Kiss Máté / DLN82Q - 2021.01.22

1) Agilis szoftverfejlesztés tesztelési lehetőségei, jellemezzen két módszertant.

a) Scrum

- nagalmas szoftverfejlesztési módszertan
- alkalmazkodik az esetleges késési változtatásokhoz
- a fejlesztési, tesztelési fázisokat kb. kétheles intervallumokra osztják, ezek a sprintek
- minden sprint előtt van egy megbeszélés arról, hogy ki mit és milyen határidővel tud elvállalni. Valamint, hogy mi az ami nem készült meg az előző sprintben és miért.
- 5-9 fős csapatokban dolgoznak.
- minden nap elején van egy 15 perces csapat megbeszélés arról, hogy ki mit csinált az előző nap.
- vannak „disznók" és „csirkék"
  - disznók között van a Scrum mester, aki felügyeli és irányítja a munkamenetet. Valamint a fejlesztők
  - csirkék között van a megrendelő, és a menedzsment.

Kiss Máté / IDLN82 - 2021.01.22

b) TDD - Test Driven Development

A tesztelési eljárás lényege, hogy először egy tesztesetet kell írni, ami biztosan hibásan fog lefutni. (1)
Ezután csak annyit kell fejleszteni a kódon, hogy az előzőleg írt tesztet lefussa. (2)
Majd a Program újragondolásán után az (1)-es ponttól ismételni. Ez fog szekvenciálisan újra futni.
Kezdetben lassúnak tűnhet.

Kiss Máté/DLN82Q - 2021.01.22

2) Rendszerteszt.

A rendszerteszt az egész rendszert együttesen tesztelni, hogy megfelel-e: - funkcionális specifikációnak
- rendszerterveknek
- hogy kiadható-e a megrendelőnek
- hogy minden komponenst teszteltek-e

A feladatban rendszergazdák tesztelik fehérdobozos teszteléssel. Tehát nem ismerik a forráskódot, így szükséges az indító fájl. Adott bemenet-kimenet párokat tesztelnek.

Átvételi teszt.

Szintén a teljes rendszert teszteli, de már a végfelhasználók által.

fajtái:

- alfa teszt: A fejlesztő cégnél végzik, de nem a fejlesztő csapat
- béta teszt: végfelhasználók szűk csoportja végzi. Ők jelentik az esetleges hibákat
- felhasználói átvételi teszt: Majdnem az összes végfelhasználó megkapja a szoftvert. Éles körülmények között tesztelik
- rendszergazdai teszt: A rendszergazdák ellenőrzik a biztonsági funkciókat. biztonsági mentés és visszaállítás.

3) Tesztelés JUnit segítségével, részei, annotációk, TDD

A JUnit (Java) keretrendszer segít az automatizált tesztelés megvalósításában. A használatához szükséges a hozzá tartozó dependency beillesztése a pom.xml fájlba.
Részei: tesztosztályok és teszt esetek.
A tesztek futtatásához szükségesek az annotációk, melyeket a @ jellel ellátott @Test hajt végre.

Annotációk fajtái:

- @Test - teszteset, tesztek futtatásakor itt hajtódnak végre a megírt tesztek.
- @Before - minden teszteset előtt lefut. Feladata a kezdeti értékek beállítása
- @After - minden teszteset után lefut. Feladata a módosított értékek alaphelyzetbe állítása

A tesztesetekben, a tesztelést az Assert osztályok segítségével lehet megvalósítani.

Pl.: assertEquals(boolean, boolean)
assertNotNull(int))
:

Futási eredmények: Sikeres(Pass), Elbukott(Failure), Hiba(Error)

Kiss Máté/DLN82C - 2021.01.22

3) TDD - Test Driven Development

- kezdetben lassúnak tűnhet
- A tesztelési eljárás lényege, hogy:
- először egy tesztesetet kell írni, mely biztosan hibára fut. (1)
- majd a kódon annyit írni, hogy az előzőleg megírt tesztesetet lefussan. (2)
- ezután a program további gondolásával ismételni kell az (1)-es ponttól. Ami így szekvenciálisan ismétlődik.

4) Kiss Máté / DLN82C - 2021. 01. 22.

- ~~Az életciklus minél hamarabbi részén el kell végezni.~~
- nem lehet minden bemeneti opciót ~~tesztelni~~.
- érdemes

Szintek:

1. - Komponenstesz - csak egységeket tesztel

2 - Integrációs teszt - komponens - komponens, vagy a rendszer - rendsz

3 - Átvételi teszt

4 - Rendszerteszt