Szoftvertesztelés

Tárgyfelelős: Dr. Bednarik László
Szak: Mérnök informatikus alapszak levelező
Kód: GEIAL31H-BL
Évfolyam: III.

ÜTEMTERV

Hét	Elmélet	Gyakorlat
1.	Követelmények ismertetése, a tesztelés alapjai, tesztelés helye a szoftver életciklusában	Csoportok kialakítása
2.	Szoftverek minőségbiztosításának alapjai	Szoftverek minőségi kritériumai és tesztelési alapelvek
3.	Funkcionális tesztelés	Használt technológiák ismertetése
4.	JUnit alapjai	JUnit gyakorlása

Kötelező irodalom:

[1] Ficsor Lajos, Dr. Kovács László, Krizsán Zoltán, Dr. Kusper Gábor: Szoftvertesztelés

Ajánlott irodalom:

- Horváth László: Szoftvertesztelés a gyakorlatban

A tárgy lezárásának módja: aláírás, vizsgajegy

Évközi számonkérések:

- A félév során a hallgatók csoportokat alkotnak.
- Két feladatot kell elkészíteni az adott témakörben (tervezés, tesztelés)

Aláírás megszerzésének feltételei:

- Aktív részvétel a csoportos feladatok megoldásában.
- Az egyéni feladatok sikeres megvédése.

Pótlás módjai:

- Az egyéni feladatok pótlása az utolsó szorgalmi héten történik.

Vizsga formája: írásbeli és szóbeli

Írásbeli: A dolgozat elkészítésére 1 óra áll rendelkezésre, a vizsga első része egy 3 elméleti kérdésből álló, 10-10 pontért.

Szóbeli: Az érdemjegy a szóbeli vizsgán kerül meghatározásra. A szóbeli vizsga a félév elméleti és gyakorlati anyagából áll.

Az írásbeli és szóbeli rész értékelése:

- 0%-50%: elégtelen
- 51%-60%: elégséges
- 61%-70%: közepes
- 71%-80%: jó
- 81%-100%: jeles.

Elégtelen írásbeli elégtelen vizsgajegyet jelent. A szóbelin a megjelenés kötelező.

Szoftvertesztelés **Név, tankör:**

Levelező tagozat vizsgadolgozat **Neptunkód:**

Minta vizsga feladatsor

1. A tesztelés alaptípusai
2. A tesztelés helye a szoftver életciklusában
3. Agilis szoftverfejlesztés módszertanai

Szoftvertesztelés **Név, tankör:**

Levelező tagozat vizsgadolgozat **Neptunkód:**

Vizsga javítókulcs

1. A tesztelés alaptípusai, felsorolás, rövid jellemzés!

„A tesztelési technikákat csoportosíthatjuk a szerint, hogy a teszteseteket milyen információ alapján állítjuk elő.
Feketedobozos (black-box) vagy specifikáció alapú, amikor a specifikáció alapján készülnek a tesztesetek.
Fehérdobozos (white-box) vagy strukturális teszt, amikor a forráskód alapján készülnek a tesztesetek.
Feketedobozos tesztelés: a tesztelő nem látja a forráskódot, de a specifikációkat igen.
Fehérdobozos tesztelés: a forráskód rendelkezésre áll.
A tesztelés szintjei:

Komponensteszt: csak a rendszer egy komponensét teszteli önmagában

Integrációs teszt: kettő vagy több komponens együttműködési tesztje

Rendszerteszt: a rendszerteszt az egész rendszert, tehát minden komponenst együtt, teszteli.

Átvételi teszt: a felhasználók a kész rendszert tesztelik.

2. A tesztelés helye a szoftver életciklusában!

Módszertanok feladata

Meghatározzák, hogy a szoftverfejlesztés életciklusának egyes lépései milyen sorrendben kövessék egymást, milyen dokumentumokat, szoftvertermékeket állítsunk elő és hogyan.
Az első szempont, hogy milyen sorrendben követik egymást a szoftver életciklusának fázisai.
Módszertanok a tesztelés részéről.
A V-modell (V-Model vagy Vee Model) a nevét onnan kapta, hogy két szára van és így egy V betűhöz hasonlít.
Az egyik szára megegyezik a vízesés modellel. Ez a fejlesztési szár.
A másik szára a létrejövő termékek tesztjeit tartalmazza. Ez a tesztelési szár.
V-modell működése
1. V-modell változatban először felmérjük az igényeket és elkészítjük a követelmény specifikációt.
2. A követelmény specifikációban jól meghatározott átvételi kritériumokat fogalmaznak meg, amik lehetnek funkcionális és nemfunkcionális igények is.
3. A funkcionális specifikáció leírja, hogyan kell majd működnie a szoftvernek. Ez lesz a rendszerteszt alapja.

3. Agilis szoftverfejlesztés módszertanai
Az agilis szoftverfejlesztésnek nagyon sok fajtája van. Két legismertebb módszertan:
eXtrém Programozás (XP)
Scrum
Közös jellemzők:
Kevesebb dokumentáció.
Növekvő rugalmasság, csökkenő kockázat.
Könnyebb kommunikáció, javuló együttműködés.
A megrendelő bevonása a fejlesztésbe.

Agilis módszertan

- elfogadja a gyakori változást,
- iteratív,
- gyakran objektumorientált,
- prototípus alapú,
- rapid,
- esetleg extrém,
- könnyűsúlyú,

- követelmény- és csapatközpontú.

Mikor alkalmazzák?

Kis rendszerek/ kis projekt

Kevés fejlesztő

Alacsony kritikussági fok

Gyakran változó követelmények

Csak gyakorlott, szenior fejlesztők esetén

Mikor nem?

Nagy és elosztott rendszerek

Kritikus rendszerek

Hosszú élettartamú rendszerek”[1]