Testowanie automatyczne - podstawy

Rodzaje testów

- Testy funkcjonalne
  - Cele biznesowe -> wykonywalne specyfikacje -> testy użytkowników (głównie akceptacyjne) **Automatyczne**
  - Cele architektoniczne -> struktura systemu -> testy komponentów/integracyjne **Automatyczne**
  - Cele implementacyjne -> struktura kodu -> testy jednostkowe **Automatyczne**

Rodzaje testów

- Testy niefunkcjonalne
  - Czy system jest ergonomiczny? -> testy użyteczności (ang. usability tests)
  - Czy system jest spójny, konsekwentny? -> testy eksploracyjne (ang. Exploratory tests)
  - Czy system jest responsywny, bezpieczny, skalowalny? -> testy własności (ang. Property tests)

Ręczne

Ręczne

Automatyczne

Cele testowania automatycznego

- Testy powinny:
  - pomagać w poprawie jakości oprogramowania
  - pomagać w lepszym zrozumieniu systemu (SUT)
  - Redukować ryzyko
  - Być łatwe w uruchomieniu
  - Być łatwe w pisaniu i pielęgnacji
  - Wymagać minimalnego nakładu konserwacyjnego w trakcie ewolucji systemu

Filozofia testowania automatycznego

- Testy przed a testy po
- Testy a przykłady działania (wykonywalne specyfikacje)
- Test za testem a testy całości
- Outside-in a Inside-out
- Weryfikacja stanu a weryfikacja zachowania

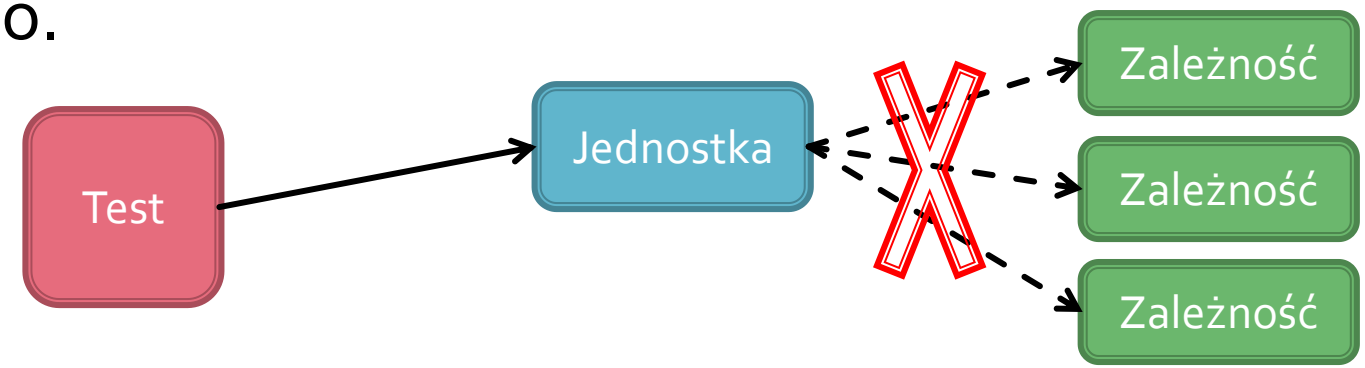
System Under Test (SUT) vs Dependent-on component (DOC)

- SUT -To co testujemy.
  - Jednostka (klasa, obiekt, metoda)
  - komponent
  - cała aplikacja
- DOT – reszta, której nie testujemy, ale jest „zamieszana” w działanie SUT
  - W automatyzacji testów może być w centrum uwagi (testujemy interakcję SUT z DOT)

http://xunitpatterns.com/SUT.html

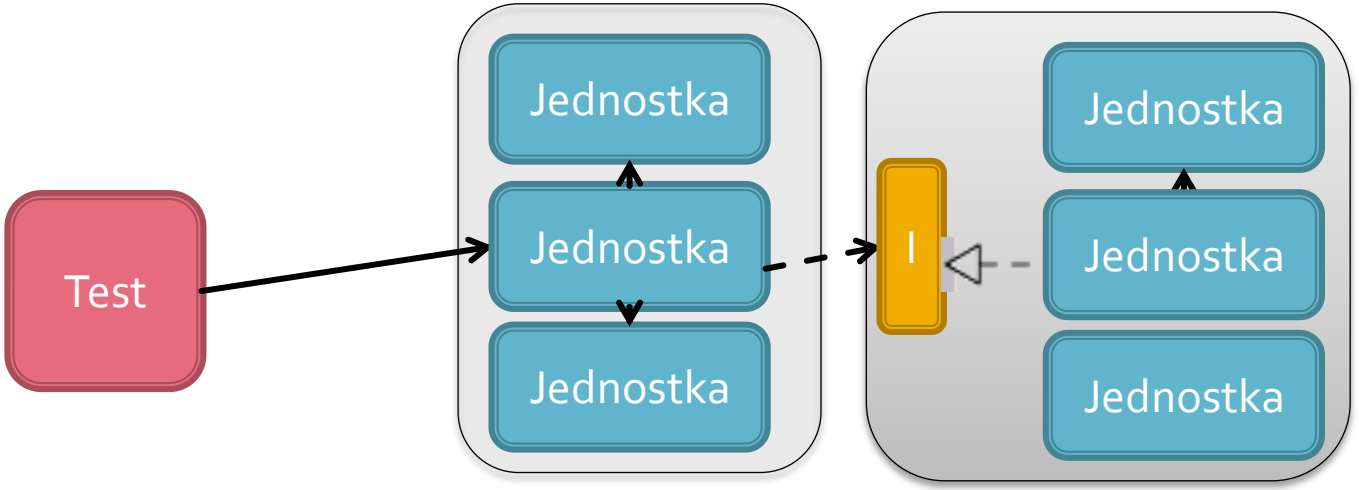
Zakres testu

- Test jednostkowy
  - Testowanie zachowania poszczególnych elementów z których zbudowane jest oprogramowanie
  - Testowanie w izolacji
  - Sprawdzenie czy logika działa **dokładnie** tak jak założono.



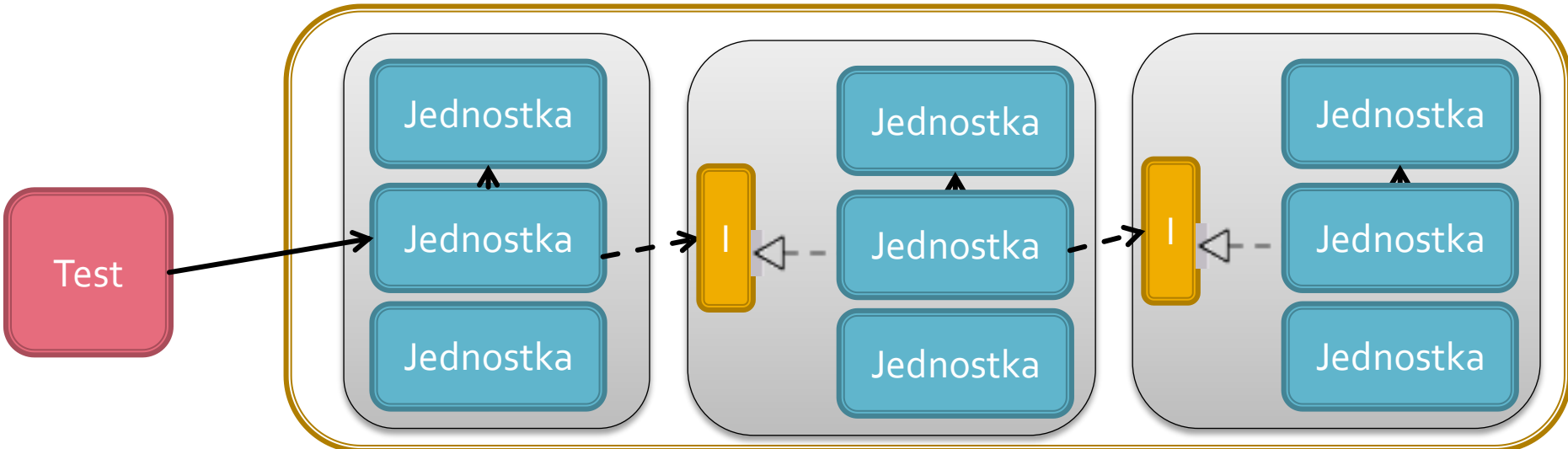
Zakres testu

- Test komponentu, systemu (integracyjny)
  - Testowanie „poprzez warstwy”
  - Sprawdzanie poprawności integracji na różnych poziomach (komponentu, systemu)
  - Sprawdzenie czy działa zgodnie z założeniami (na niższym poziomie dokładności niż test jednostkowy).



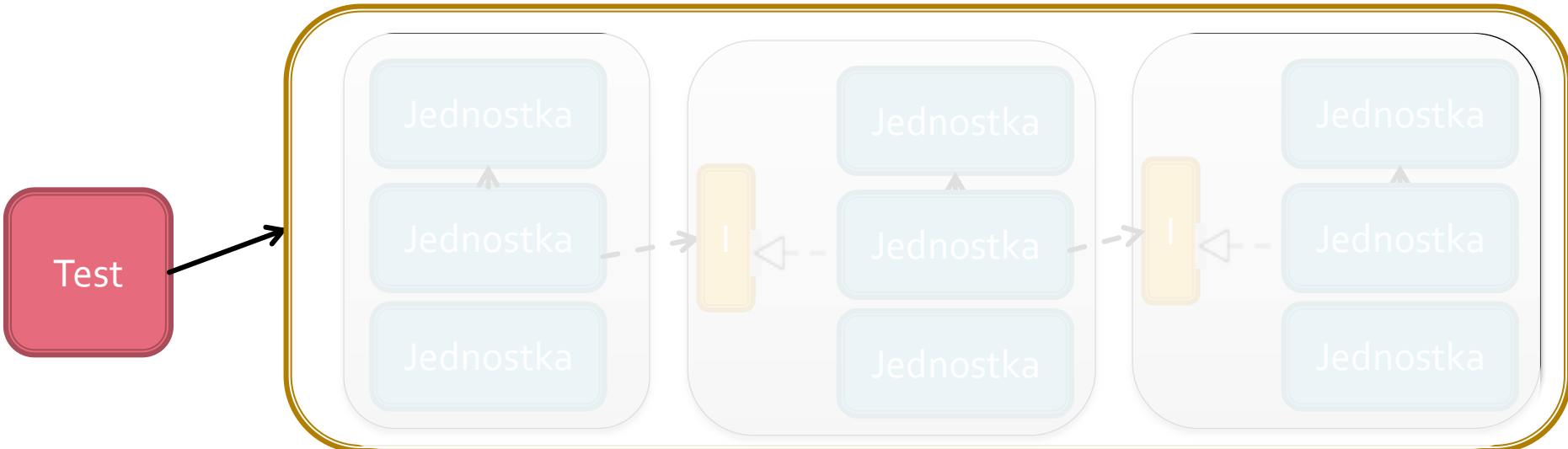
Zakres testu

- Test systemu (End2End)
  - Testowanie „poprzez warstwy”
  - Sprawdzanie poprawności integracji na różnych poziomach (komponentu, systemu)
  - Sprawdzenie czy działa zgodnie z założeniami (na niższym poziomie dokładności niż test jednostkowy).



Zakres testu

- Test akceptacyjny
  - Testowanie scenariuszy
  - Testowanie całego systemu
  - Test **pokazujący**, że system działa zgodnie z wymaganiami.



Rola testu

- Sprawdzanie czy kod działa (perfekcyjnie) zgodnie z założeniami
- Wyznaczanie zadań i kryteriów akceptacji systemu
- Sprawdzanie czy kod po zmianach **dalej działa** zgodnie z założeniami

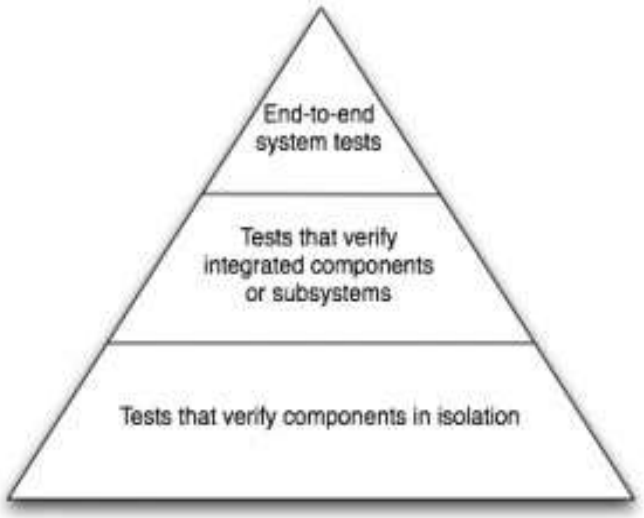
Typ testu

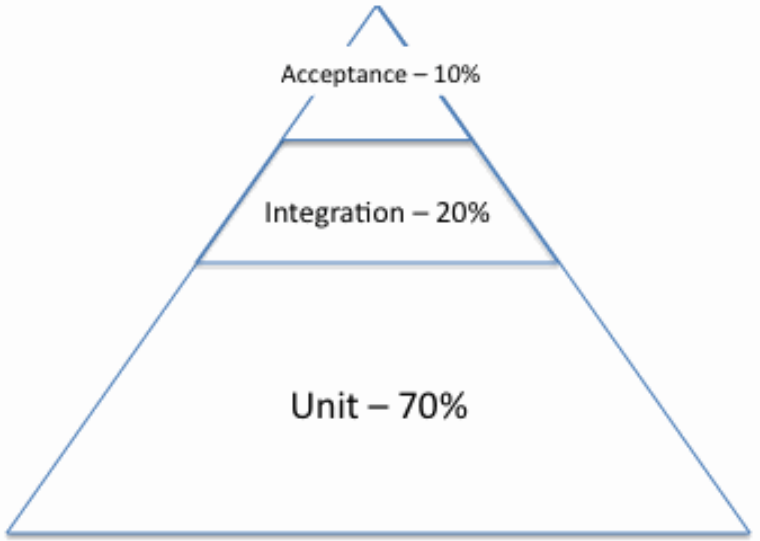
- Test stanu
  - Weryfikacja poprawności na podstawie stanu SUT (lub DOT) po wykonaniu testu.
- Test zachowania
  - Weryfikacja poprawności na podstawie poprawności procesu komunikacji z DOC

Pokrycie kodu testami

- Procent
  - kodu, ścieżek, stanów obiektu
    - uruchamianych przez testy
- Metryki zależne od typu projektu
  - Np.: 80% dla projektów biznesowych
- Równomierność vs nierównomierność rozkładu testów w kodzie
  - Kod krytyczny
  - Kod „mniej istotny”

Piramida testów





- Kontekst architektury może dawać dodatkowe wskazówki dotyczące tego kiedy i gdzie stosować dane testy

Bibliografia

- Gerard Meszaros: xUnit Test Patterns Refactoring Test Code
- Sławomir Sobótka: Kompendium testowania aplikacji opartej o DDD – problemy, strategie, taktyki i techniki
- http://xunitpatterns.com