Szoftver fejlesztő vizsga

szóbeli tétel

(2.tétel)

Készítette

Nagy Ferenc

Közreműködött

Laki László
Molnár Viktor
Wahl Zoltán
Zorinátz Orsolya

[2020-2021]

Egy desktop ügyviteli szoftver tesztelését kell megvalósítani. Készítse el a tesztelési tervet, tegyen javaslatot a tesztelési környezet kialakítására!

- Definiálja a tesztelés fogalmát és jellemzőit!
- Határozza meg a szükséges teszttípusokat!
- Határozza meg a tesztelés szintjeit!
- Jellemezze a tesztelési módszereket!
- Definiáljon egy teljes tesztelési környezetet!

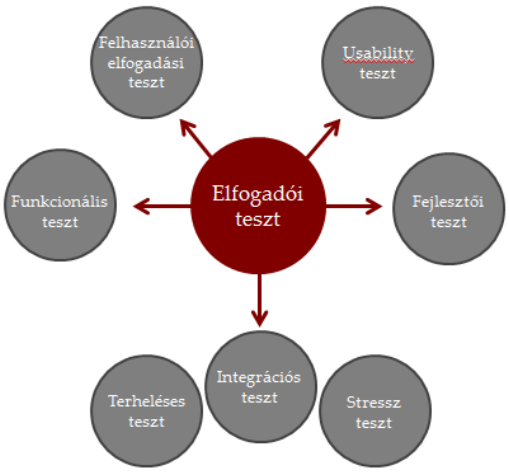
Kulcsszavak, fogalmak:

- *Validáció és verifikáció.*
- *Tesztelés szintjei: komponens teszt, modul teszt, integrációs teszt (alrendszer, rendszer teszt), elfogadási teszt.*
- *Tesztelési módszerek: statikus és dinamikus tesztelés, fekete doboz, fehér doboz.*
- *Szoftvertesztelés módszertana és folyamata.*
- *Tesztelési vezérlevek.*
- *Rendszertesztelés, integrációs tesztelés, végtesztelés.*
- *Teljesítménytesztek (volumen, stressz teszt).*
- *Automatikus tesztelési eszközök (pl. JUnit, NUnit, xUnit).*
- *Tesztvezérelt fejlesztés (napi build, release).*

---

1) Definiálja a tesztelés fogalmát és jellemzőit!

A rendszerfejlesztési és bevezetési projektek egyik kiemelt feladata az elkészült rendszer éles üzembe állítása. Ennek részeként az elkészült rendszert alapos tesztelési eljárásoknak kell alávetni annak érdekében, hogy annak üzembiztos és hibamentes működése szavatolható legyen. A tesztelés tevékenységén keresztül tudjuk biztosítani a lefejlesztett rendszer tervek szerinti működését, illetve a tesztelés során történik meg a rendszer megrendelő és felhasználók általi elfogadása is. A rendszer elfogadási tesztje rendkívül összetett feladat, amely általában az alábbi ábrán bemutatott egységekből áll:



1. ábra- Tesztfázisok

Jellemzői:

- Szelektivitás / specificitás
- Pontosság és precizitás
- Újra ismételhető folyamat
- Reprodukálhatóság
- Hibák nagy része a program egy kis részében található
- Fordított arányosság van a hibák megtalálása és a ráfordított erőforrás között
- Rendszerkompatibilitás

2) Határozza meg a szükséges teszttípusokat!

- Fejlesztői teszt
- Integrációs teszt
- Terheléses teszt
- Stressz teszt
- Funkcionális teszt
- Felhasználó elfogadási teszt
- Usability teszt

3) Határozza meg a tesztelés szintjeit!

A tesztelés szintjei a következők:

- komponensteszt,
- integrációs teszt,
- rendszerteszt,
- átvételi teszt.

A komponensteszt csak a rendszer egy komponensét teszteli önmagában. Az integrációs teszt kettő vagy több komponens együttműködési tesztje. A rendszerteszt az egész rendszert, tehát minden komponenst együtt, teszteli. Ez első három teszt szintet együttesen fejlesztői tesztnek hívjuk, mert ezeket a fejlesztő cég alkalmazottai vagy megbízottjai végzik. Az átvételi teszt során a felhasználók a kész rendszert tesztelik. Ezek általában időrendben is így követik egymást.

A komponensteszt a rendszer önálló részeit teszteli általában a forráskód ismeretében (**fehér dobozos tesztelés**). Gyakori fajtái:

- **unit-teszt,**
- **modulteszt.**

Unit-teszt, vagy más néven egységteszt, a metódusokat teszteli. Adott paraméterekre ismerjük a metódus visszatérési értékét (vagy mellékhatását). A unit-teszt megvizsgálja, hogy a tényleges visszatérési érték megegyezik-e az elvárttal. Ha igen, sikeres a teszt, egyébként sikertelen. Elvárás, hogy magának a unit-tesztnek ne legyen mellékhatása.

A unit-tesztelést minden fejlett programozási környezet (integrated development environment, IDE) támogatja, azaz egyszerű ilyen teszteket írni. A jelentőségüket az adja, hogy a futtatásukat is támogatják, így egy változtatás után csak lefuttatjuk az összes unit-tesztet, ezzel biztosítjuk magunkat, hogy a változás nem okozott hibát. Ezt nevezzük regressziós tesztnek.

A modulteszt általában a modul nem-funkcionális tulajdonságát teszteli, pl. sebességét, vagy, hogy van-e memóriaszivárgás (memory lake), van-e szűk keresztmetszet (bottleneck).

Az integrációs teszt során a komponensek közti interfészeket, az operációs rendszer és a rendszer közti interfészt, illetve más rendszerek felé nyújtott interfészeket tesztelik. Az integrációs teszt legismertebb típusai:

1. **Komponens – komponens integrációs teszt:** A komponensek közötti kölcsönhatások tesztje a komponensteszt után.
2. **Rendszer – rendszer integrációs teszt:** A rendszer és más rendszerek közötti kölcsönhatásokat tesztje a rendszerteszt után.

Az integrációs teszt az összeillesztés során keletkező hibákat keresi. Mivel a részeket más-más programozók, csapatok fejlesztették, ezért az elégtelen kommunikációból súlyos hibák keletkezhetnek. Gyakori hiba, hogy az egyik programozó valamit feltételez (pl. a metódus csak pozitív számokat kap a paraméterében), amiről a másik nem tud (és meghívja a metódust egy negatív értékkel). Ezek a hibák kontraktus alapú tervezéssel elkerülhetők.

Az integrációs teszteket érdemes minél hamarabb elvégezni, mert minél nagyobb az integráció mértéke, annál nehezebb meghatározni, hogy a fellelt hiba (általában egy futási hiba) honnan származik. Ellenkező esetben, azaz amikor már minden komponens kész és csak akkor tesztelünk, akkor ezt a „nagy bumm tesztnek” (big bang tesztnek) nevezzük, ami rendkívül kockázatos.

A **rendszerteszt** a már kész szoftverterméket teszteli, hogy megfelel-e:

- *a követelmény specifikációnak,*
- *a funkcionális specifikációnak,*
- *a rendszertervnek.*

A rendszerteszt nagyon gyakran **feketedobozos teszt.** Gyakran nem is a fejlesztő cég, ahol esetleg elfogultak a tesztelők, hanem egy független cég végzi. Ilyenkor a tesztelők és a fejlesztők közti kapcsolat tartást egy hibabejelentő (bug trucking) rendszer látja el. A rendszerteszt feladata, hogy ellenőrizze, hogy a specifikációknak megfelel-e a termék. Ha pl. a követelmény specifikáció azt írja, hogy lehessen jelentést készíteni az éve forgalomról, akkor ezt a tesztelők kipróbálják, hogy lehet-e, és hogy helyes-e a jelentés. Ha hibát találnak, azt felviszik a hibabejelentő rendszerbe.

Fontos, hogy a rendszertesztheez használt környezet a lehető legjobban hasonlítson a megrendelő környezetére, hogy a környezet-specifikus hibákat is sikerüljön felderíteni.

Az átvételi teszt hasonlóan a rendszertesztheez az egész rendszert teszteli, de ezt már a végfelhasználók végzik. Az átvételi teszt legismertebb fajtái a következők:

- **alfa teszt,**
- **béta teszt,**
- **felhasználói átvételi teszt,**
- **üzemeltetői átvételi teszt.**

Az alfa teszt a kész termék tesztje a fejlesztő cégnél, de nem a fejlesztő csapat által. Ennek része, amikor egy kis segédprogram több millió véletlen egérkattintással ellenőrzi, hogy össze-vissza kattintgatva sem lehet kifektetni a programot.

Ezután következik a béta teszt. A béta tesztet a végfelhasználók egy szűk csoportja végzi. Játékoknál gyakori, hogy a kiadás előtt néhány fanatikus játékosnak elküldik a játékot, akik rövid alatt sokat játszanak vele. Cserébe csak azt kérik, hogy a felfedezett hibákat jelentsék.

Ezután jön egy sokkal szélesebb béta teszt, amit felhasználói átvételi tesztnek nevezünk. Ekkor már az összes, vagy majdnem az összes felhasználó, megkapja a szoftvert és az éles környezetben használatba veszi. Azaz installálják és használják, de még nem a termelésben. Ennek a tesztnek a célja, hogy a felhasználók meggyőződjenek, hogy a termék biztonságosan használható lesz majd éles körülmények között is. Itt már elvárt, hogy a fő funkciók mind működjenek, de előfordulhat, hogy az éles színhelyen előjön olyan környezet függő hiba, ami a teszt környezetben nem jött elő. Lehet ez pl. egy funkció lassúsága.

Ezután már csak az üzemeltetői átvételi teszt van hátra. Ekkor a rendszergazdák ellenőrzik, hogy a biztonsági funkciók, pl. a biztonsági mentés és a helyreállítás, helyesen működnek-e.

4) Jellemezze a tesztelési módszereket!

A szoftver verifikáció és validáció (V & V):

Megmutatja, hogy a kifejlesztett rendszer megfelel-e a szoftverspecifikációnak. A megrendelő részéről megfelel-e pedig az általa támasztott követelményeknek. Különböző szemléket és felülvizsgálatokat foglal magában. A legnagyobb validációs költségek az implementáció után jelentkeznek, amikor a működő rendszert teszteljük.

- **Fejlesztői teszt:**

  Ahogy a neve is mutatja, elvégzése a rendszer fejlesztőjének feladata. A fejlesztői teszt elvégzésévével és annak eredményeinek dokumentálásával igazolja a fejlesztő, hogy az általa elkészített rendszer megfelel a meghatározott követelményeknek.

- **Integrációs teszt:**

  Célja a rendszer-rendszer kapcsolatok megfelelőségének ellenőrzése, az egyes rendszerkomponensek együttműködésének vizsgálata.

- **Terheléses teszt:**

  A rendszer működési környezetét biztosító hardver és szoftver erőforrások megfelelőségének vizsgálata, az üzemi terhelést meghaladó mértékű terhelés mellett.

- **Stressz teszt:**

  A rendszer hibatűrő képességének ellenőrzésére hivatott, az egyes rendszerkomponensek váratlan kiesésének hatását vizsgálja a rendszer működése szempontjából.

- **Funkcionális teszt:**

  A rendszer gyors funkcionális tesztje (általában az éles környezetben), amelynek célja, hogy a rendszer felhasználója meggyőződhessen arról, hogy a rendszer működik és kész a felhasználói elfogadási tesztre; vagyis cél annak ellenőrzése, hogy a lefejlesztett szoftver elem megfelel-e a specifikációkban (üzleti specifikációban és informatikai rendszertervben) megadott funkcionális és nem funkcionális elvárásoknak.

- **Felhasználói elfogadási teszt** (UAT – User Acceptance Test):

  A rendszer átadás-átvételi folyamatának legfontosabb mérföldköve. Az UAT során kell elvégezni a rendszer funkcionális és nem funkcionális követelményeknek történő megfelelőségének vizsgálatát.

- **Usability teszt:**

  Regressziós tesztek során azt vizsgáljuk, hogy egy korábban kifejlesztett modul, vagy funkció hogyan viselkedik, működik az új modul vagy funkció üzembe helyezésével.

6) Definiáljon egy teljes tesztelési környezetet!

Tesztkörnyezet:

Az a környezet, ami hardvert, beműszerezést, szimulátorokat, szoftvereszközöket és egyéb tesztelést segítő elemeket tartalmaz annak érdekében, hogy a tesztelést le lehessen folytatni. A tesztkörnyezetek különböző szolgáltatásokat nyújtanak a tesztek módszeres elvégzéséhez.

Komponensei:

- Up to Date" rendszer – a módosítások ide is átvezetésre kerülnek, valid adatok vannak benne, patch-ek mint az élesen, adatbázisa is naprakész stb.)
- teszteket végrehajtó komponens (test driver, test harness, test bench),
- tesztkonfiguráció manager komponens (test CM),
- tesztelendő egység „beműszerezésére" szolgáló komponens (instrumentation)
- tesztelt egység külső szoftver környezetét szimuláló csonkok (stubs).

A szolgáltatások köre általában a következő:

- Automatizált tesztelés – mechanikus lépések, gyors végrehajtása
- Debugger –részletek nyomon követése
- Teljesítménytesztek és profilok készítése
- Statikus analízis
- Eredmények dokumentálása és megjelenítése.

Fel kell mérni, hogy mire van szükség. El kell dönteni, hogy pontosan milyen szolgáltatások szükségesek, ezt pedig csak alapos felmérés útján lehet eldönteni. Ezt teszt konfiguráció managementnek (CM) hívjuk.

A teszt konfiguráció management célja:

szabályozottan biztosítsa a megfelelő szoftver egységek tesztelésé ta megfelelő teszt esetekkel és megfelelően dokumentálja.Ez a szolgáltatás teremti meg a teszteket végrehajtó komponens működésének feltételeit, illetve magáért a működtetésért is felelős.

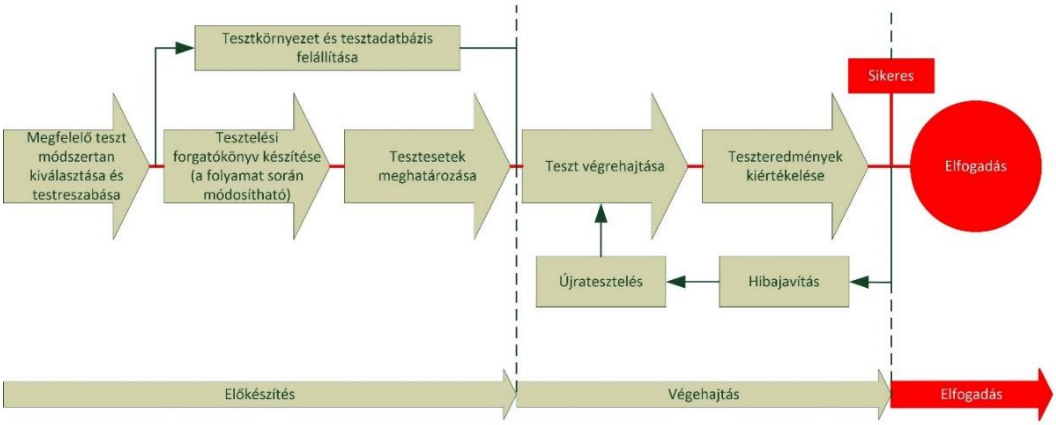
Az alábbi folyamatok adminisztrációját kell elvégeznie:

- cél egység "beműszerezése"[1]
- tesztelő által megadott tesztesetek és teszt paraméterek alapján a teszt beállítása
- teszt elvégzése és az eredmények értékelése, kezelése.

A teszteléshez a szoftveregységet felszereljük a működéséről tájékoztató, a teszt megkívánta helyzetek azonosításához, lefedettségi adatok szolgáltatásához, valamint a teszt közbeni beavatkozáshoz szükséges kapcsolódási pontokkal, „jeladókkal". Ez a "műszerezettség" teremti meg a kapcsolódási pontokat a tesztvégrehajtó komponens és a tesztelt egység között.

[1] forráskód vagy végrehajtói kód beműszerezése, előbbi esetén a szoftverbe helyezzük, mely rugalmasabb, de nehéz karbantartani, és módosítja a kódot, melynek mellékhatásai lehetnek, utóbbinál a forráskód mellé csatolunk debug információkat, csak korlátozottan használható, de a szofter működését csak minimálisan zavarja.

Szoftvertesztelés módszertana és folyamata:



2. ábra - Tesztelés folyamata

I. Előkészítés

Ebben a fázisban összegyűjtjük a fejlesztéssel és teszteléssel kapcsolatos minden információt és elvárást, majd azokra építve kiválasztjuk a feladat végrehajtásához leginkább illeszkedő teszt módszertant, és meghatározzuk a tesztelés alapelveit, amelyek a következők lehetnek:

- Teszt hatóköre
- Erőforrás szükséglet
- Átfutási idő
- Hiba kezelési alapelvek (hiba súlyossági kategóriák és javítási válaszidők meghatározása)
- Dokumentációs követelmények és sablonok
- Alkalmazandó gépi teszt eszközök (opcionálisan)
- Alkalmazandó teszt menedzsment alkalmazások (opcionálisan)

Az előkészítés következő lépése a teszt lépések tervezése – ebben a fázisban történik meg a teszt részletes megtervezése. A rendelkezésre álló dokumentáció (követelményjegyzék, üzleti specifikáció, az esetleges korábbi teszt fázisok dokumentációja, valamint a rendszer leendő felhasználói által szolgáltatott információk alapján). Lépései a következők:

- Követelmény lista összeállítása - a funkcionális, nem funkcionális és design követelmények összegyűjtése.
- Teszt forgatókönyvek elkészítése – az egyes követelmények ellenőrzésére alkalmas forgatókönyvek kidolgozása, beleértve a teszt lépés végrehajtásának pontos leírását, annak szereplőit, illetve az elvárt eredmény megfogalmazását is.
- Tesztadat szükségletek felmérése – az egyes tesztesetek végrehajtásához szükséges teszt adatok körének meghatározása.

II. Végrehajtás

- Teszt lebonyolítása
  - A végrehajtás során kezdődik meg a tényleges tesztelés, melynek során az előzetesen előkészített teszt forgatókönyvek kerülnek végrehajtásra. Ez utóbbi a megfogalmazott teszt feladatok végrehajtásának segítségével a követelmények teljesülésének ellenőrzését jelenti.

- Hibák azonosítása, jelentése a fejlesztőnek – a tesztelés során feltárt hibákat a fejlesztőnek jelezni kell, erre adott esetben egy előre kiválasztott szoftvert használunk. Ennek funkcionalitása biztosítja az egyes hibák részletes dokumentálhatóságát. A cél, hogy a fejlesztő számára minden információt biztosítsunk, amely támogatja a hiba azonosítását, illetve javítását.
- Változtatási igények azonosítása – a tesztelés során gyakran előfordul, hogy a felhasználók az eredeti követelményektől eltérő működési javaslatot fogalmaznak meg. Fontos ezeknek a változtatási igényeknek a gyűjtése, hiszen ezek alapozhatják meg azt, hogy a végleges rendszer minél jobban megfeleljen a felhasználók igényeinek.

o Hibajavítás ellenőrzése, menedzsmentje

- Az azonosított hibák javítását nyomon kell követni. A projekt során nyomon követjük az egyes hibák állapotát, szükség esetén egyeztetéseket szervezünk a fejlesztő és Megrendelőnk szakértőinek közreműködésével annak érdekében, hogy a hibaelhárítás gördülékenyen menjen. Szükség esetén eszkaláljuk a problémát a megfelelő vezetői szintre.
- A hibajavítás és változtatások végrehajtásának ellenőrzése – az elvégzett hibajavítást az adott teszteset ismételt végrehajtásával ellenőrizni kell.
- A tesztelés során szükség lehet ismételt regressziós tesztek elvégzésére is, hiszen egyesesetekben előfordulhat, hogy egy hiba javításával összefüggésben olyan változtatások kerülnek a rendszerbe, amelyek korábban jól működő funkciók meghibásodását is okozhatják. Ezek a hibák a regressziós teszteléssel kiszűrhetők.

III. Elfogadás

A teszt lezárását követően, amikor már nincs hiba, vagy csak a megrendelő által elfogadott hibák vannak a rendszerben, szükség van a teszt eredményeinek összefoglaló értékelésére. Ennek érdekében összefoglaló teszt jelentést készítünk (tesztelési jegyzőkönyv), amely megalapozza a rendszer éles üzembe állításáról szóló döntést.
A teszt eredményeinek, tapasztalatainak elemzése – a teszt dokumentáció, illetve a tesztelésben résztvevők tapasztalatai alapján a rendszer működését értékelni kell. Meg kell határozni, hogy az megfelel-e az előzetes elvárásoknak, illetve alkalmas-e az éles üzemi működésre.
A teszt dokumentáció feldolgozásával és a tesztelésben résztvevőkkel folytatott interjúk alapján döntési javaslatot fogalmazunk meg. A döntési javaslat alapján lehetőség van a rendszer éles üzembe állításáról vagy annak elhalasztásáról szóló döntés meghozatalára.

A tesztelési fázisok végrehajtása során szorosan együttműködünk a projektben közreműködő munkatársakkal. Nagy hangsúlyt fektetünk a tesztelésben részt vevő kollégák közötti gyors és hatékony kommunikációra.

A tesztelés nyomon követésére az ügyfél igényeinek megfelelően tesztmenedzsment eszközöket is használunk, a legmegfelelőbb eszköz kiválasztásában gyakorlati tapasztalataink alapján segítséget nyújtunk az ügyfél számára. A tesztmenedzsment eszközök használata megkönnyíti a tesztesetek adminisztrálását, a teszttervek, teszt futtatások és hibajegyek kezelését is. A tesztesetek, tesztforgatókönyvek és hibák dokumentálásán túl fontos a tesztelés eredményeinek és a felmerült hibák státuszának elemzése, nyomon követése, valamint a tervekhez viszonyítása, melyekhez a megfelelően kiválasztott tesztmenedzsment eszköz támogatást nyújt.

Az egyes tesztesetek elvégzése során a feltárt hibákról és azok állapotairól, észrevételekről átfogó riportokat készítünk, mely segítséget nyújt a folyamatos ellenőrzéshez, nyomon követéshez.