

# Automatycznie testowalne aplikacje w środowisku rozproszonym

czyli jak testować mikro serwisy (i nie tylko)

Piotr Litwiński

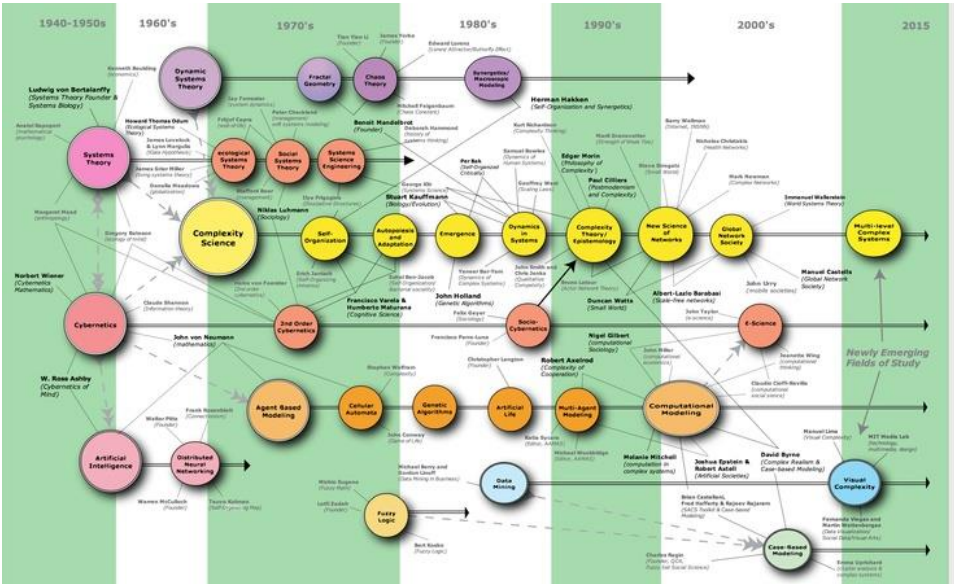
Software Engineer, NewVoiceMedia

github: <https://github.com/plitwinski>

# Agenda

- W czym tkwi problem?
- Rodzaje testów
- Jakich narzędzi użyć?
- Jak pisać testy automatyczne?
- Jak testować testy?
- Podsumowanie - wady i zalety

# W czym tkwi problem?



# Rodzaje testów



# W trakcie budowania

- Jednostkowe
- Kontraktowe
- Serwisowe/Komponentowe

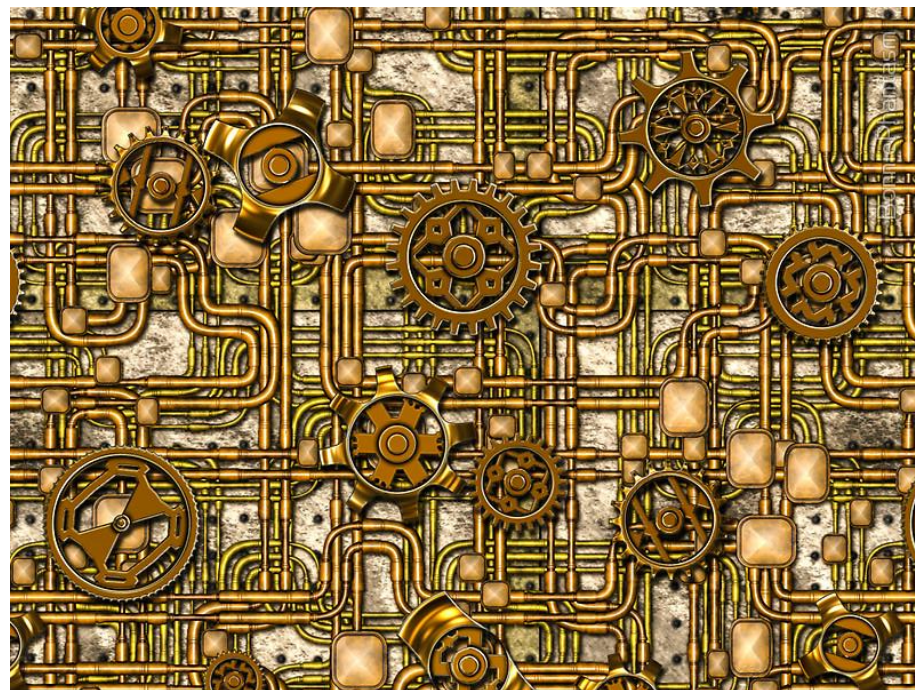
# Środowiskowe

- Deployed / integracyjne
- E2E
- Eksploracyjne

# Systemowe

- Wydajnościowe
- Odpornościowe/Dostępnościowe (resilience/availability)

# Jakich narzędzi możemy użyć?





# Narzędzia dla developera

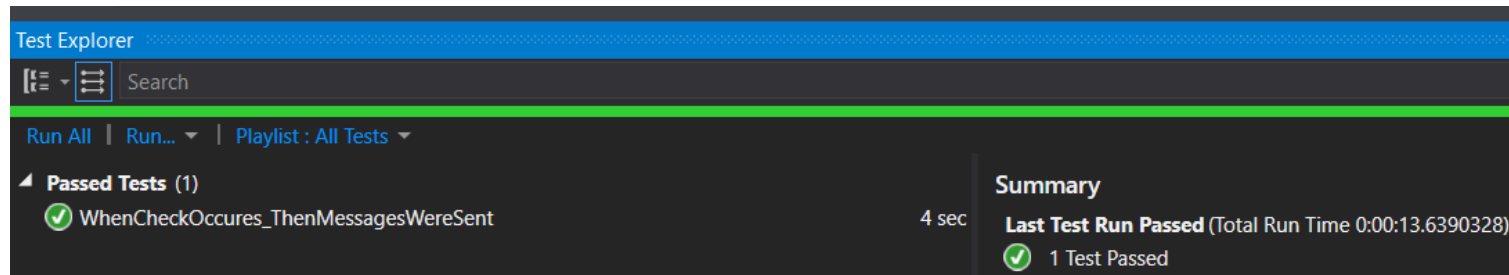
- Narzędzia do wykorzystania wewnątrz procesu (w pamięci)
  - Stubs & mocks
  - Coarse grained „unit” tests
  - Zdefiniowane kontrakty
  - Odpowiedniki odpalane w pamięci (np.: in-memory databases, test server)
- Narzędzia do wykorzystania na agencie do budowania
  - Weryfikacja kontraktów
  - Web driver + (Headless) browsers
  - Lokalne odpowiedniki (np.: local sql, json server, etc.)

# Narzędzia dla developera/opsa

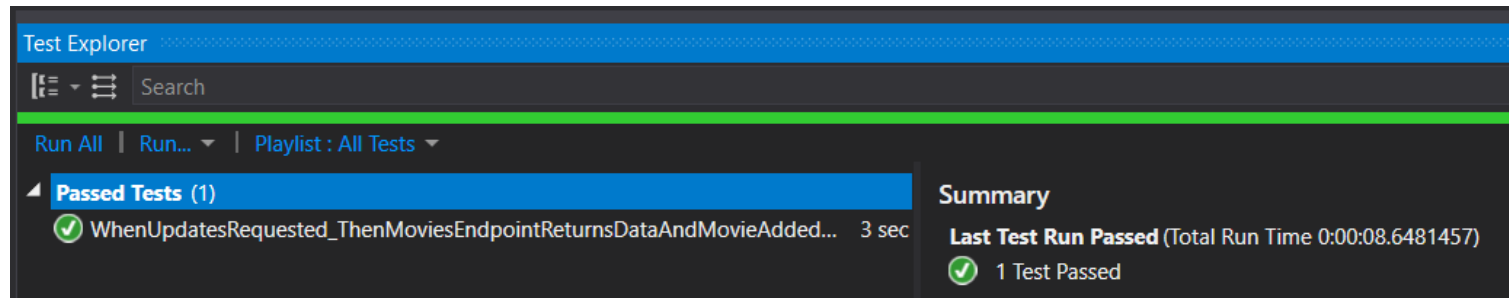
- Narzędzia do wykorzystania podczas uruchomienia w danym środowisku
  - Agent CD uruchamia kod z testami, które sprawdzają integrację z zewnętrznymi serwisami (np.: sprawdzenie endpoint'u, etc)
  - Uruchomienie diagnostyki zależności przy starcie serwisu

# Narzędzia dla developera – przykłady 1,2

- Coarse grained unit tests



- InMemory database and test server



# Narzędzie dla developera – przykład 3

- Selenium WebDriver + Http Server + Json Server + headless Firefox
  - Kroki:
    - npm run startWebServer (w osobnym procesie)
    - npm run startApi (w osobnym procesie)
    - npm run selenium-test (w osobnym procesie)

```
PS C:\Projects\presentations\automated-tests-presentation\examples\03 - local-headless json-server> npm run selenium-test
> example_3@0.0.1 selenium-test C:\Projects\presentations\automated-tests-presentation\examples\03 - local-headless json-server
> jest

PASS src\webdriver.test.js
  ✓ make sure movies list has been rendered (3417ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        6.097s, estimated 12s
Ran all test suites.
```

# Narzędzia dla ops'a

- Pipelines
- Monitoring
- Dashboards

# Jak pisać testy automatyczne?



- Wybór strategii – co chcemy osiągnąć poprzez napisanie testów autoamtycznych?
- Wybór metodologii – testowanie funkcji/klas/komponentow vs testowanie scenariuszowe
- Planowanie – zanim zaczniemy pisać kod, zastanówmy się jakiego rodzaju testy będą nam potrzebne
- Modułowość – budowanie z „klocków”

# Metodologia – przykład 4

Test Explorer

Search

Run All | Run... | Playlist : All Tests

▲ **Passed Tests (5)**

✓ ThenCinema1IsReturned	71 ms
✓ ThenCinema1MoviesCollectionIsNotEmpty	31 ms
✓ ThenCinema2IsReturned	1 ms
✓ ThenCinema2MoviesCollectionIsNotEmpty	1 ms
✓ ThenTwoEventsWerePublished	90 ms

**Summary**

**Last Test Run Passed** (Total Run Time 0:00:07.5563702)

✓ 5 Tests Passed



# Modułowość - przykład 5

Test Explorer

Search

Run All | Run... | Playlist : All Tests

Passed Tests (10)

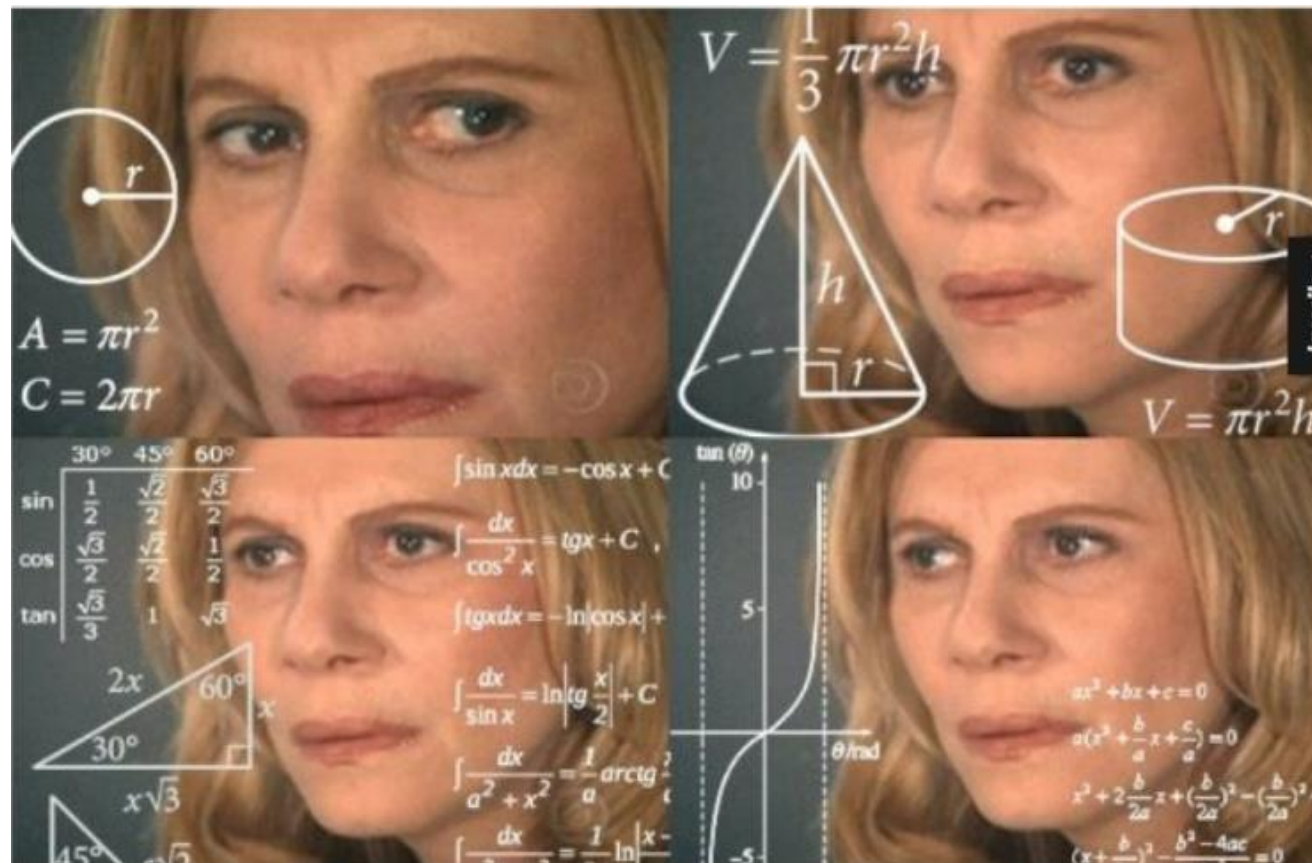
✓ ThenCinema1IsReturned	70 ms
✓ ThenCinema1IsReturned	1 ms
✓ ThenCinema1MoviesCollectionIsNotEmpty	26 ms
✓ ThenCinema1MoviesCollectionIsNotEmpty	1 ms
✓ ThenCinema2IsReturned	1 ms
✓ ThenCinema2IsReturned	1 ms
✓ ThenCinema2MoviesCollectionIsNotEmpty	1 ms
✓ ThenCinema2MoviesCollectionIsNotEmpty	1 ms
✓ ThenTwoEventsWerePublished	85 ms
✓ ThenTwoEventsWerePublished	4 ms

Summary

Last Test Run Passed (Total Run Time 0:00:07.5279641)

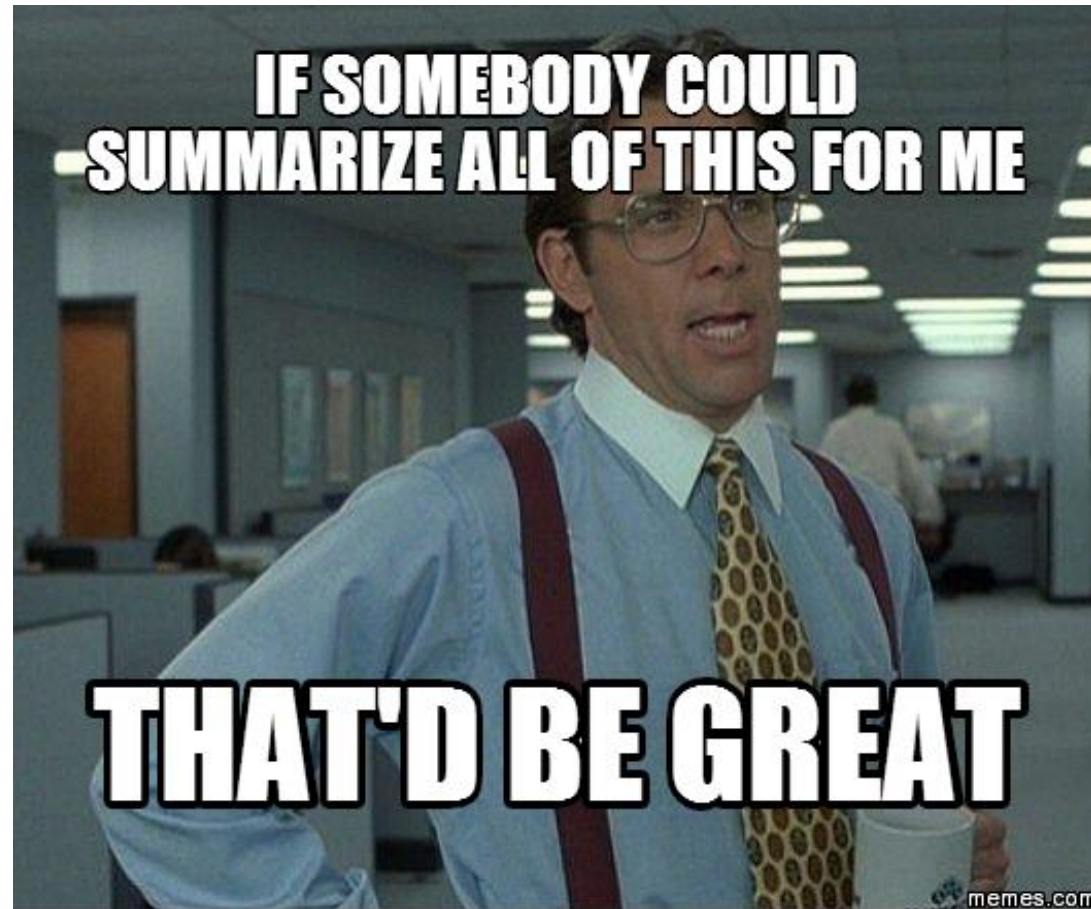
✓ 10 Tests Passed

# Jak testować testy?



- Planowanie
- Code reviews
- Ilość wprowadzeń nowych wersji na produkcję vs wycofanych wersji spowodowanych błędami implementacji
- Test coverage
- Mutation tests (testy mutacyjne)

# Podsumowanie – trudności i zalety



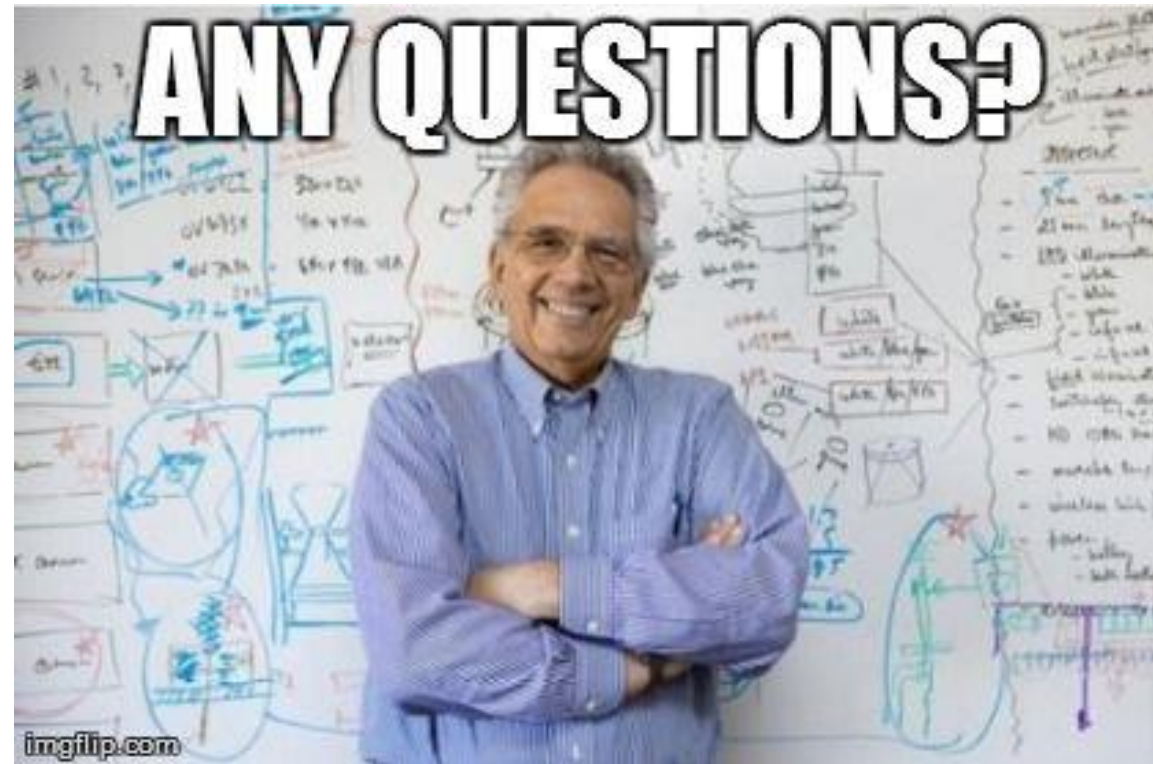
# Trudności

- Standaryzacja i automatyzacja procesów na poziomie całej firmy
- Wprowadzanie większych zmian może być bardziej złożone
- Koszt - czas, narzędzia, krzywa nauki, szkolenia, etc. w krótkim okresie czasu

# Zalety

- Oszczędności przy regresji
- Wychwytywanie błędów znacznie wcześniej
- Znacznie większa pewność przy refactoring'u i wprowadzaniu zmian
- Standaryzacja na poziomie firmy

Pytania?



# Przydatne linki

<https://github.com/plitwinski/automated-tests-presentation>

<https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116>

[https://en.wikipedia.org/wiki/Mutation\\_testing](https://en.wikipedia.org/wiki/Mutation_testing)

<https://martinfowler.com/articles/microservice-testing>