M&I SYSTEMS GROUP

VOJVODINA ICT CLUSTER

Testiranje Softvera

GO4Code

Marko Cvijanović
Mihailo Bubnjević

Testiranje

- Proces proveravanje funkcionalnosti, kvaliteta i performansi softvera
- Verifikacija da je softver ili aplikacija *bug-free*, popunjava sve tehničke i korisničke zahteve i pokriva sve slučajeve
- Krucijalan deo razvoja softvera
- Izbegavanje problema pre nego što softver ode u produkciju

Zašto je testiranje bitno

- Otkrivanje bugova, problema i propusta kako bi se što pre rešili
- Održavanje i poboljšanje kvaliteta softvera
  - Održavanje
    - Regression testing
    - Periodično testiranje drugih aspekata
  - Poboljšanje
    - Pronalaženje i rešavanje skrivenih mana
    - Pronalaženje delova koda koji mogu da se optimizuju
- Identifikovanje vulnerabilnosti koji mogu da sačuvaju troškove a nekad i ljudske živote

Vrste testiranja

- **Funkcionalno testiranje**
  - Unit testiranje
  - Integraciono testiranje
  - End to End testiranje
  - User Acceptance testiranje
- **Ne-funkcionalno testiranje**
  - Security testiranje
  - Performance testiranje
  - Load testiranje

Pristupi testiranju

- **Manualno testiranje**
  - Testiranje softvera ručno od strane ljudi bez korišćenja alata ili skripti
  - Exploratory testing, Usability Testiranje, Ad-hoc Testiranje
- **Automatizovano testiranje**
  - Testiranje softvera koje podrazumeva korišćenje alata ili skripti koji automatski interaguju sa sistemom.
  - Load i Performance Testiranje
  - Unit i Integraciono Testiranje

Manualno testiranje

- **Prednosti**
  - Lako uočavanje bug-ova koje automatizovane skripte ne mogu da pronađu
  - Fleksibilni
- **Mane**
  - Ljudski faktor
  - Vremenski intenzivno i dosadno
- **Koristi se**
  - u inicijalnim fazama testiranja radi razumevanja rada aplikacije i učenje domena i sistema
  - za specifične scenarije koje je lakše ručno testirati
  - za testiranje UI/UX

Automatizovano testiranje (Automated Testing)

- **Prednosti**
  - Brzo i efektivno
  - Isplativno
  - Transparentnost rezultata testiranja
- **Mane**
  - Zahtevaju investiciju resursa
  - Nemaju informaciju kako

Unit Testiranje

- Testiranje najmanjih delova aplikacije koje nazivamo Unit
- Unite testiramo individualno i nezavisno od ostatka sistema
- Dobri unit testovi su:
  - Nezavisni i izolovani
  - Deskriptivni
  - Ponovljivi
  - Brzi

Izolovanost Unit Testova

REAL SYSTEM

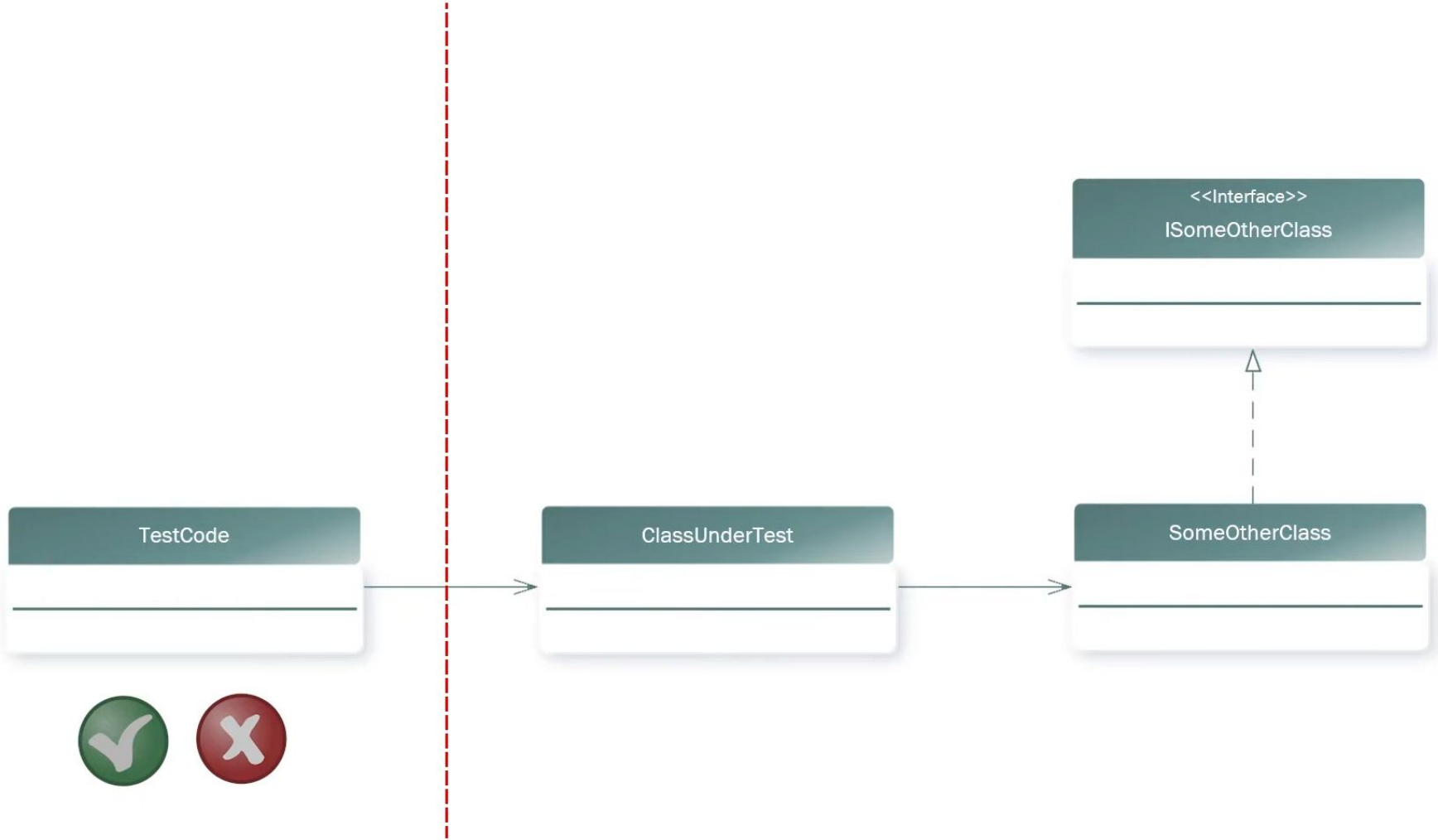
Green = class in focus
Yellow = dependencies
Grey = other unrelated classes

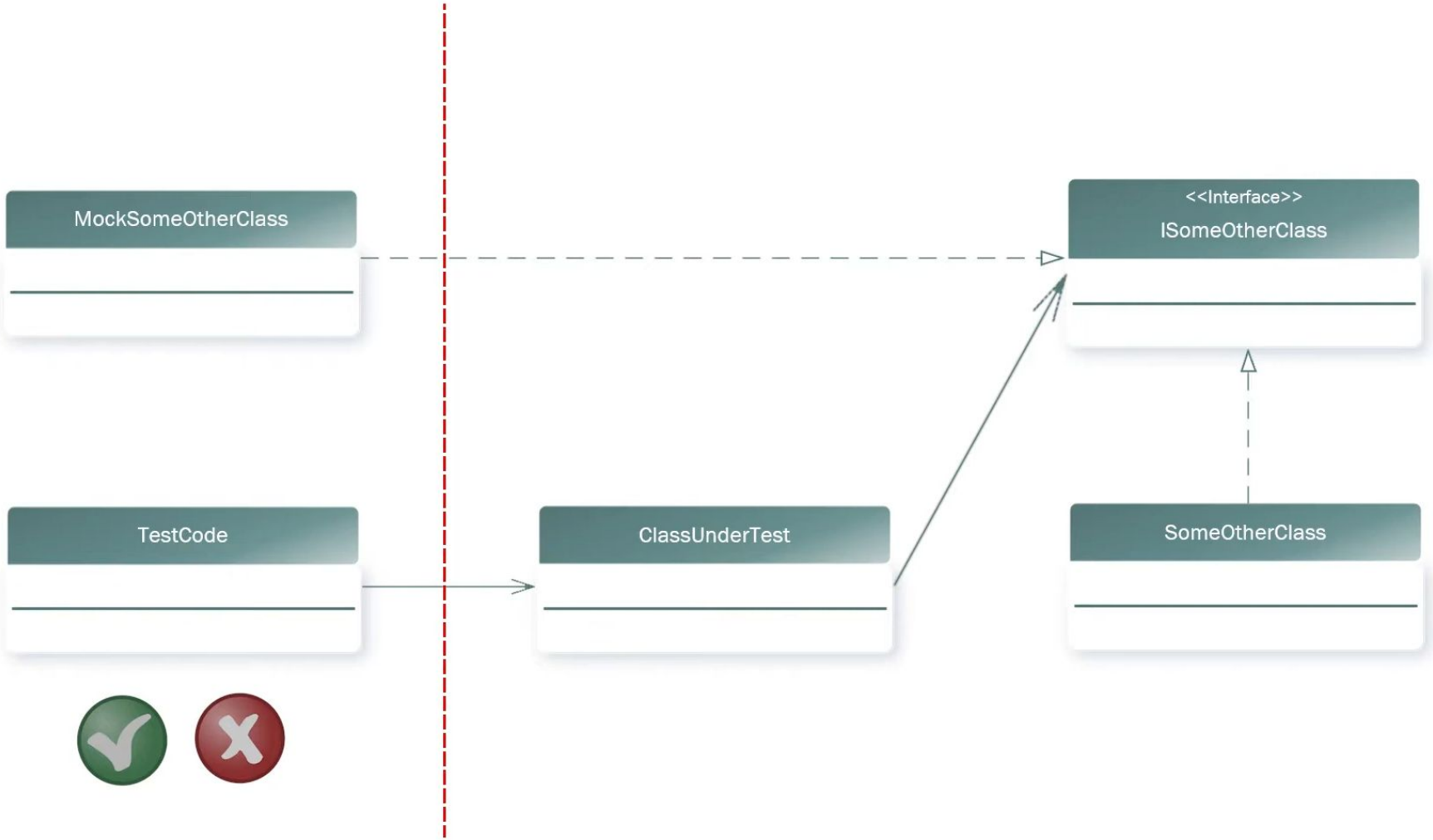
CLASS IN UNIT TEST

Green = class in focus
Yellow = mocks for the unit test

Mocking

- Kada Unit ima zavisnosti na druge kompleksne objekte koristimo mocking
- Menjamo zavisne objekte sa mockovima tokom testiranja
- Pomaže nam da izolujemo ponašanje i funkcionalnost
- Fokusiramo testiranje na samo kod jednog Unit-a
- Ako nešto ne radi kako treba znamo da je unutar Unit-a


<<Interface>>
ISomeOtherClass
TestCode
ClassUnderTest
SomeOtherClass


MockSomeOtherClass
<<Interface>>
ISomeOtherClass
TestCode
ClassUnderTest
SomeOtherClass

Integraciono testiranje

- Testiranje grupe Unita
- Testiranje individualnih funkcionalnosti i svih njenih povezanih delova
- Da li svaka komponenta obavlja svoj deo i daje krajnji rezultat
- Daje nam informaciju da li se naš sistem i svi njegovi delovi ponašaju kako je očekivano umesto samo jednog malog dela
- Najčešće se vodimo Top-Down pristupom