Szoftverellenőrzési technikák (vimim148)

Tesztelés a fejlesztés különböző fázisaiban

Majzik István, Micskei Zoltán

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

1

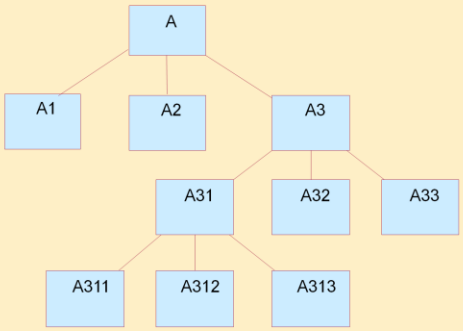
Utolsó módosítás: 2010.10.25.

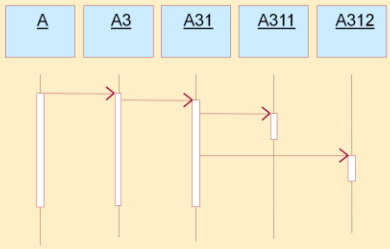
Tartalom

- Modul / Unit tesztelés
- Integrációs tesztelés
- Rendszer tesztelés
- Elfogadás tesztelés

Modul / Unit

- Modulok:
  - logikailag egy egységként kezelhető elemek
  - jól meghatározott interfésszel rendelkeznek
  - OO fejlesztés: Osztályok (csomagok, komponensek)
- Modul hívási hierarchia (ideális eset):





3

Miért van szükség modultesztelésre?

- Legalacsonyabb szintű tesztelés
  - integrációs fázis hatékonyabb, ha a modulok teszteltek
- Modulok külön-külön tesztelhetők
  - komplexitás kézbentartható
  - hibák helye egyszerűbben felderíthető, javítás olcsóbb
  - párhuzamosítható folyamat

4

Modul / Unit tesztelés jellegzetességei

- Unit teszt
  - a kód egy speciális funkcióját ellenőrzi
  - „szerződés” a modul működéséről
  - példaként szolgálhat az egység használatához
  - (nem feltétlenül automatikus)
- Gyakran futtatjuk
  - Fejlesztés közben, kód változásakor (refactoring)
  - Környezet változása során
  - Épp ezért gyorsnak kell lennie
- Általában a modul fejlesztője írja

Unit teszt keretrendszerek

- Jó eszköztámogatás
  - Keretrendszerek (JUnit, xUnit, TestNG…)
  - Támogatás az IDE-ben (Eclipse, Netbeans..)
- Egyszerű funkcionalitás általában
  - Tesztek és ellenőrzések megadása
  - Tesztek futtatása
  - Eredmény megjelenítése (red-green)

6

Egy egyszerű JUnit teszt

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

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

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

Test fixture

SUT meghívása

Ellenőrzés

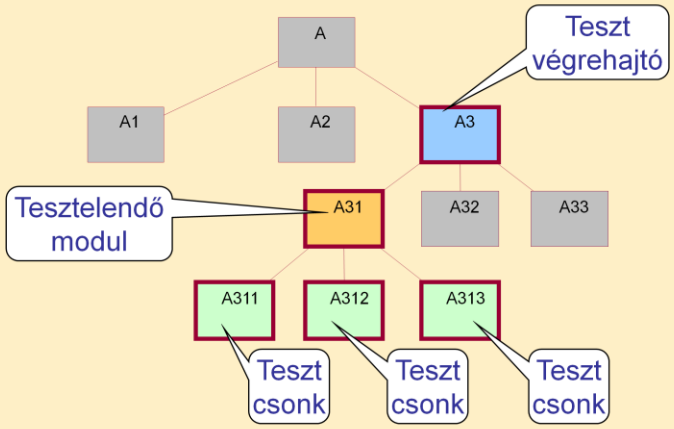
7

Jó unit teszt jellegzetessége

- Egyszerű, megbízható
  - Nincs benne logika (ciklus, feltétel)
- Egy teszt egy dolgot vizsgál
- Ez is jó minőségű kód
  - Duplikáció elkerülése
  - Olvasható
- Tesztek függetlenek egymástól
- Ellenőrzések nincsenek túlspecifikálva
- …

Cél: Modulok izolációs tesztelése

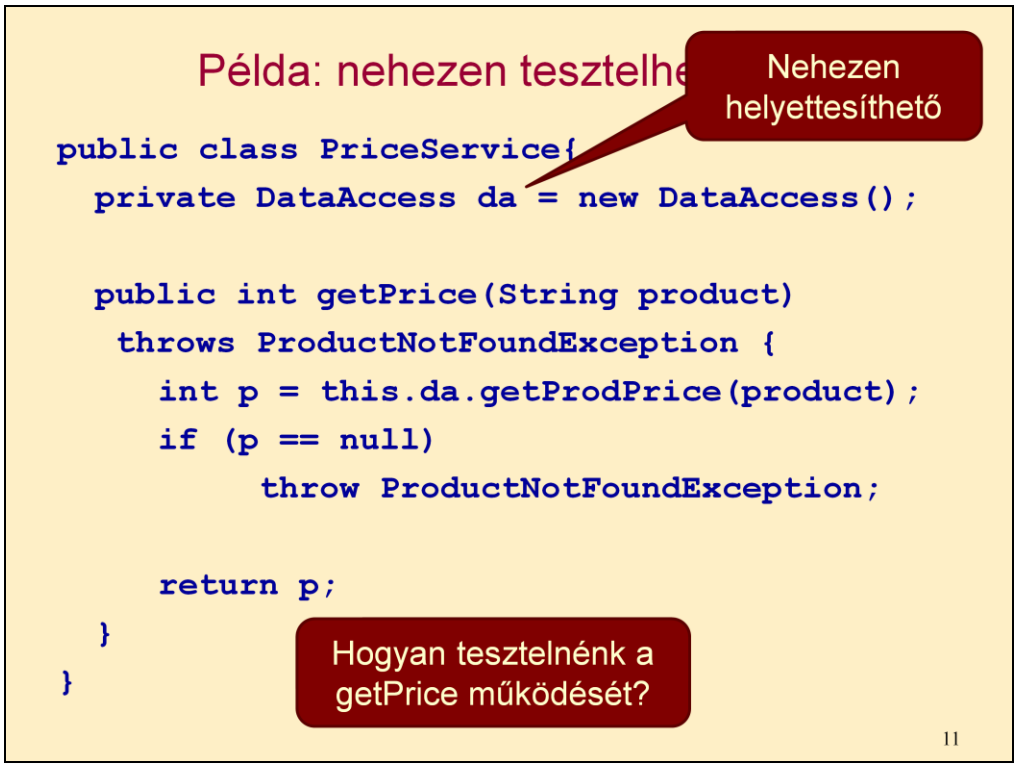
- Modulok egyenként, elszigetelten teszteltek
- Teszt végrehajtó és teszt csonkok szükségesek



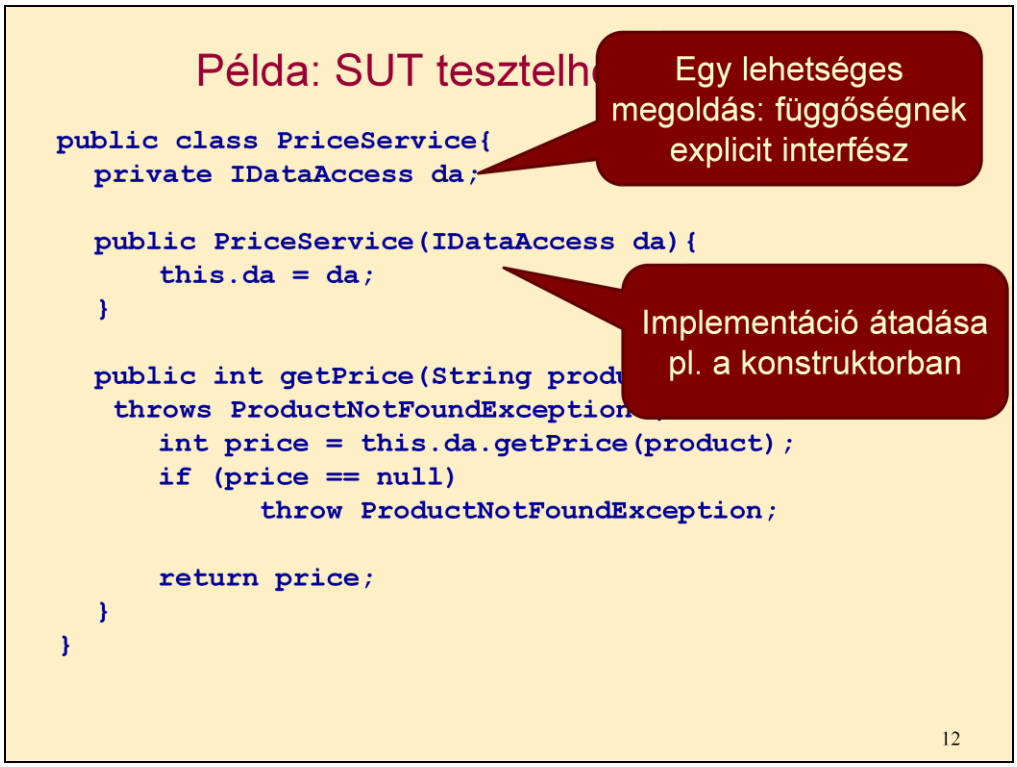
9

Probléma: függőségek kezelése

- **Függőség**: bármi, amivel
  - a SUT együttműködik, de
  - nem tudjuk befolyásolni
- Példa:
  - másik modul,
  - fájlrendszer hívás,
  - hét napjának lekérdezése,
  - …



A kód csak illusztráció jellegű!



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

-…

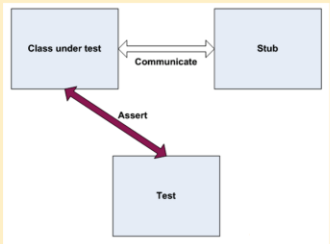
Példa: unit tesztek a

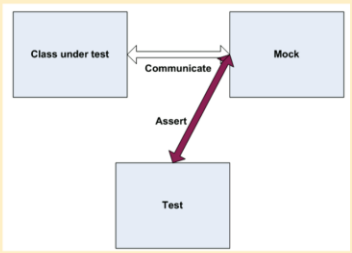
Tesztben az igazi helyett teszt csonk használata

```
public class PriceServiceTests{
   @Before init(){
       DataAccessStub das = new DataAccessStub();
       da.Add(„A100”, 50);
       PriceService ps = new PriceService(das);
   }

   @Test public void testSuccessfulPriceQuery(){
       int p = ps.getPrice(„A100”);
       assertEquals(50, p);
   }

   @Test(expected = ProductNotFoundException.class)
   public void testNotExistingProduct(){
       ps.getPrice(„notExists”);
   }
}
```

Mocks, Fakes, Stubs, Dummies…

- Sokféle technika a helyettesítésre
  - Eltérő elnevezések, lásd: xUnit Patterns
  - összefoglaló név: test double
- Stub
  - SUT állapotának ellenőrzése
  - Nem hiúsíthatja meg a tesztet
- Mock Object
  - SUT interakcióinak ellenőrzése





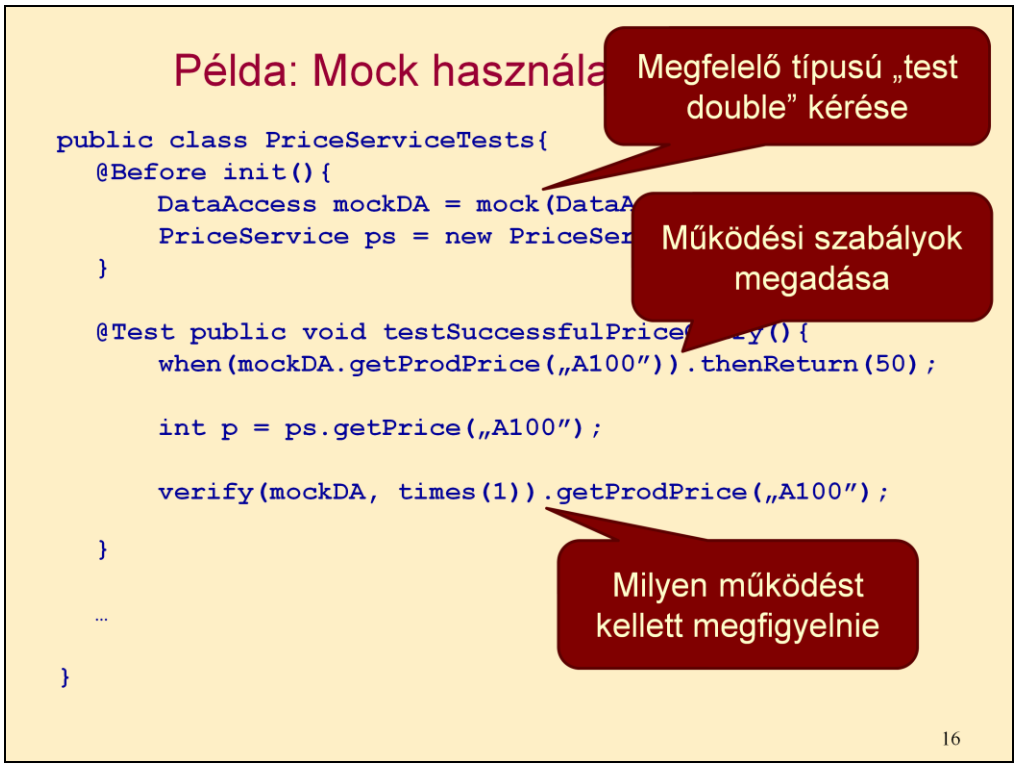
Képek forrása:

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

Isolation frameworks

- 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…

15



Egy jóval részletesebb példa a Mockito keretrendszer használatával:

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

Megéri ilyen unit teszteket 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.

17

Olvasnivalók

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

Tartalom

- Modul / Unit tesztelés
- **Integrációs tesztelés**
- Rendszer tesztelés
- Elfogadás tesztelés

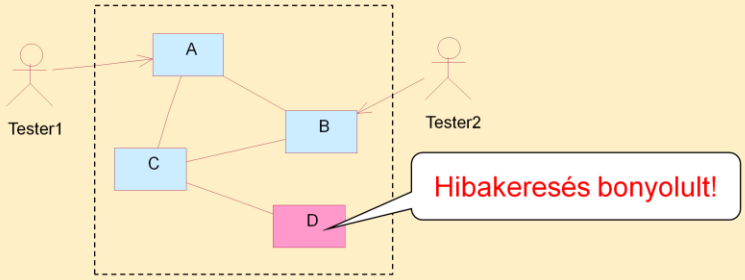
Integrációs tesztelés

Modulok együttműködésének ellenőrzése

- Kapcsolódási interfészek tesztelése
  - A rendszer annak ellenére hibás lehet, hogy minden modul egyenként hibátlan!
- Módszerek:
  - Funkcionális tesztelés: Forgatókönyvek tesztje
    - Ez sokszor a specifikáció része (scenario)
  - (Strukturális tesztelés csak modulszinten!)
- Megközelítés:
  - “Big bang”: minden modult egyszerre integrálni
  - Inkrementális: egyenként összerakni a modulokat

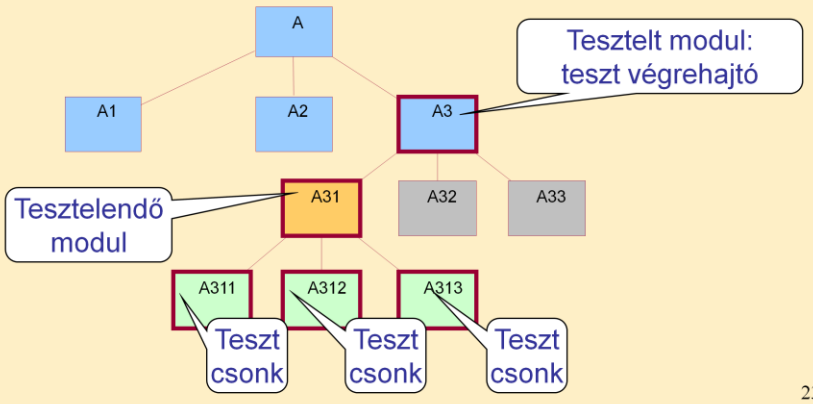
“Big bang” integrációs tesztelés

- Tesztelés a külső interfészeken keresztül
- Külső teszt végrehajtó
- Rendszer funkcionális specifikáció alapján történik
- Modul módosítás: Újratesztelés
- Kis rendszerek esetén alkalmazható



Felülről lefelé történő integrációs tesztelés

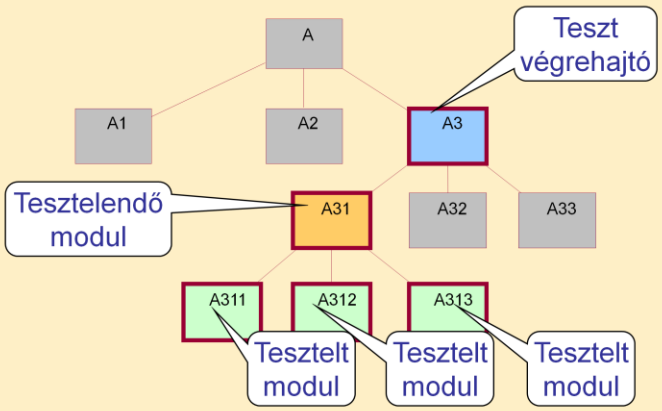
- Modulok a hívó modulokból kerülnek tesztelésre
- Csonkok helyettesítése tesztelendő modulokkal
- Erősen követelmény-orientált (“fentről” tesztelünk)
- Modul módosítás: alatta lévők tesztelését módosítja



23

Alulról felfelé történő integrációs tesztelés

- Tesztelendő modul a már teszteltek használja
- Teszt végrehajtó szükséges
- Integrációval párhuzamosan megtehető
- Modul módosítás: felette lévők tesztjére hatással van



24

Felülről lefelé vs. alulról felfelé

- Felülről lefelé
  - + Hamar összeállhat egy demonstrálható „szkeleton”
  - - Csonkok készítése általában nehezebb
  - - Teszt bemenetek távol lehetnek az épp integrálandó modultól
- Alulról felfelé
  - + Könnyebb megfigyelni és irányítani a teszteket
  - - A rendszer maga csak a legvégén áll össze

25

Futtató rendszer integrációja

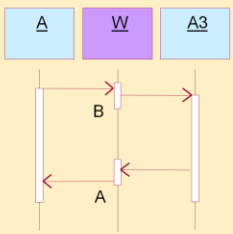
Motiváció: Nehéz csonkokat írni a futtató rendszerhez (OS, J2EE konténer…)

Stratégia:

1. Alkalmazás modulok integrációja felülről lefelé, a futtató rendszer szintjéig
2. Futtató rendszer alulról felfelé történő tesztelése
   - Funkciók izolációs tesztje (ha szükséges)
   - „Big bang” tesztelés az alkalmazás hierarchia legalsó rétegével
3. Alkalmazás és futtatórendszer integrációja, a felülről lefelé történő tesztelés befejezése

Eszközök az integrációs teszteléshez

- Wrapper (csomagoló) kódrészletek
  - „before” wrapper: A hívás végrehajtása előtt
    - paraméterek vizsgálata
    - hívási szekvencia ellenőrzése
  - „after” wrapper: A hívás végrehajtása után
    - visszaadott érték ellenőrzése vagy módosítása
  - „replace” wrapper: A hívott helyettesítése
    - teszt csonk megvalósítás



27

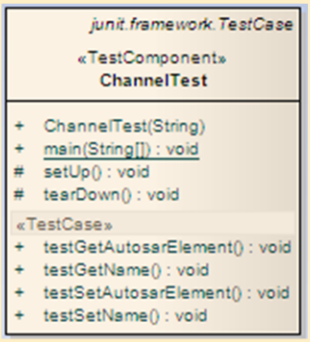
Tartalom

- Modul / Unit tesztelés
- Integrációs tesztelés
  - Integrációs tesztelési megközelítések
  - **Tesztek leírása: U2TP**
- Rendszer tesztelés
- Elfogadás tesztelés

UML 2.0 Testing Profile (U2TP)

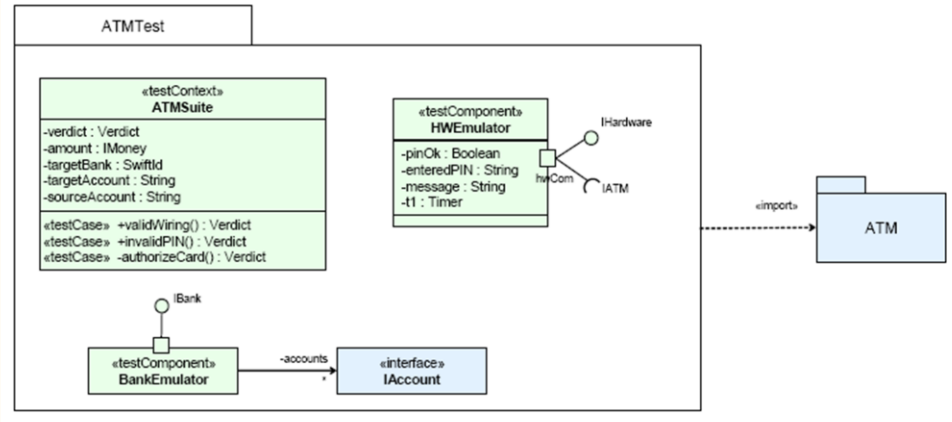
- UML Profile elemei:
  - sztereotípia
  - tagged value

- UML 2.0 Testing Profile elemei:
  - Test Architecture
  - Test Behaviour
  - Test Data



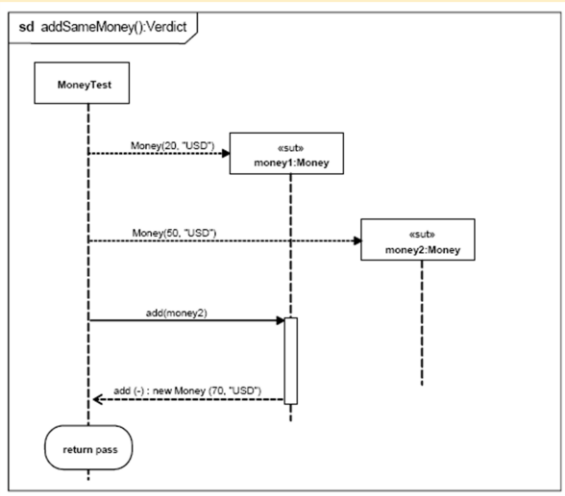
U2TP: Test Architecture

- *TestComponent, TestContext, SUT, Arbiter*
- Példa: nem triviális teszt architektúra:



U2TP: Test Behavior

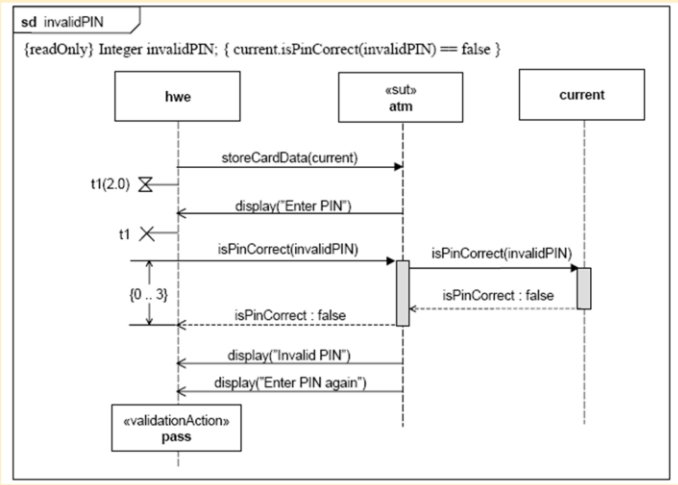
- *TestObjective, TestCase, Default, Verdict…*



31

U2TP: Test Data, Time

- *DataPool, DataSelector, CodingRule, Timer, Timeout*



32

Specifying timeouts on a Sequence Diagram

Tartalom

- Modul / Unit tesztelés
- Integrációs tesztelés
- **Rendszer tesztelés**
- **Elfogadás tesztelés**

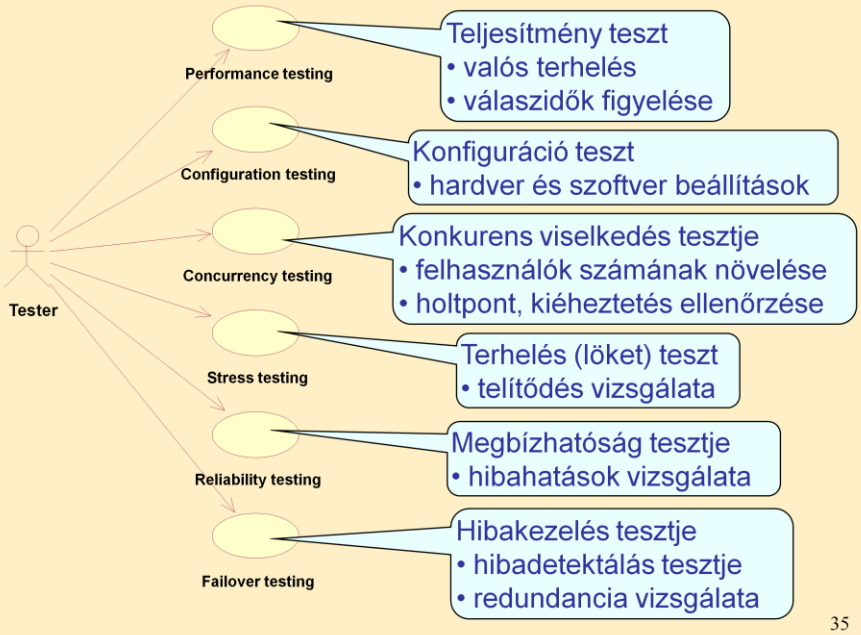
3. Rendszertesztelés

Tesztelés a rendszerszintű specifikáció alapján

- Jellemzők:
  - hardver-szoftver integráció után végezhető
  - funkcionális tesztek +
    nem-funkcionális jellemzők tesztje is!
- Kiemelhető:
  - Adat integritás vizsgálata
  - Felhasználói profil figyelembe vétele (terhelés)
  - Rendszer alkalmazhatósági korlátok megállapítása (erőforrás-használat, telítődés)
  - Hibahatások vizsgálata

34

Rendszerteszt típusok



35

Tesztelés a fejlesztés különböző fázisaiban

1. Modultesztelés
   - izolációs tesztelés
2. Szoftver integráció tesztelése
   - „big bang” tesztelés
   - top-down (felülről-lefelé) tesztelés
   - bottom-up (alulról-felfelé) tesztelés
   - futtató környezet integrációja
3. Rendszertesztelés
   - teljes rendszer együttes tesztelés
4. Validációs tesztelés
   - felhasználói elvárások
   - környezeti szimuláció

36