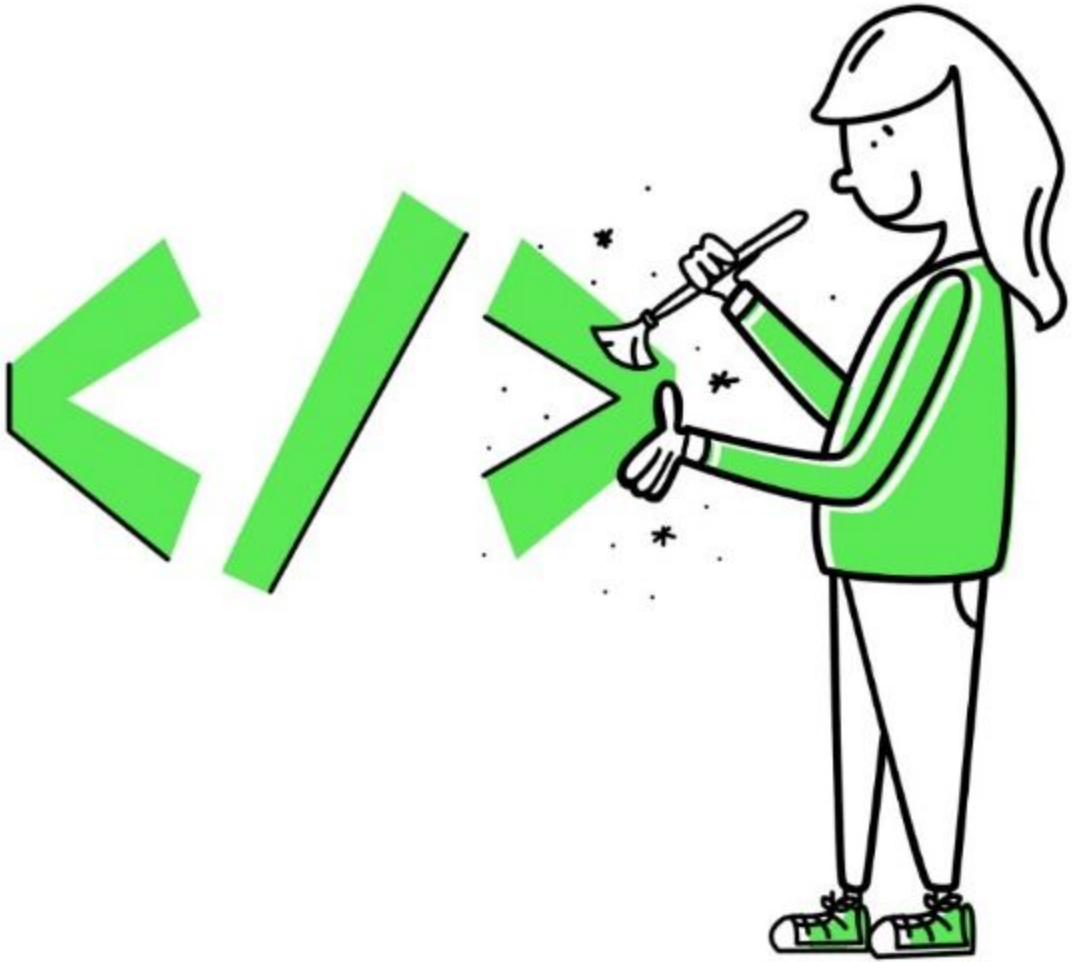
1. Poprawność kodu

2. Łatwość zmiany

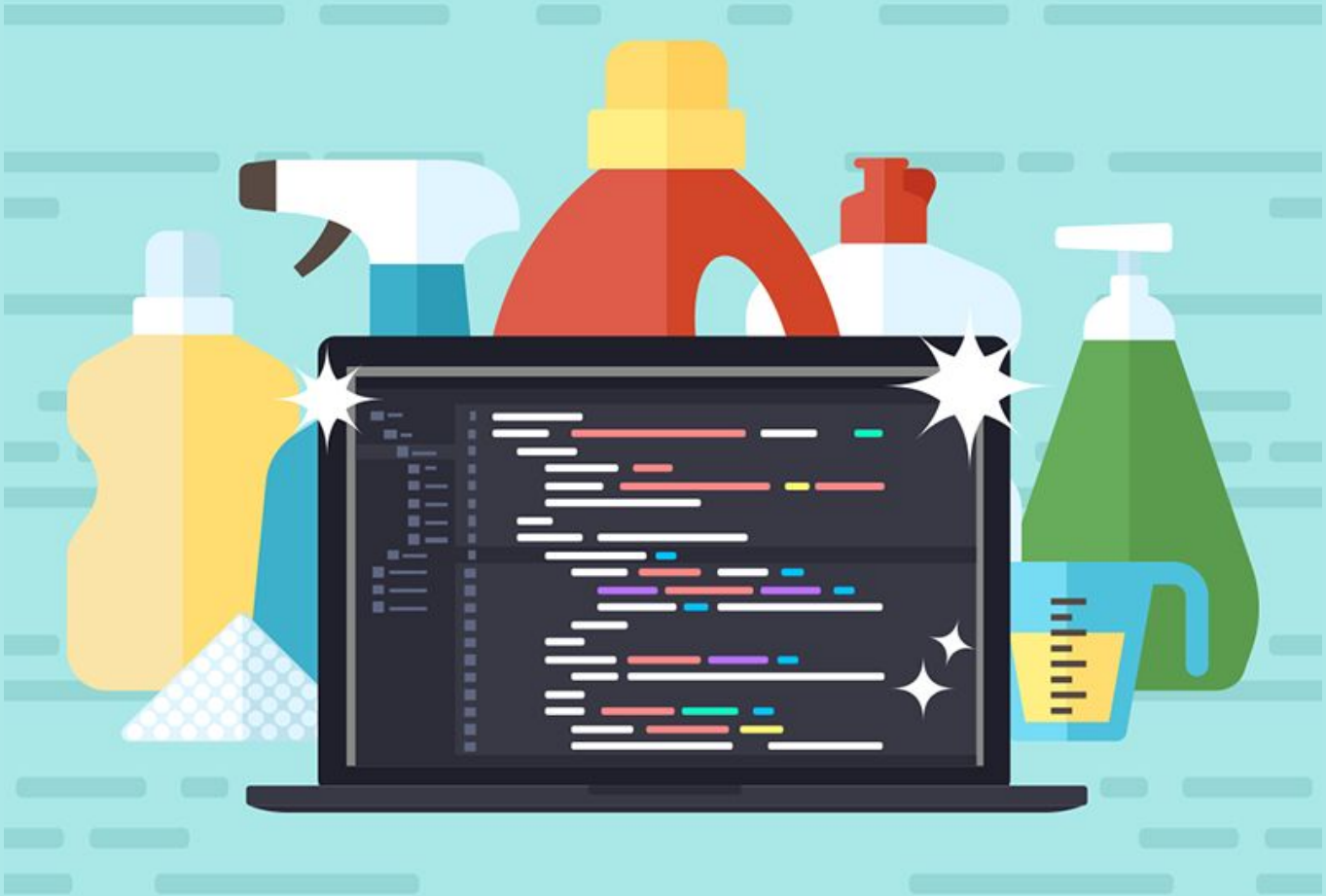

Nothing is constant except change.
Heraclitus
quotefancy

1. Poprawność kodu
2. Łatwość zmiany
3. **Refaktoring**


LEGACY CODE



1. Poprawność kodu
2. Łatwość zmiany
3. Refaktoring
4. **Testowalny kod == “czysty kod”**



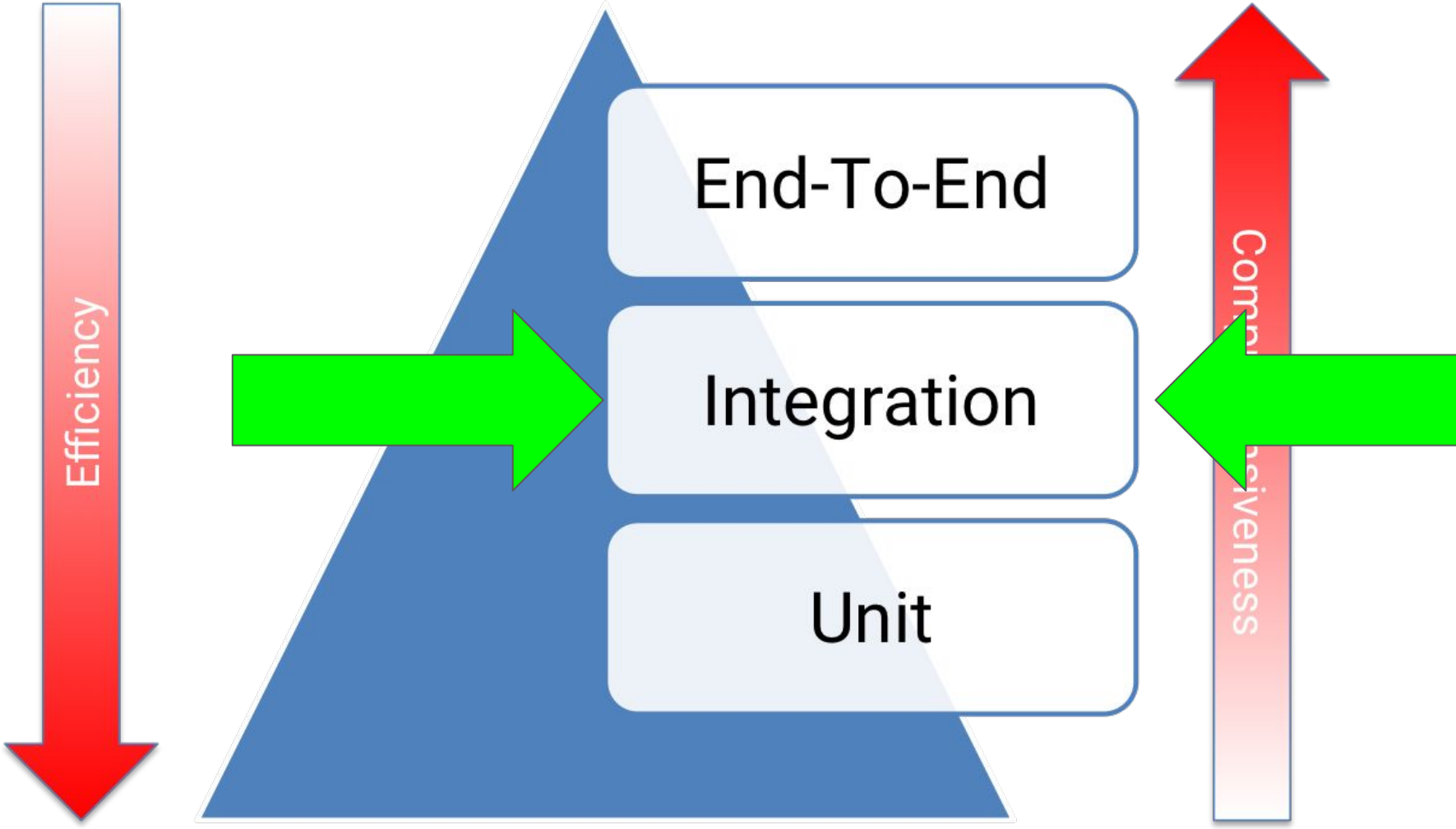
1. Poprawność kodu
2. Łatwość zmiany
3. Refaktoring
4. Testowalny kod == “czysty kod”
5. **Uruchamialna dokumentacja**

1. Poprawność kodu
2. Łatwość zmiany
3. Refaktoring
4. Testowalny kod == “czysty kod”
5. Uruchamialna dokumentacja
6. **Niska bariera wejścia**


WARNING
LOW
BARRIER

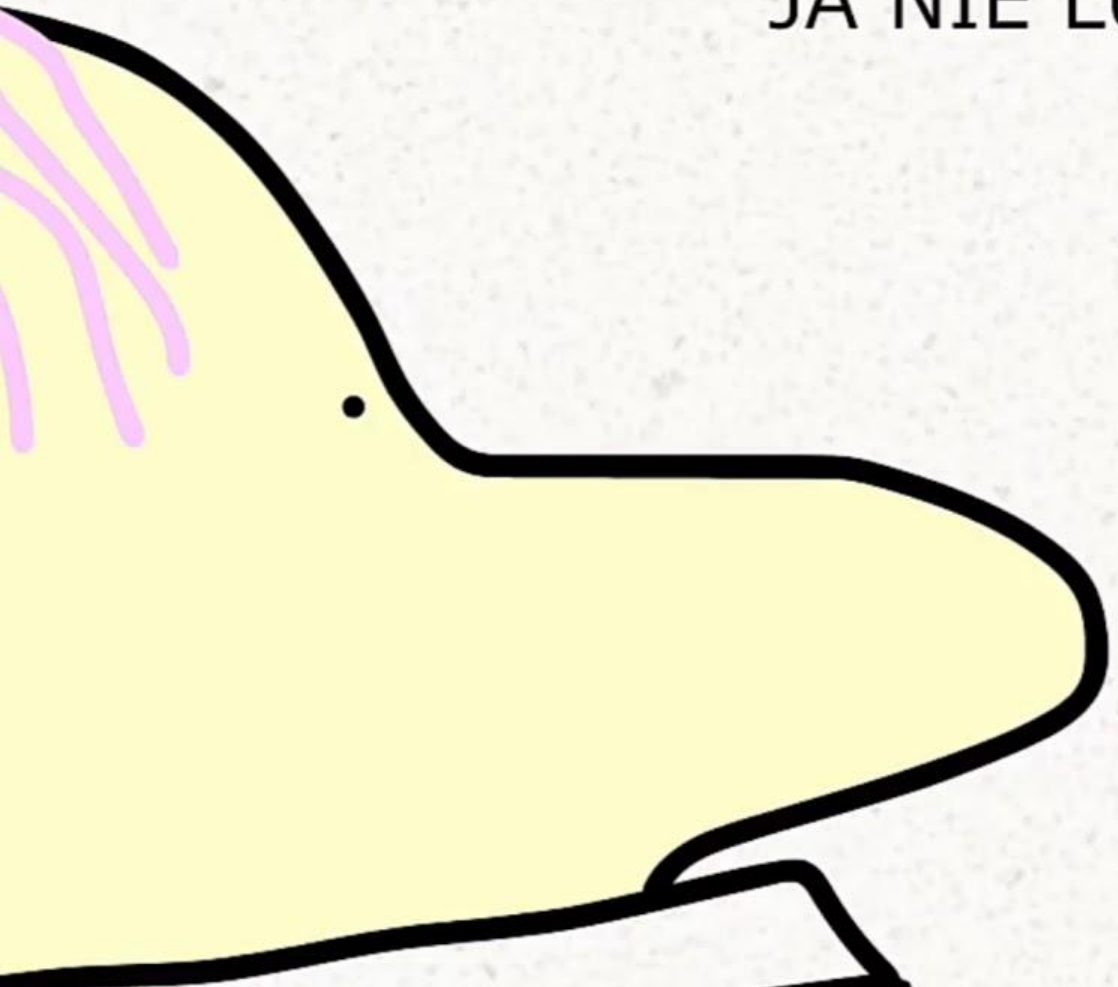
Jak robić to dobrze?


ATN_STAGE_FK (FK)
ATN_STAGE_MODIFIER_FK (FK)
ATN_PUBLIC_ID
asyn_obj
ANA_OBJECT
OBJ_OID
OBJ_CREATION_DATETIME
OBJ_CREATOR_FK
arel_obj
ANA_RELATIONSHIP
REL_OID (FK)
REL_CHILD_FK (FK
REL_PARENT_FK (
REL_SEQUENCE
ANO_OID (FK)
ANO_SPECIES_FK (FK)
ANO_COMPONENT_NAME
ANO_IS_PRIMARY
ANO_IS_GROUP
ANO_PUBLIC_ID
ANO_DESCRIPTION
RSP_NAME
RSP_LATIN_NAME
RSP_TIMED_NODE_ID_PREFIX
RSP_NODE_ID_PREFIX
aeditor_obj
attrib_obj
ANA_STAGE
STG_OID (FK)
STG_SPECIES_FK (FK)
STG_NAME
STG_SEQUENCE
STG_DESCRIPTION
ANA_ATTRIBUTION
ATR_OID (FK)
ATR_OBJECT_FK (FK)
ATR_SOURCE_FK (FK)
ATR_EVIDENCE_FK (FK)
asrc_obj
ANA_SOURCE
SRC_OID (FK)
SRC_NAME


Efficiency
End-To-End
Integration
Unit
Comp
iveness

1. Szybkie testy jednostkowe

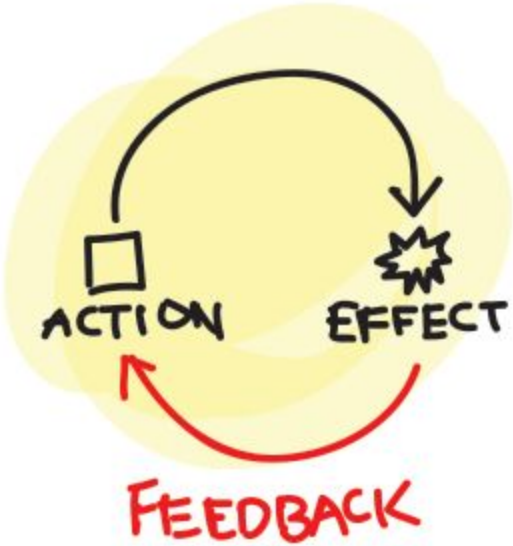
JA NIE LUBJE SZYPKO



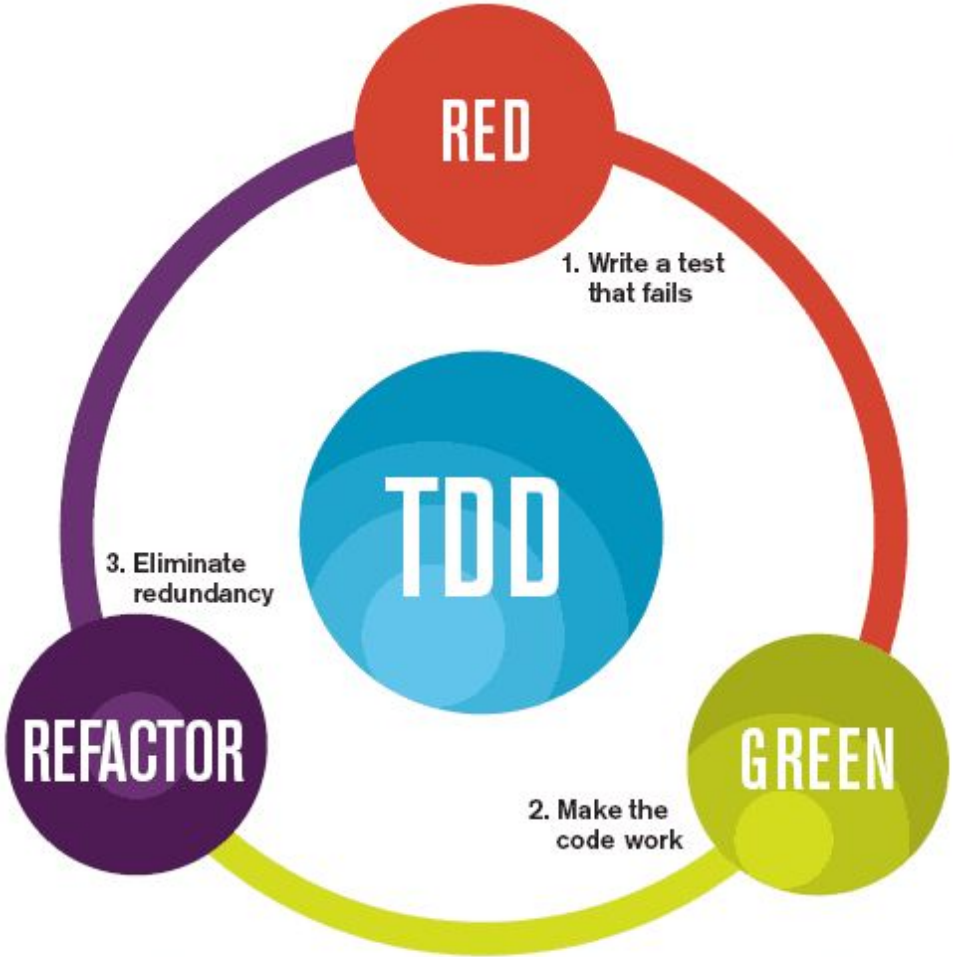
JA LUBIE
WOLMO

Dlaczego testy jednostkowe muszą być szybkie?

Quick feedback loop


ACTION
EFFECT
FEEDBACK

Test Driven Development

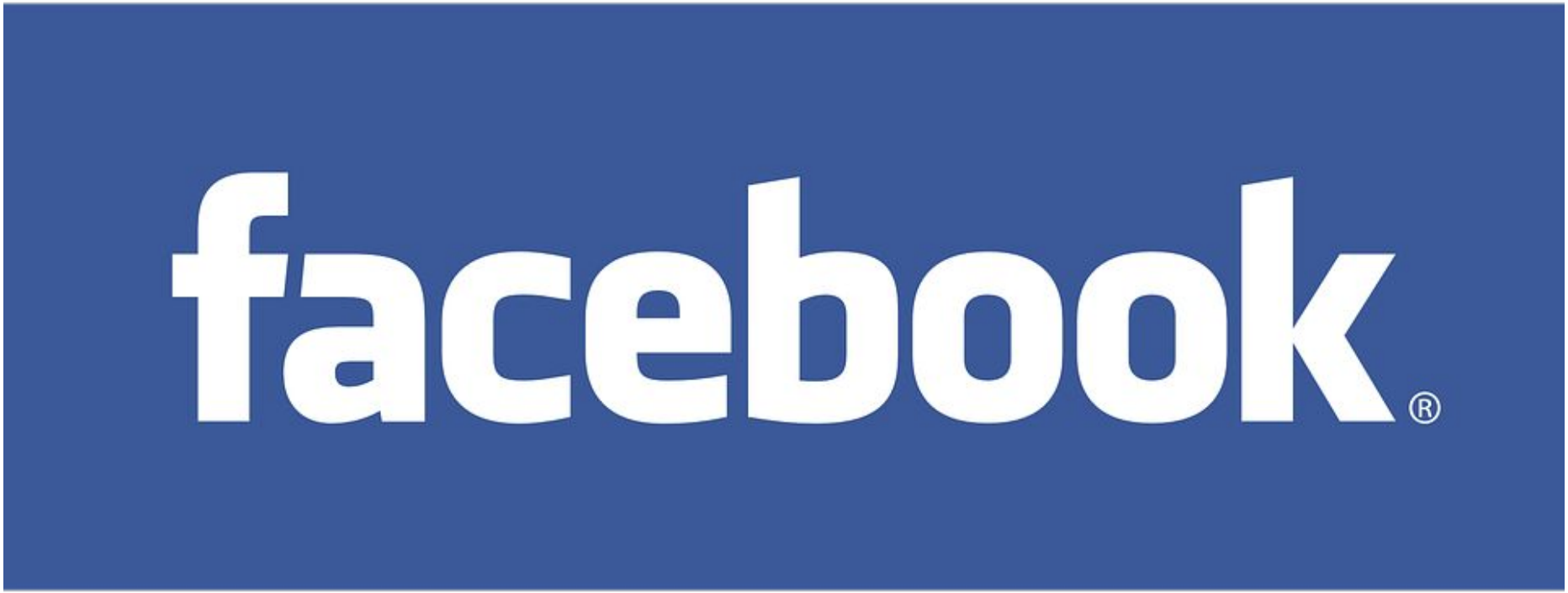


The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Jak szybkie?

- us?
- ms?
- s?
- min?

Zdroworozsądkowo...


facebook®

Zachowanie “flow”

Jak to osiągnąć?

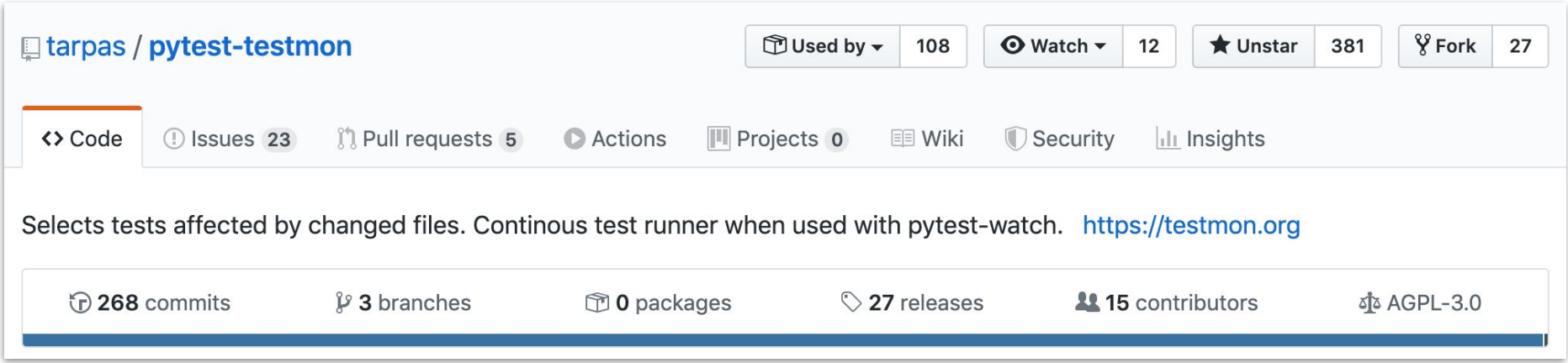
a) Test Doubles

- **Fake**
- **Mock**
- **Stub**
- **Spy**

b) Poprawna architektura

- **Modularyzacja**
- **Trójwarstwowa**
- **Heksagonalna**
- **...**

c) Nie cały *suite*, a część


tarpas / pytest-testmon
Used by 108
Watch 12
Unstar 381
Fork 27
Code
Issues 23
Pull requests 5
Actions
Projects 0
Wiki
Security
Insights
Selects tests affected by changed files. Continous test runner when used with pytest-watch. https://testmon.org
268 commits
3 branches
0 packages
27 releases
15 contributors
AGPL-3.0

d) Wielowątkowość

pytest-xdist 1.31.0

```
pip install pytest-xdist
```

pytest xdist plugin for distributed testing and loop-on-failing modes

1. Szybkie testy jednostkowe

2. Uruchamiaj je często

- **Automatycznie**
  - **PyCharm**
  - **pytest-watch**
- **Ręcznie skrótem**
  - **⇧F10 // ^F5**

Serwer CI → za późno!!

1. Szybkie testy jednostkowe
2. Uruchamiaj je często
3. **Testuj publiczny kontrakt**

Jeśli nie...

Testujemy “za nisko”

Kod staje się ”sztywny”

```python
class TaskStatus(Enum):
    TODO = 1
    IN_PROGRESS = 2
    CODE_REVIEW = 3
    QA = 4
    DONE = 5


class Task:
    def __init__(self, name: str, status: TaskStatus):
        self._name: str = name
        self._status: TaskStatus = status
```

```python
class Task:
    def __init__(self, name: str, status: TaskStatus):
        self._name: str = name
        self._status: TaskStatus = status

    def change_status(self, new_status: TaskStatus):
        self._status = new_status

    @property
    def status(self):
        return self._status
```

```
def test_status_change():
    # Given
    task = Task("Prepare presentation", TaskStatus.TODO)
    # When
    task.change_status(TaskStatus.CODE_REVIEW)
    # Then
    assert TaskStatus.CODE_REVIEW == task.status
```

```
class Task:
    def __init__(self, name: str, status: TaskStatus):...

    def change_status(self, new_status: TaskStatus):
        if self._is_move_possible(new_status):
            self._status = new_status
        else:
            raise TaskError
```

```python
def test__is_move_possible():
    # Given
    task = Task("Prepare presentation", TaskStatus.TODO)
    # When
    result = task._is_move_possible(TaskStatus.DONE)
    # Then
    assert not result
```

```python
def test__is_move_possible():
    # Given
    task = Task("Prepare presentation", TaskStatus.TODO)
    # When
    result = task._is_move_possible(TaskStatus.DONE)
    # Then
    assert not result
```

```python
def test_invalid_status_change():
    # Given
    task = Task("Prepare presentation", TaskStatus.TODO)
    # Then
    with pytest.raises(TaskError):
        task.change_status(TaskStatus.DONE)
```

```python
@pytest.mark.parametrize('initial_status,to_status', [
    (TaskStatus.TODO, TaskStatus.DONE),
    # ...
])
def test_invalid_status_change(initial_status, to_status):
    # Given
    task = Task("Prepare presentation", initial_status)
    # Then
    with pytest.raises(TaskError):
        task.change_status(to_status)
```

1. Szybkie testy jednostkowe
2. Uruchamiaj je często
3. Testuj publiczny kontrakt
4. Zawsze Red → Green

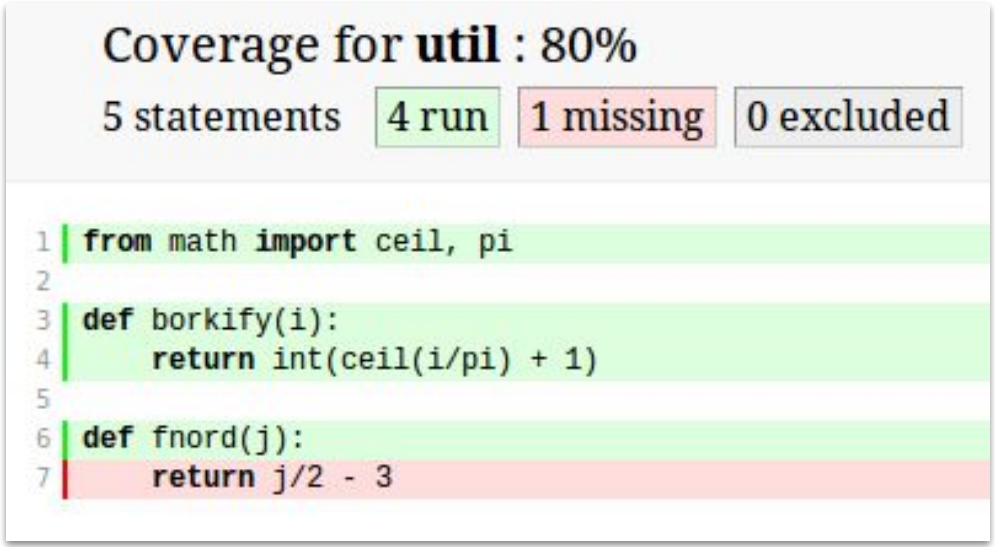
“False positives”

Błąd: nowy “if” w kodzie...

...a test bez zmian

```
class Task:
    def __init__(self, name: str, status: TaskStatus):...

    def change_status(self, new_status: TaskStatus):
        if self._is_move_possible(new_status):
            self._status = new_status
        else:
            raise TaskError
```

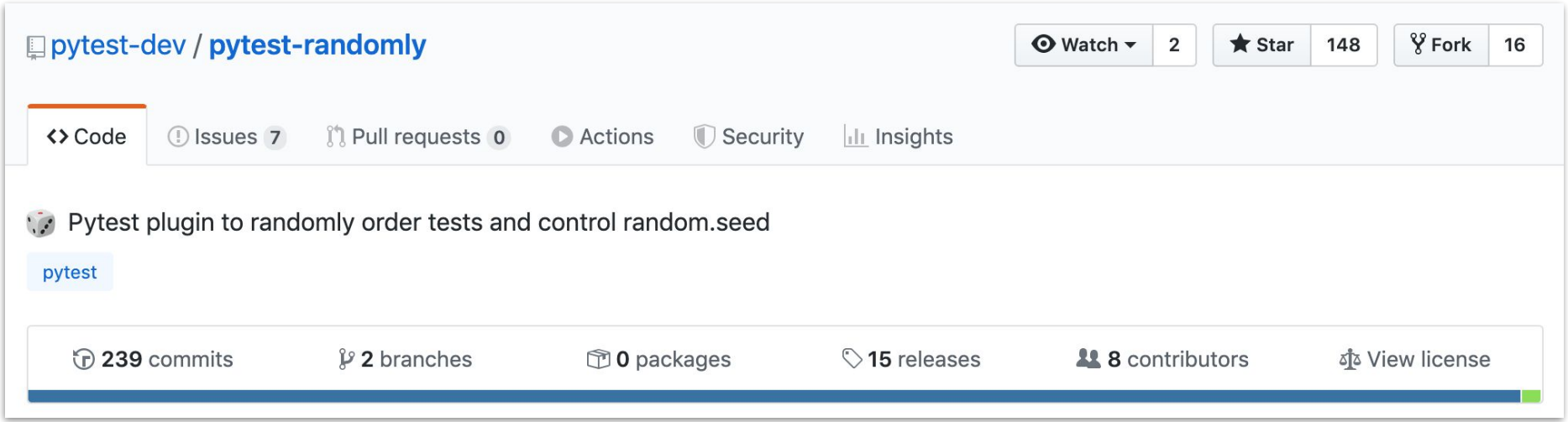
Code coverage


Coverage for **util** : 80%
5 statements
4 run
1 missing
0 excluded
1 from math import ceil, pi
2
3 def borkify(i):
4 return int(ceil(i/pi) + 1)
5
6 def fnord(j):
7 return j/2 - 3

- **PyCharm**
- **coveragepy**

1. Szybkie testy jednostkowe
2. Uruchamiaj je często
3. Testuj publiczny kontrakt
4. Zawsze Red → Green
5. **Testy w losowej kolejności**

Zależność od side effectów


pytest-dev / pytest-randomly
Watch 2
Star 148
Fork 16
Code
Issues 7
Pull requests 0
Actions
Security
Insights
Pytest plugin to randomly order tests and control random.seed
pytest
239 commits
2 branches
0 packages
15 releases
8 contributors
View license

1. Szybkie testy jednostkowe
2. Uruchamiaj je często
3. Testuj publiczny kontrakt
4. Zawsze Red → Green
5. Testy w losowej kolejności
6. Dbaj o jakość testów

Testy to też kod!

Musi być...

Czytelny

Przejrzysty

Zrozumiały

Cechy dobrych testów:

a) Given, When, Then

```
def test_status_change():
    # Given
    task = Task("Prepare presentation", TaskStatus.TODO)
    # When
    task.change_status(TaskStatus.CODE_REVIEW)
    # Then
    assert TaskStatus.CODE_REVIEW == task.status
```

b) Mały setup

pytest fixtures

c) Zasady Clean Code

Dobre nazwy

Wydzielanie metod

etc.

Jak przekonać innych?

1. Koledzy z zespołu


WARTO
ROZMAWIAĆ

Co stoi na przeszkodzie?

- Czy są umiejętności?
- Czy korzyści są jasne?
- Złe doświadczenia z przeszłości?

1. Koledzy z zespołu
2. Interesariusze

1. Koledzy z zespołu
2. Interesariusze
    - Manager / PO
    - Biznes
    - Klient


55
5
10
50
15
45
20
40
25
35
30

Są dwie drogi

a) Metryki

[YouTube] [Unit Testing and TDD: Why you should care and how to make it happen]

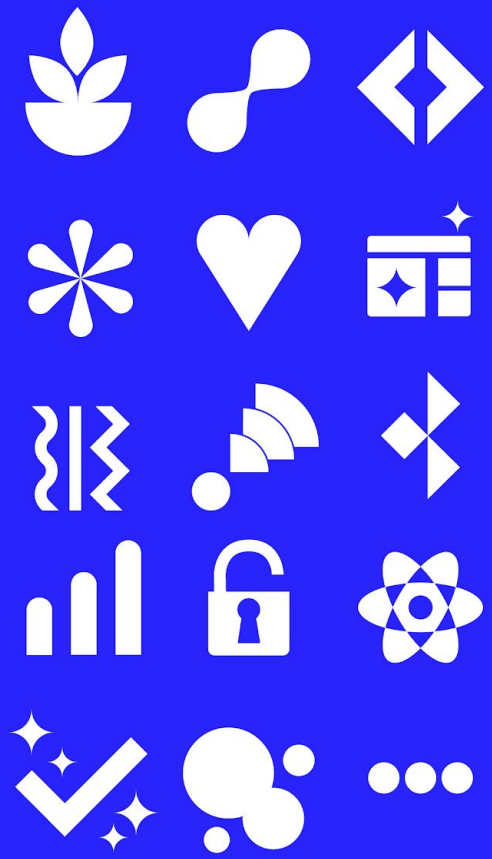
b) Nie pytać o pozwolenie


TIME OUT

Do brzegu

Testowanie jednostkowe

- 👉 Dlaczego warto? ✔️
- 👉 Jak robić to dobrze? ✔️
- 👉 Jak przekonać innych? ✔️



Thanks!

Polidea

szymon.przedwojski@polidea.com

szymon@umniedziala.com

Szymon Przedwojski

- umniedziala.com
- pythonowiec.pl
- facebook
- twitter
- instagram
- tiktok

Polidea

- blog
- twitter
- linkedin

Prezentacja:

pythonowiec.pl/pywaw