Szoftverellenőrzési technikák

Fejlesztői tesztelés
Modultesztelés és izoláció

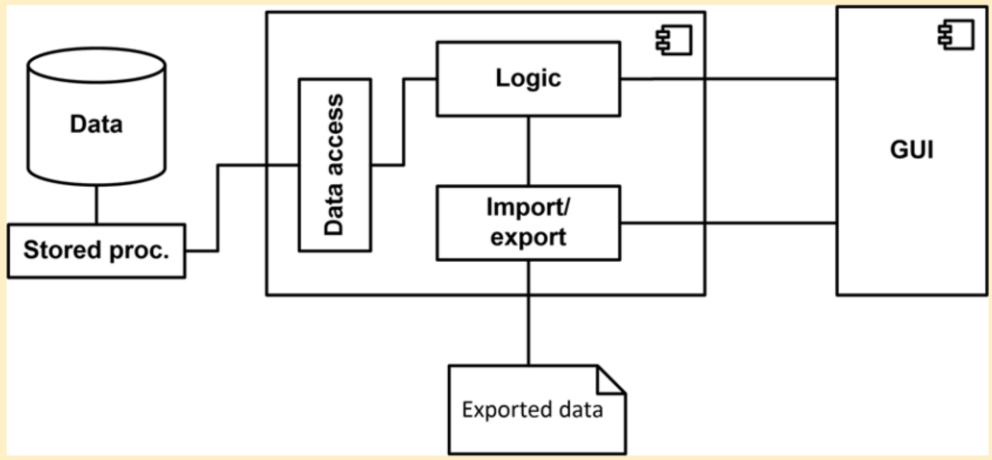
Majzik István, Micskei Zoltán

http://www.inf.mit.bme.hu/

1

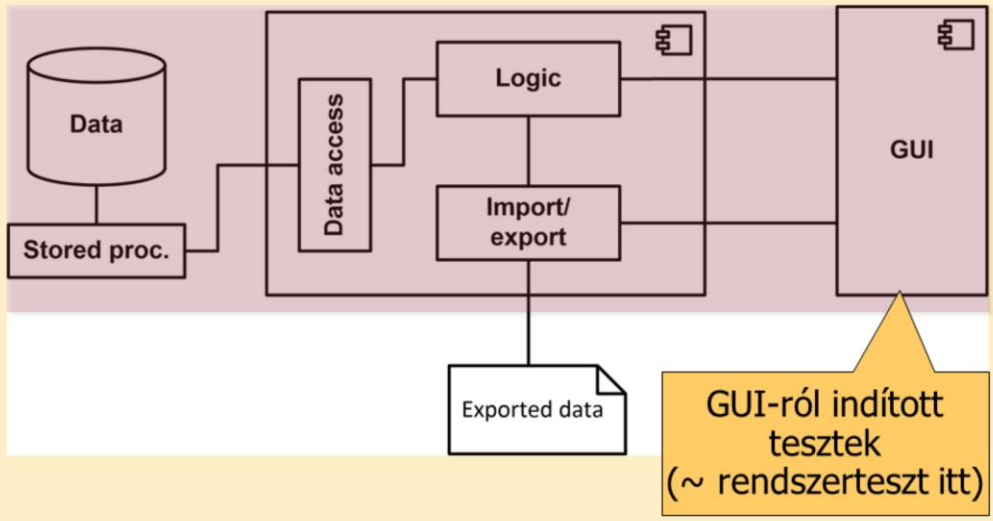
Utolsó módosítás: 2014.10.12.

Példa: hol/hogyan/mit lehet tesztelni?



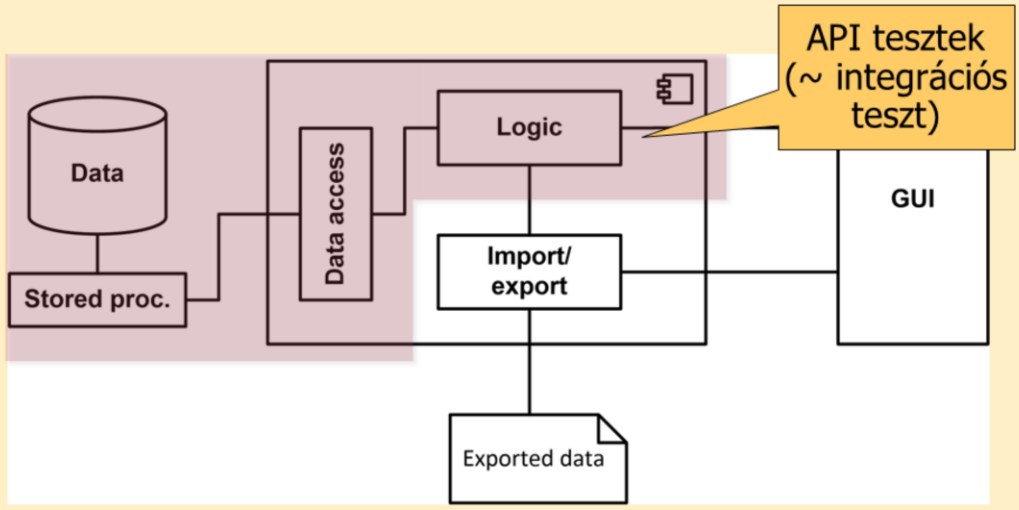
2

Példa: hol/hogyan/mit lehet tesztelni?

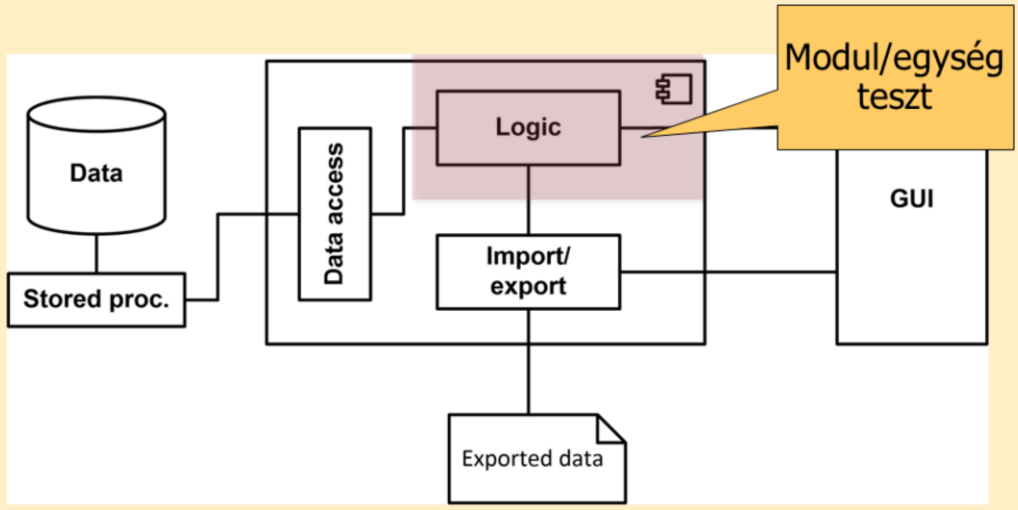


3

Példa: hol/hogyan/mit lehet tesztelni?



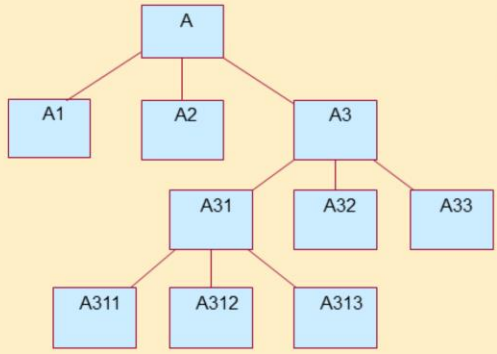
Példa: hol/hogyan/mit lehet tesztelni?

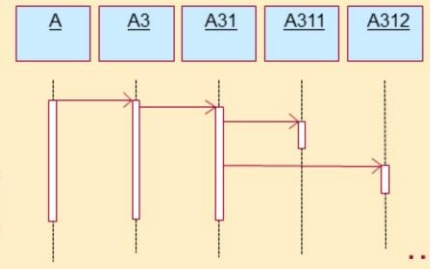


5

Modul / egység tesztelés

- Modul / Egység (Unit):
  - Logikailag egy egységként kezelhető elem
  - Jól meghatározott interfésszel rendelkezik
  - OO fejlesztés: Osztály (csomag, komponens is lehet)
- Modul hívási hierarchia (ideális eset):





6

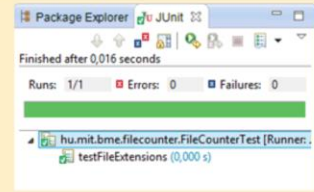
Miért van szükség modultesztelésre?

- Cél: Hibák gyors kijavítása a fejlesztés után (legalacsonyabb szinten)
  - Integrációs fázis hatékonyabb, ha a modulok teszteltek
  - A modul fejlesztője által leggyorsabb a javítás
- Modulok külön-külön tesztelhetők
  - Komplexitás kézbentartható
  - Hibák helye egyszerűbben felderíthető, javítás olcsóbb
  - Párhuzamosítható folyamat
- Egységtesztek jellegzetességei
  - A kód egy-egy speciális funkcióját ellenőrzi
  - „Szerződés” ellenőrzése a modul működéséről
  - Példaként szolgálhat az egység használatához
  - (Nem feltétlenül automatikus)

7

Egységteszt futtató keretrendszerek

- Egységtesztek: Gyakran kell futtatni
  - Fejlesztés közben, kód változásakor (pl. refactoring)
  - Környezet változása esetén
  - → Épp ezért gyorsnak kell lennie
- Jó eszköztámogatás
  - Keretrendszerek (JUnit, xUnit, TestNG, …)
  - Támogatás az IDE-kben (Eclipse, VS, ...)
- Általában egyszerű funkcionalitású eszközök
  - Tesztek és ellenőrzések megadása
  - Tesztek futtatása
  - Eredmény megjelenítése (red-green)



Egy egyszerű JUnit teszt

```
public class ListTest{
    List list; // SUT

    @Before public void setUp(){
        list = new List();
    }

    @Test public void add_EmptyList_Success(){
        list.Add(1);
        assertEquals(1, list.getSize());
    }
}
```

Teszt előkészítése

SUT meghívása

Ellenőrzés

JUnit annotációk (részlet)

Annotáció	Leírás
`@Test`	Teszt metódus definiálása
`@Before`	Minden teszt előtt lefut, pl. teszt környezet előkészítése
`@After`	Minden teszt után fut le
`@BeforeClass`	Csak egyszer fut le az összes teszt előtt (vigyázat, tesztek nem lesznek függetlenek)
`@AfterClass`	Tesztek befejezése után fut le, pl. takarítás
`@Ignore`	Kihagyja az adott tesztet, pl. akkor érdemes használni, ha megváltozott a kód
`@Test(expected=IllegalArgument Exception.class)`	Akkor sikeres a teszt, ha ezt a kivételt dobja
`@Test(timeout=100)`	Teszt futási idejét korlátozza

További: Suite létrehozása, Category, Parametrized…

10

Részletesebben lásd pl.: http://www.vogella.de/articles/JUnit/article.html

Jó egységteszt: ajánlások (!)

- Egyszerű, megbízható
  - Nincs benne bonyolult logika (pl. ciklusok, try/catch)
- Egy teszt egy dolgot vizsgál
  - (de ez nem mindig 1 assert hívás)
- Ez is jó minőségű kód
  - Duplikáció elkerülése
  - Jól áttekinthető, módosítható
- Tesztek függetlenek egymástól
- Ellenőrzések nincsenek túlspecifikálva
- …

11

Multiple assert:

- Roy Osherove: „My guideline is usually that you test one logical CONCEPT per test. you can have multiple asserts on the same *object*. they will usually be the same concept being tested.”
- http://xunitpatterns.com/Assertion%20Roulette.html
- An NUnit Addin for Running One Assert Per Test, http://rauchy.net/oapt/

Egységteszt konvenciók

- Teszt osztály neve: `[Unit_neve]Test`
- Teszt metódus neve:
  - `Method_StateUnderTest_ExpectedBehavior`
  - `MethodName_DoesWhat_WhenTheseConditions`
  - `[feature being tested]`
- Teszt szerkezete:

```
// Arrange        // Given
// Act            // When
// Assert         // Then
```

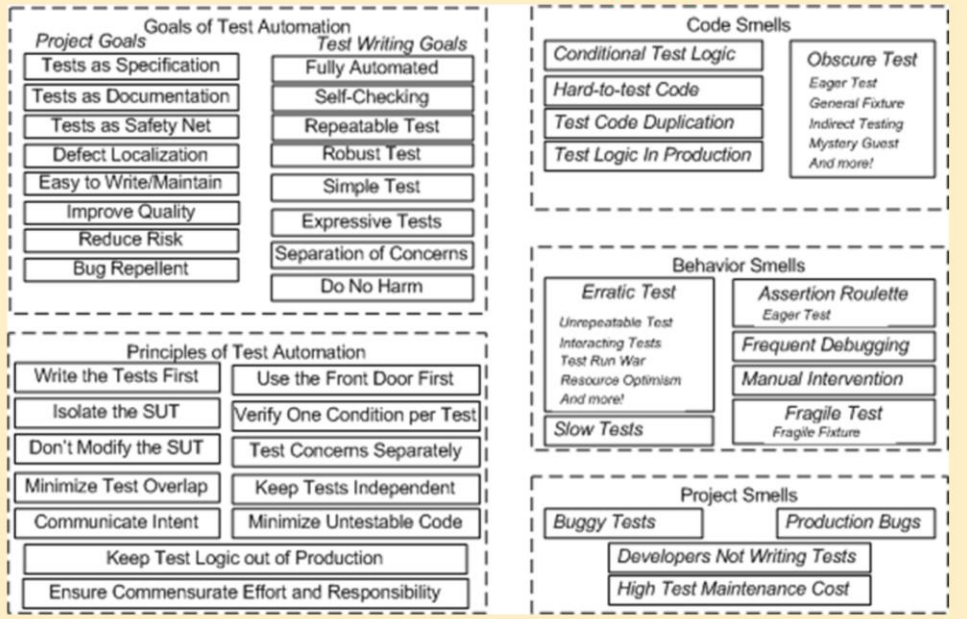
12

Elnevezési konvenciókat lásd pl.:

- Unit test naming best practices, http://stackoverflow.com/questions/155436/unit-test-naming-best-practices
- What are some popular naming conventions for Unit Tests?, http://stackoverflow.com/questions/96297/what-are-some-popular-naming-conventions-for-unit-tests
- Testing on the Toilet: Writing Descriptive Test Names, http://googletesting.blogspot.com/2014/10/testing-on-toilet-writing-descriptive.html

A Given/When/Then a Behavior-Driven Development (BDD) esetén használt elnevezés, de nagyjából ugyanaz a javaslat.

xUnit Test Patterns könyv ajánlásai



Forrás: Gerard Meszaros, http://xunitpatterns.com/

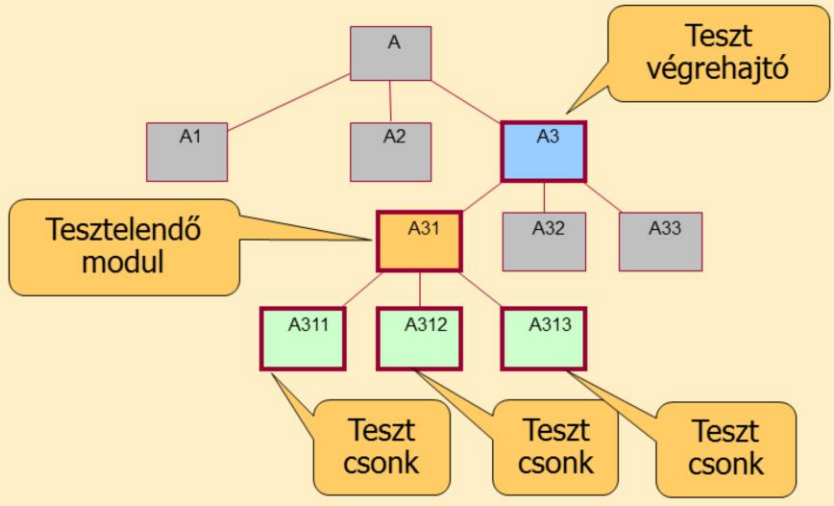
13

Forrás: http://www.xunitpatterns.com

IZOLÁCIÓ MEGVALÓSÍTÁSA

Modulok izolált tesztelése

- Modulokat egyenként, elszigetelten teszteljük
- Teszt végrehajtó és teszt csonkok szükségesek



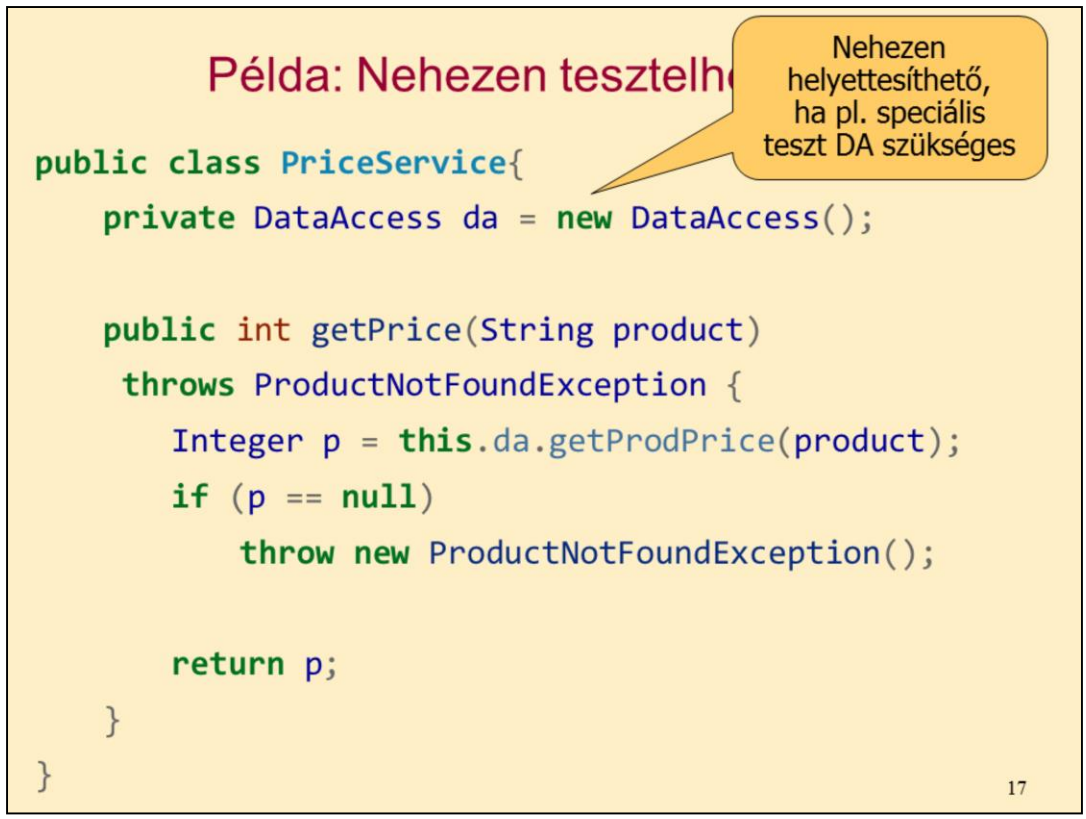
15

Probléma: Függőségek kezelése

- Mi tekinthető függőségnek?

  Bármi, amivel
  - a SUT együttműködik,
  - de nem hozzá tartozik (nem közvetlenül befolyásolja)
- Példák:
  - Másik modul
  - Fájlrendszer hívás
  - Dátum lekérdezése
  - …

16



A kód csak illusztráció jellegű!

Példa: SUT tesztelhetővé tétele

```
public class PriceService{
    private IDataAccess da;

    public PriceService(IDataAccess da){
        this.da = da;
    }

    public int getPrice(String product)
     throws ProductNotFoundException {
        Integer p = this.da.getProdPrice(product);
        if (p == null)
            throw new ProductNotFoundException();
        return p;
    }
}
```

Implementáció átadása pl. a konstruktorban

18

A konkrét implementációt sok féle módon lehet átadni:

-konstruktor

-új tulajdonságot felvenni a PriceService-be

-készíteni egy factory-t, és az gyártja le az épp szükséges DataAccess komponenssel rendelkező PriceService-t

-…

Példa: Egyégtesztek a SUT-hoz

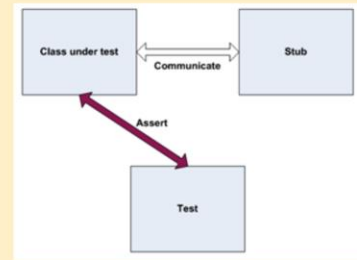
Tesztben egy csonk használata

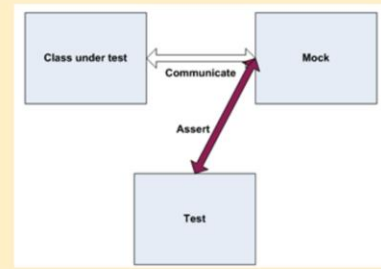
```
public class PriceServiceTest{
    @Before public void init(){
        DataAccessStub das = new DataAccessStub();
        das.add("A100", 50);
        ps = new PriceService(das);
    }

    @Test public void SuccessfulPriceQuery(){
        int p = ps.getPrice("A100");
        assertEquals(50, p);
    }

    @Test(expected = ProductNotFoundException.class)
    public void NotExistingProduct(){
        ps.getPrice("notExists");
    }
}
```

19

Stub, Mock, Dummy, Fake, … objektumok

- Sokféle technika a helyettesítésre
  - Eltérő elnevezések (ld. *xUnit Patterns* könyv)
  - Összefoglaló név: Test double
    - Helyettesítő elem teszteléshez
- Stub (csonk)
  - SUT állapotának ellenőrzésére
  - Rögzített válaszok adott hívásokra
- Mock
  - SUT interakcióinak ellenőrzésére
  - Elvárt és ellenőrzött viselkedés
- Dummy
  - Nem használt („kitöltő”) objektum
- Fake
  - Működő, de nem az „éles” objektum





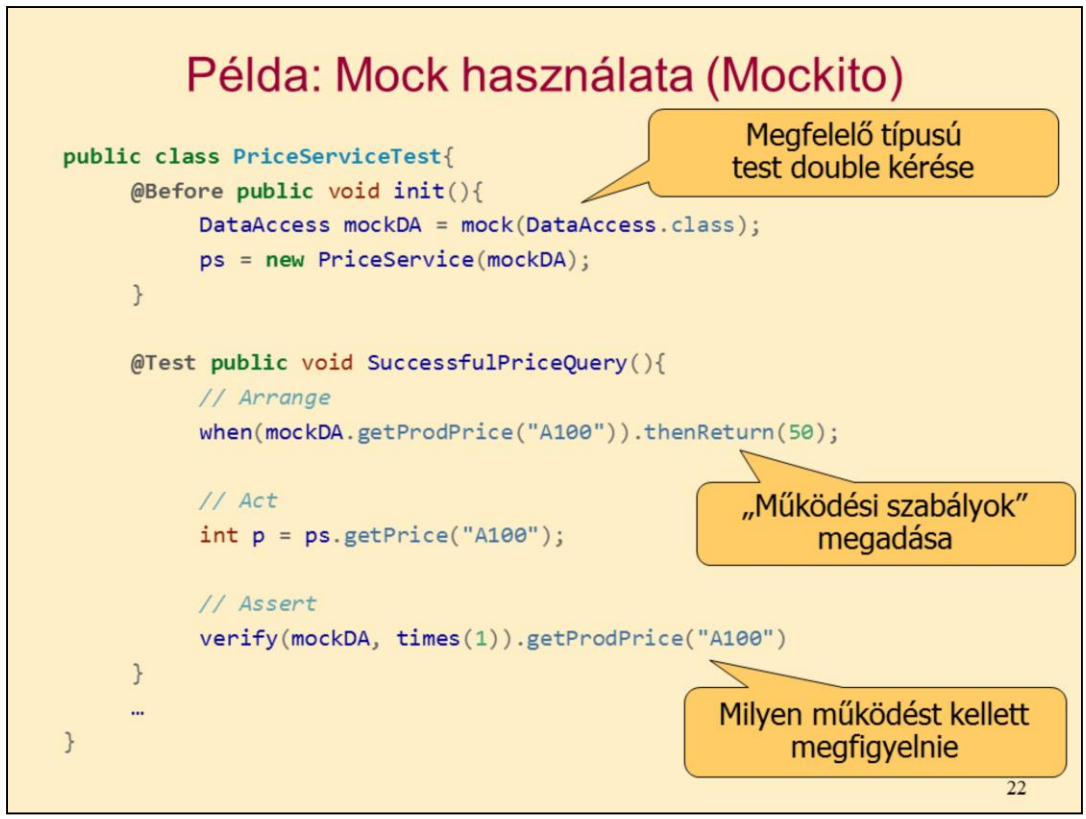
Forrás: Roy Osherove, The Art of Unit Testing: With Examples in .Net, Manning Publications; 1 edition (June 3, 2009), URL: http://artofunittesting.com/

Lásd még: „Testing on the Toilet: Testing State vs. Testing Interactions”, http://googletesting.blogspot.hu/2013/03/testing-on-toilet-testing-state-vs.html

Izolációs keretrendszer (Isolation framework)

- Csonkok és mockok kézi elkészítése nehézkes
  - Sok manuális munka
  - Karbantartása időigényes lehet
- Keretrendszerek:
  - Osztály/interfész leírás alapján
  - Dinamikus csonkok vagy mockok készítése
- Példák:
  - JMock, Mockito, Rhino Mocks, Typemock…

21



Egy jóval részletesebb példa a Mockito keretrendszer használatáról:

-Brett L. Schuchert, Mockito.LoginServiceExample, URL: http://schuchert.wikispaces.com/Mockito.LoginServiceExample

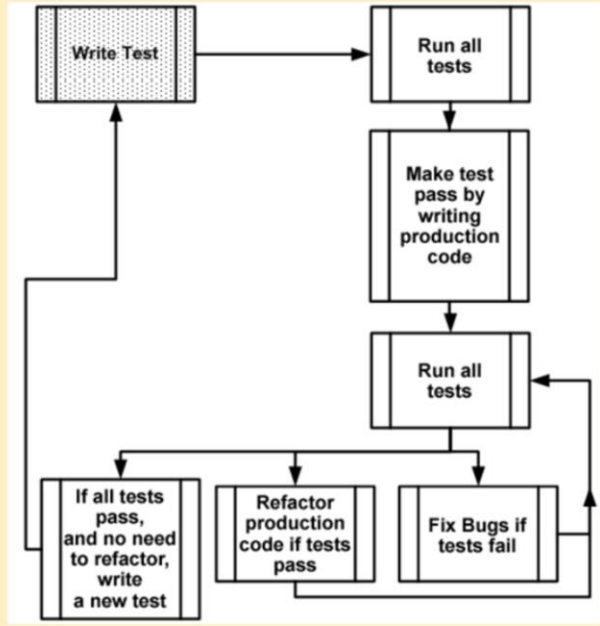
Megéri ilyen egységteszteket készíteni?

Egy példa pilot projekt számai:

Stage	Team without tests	Team with tests
Implementation (coding)	7 days	14 days
Integration	7 days	2 days
Testing and bug fixing	Testing, 3 days Fixing, 3 days Testing, 3 days Fixing, 2 days Testing, 1 day Total: 12 days	Testing, 3 days Fixing, 1 day Testing, 1 day Fixing, 1 day Testing, 1 day Total: 8 days
Overall release time	26 days	24 days
Bugs found in production	71	11

Forrás: Roy Osherove, The Art of Unit Testing, 2009.

Test-Driven Development (TDD)



24

Olvasnivalók

- Martin Fowler: Mocks Aren't Stubs, 2007
  - URL: http://martinfowler.com/articles/mocksArentStubs.html
- Martin Fowler: UnitTest,
  - http://martinfowler.com/bliki/UnitTest.html
- Roy Osherove: The Art of Unit Testing: With Examples in .Net.
  - Manning Publications; 1st edition (June 3, 2009)
  - URL: http://artofunittesting.com/
- Brett L. Schuchert: Mockito.LoginServiceExample
  - URL: http://schuchert.wikispaces.com/Mockito.LoginServiceExample