

Czego programista μS może się nauczyć od dostawcy części samochodowych?

Czyli czym są testy kontraktowe

Marcin Baranowski

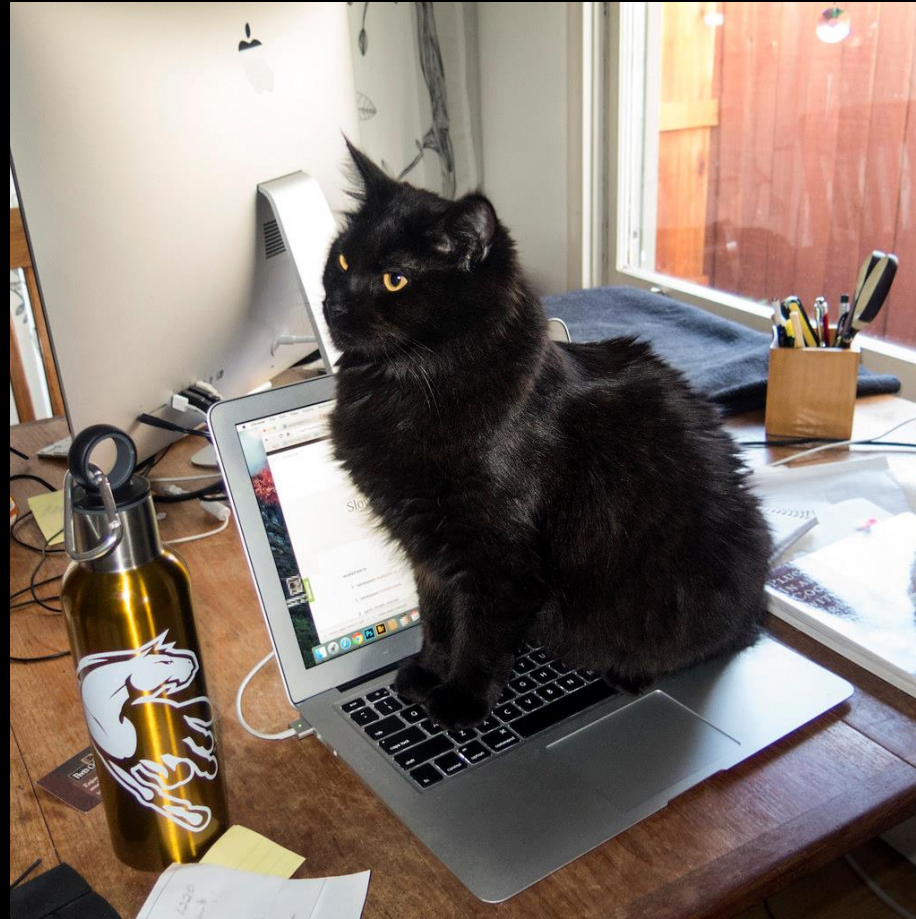
Java developer / Chapter leader, Applica.ai

Co zrobić, by testy były...

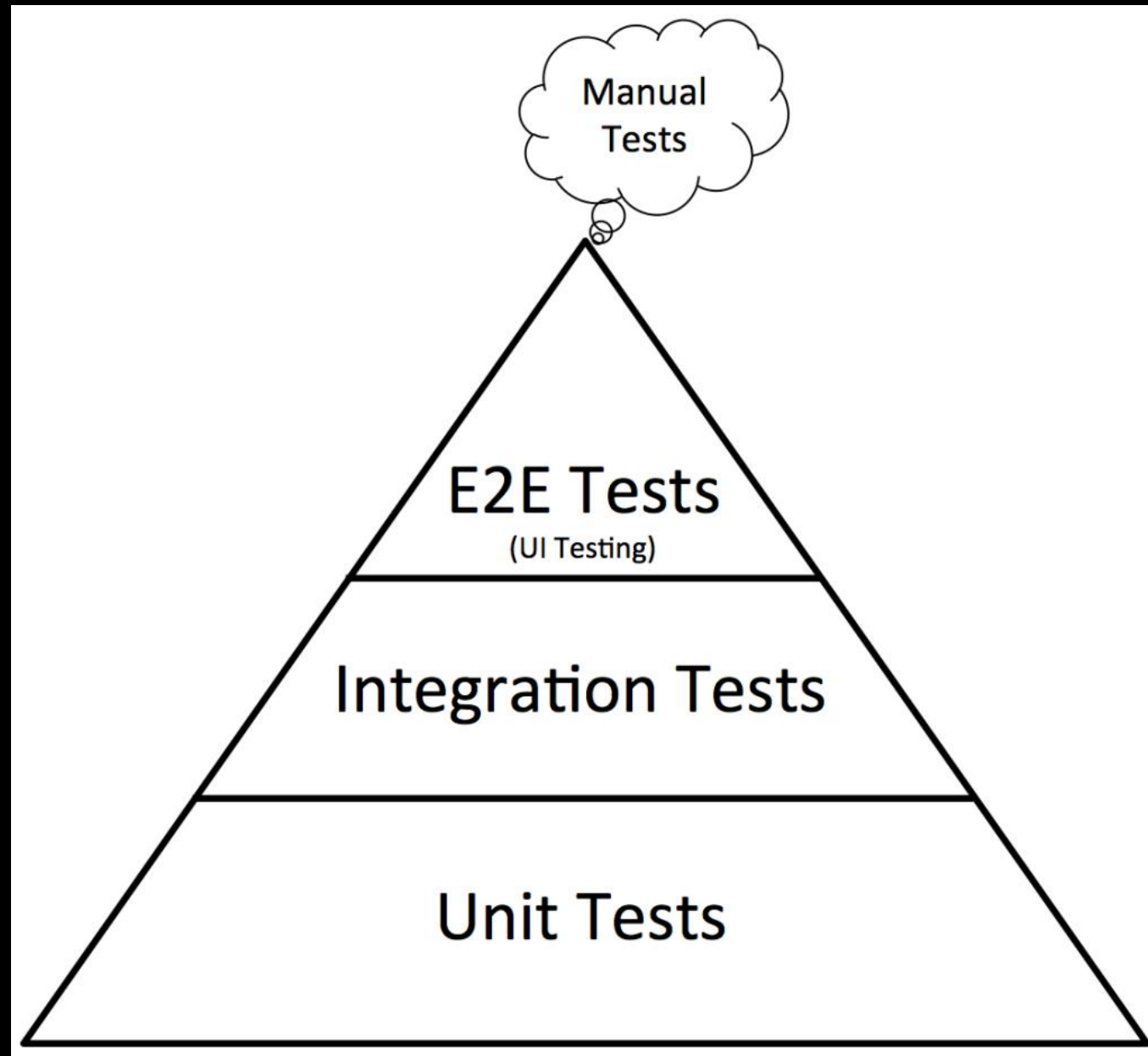
- Stabilniejsze
- Wiarygodne
- Łatwiejsze do pisania
- Szybsze

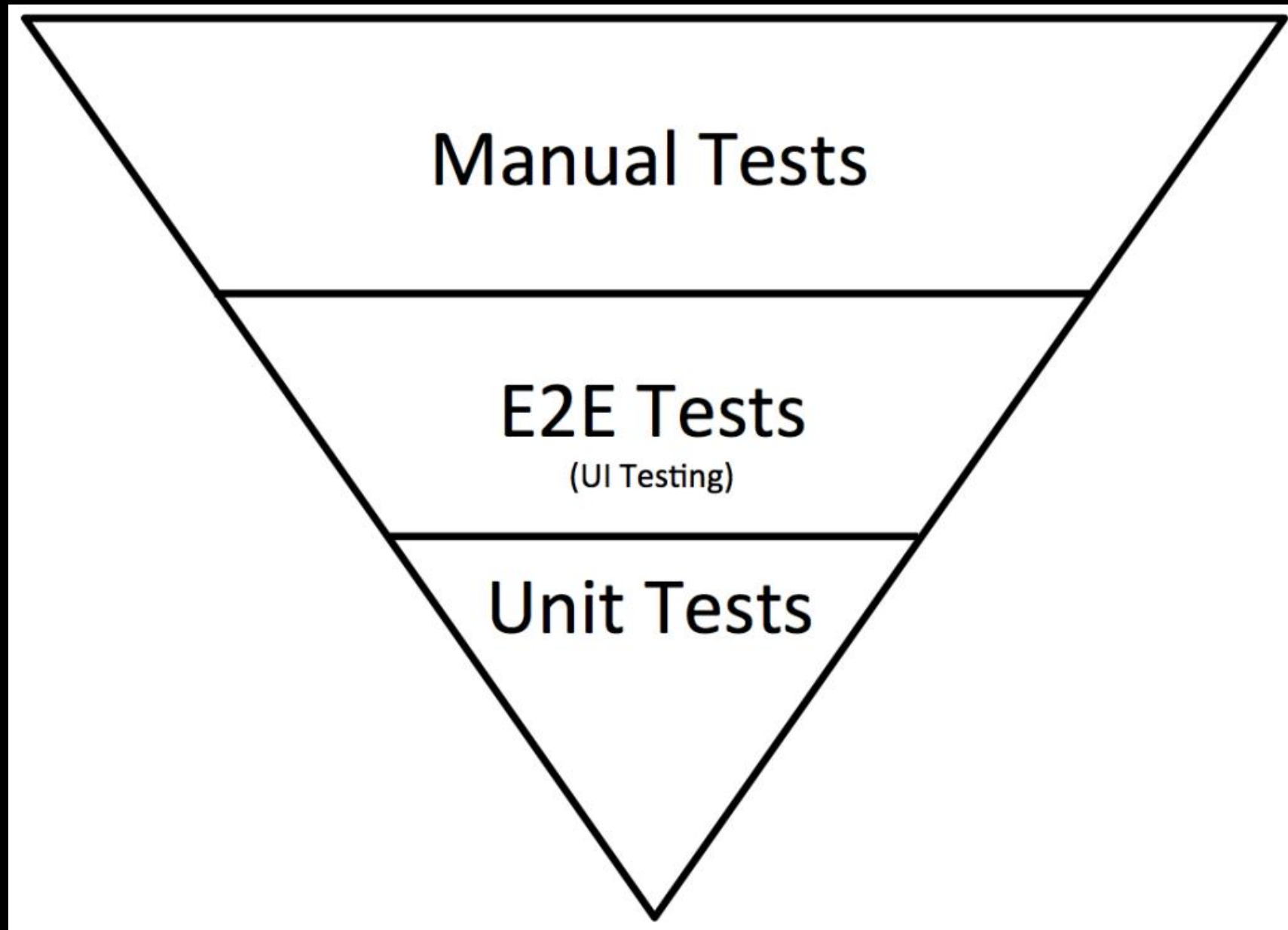
...i by wdrożenia były tak łatwe, że...

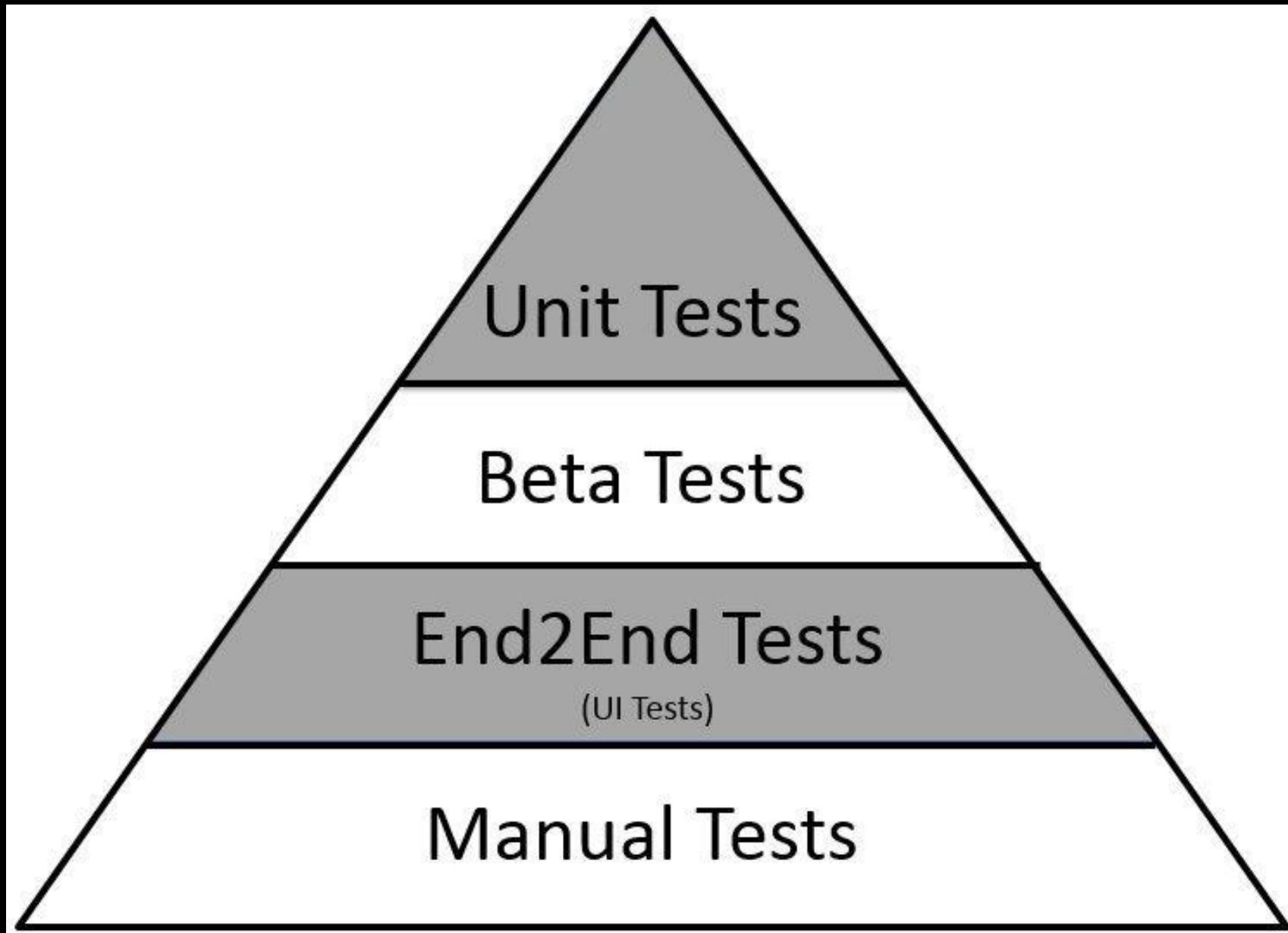
Twój kot mógł je zrobić?



Piramida testów może mieć różne oblicza







Mam nadzieję, że
po przejrzaniu tej prezentacji
wszystko stanie się dla Ciebie...

Jeszcze bardziej skomplikowane

...ale też lepiej zrozumiały

Testy jednostkowe

Szybkie

Niezależne

Proste

Testy jednostkowe

Niezależne

Niewystarczające

Proste

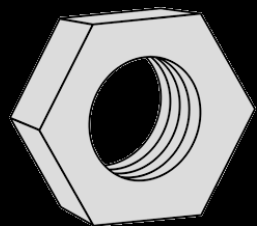
Testy integracyjne - zalety

- Sprawdzają komunikację między komponentami
- Testują realny kody (bez mocków)

Testy integracyjne - wady

- Składają się z wielu składników
- Trudno je napisać (trudniej niż testy jednostkowe)
- Długo się wykonują
- Wyniki mogą być niezgodne z prawdą (zależność od czynników losowych)

Jak to jest z samochodami?



x 30 tys. \approx



x



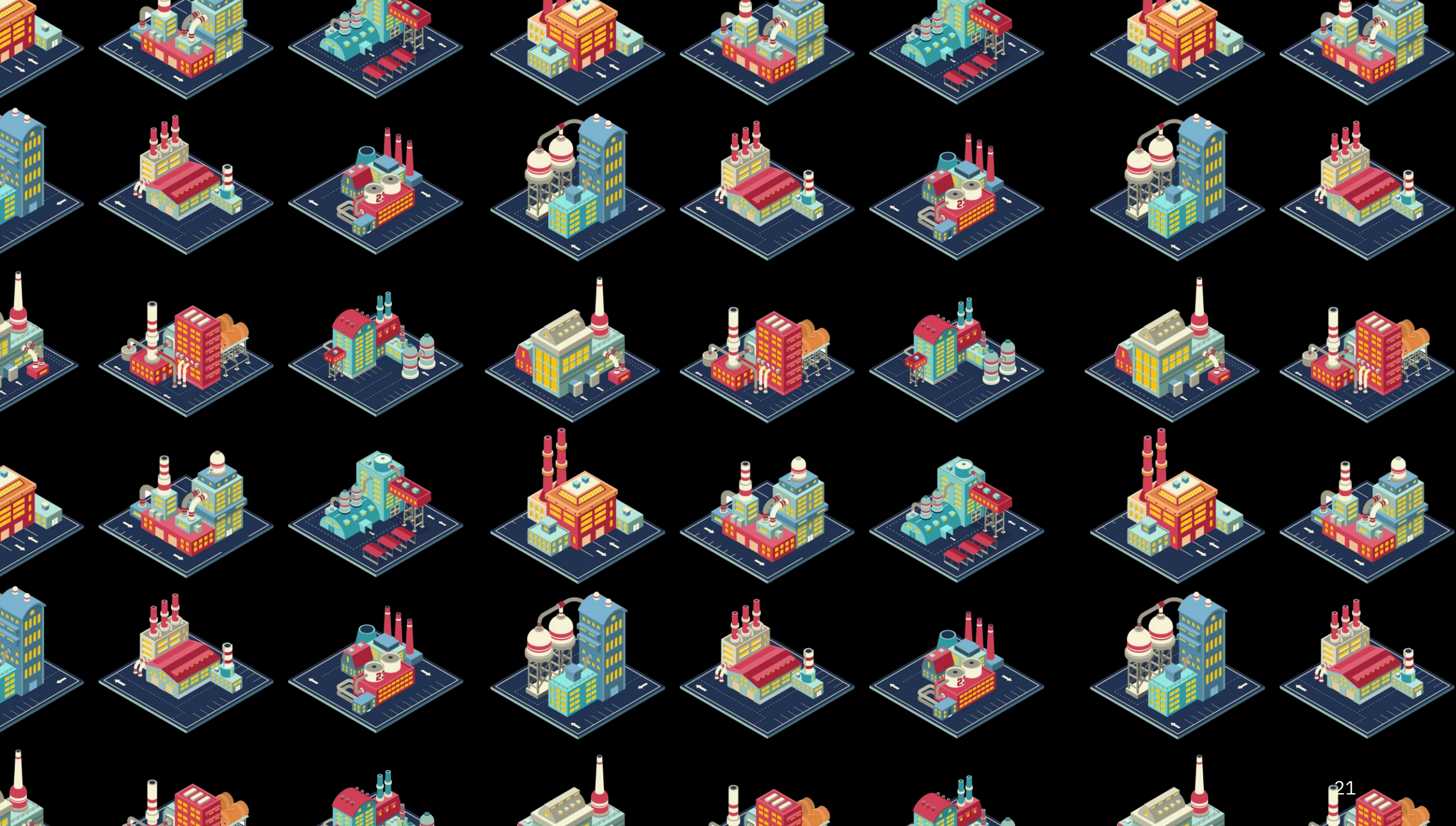
x 3


Jak przetestować 30 tys. części?



Samochody nie powstają
w jednej fabryce

Ciąg dostawców





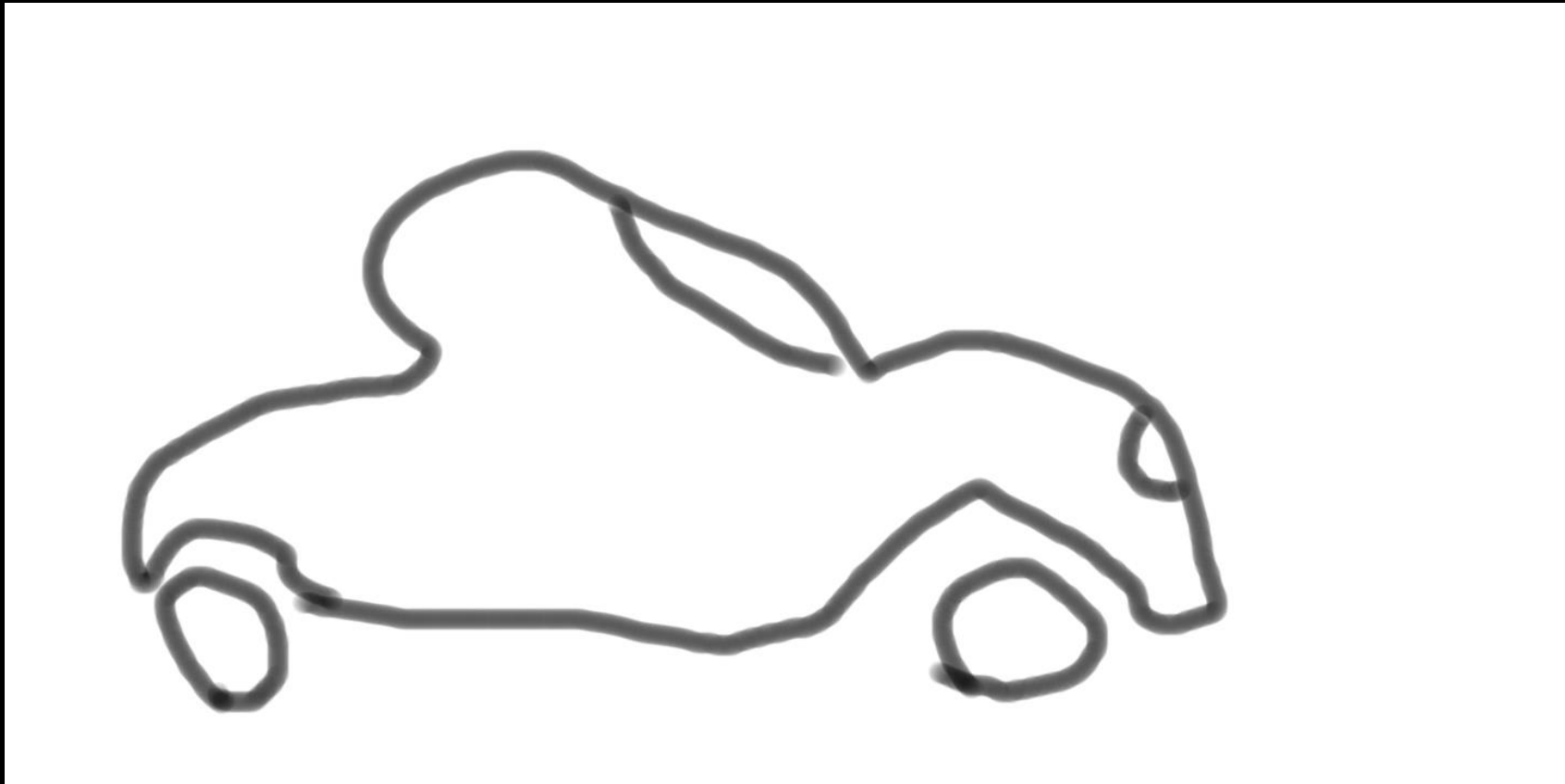
Teraz chodzisz lub jeździsz do marketu, żeby zrobić zakupy. Wyobraź sobie, że po każdy produkt, który chcesz kupić musisz wybrać się do innego sklepu...

Dostawców dla jednego samochodu
są setki

Fabryka składająca musi
się dogadać z dostawcami

Jak to wygląda?

Producent auta przedstawia dostawcy swoją wizję i powstaje szkic



Produkowane jest auto z napędem hybrydowym...



Albo gorzej



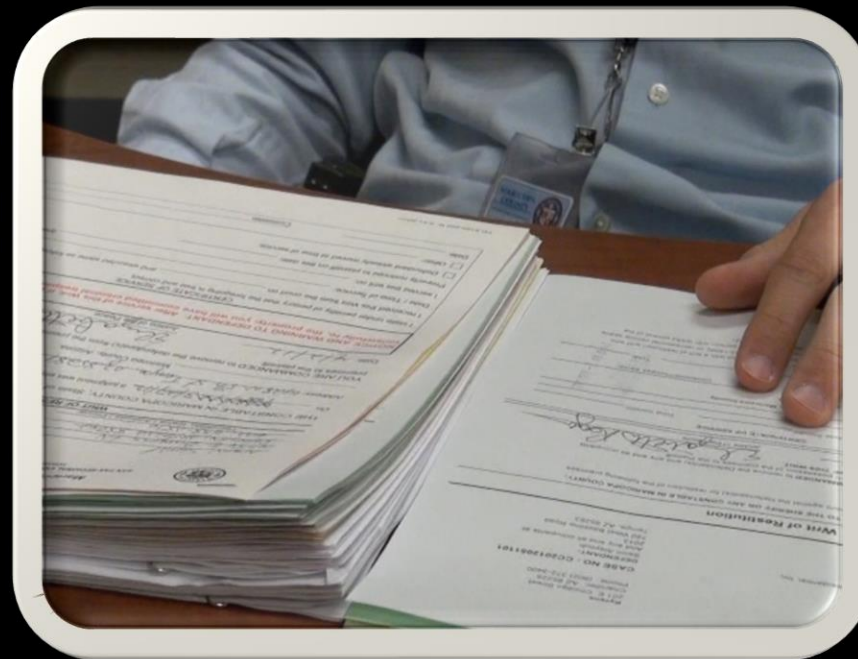
Img: Rudolf Stricker

Na szczęście jest inaczej

(Multipla jest wyjątkiem)

Wielostronicowe umowy

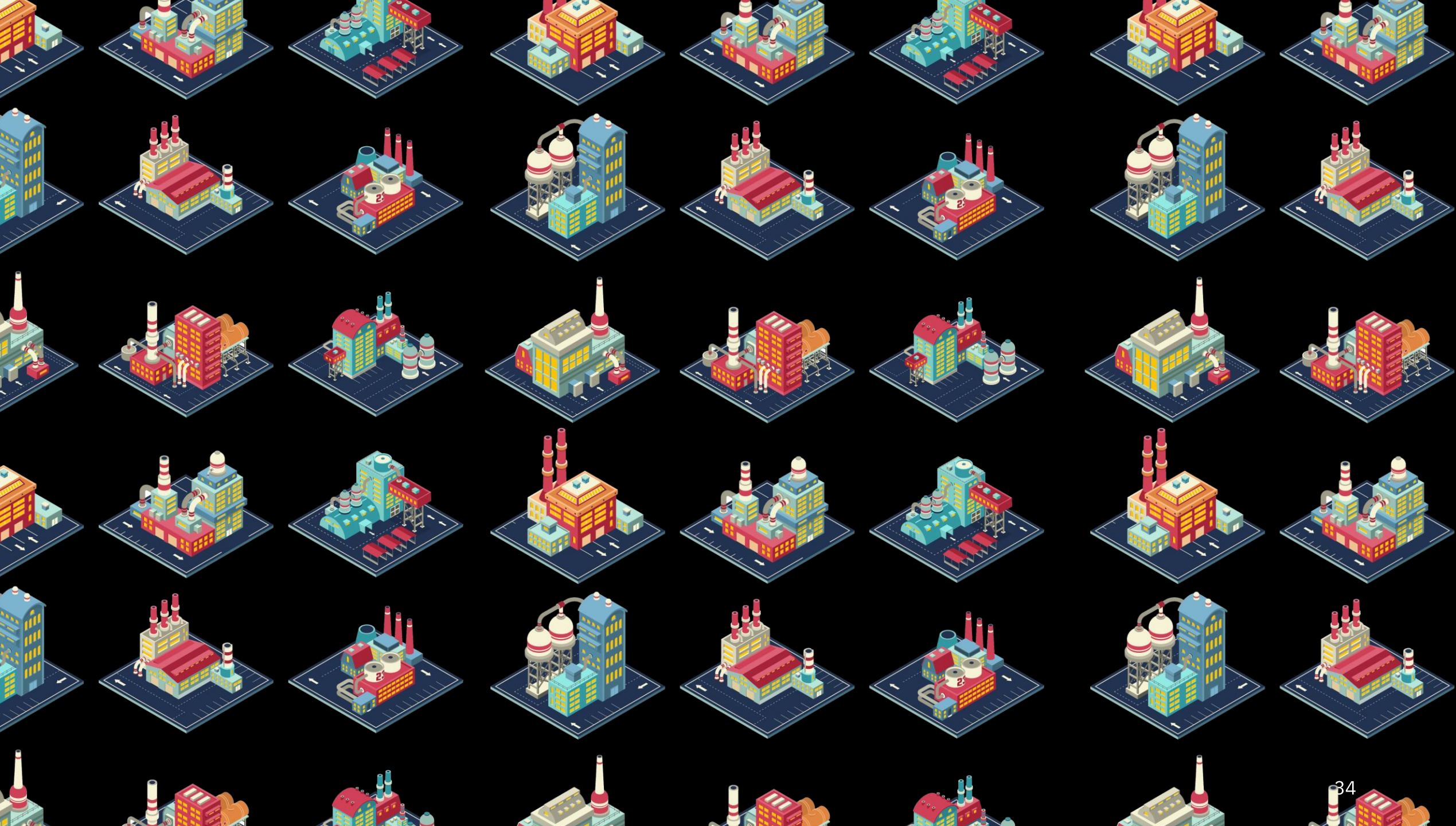
- Kary umowne
- Audyty
- Standardy

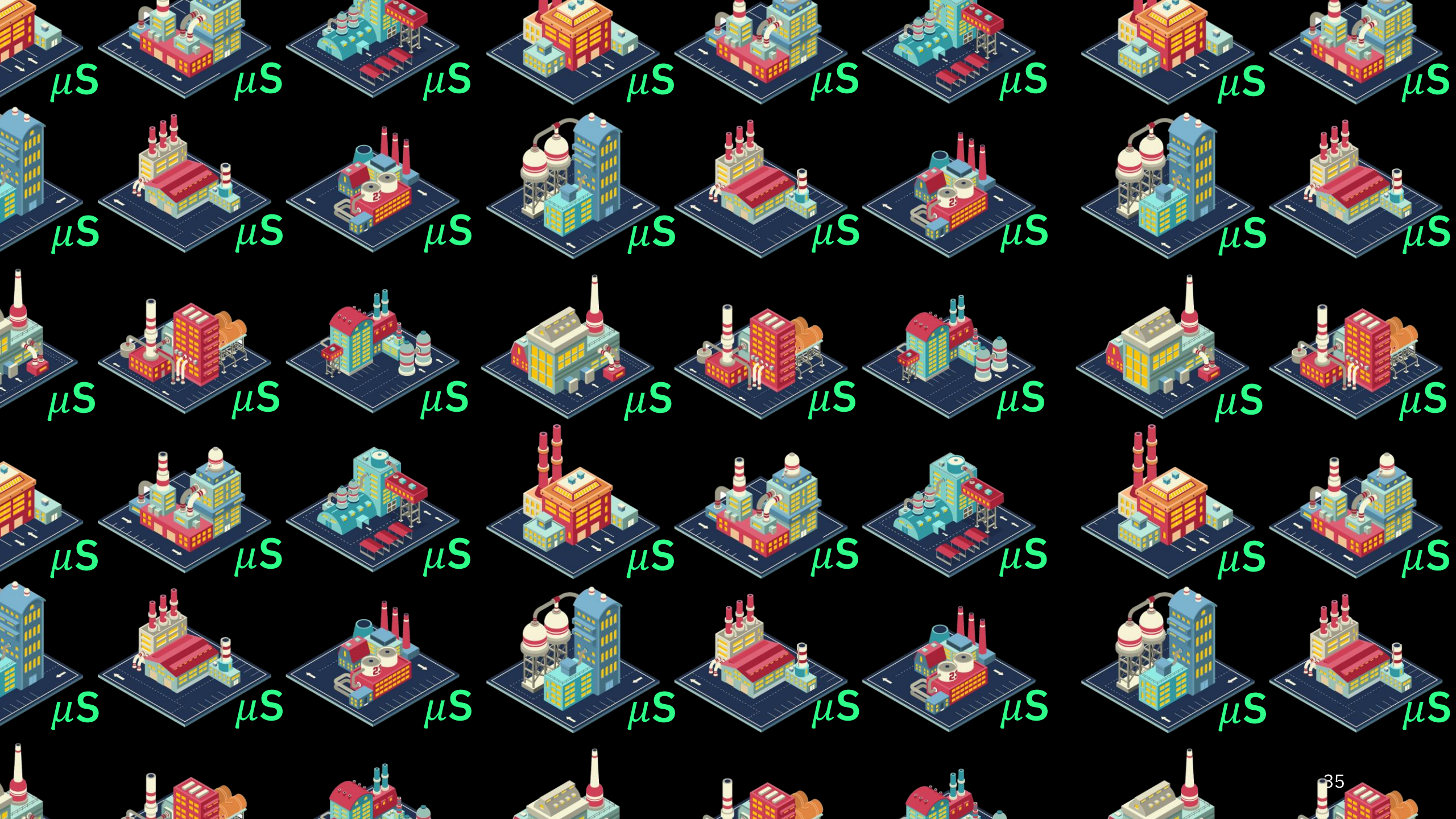


Producent i dostawca zawierają
kontrakt

Producent i dostawca zawierają
kontrakt

Jak to przełożyć na programowanie?





uS mogą tworzyć
między sobą kontrakty

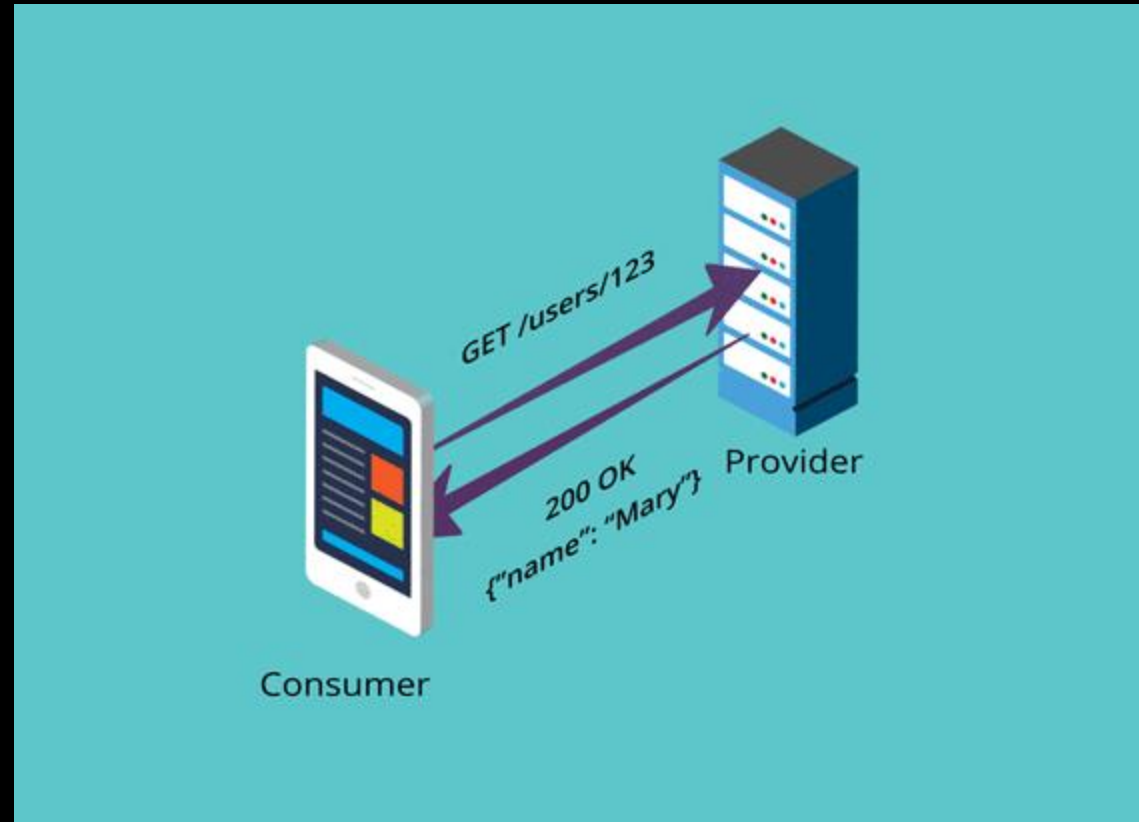
Frontend <-> Backend

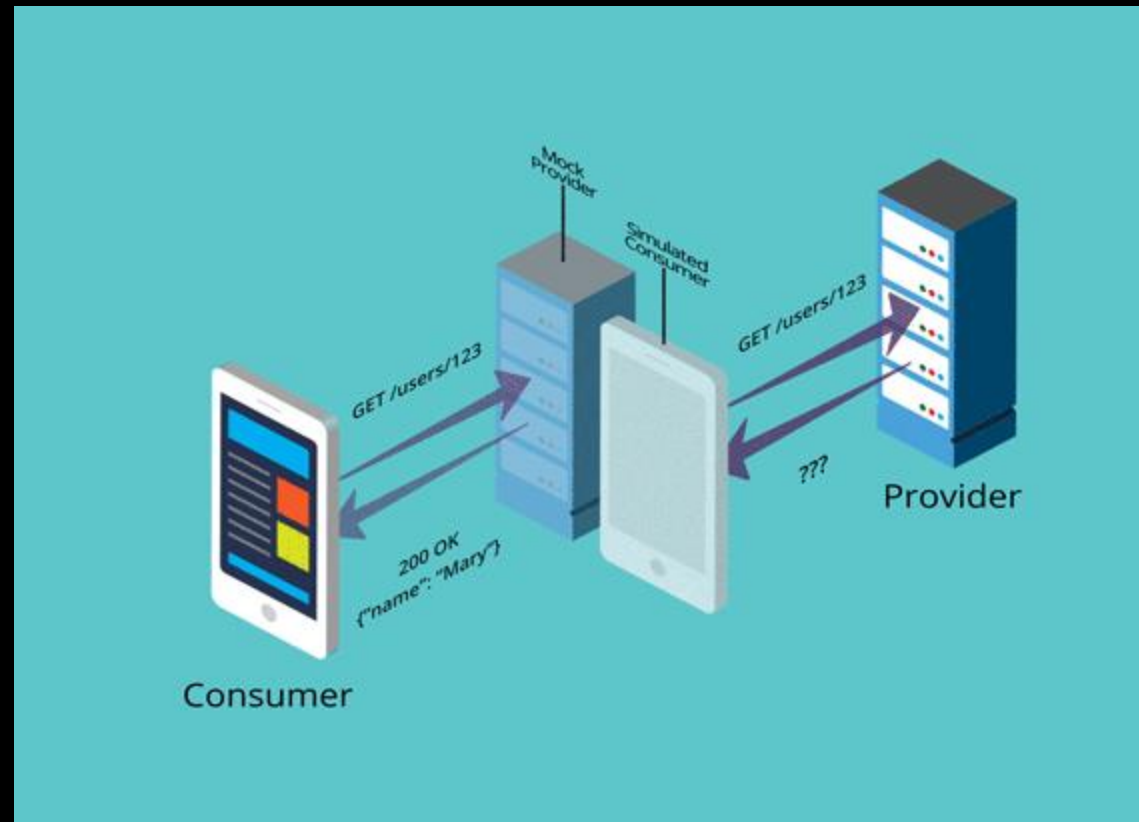


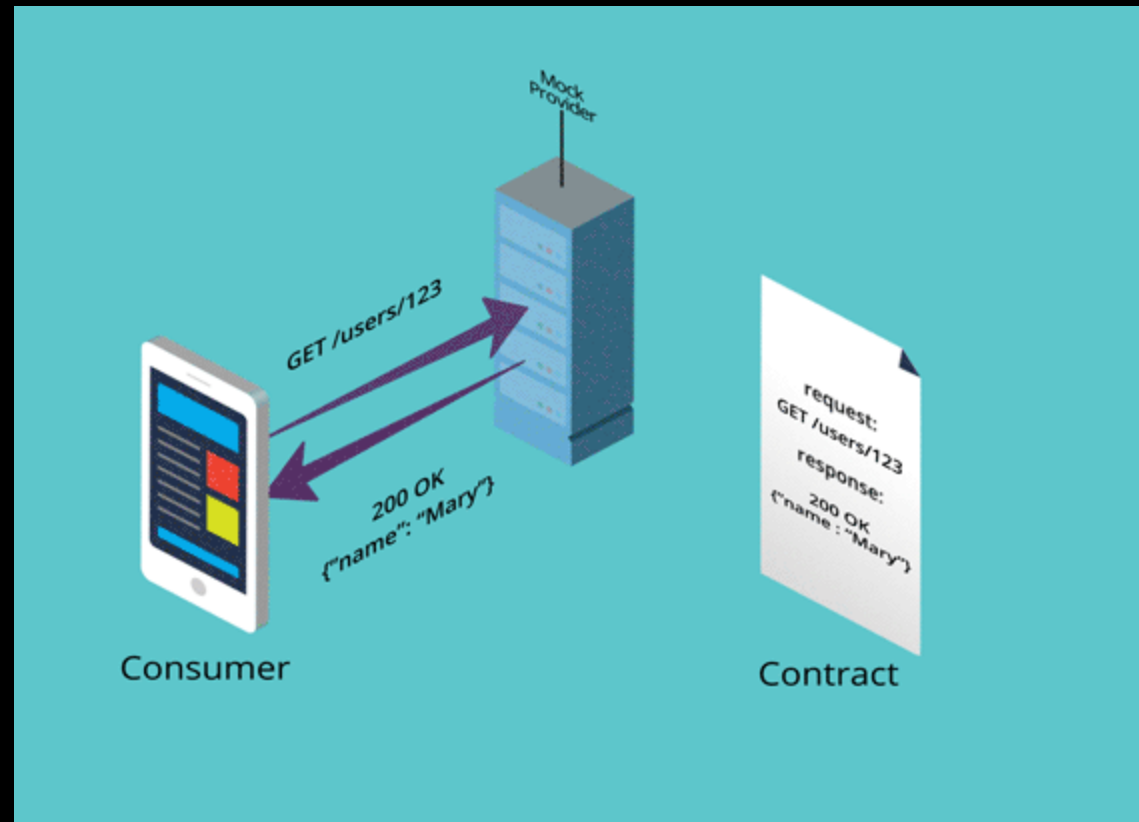
Consumer

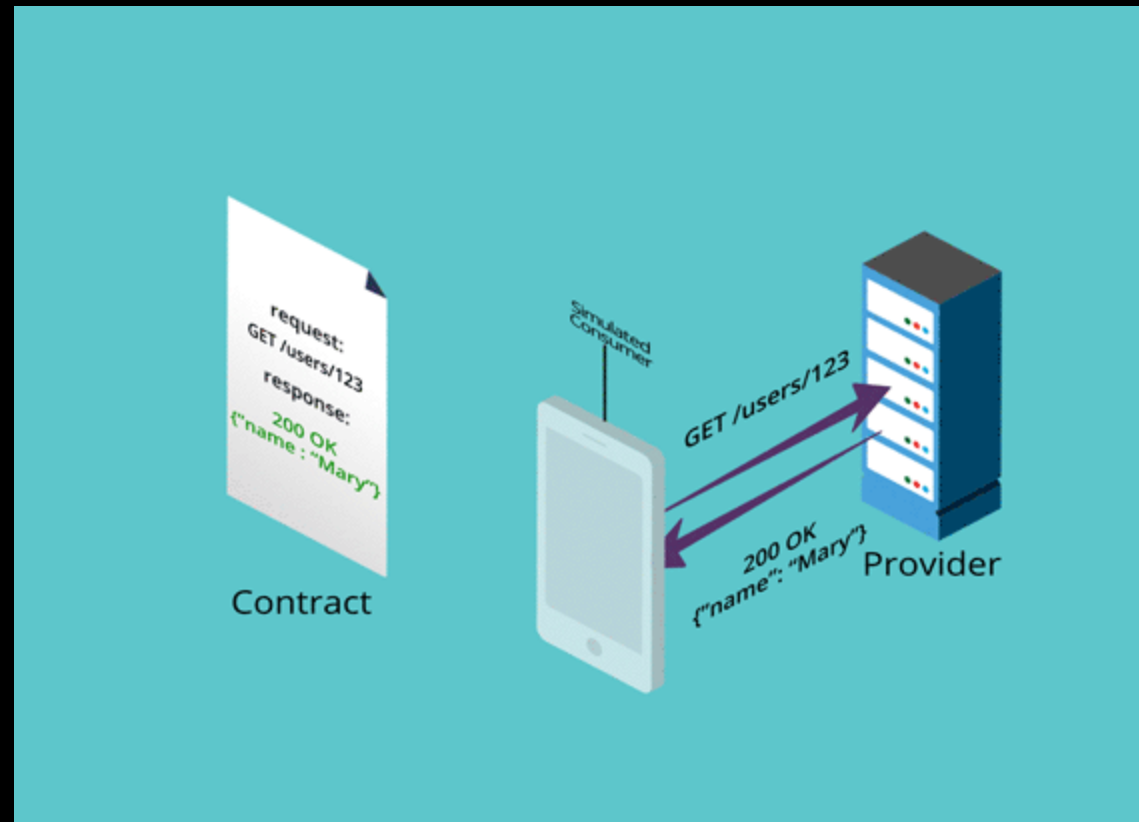


Provider









Typy testów kontraktowych

- **Provider driven contract** – dostawca określa kontrakt, do którego konsumenci muszą się dostosować
- **Consumer driven contract** – konsument określa kontrakt, do którego dostawcy muszą się dostosować



**CLIENT
EXPECTATION**



**CLIENT
BUDGET**



Pact jest językowo agnostyczny

Można pisać kontrakty niezależnie od używanego języka programowania

...ale niektóre języki mają gotowe
przykłady/biblioteki do wykorzystania



Wspierane formy komunikacji

- Http rest
- Grpc
- GraphQL
- Messaging

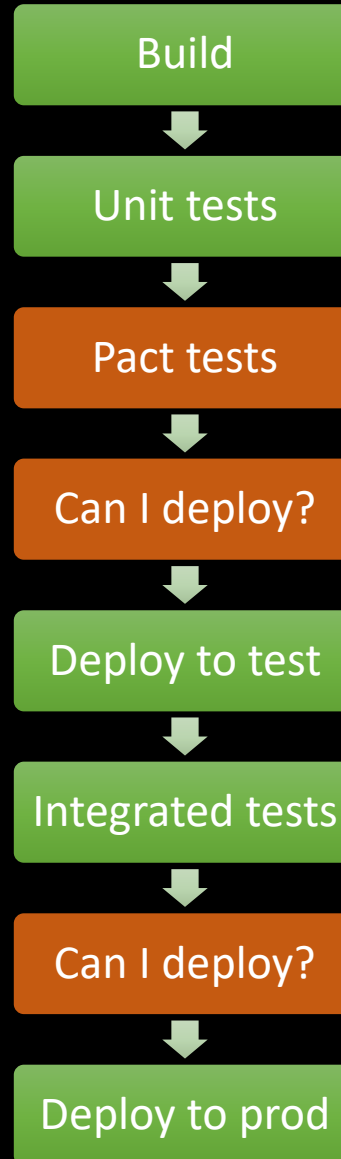
Gdzie trzymać kontrakty?

Pact broker

Flow CI bez Pacta



Flow CI z Pactem



Can-i-deploy

Narzędzie do weryfikacji, czy dany komponent (mikroserwis) może być wdrożony

Kazali szybko wgrać na proda...



Mają za swoje

```
pact-broker can-i-deploy --pacticipant frontend --version 1 --pacticipant backend --version 5
```



Wady i pułapki testów kontraktowych w Pact

- Dodatkowa infrastruktura
- Darmowe rozwiązanie nie będzie miało nowych, przełomowych funkcjonalności
- Sporo konfiguracji na początku
- Dopiero używając can-i-deploy mamy „bezpieczny” pipeline CI/CD

Zalety testów kontraktowych w Pact

- Dobry brakujący element piramidy testowej (sprawdza więcej niż test jednostkowy, jest szybszy niż integracyjny)
- Silne community
- Wiele tutoriali / przykładów
- Pręźnie się rozwija

Bi-directional contracts

NAJWAŻNIEJSZA JEST KOMUNIKACJA



Marcin Baranowski

`mrsilversheep@gmail.com`

`https://github.com/SilverSheep`