Automatycznie testowalne aplikacje w środowisku rozproszonym

czyli jak testować mikro serwisy (i nie tylko)

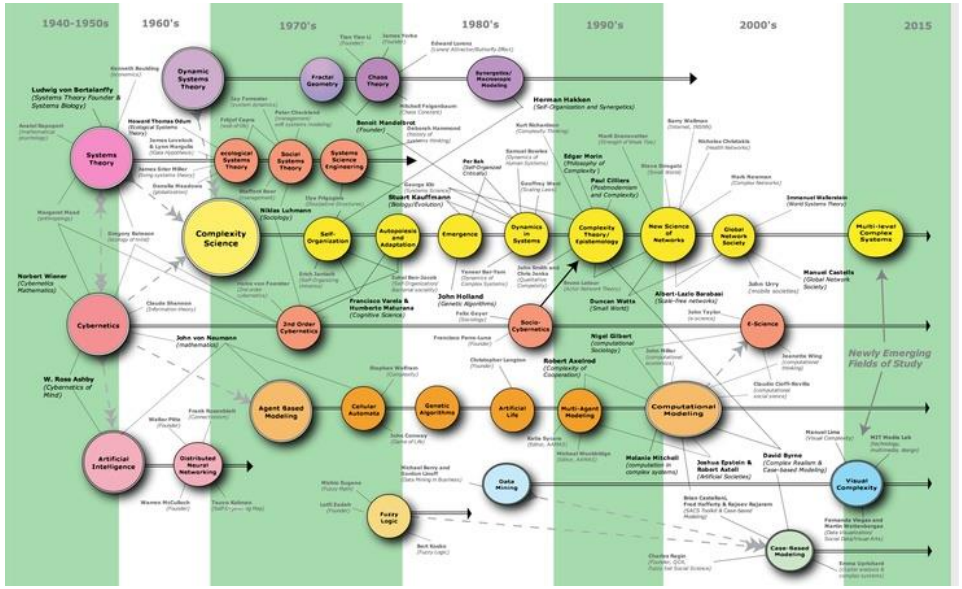
Piotr Litwiński

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
- Klasyczne testy integracyjne

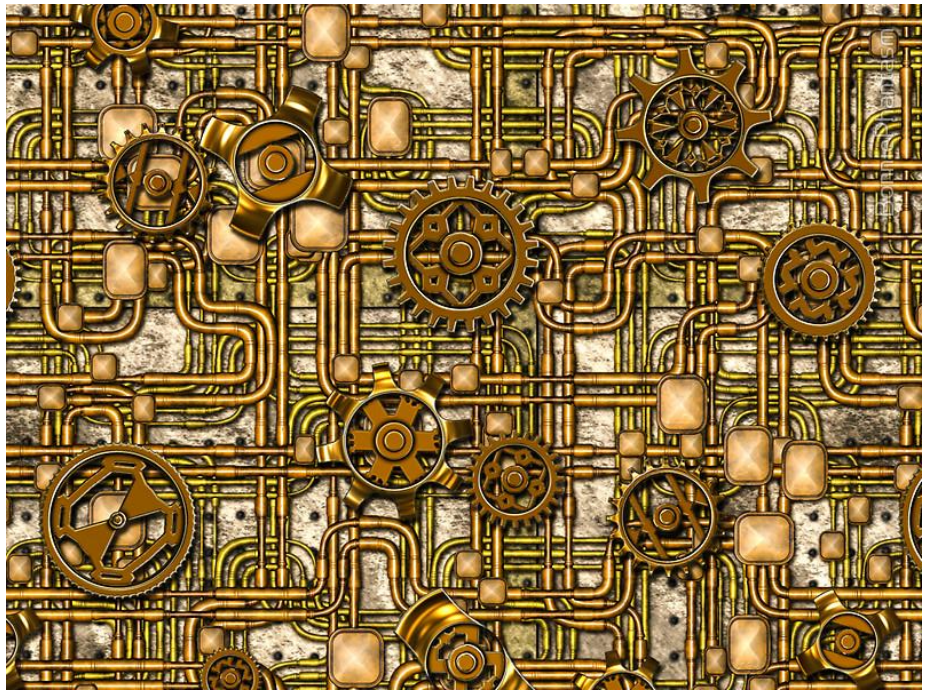
Środowiskowe

- Deployed / integracyjne
- E2E
- Eksporacyjne

Systemowe

- Wydajnościowe
- Odpornościowe/Dostępnościowe (resilience/availability)

Jakich narzędzi możemy użyć?



Narzędzia dla developera

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

Narzędzia dla developera – przykłady

- Coarse grained unit tests
- InMemory database and test server
- Selenium WebDriver + Http Server + Json Server + headless Firefox

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 ops’a

- Pipelines
- Monitoring
- Dashboards

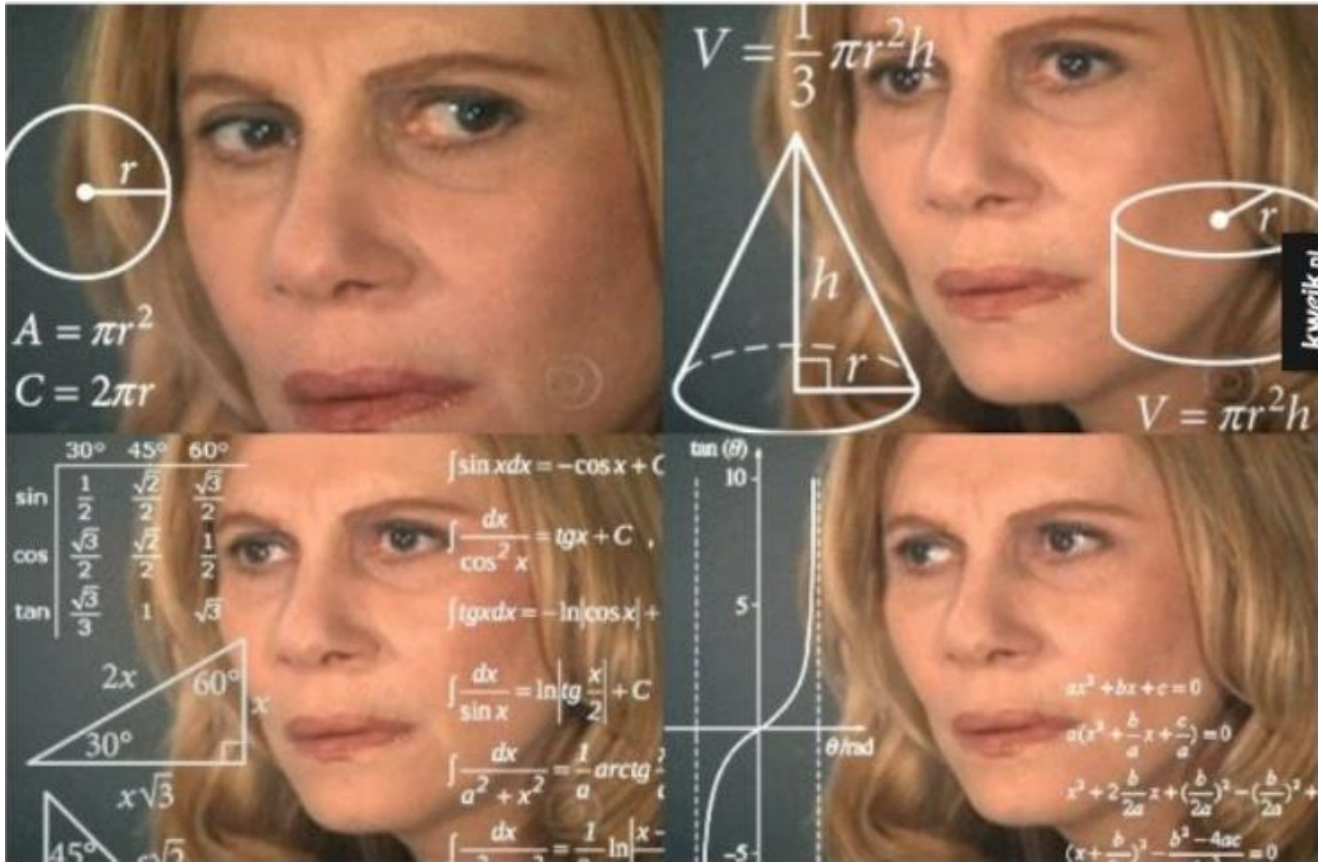
Jak pisać testy automatyczne?

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

Jak pisać testy - przykłady

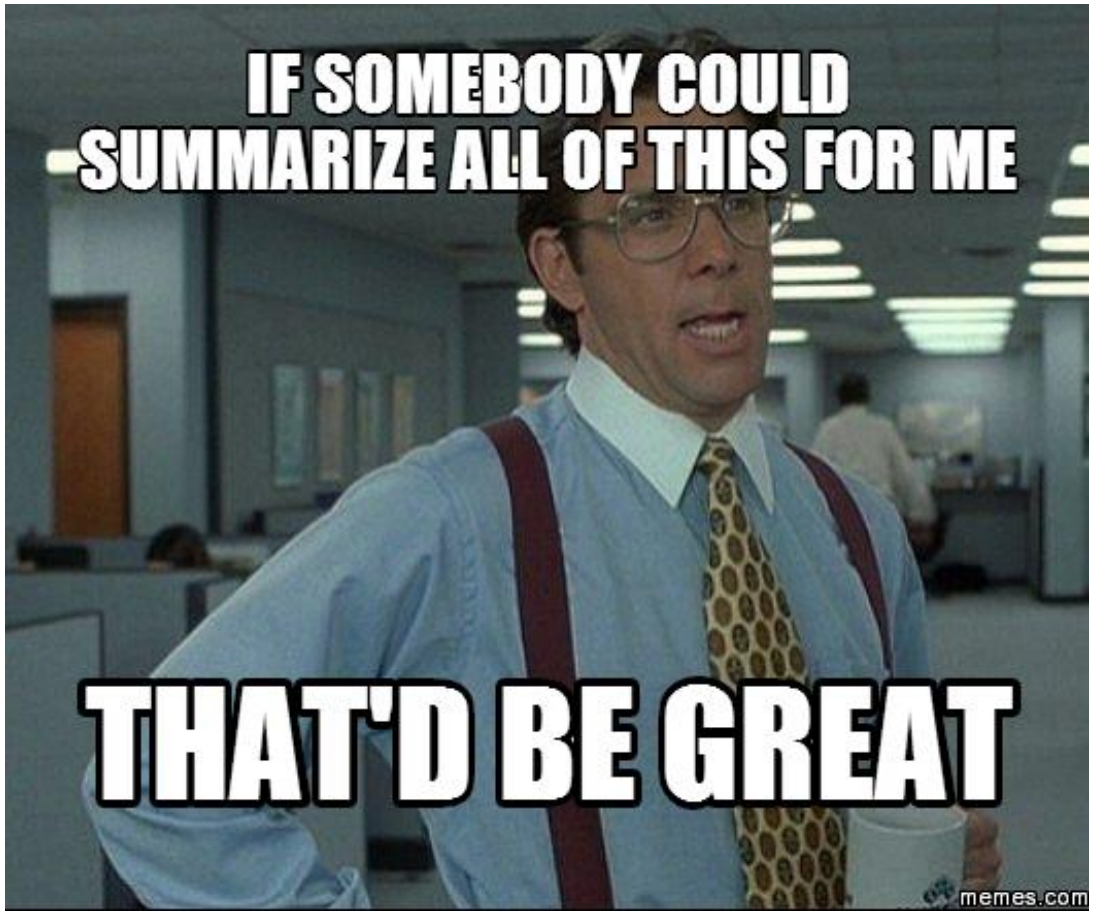
- Testy scenariuszowe
- Modułowość

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



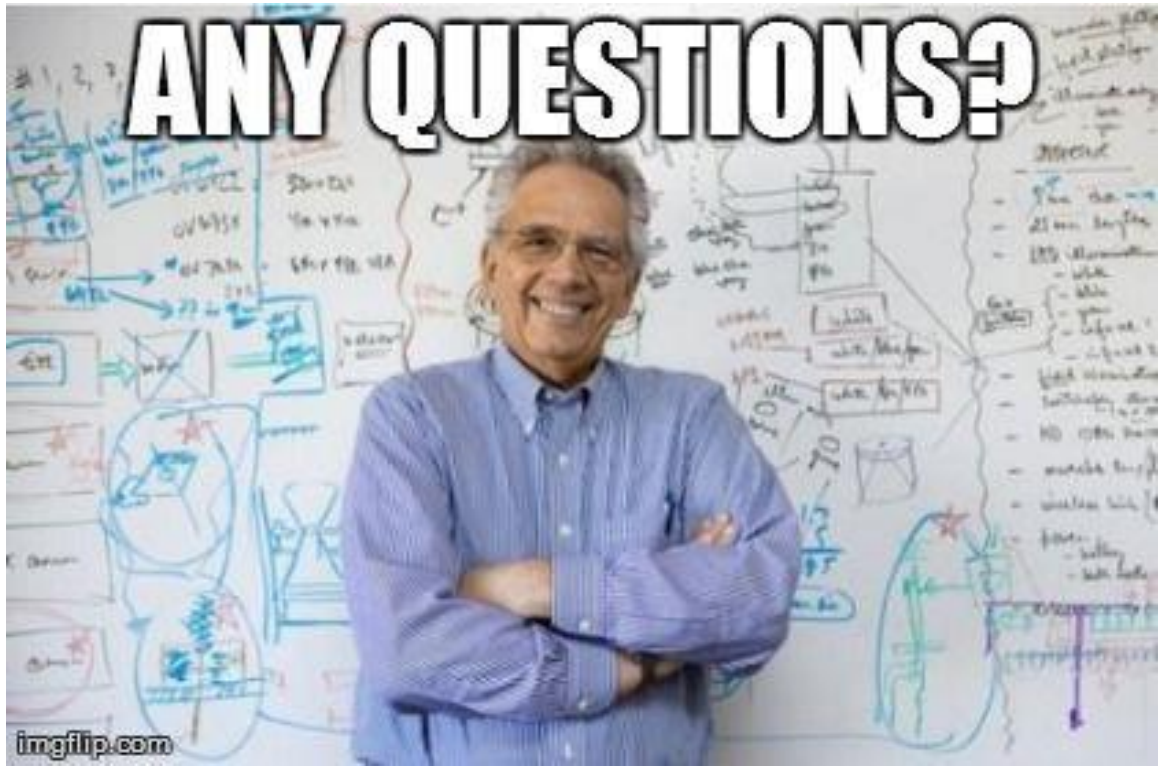
Trudności

- 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 błedów 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://martinfowler.com/articles/microservice-testing