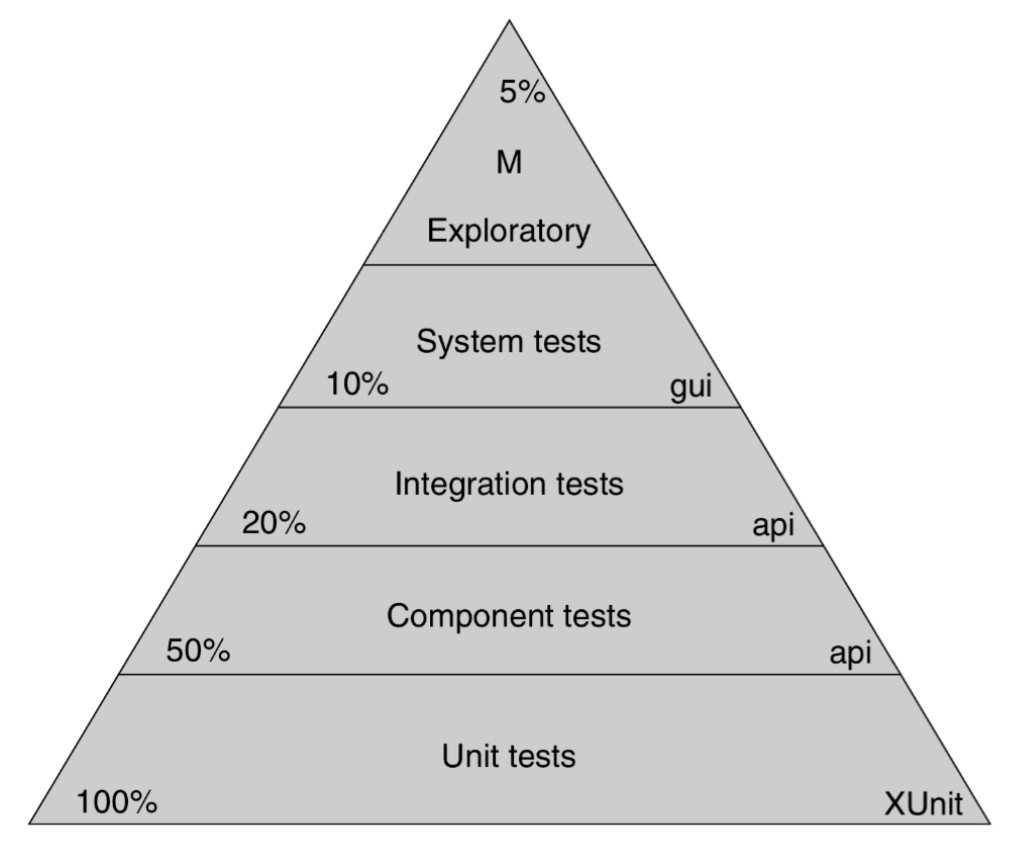
Smoke test:

* megbizonyosodni arról, hogy a jelenlegi változtatások nem okoztak semmilyen súlyos hibát az applikációban.
* A smoke teszt olyan tesztforgatókönyv, amelyeket egy fejlesztőcsapat azzal a céllal készít, hogy egy alkalmazás alapvető funkcióinak működését ellenőrizze. Bármely ilyen teszt sikertelensége az aktuális release visszautasítását vonja maga után.

Regressziós teszt:

* célja, hogy meggyőződjünk arról, hogy a korábban működő funkciók az elvégzett módosítások mellett is működőképesek maradnak.
* Ezt általában a meglévő tesztkészlet ismételt futtatásával érjük el.



# 1szoftvertesztelés célja

Bevezetés.

Ismertesse a szoftvertesztelés célját, fontosságát. Miért szükséges tesztelni?

Fogalmazzon meg 2 lehetséges

teszt célt.

Miben különbözik a szoftvertesztelés a hibakereséstől

és a minőségbiztosítástól?

A tesztelés célja, Miért szükséges tesztelni?

A munkatermékek, user storik, műszaki tervek és kód kiértékelése azért, hogy ne legyen hiba

Teljesül-e az összes követelmény

Ellenőrizni hogy a teszt tárgya impelmentálásra került-e és hogy az elvárásnak megfelelően működik-e

A termék által a bizalom kiépítése

A hibák megtalálása ezáltal a kockázat csökkentése

Infomráció biztosítása a minőségről

A szerződésben foglaltak biztosítása

Miben különbözik a szoftvertesztelés a hibakereséstől

és a minőségbiztosítástól?

tesztelés: kimutatja a hibákat nem csak a program futtatásával

hivakeresés: Fejlesztési tevékenység, amelynek során megtalálják, elemzik és kijavítják ezeket a hibákat

minőségbiztosítás: kapcsolódik a tesztelés és a minőségbiztosítás, de nem ugyanaz, a minőségmenedzsment fogja őket össze. A minőségmenedzsment sok folyamatot ölel fel, például a minőségirányítást is. A teszttevékenyésgek a minőségirányítás részei, a teljes fejlesztés vagy karbantartási folyamat részeit képezik.

# 2 hiba, stressz teszt

Alapfogalmak. Ismertesse a következő tesztelési alapfogalmakat: emberi eredetű hiba,

meghibásodás, hibaok, hibamaszkolás, stressz-teszt. Mi a különbség a teszt eset és a teszt adat között? Melyek egy teszt eset legfontosabb elemei?

emberi eredetű hiba: egy ember tévedhet, ami a szoftverben hibát okoz, és ez meghibásodáshoz vezethez

meghibásodás: ua

hibaok: azok a legkorábbi tevékenységek vagy feltétleek, melyek hozzájárultak a hibák kialakulásához

hibamaszkolás: amikor egy hiba jelenléte elrejti egy másik hoba jelenlétét

stressz-teszt: a terheléses teszt azt jelenti, hogy a szv-t úgy terheljük, mintha egy időben sok felhasználó lenne. Ha a terhelés a normálisan elvárt fölé emelkedik akkor van stressz teszt.

teszt eset: inputja és outputja van

teszt adat: konkrét értéket rendel a tesztesetek bemenetaihez és az elvárt eredményekhez

teszt eset elemei:

* azonosítós
* cél
* előfeltételek (precondi )
* input adatok
* elvárt output:
* utófeltétel
* teszt eset végrehajtásának története

# 3 7 alapelv

Ismertesse a tesztelés 7 alapelvét.

1. A tesztelés a hibák jelenlétét mutatja, nem a hiányukat
2. Nem lehetséges kimerítő teszt
3. A korai tesztelés időt és pénzt spórol
4. Hibafürtök megjelenése
5. Kísérd figyelemmel a féregirtó paradoxont
6. A tesztelés függ a körülményektől
7. A hibamentesség egy téveszme

# 4 A tesztelés pszichológiűja

A tesztelés pszichológiája. Miben különbözik a tesztelői viselkedés a szoftverfejlesztői

viselkedéstől? Melyek a tesztelő legfontosabb képességei? Mit tud a tesztelő hozzáadni a

fejlesztendő rendszerhez? Mit jelent a megerősítési torzítás?

A talált hibákat kritikának érezhetjük.

Miben különbözik a tesztelői viselkedés a szoftverfejlesztői viselkedéstől?

A fejlesztő helyesnek véli a kódját, a tesztelő hibákat keres, de a cél ugyanaz, és a fejlesztők is tesztelhetik saját kódjukat tesztelői szemléletmóddal.

Mit jelent a megerősítési torzítás?

A fejlesztők helyesnek vélik a kódjukat, a megerősítési torzítás miatt nehéz elfogadniuk a kód helytelenségét. A megerősítési torzítás azt jelenti, hogy azokat az információkat részesítjük előnyben, amik igazolják a saját igazunkat. Továbbá egy gyakori emberi jellemvonás, hogy a rossz hír hozóját okolják, és a tesztelés által szolgáltatott információ gyakran tartalmaz rossz hírt.

Melyek a tesztelő legfontosabb képességei?

* Éles megfigyelőképesség
* Részletekre figyelés
* Destruktív kreativítás
* A terméket úgy felfogni, mint részeinek integrációját
* Vevő-központú hozzáállás
* Cinikus, de odafigyelő hozzáállás
* Szervezettség, rugalmasság, türelem Organized
* Objektív & semleges hozzáállás

A jó kommunikáció kulcsfontosságú

A tesztelő legfontosabb feladata a megfelelő tesztelési technikák kiválasztása és kombinálása

# 5 A tesztelési folyamat

A tesztelési folyamat. Ismertesse, hogyan jelenik meg a tesztelés az alapvető szoftverfejlesztési életciklus modellekben. Miről szól az ISO 12207 szabvány? Hogyan kapcsolódik a V-modellhez? Hogyan használható az ISO 12207 egy szoftverfejlesztési életciklus modell kialakításában?

Alapvetően a szv fejlesztés az alábbiakat tartalmazza:  
Követelmények fejlesztése és menedzsmentje

Tervezés

Kódolás

Tesztelés

Átadás

Karbantartás

Szoftver életciklus modellek

Szekvenciális - lineárisan ábrázolja, ez előző fázis után jöhet csak a következő

* vízesés, v modell

Inkrementális - követelmények kisebb csoportokban és a követelménykezelés, kódolás, tesztelés erre a csoportra történik, rövidebb idő alatt, fejhasználó hamarabb kap visszajelzést

* rup, spirál

Iteratív - ciklusos, a követelmények egy kisebb csoportját implementáljuk adott idő alatt

* agilis megközelítések: scrum, kanban, XP, lean

12207

Ez foglalja össze amit tudunk a szv fejlesztési folyamatokról. Minden szv fejlesztési életciklus modell többé kevésbé ugyanazokból a folyamatokból épül fel, a teljes menü, melyekből a folyamatokat kiválaszthatjuk, szabvánnyá vált. Ez teljesen kompatibilis a V modellel

Hogyan használható az ISO 12207 egy szoftverfejlesztési életciklus modell kialakításában?

Azt mondja, hogy a szv fejlesztésben adott folyamatok lehetnek, milyen bemenet, milyen tevékenységek mit kell létrehozni, kik a résztvevők. Folyamatokfelsorolása nem életciklus modell. A használó ki tudja váasztnia azoka t afolyamtokat amik nekem fontosak és olyan sorrendbe tudom tenni. Azért jó mert ezek a folyamatok szabványosak, bárki érti őket.

Nem életciklust modellt ír de de bármilyen életciklus modellt lére lehet hozni vele.

# 6 alapvető tesztelési folyamat

Az alapvető tesztelési folyamat. Ismertesse az alapvető tesztelési folyamatot az ISTQB

ajánlása alapján.

Teszttervezés

* tesztcél definiálása
* tesztechnikák, feladatok specifikálása
* ütemterv
* újra lehet gondolni szükség esetén

Tesztfelügyelet és -irányítás

* aktuáis előrehaladás és tervezett előrehaladás összehasonlítása
* felhsaznála a teszttervben meghatározott felügyeleti metrikákat
* DoD, a kilési feltételek kiértékelése támogatja

Tesztelemzés

* tesztbázis elemzése azért, hogy a tesztelhető dolgok azonosítva legyenek és meg legyenek határozva a tesztfeltételek
* meghatározza, hogy mit teszteljünk

Műszaki teszttervezés

* magas szintű tesztesetek kialakítása
* megválaszolja az, hogy hogyan teszteljünk
* tesztesetek tervezése, priorizálása
* tesztfeltételek, tesztadatok meghatározása
* tesztkörnyezet megtervezézse
* tesztbázis, tesztfeltétele és tesztesetek közötti kétirányú nyomonkövethetőség meghatározása

Tesztmegvalósítás

* tesztek megírása, tesztkészletek létrehozás,automata tesztek megírása
* ütemterv összeállítása
* tesztkörnyezet összeállítása, ellenőrzése
* teszteadatok előkészítése
* kétirányú nyomonkovethetőség ellenőrzése a tesztbázis, tesztfeltételek, tesztesetek, teszteljárások és tesztkészletek közötti

Tesztvégrehajtás

* A tesztek manuálisan vagy tesztvégrehajtási eszközök segítségével történő végrehajtása
* A várt és a tényleges eredmények összehasonlítása
* Hibák jelentése
* A tesztvégrehajtás kimenetének naplózása
* Ismétlés, ha hiba van vagy megköveteli a terv

Tesztlezárás

* Adatokat gyűjtenek a befejezett teszttevékenységekből
* Projekt mérföldköveinek elérésekor hajtják végre
* Annak ellenőrzése, hogy minden hibajelentést lezártak
* Összefoglaló tesztjelentés készítése
* A tesztver átadása a karbantartást végző csapatoknak, más projektcsapatoknak
* tanulságok elemzése
* tesztfolyamat érettségének fejlesztése a gyűjtött információk alapján

# 7 Tesztszintek

Tesztszintek. Ismertesse az ISTQB ajánlásaiban szereplő teszt szinteket.

Komponens (Unit) teszt

* Alapok célok:
  + Komponens: a legkisebb önállóan tesztelhető szoftver egység
  + A forráskód egy adott egységének helyességét vizsgáló tesztmódszer
  + Annak megelőzése, hogy a hibák magasabb tesztszintre is bekerüljenek
  + Általában a fejlesztők írják
* tesztbázisként használható munkatermékek
  + Részletes terv, kód , adatmodell, komponens
* jellemző hibák:
  + helytelen kód és logika
* teszt tárgy:
  + Komponensek, egységek vagy modulok, kód és adatstruktúrák, osztályok

Integrációs teszt

* Alapok célok
  + Célja, hogy a nagyobb elemekkel szemben támasztott elvárásokat ellenőrizze
  + Az egységteszten sikeresen „átesett” modulokat veszi alapul
  + A modulok közötti interfészek kerülnek kipróbálásra
* tesztbázis
  + Szoftver- és rendszerterv , szekvencia diagram, specifikációk, use case-ek
* teszt tárgy
  + Alrendszerek , adatbázisok , infrastruktúra , interfészek, API-k , mikroszolgáltatások
* jellemző hibák
  + Helytelen vagy hiányzó adatok, interfész hibás illesztése, komponensek közötti kommunikáció meghibásodással

Rendszerteszt

* Alapok célok
  + A rendszertesztet a felhasználónak is értenie kell
  + A teljes rendszert teszteljük a specifikációval szemben
  + Annak validálása, hogy a rendszer teljes és az elvártaknak megfelelően működik
  + A rendszer egészének minősége iránti bizalom megteremtése
* tesztbázis
  + Rendszer- és szoftverkövetelményspecifikáció, kockázatelemzés , használati esetek, felhasználói történetek
* teszt tárgy
  + Alkalmazások, hardver/szoftver rendszerek, operációs rendszer
* jellemző hibák
  + A rendszer helytelen vagy váratlan viselkedése
  + Meghibásodás az adott rendszerkörnyezet(ek)ben
  + A rendszer nem a rendszer- és felhasználói kézikönyvekben leírtak szerint működik

Átvételi teszt/elfogadási teszt

* Egy egész rendszer vagy termék viselkedésére és képességeire összpontosít
* Bizalom kialakítása a teljes rendszer minőségével kapcsolatban
* Annak validálása, hogy a rendszer teljes és az elvártaknak megfelelően fog működni • Annak verifikálása, hogy a rendszer funkcionális és nemfunkcionális specifikációkban meghatározottak szerint készült-e el

# 8 Funkcionális és nemfunkcionális

Funkcionális és nemfunkcionális jellemzők tesztelése. Mutassa be a nemfunkcionális szoftvertermék jellemzők tesztelésére alkalmazható megközelítést.

Funkcionális

* Olyan teszteket foglal magában, melyek kiértékelik a rendszer által végzendő funkciókat.
* A funkció az, amit a rendszernek tennie kell.
* minden tesztszinten el kell végezni
* A szoftver viselkedését veszi figyelembe, így a feketedoboz technikák felhasználhatók a tesztesetek létrehozásakor

nemfunkcionális

* a rendszerek és szoftverek jellemzőit vizsgáljuk, például a használhatóságot, a teljesítmény-hatékonyságot vagy a biztonságot
* azt vizsgáljuk, hogy miként viselkedik a rendszer
* minden tesztszinten elvégezhető
* A feketedoboz technikákat használhatjuk a teszteseteinek meghatározásához, például teljesítményteszteknél határérték elemzés
* speciális tudást igényelhet például a tehcnológia vagy felhasználói bázis ismerete (egsészségügy)

A funkcionális és nemfunkcionális követelményeken alapszik.

funkcionális követelmény:

* olyan követelmény, amely a szoftverrel szemben támasztott funkcionális elvárást írja le

nem funkcionális követelmény:

* „Ahogyan a rendszer csinálja”
* Minőségi követelményeknek is szokás nevezni őket
* Teljesítmény Biztonság Megbízhatóság Használhatóság Karbantarthatóság

# 9 regressziós és a progressziós

Karbantartási tesztelés. Ismertesse a karbantartási tesztelés legfőbb típusait. Beszéljen a regressziós és a progressziós tesztelés jelentőségéről.

karbantartási tesztelés

Az átadott rendszert karban kell tartani

Hibajavítás, új funkció, funkció törlés esetén változás van a rendszerben

Meg kell őrizni a rendszer minőségét

Kiváltó okok

* Módosítások
* Migráció

Karbantartási hatáselemzés

* Kiértékeli a várható következményeit a változtatásoknak és azonosítja a rendszer azon részeit, amik érintettek
* Segít azonosítani a változás kihatását a meglévő tesztekre

Típusok

* regressziós és progressziós tesztek fontosak

Regressziós tesztelés:

* az egyik változás befolyásolhatja a kód más részeit
* ezeket a nem kívánt változtatásokat nevezzük regressziónak
* azt bizonyítjuk, hogy ami az előző buildben működött, most is működik

Progressziós tesztelés:

* ellenörző tesztelés, célja annak megerősítése, hogy a hibákat kijavították-e
* feltételezzük, hogy az integrációs teszt rendben lefutott és az új funkciókat tesztelhetjük
* új területet tesztelünk, több hibára számítunk
* az összes olyan tesztet végrehajtjuk ami miatt korábban sikertelen volt

# 10 Tesztek típusai

Tesztek típusai. Ismertessen a tesztelésben alkalmazott teszt típusok közül legalább 3 lehetséges csoportosítást és elemeiket. A csoportok közül egyik legyen a tesztelési technikák szerinti csoportosítás.

Csoporttosítás:

tesztelt elem szerint - lásd tesztszintek

* komponens
* integrációs
* rendszerteszt

Egyéb kritérium szerint

* Manuális tesztelés
* Automata tesztelés
  + szkritpek alapján történő tesztelés
  + tesztelési eszközt használó tesztelés

Életciklusban elfoglalt hely szerint

* Kifejezi, hogy a készülő szoftver mennyire van készen
  + Pre alfa teszt: egyes funkciók fejlesztői tesztje
  + Alfa teszt: fejlesztő belső rendszertesztje
    - szimulált vagy tényleges teszt
    - a fejlesztés helyszínén, de függetlenül a fejlesztőktől
  + Béta teszt
    - szűkebb felhasználói tesztelés a végső kiadás előtt
    - megfelel-e a szoftver a felhasználók piaci igényeinek
  + gamma, delta, omega teszt: egyre fejletteb release-ek tesztje

tesztelési technikák szerint:

* Különböző szinten különböző tehcnikát használunk
* Bármely szint esetében igaz hogy az elvárt eredményeket több tehcnika ötvözésével lehet eléreni
  + statikus tesztelés
  + dinamikus tesztelés
    - strukturális
    - funkcionális

# 11 Dokumentumok

Ismertesse a tesztelési folyamat alapvető dokumentumait. Mutasson rá, hogy melyik dokumentum tipikusan melyik teszt fázisban kerül kidolgozásra.

A tesztelés minden elemét dokumentálni kell

A felhasználók is kérnek dokumentumokat

Dokumentumok:

Tesztelési irányelvek

* Magas szintű, megközelítésmód, cél

Tesztterv

* erőforrás, ütemezés összeszedése
* tesztelemek, funkciók, feladatok, személyek, környzeet, technikák, belépési kilépési feltételek, kockázatok

Teszt forgatókönyv

* Teszteseteket tartalmaz

Teszteset

* …

Tesztzkript

* elsősorban automatizált tesztre használjuk

Tesztadat

* inputok, előfeltételek, outputok
* olyan adatok amik léteznek tesztelés előtt és felhasználjuk őket tesztelés során

Tesztelési jegyzőkönyv

* név, funkció, dátum, sikeres/bukott, környezet

hibalista

* ...

Átadási átvételi dokumentumok

* teszt jelentés, összegző dokumentum, tevékenység + eredmény, tapasztalatok

# 12

Mutassa be, hogy a szoftverfejlesztési életciklusban általában milyen tesztelési technikák alkalmazása szokásos az egyes fázisokban. Javaslat: vegye alapul a V-modellt.

18-ashoz hasonló

# 13 Statikus tesztelés

Statikus tesztelés. Mutassa be a statikus tesztelés jellemzőit és legfontosabb, az előadásokon vizsgált technikáit. Milyen munkatermékekre alkalmazhatunk statikus tesztelést? Melyek a statikus tesztelés előnyei, és milyen hátrányai lehetnek?

A tesztelés az a formája, amikor a szv-t nem használjuk, például kódelemzés

Emberi megfigyelésen alapul, bármilyen termékre alkalmazható

Alapelv: futtatással nem tudunk mindent lefedni, de a teljes kódot át lehet nlézni

Egy kompens vagy rendszer tesztelése specifikáció vagy implementáció szinten

Munkatermékek:

* specifikációk, user storyk, kritériumok, kód, tesztterv, teszteset, felhasználói kézikönyv, weboldal, szerződés, konfiguráció terv

Előnyök:

* hibák korai megtalálása, javítás
* költségek és időigény csökkentése
* szoftver minőségének növelése, nem kerül elő meghibásodás később
* a felülvizsgálatokban a részvétel javítja a kommunikációt

Statikus tesztelés hátrányai

* Nem könnyű rávenni a szervezetet (érettséget és szervezést igényel)
* Emberi „megfigyelésen” alapul – sokat számíthat a tapasztalat

Különbségek dinamikus és statikus között:

* a statikus közvetlenül talál meg hibákat
* dinamikussal meghibásodást találunk
* statikussal találunk kódolási hibákat, szabányoktól való eltérést, biztonsági sebezhetőség

Statikus analízi technikák:

* Statikus kódelemzés
  + a forráskód elemzése, hogy megértsük, mit csinál a szoftver
  + veszélyesnek látszó elemket is kereshetünk
  + vagy formális módszer, amivel a program matematikailag leírható
* Statikus analízis:
  + például egy debugguló, ami a futási hibákat segít megtalálni
  + eredménye nem egyértelmű, nem tudjuk hogy futás közben megjelennek-e a hibák
* Átvizsgálás (walkthrought):
  + kód felosztása részekre, célhoz kapcsolódóan, majd átvizsgálása

Felülvizsgálat:

formálistól az informálisig

* informális felülvizsgálat
* átvizsgálás
* technikai felülvizsgálat
* inspekció

Szerepek lehetnek:

* szerző - létrehozza a munkaterméket
* management - megtervezi, időkeretet teremt
* moderátor - hatékony levezetése a felülvizsgálatot
* felülvizsgálati vezető - általános felelősség a felülvizsgálat végrehajtásáért, résztvevők meghatározása
* Felülvizsgáló - szakértők, jártas személyek
* írnok - jegyzőkönyvvezető

Felülvizsgálati technikák:

* ad hoc - nincs iránymutatás
* ellenőrzőlista alapú - előnye, hogy szisztematikusan lefedi a gyakori hibákat
* forgatókönyvek és száraz tesztelés (elméleti futtatás) - strukturális iránymutatást ad, pl user storyk alapján
* perspektíva alapú - különböző érintett felek szempontjaiból (marketing, tervezői, üzemeltetői)
* szerep alapú - különböző érintett felek, hasonló a perspektvához, főként különböző user szemszög (tapasztalt, gyerek, idős)

# 14 Dinamikus tesztelési technikák

Dinamikus tesztelési technikák. Röviden mutassa be a dinamikus tesztelésben a teszt esetek meghatározásához alkalmazott két alapvető megközelítést. Vannak tiltott technikák bizonyos szoftverek esetében?

A rendszer futtatása közben tesztelünk

Kétféle lehet:

1. fekete (funkcionális)
   1. teszt esetek specifikáció alapján vannak meghatározva
   2. nem feltételei a kód belső ismeretét
2. fehérdoboz (strukturális)

* Funkcionális elvárások alapján: funkcionális / fekete doboz tesztelésnek nevezzük. A teszt eseteket a specifikáció alapján határozzuk meg.
* A szoftver belső szerkezetének alapján: strukturális / fehér doboz tesztelésnek nevezzük. A teszt eseteket a kód alapján határozzuk meg.

**Dinamikus tesztelés**

* A rendszer / modul futtatását feltételezi
* A rendszer próbafuttatása, teszt környezetben, teszt adatokkal

Kétféle lehet:

* Fekete doboz teszt (funkcionális teszt)
* Fehér doboz teszt (strukturális teszt)

**Funkcionális tesztelés**

* Funkcionális elvárások alapján
* A teszt eseteket a specifikáció alapján határozzuk meg
* Nem feltételezi a kód belső szerkezetének ismeretét
* A halmazelmélet és függvényelmélet alapfogalmait felhasználjuk a funkcionális tesztelésben
* Bármely program tekinthető egy függvénynek, amely az input értékeket (értelmezési tartomány) az output értékekre (képtér) képezi le
* A funkcionális tesztelés alapvetően az input értékek felhasználásával kialakított teszt esetek halmazát jelenti. A teszt esetekkel az output értékek viselkedését figyeljük.

Funkcionális tesztelés előnyei:

* Független a szoftver implementálási módjától
* A teszt esetek fejlesztése a szoftver fejlesztésével egy időben történhet (rövidül a projekt átfutási ideje)

Funkcionális tesztelés hátrányai:

* A teszt esetek között lényeges redundancia fordulhat elő
* Nagy arányban maradhat teszteletlen a szoftver
* tehát kombinálni kell más technikákkal

Strukturális tesztelés

* A forráskódon alapszik
* „Abszolút” alapokon nyugszik
* Technikáiban pontos definíciókat, matematikai analízist alkalmaz
* Az alkalmazott technikák már a ’70-es években léteztek
* Egészen pontos méréseket tesz lehetővé

Strukturális tesztelés előnyei

* Szilárd elméleti alap: a gráfelmélet elemeit használva egészen pontosan leírható, hogy mit tesztelünk
* Pontos lefedettségi metrikákat tesz lehetővé. Ezekkel pontosan mérhető, hogy egy szoftver elemet milyen mértékben teszteltünk.

Strukturális tesztelés hátrányai

* Befektetést igényel, hogy a tesztelők az elméleti alapokat elsajátítsák
* A szilárd elméleti alap nélkül nem lehet jól csinálni
* Nagyon hangsúlyozza, hogy a tesztelés komoly „szakmai” feladat

# 15 Funkcionális és strukturális tesztelés

Funkcionális és strukturális tesztelés. Mutassa be, milyen alapvető különbség van a két megközelítésmód között. Kombinálható-e a két megközelítés? Mondjon példát a funkcionális és strukturális tesztelés hatékony kombinálására.

fekete (funkcionális)

* teszt esetek specifikáció alapján vannak meghatározva
* nem feltételei a kód belső ismeretét

fehérdoboz (strukturális)

* a rendszer belső ismeretét feltételezi

A jó tesztelési módszer a technikákat megfelelő arányban komibálva alakul ki.

Hitviták alakultak ki a kérdéskör körül:

* Funckionális tesztelés előnyei
  + független a szv implementálási módjától
  + a teszt esetek a szv fejlesztésével egyidőben történhetnek, rövidebb idő
* Hátrányok
  + nagy arányban maradhat teszteletlen a szv
* Strukturális tesztelés hátrányai
  + gráfelméletet használva potnosan leírható, hogy mit tesztelünk
  + lefedettségi metrikákat tesz lehetővé
* Srukturális tesztelés hátrányai
  + elméleti alap nélkül nem lehet megcsinálni, szaktudást igényel

kombinálás?

strukturális jelelggel meghatározom hogy integrációs tesztelésnél melyik elemeket fogom használni

strukturálisnál valamilyen modellel közelítek a rnedszerhez

# 16 ekvivalencia, határérték, döntési tábla, állapotát., use case

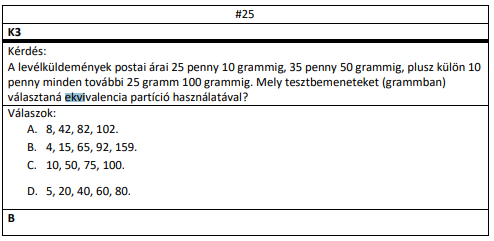
Funkcionális tesztelési technikák. Ismertesse a tananyagban szereplő 5 alapvető funkcionális tesztelési technikát. Hogyan viszonyulnak ezek egymáshoz a kialakított teszt esetek száma és a tesztelési ráfordítás szempontjából?

Funkcionális tesztelés:

* funkcionális elvárások alapján
* a teszt eseteket a specifikáció alapján határozzuk meg
* nem feltételezi a kód belső ismeretét

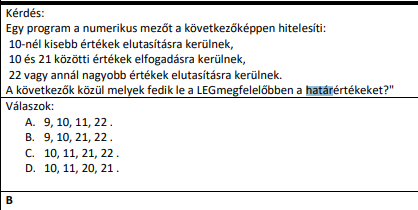
Ekvivalencia particionálás

* adatok partíciókra osztása, egy partícióban lévő elemek viselkedése ugyanolyan
* érvényesek azok, amiket a rendszernek el kell fogadnia
* érvénytelenek azok, amiket a rendszernek el kell utasítania



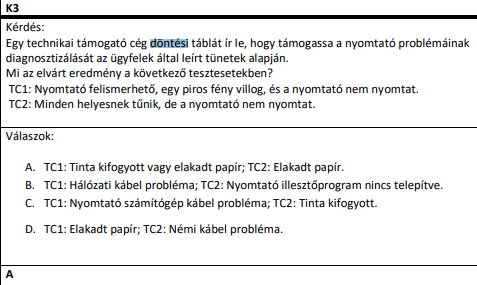
Határérték elemzés

* ekvivalenciaparticionálás kiterjesztése
* a partíció minimum és maximum értékei a határértékek
* határonként a a határnél kisebb és a nagyobb az érvénmytelen
* azért hasznos mert határonként nagyobb valószínűséggel lehet rossz a viselkedés



Döntési tábla alapú tesztelés

* olyan esetek leírására, amikor különböző akciók kombinációira kerül sor változó feltételek mellett



Állapotátmenet tesztelés

* a rendszer különbözőképpen reagálhat eseményekre a jelenlegi állapot vagy előzmény függvényében
* az állapotátmenet diagram vagy tábla megmutatja a lehetséges szv állapotokat, hogy a szv hogyan lép be és ki egy állapotba vagy mik az átmenetek az állapotok között
* menü alapú alkalmazások tesztelésére hasznos
* úgy tervezzük meg a teszteseteket, hogy érvényes és érvénytelen állapotátmeneteket generáljanak

Használati eset tesztelés

* a tesztek használati esetekből származtathatók
* minden használati eset egy viselkedést határoz meg, melyet a tárgy hajt végre együttműködésben egy vagy több szereplővel
* a teszteket a meghatározott viselkedések végrehajtására tervezik
* hasonló a user story alapú teszteléshez

# 17 Strukturális tesztelési technikák

Strukturális tesztelési technikák: utasításalapú, döntésalapú, adatfolyamalapú tesztelés.

Strukturális tesztelés:

* forráskódon alapszik
* pontos definíciókat, matematikai analízist használ
* pontos méréseket tesz lehetővé

utasításalapú:

* a kódban lévő utasításokat hajtja végre
* a lefedettség a tesztek által végrehajtott utasítások és az összes utasítás hányadosa

döntésalapú tesztelés:

* a kódban lévő döntéseket hajtja végre
* minden lehetséges true, false, case ágat letesztel
* DD-Path gráf alakul ki

100% döntési lefedettség 100%-os utasításlefedettséget jelent, de ez fordítva nem igaz

adatfolyam alapú tesztelés:

* azokra a pontokra fókuszál ahol a változók értéket kapnak és azokra amelyekben ezeket az értékeket felhasználják
* felfeedzhető hibák: változó amit sehol nem használunk, többször definiált változó
* Alfajai:
  + define / use tesztelés: programgráf segítségével vizsgáljuk a programot
  + szeletelés: programgráfot kisebb szeletekre bontjuk, pl szeletek a csúcsok + 1 él

# 18

A különböző tesztelési szintek technikái. Mennyire kapcsolódnak a tesztelési szintekhez bizonyos technikák? Vannak olyan technikák, amelyek tiltottak egy tesztelési szinten?

A tesztelt rendszer, fejlesztési életciklus meghatározza, hogy milyen technikákkal tudunk tesztelni

Vízesés modell

* az egyésgteszt a legegyértelműbb
* teszteljük az egységeket, komponenseket
* integráljuk a komponenseket alrendszerekbe, teszteljük ezeket
* folytassuk az integrációt, amíg a teljes rendszert tesztelnie tudjuk

Inkrementális modell

* buildekben gondolkodunk
* követhető a vízesés modellben alkalmazott rendszerteszt akár
* regressziós teszteléssel bizonyítjuk, hogy ami eddig működött most is működik

Alternatív életciklus modell

* prototípus?

tiltott?

nincs tiltott, szokások alakultak ki

Integráció és rendszerteszt megkülönböztetése

* ez kell ide?

V modellen a fókusz

# 19

Strukturális technikák integrációs tesztelésben. Ismertessen legalább 2 példát arra, hogy integrációs teszteléshez hogyan lehet teszt eseteket létrehozni strukturális technikák alkalmazásával.

semmi nincs a diában

integrációs tesztelés:  
felbontás alapú top dw bottom up

composite alapú

Hány elemet rakunk össze az integrációs tesztelésnél?

ha csak 2-t teszünk össze akkor kevés hiba, de sokáig tart

ha egyszerre

csonk, driver, top down, botton up

szomszédsági

strukturális mert a rendszert modellezem egy gráffal, nem kód, hanem elemei alapján

modellek, pénzkiadó automata

# 20 tesztmenedzsment

A tesztelés menedzsmentje. Ismertesse a legfontosabb teszt menedzsment tevékenységeket. Mit jelent a „defect leakage” és mire alkalmazható?

Legfontosabb menedzsment tevékenységek:

* tesztstartégia kialakítása
* tesztcélok, kockázatok felmérése, megértése
* tesztterv készítése
* tesztelés vezetése, DOD ellenőrzése
* tesztjelentés előkészítése, átadása
* keretrendszerek kialakítása
* metrikák bevezetése, használata
* koordináció
* képességfejlesztés, karrierfejlesztés

Defect leakage:

* mennyi hiba jut át
* release előtt a tesztelés során talált hiba és a release után vagy userek által talál hiba aránya

"Maga a fogalom azt a jelenséget takarja, hogy bizonyos hibák észrevétlenül a következő fázisba haladnak tovább. Ugyanezen a néven hatékonysági metrika is, mellyel mérhető egy szervezet QA hatékonysága. Gyakorlatilag az átvételi tesztek alatt talált hibák száma, osztva a QA által korábban megtalált hibák számával"

# 21 Kockázatkezelés, konfigurációkezelés, mérés

Kockázatkezelés, konfigurációkezelés, mérés a tesztelés szolgálatában. Milyen mérőszámokat szokás alkalmazni a tesztelés hatékonyságának figyelésére?

Általánosan akkor jó a tesztelés

* ha a hibák minél korábban derülnek ki
* kód alapú lefedettség mérhető
* elkészült termék az elfogadái kritériumnak megfelel
* vevő elégedett

Konfigurációmanagement

* célja a munkatermék integritásának biztosítása, a konfiguráció azonosítéséval, elenőrzésével, állapotának követésével
* mindent ami fontos konfigurációkezelés alá kell vonni
* lehetnek alapverziók, amik már jóvá lettek hagyva
* a verziókezelés fontos eleme, ha lehet ezt automatizáljuk (pl SVN, Git)

Kockázatmanagement

* kockázat: annak a valószínűsége, hogy előre nem látható eremény fodul elő
* kockázatmanagement: tevékenyésgek összessége, melyek elősegítik, hogy egy tevékenység tényleges eredménye a lehető legjobban közelítse az elvárt eredményt
  + termékkozkázat: a projekttermék nem felel meg a felhasználók igényeinek, a minőségi jellemzők nem megfelelőek
  + projektkockázat: olyan helyzetetk, melyek negatív hatással lehetnek a projekt céljainak elérésére, pl késedelem, pontatlan becslés, késői változtatás
* lépések: kockázat azonosítása, elemzése, tényezők priorizálása, intézkedések összeírása végrehajtása, eredmények követése
* risk based testing

Hibamegtalálási százalék:

* (ebben a tesztelésben megtalált hibák / összes hiba)\*100
* ez előrejeltésre is jó, ha tudjuk, hogy eddig milyen %-ok voltak a korábbi release-ek hasonló fázisában

Hibamegtalálási arány:

* bármely fáziban megtalál, összes hibák száma/abban a fázisban eltöltött tesztelési, szemlézési idő

Defect leakage:

* mennyi hiba jut át
* release előtt a tesztelés során talált hiba és a release után vagy userek által talál hiba aránya

Fegyelmezett munkamódzser nagyon sokat számít, hogy értelmezni tudjuk és javítani tudjuk. Sokszor kell csinálni és fel kell jegyezni minden eredményt.

# 22 Kockázatkezelés, konfigurációkezelés

Kockázatkezelés, konfigurációkezelés, mérés a tesztelés szolgálatában. Milyen kockázatokkal foglalkozunk a szoftvertesztelés során?

A kockázatmenedzsment olyan tevékenységek összegzése, amelyek elősegítik azt, hogy egy tevékenység tényleges eredménye a lehető legjobban közelítse meg az elvárt eredményt.

* **termékkockázat**: a projekttermék nem felel meg a felhasználók, érdekelt felek igényeinek. A termék sajátos minőségi jellemzőivel kapcsolatos kockázatokat minőségi kockázatoknak is nevezik.
* pl nem a specifikációnak megfelelően működik a szoftver, vagy nem támogatja kellőképpen a nemfunkcionális követelményeket (UX, teljesítmény stb)
* termékkockázatokat a tesztelés hivatott megoldani
* **projektkockázat**: olyan helyzetek, melyek negatív hatással lehetnek a projekt céljainak elérésére, pl. késedelem, pontatlan becslés, késői változtatás, szervezeti és politikai problémák, technikai problémák

Kockázatmenedzsment

* Elkerülés
* ne indítsuk el a veszélyeztetett tevékenységet (aminek a kimenetele kétséges)
* Csökkentés
* a tevékenység megkezdése előtt tegyünk lépéseket, hogy a kockázati elem hatása minimálisra (elfogadható nagyságúra) csökkenjen
* Kompenzálás
* fogadjuk el a kockázati tényező negatív hatását a tevékenységre, de egyéb tényezőkre figyelve igyekezzünk ezt a negatív hatást elfogadható mértékűre csökkenteni
* Megegyezés
* tételezzük fel, hogy a kockázati tényező kifeji hatását és készüljünk fel a negatív hatás kezelésére

**Agilis környezetben**

Szedjük össze a súlyozott esélyeket és a súlyozott hatásokat egy mátrixban. helyezzük el a kockázatokat és kezdjük azokkal, ahol a súlyértékek szorzata a legnagyobb és haladjunk a legkisebbek felé.

# 23

Miben különbözik a teszt menedzsmentben használt teszt terv a műszaki teszt tervtől? A hibák bizonylatolásán kívül mit szükséges még bizonylatolni a tesztelés végrehajtása során?

# 24. agilis tesztelés

Az agilis tesztelés elemei. Az agilis tesztelés és a hagyományos tesztelés kapcsolata. Agilis tesztelési módszerek. Miben különbözik az agilis tesztelői szervezet a hagyományostól?

* Az agilis tesztelés a lehető legkorábbi tesztelést hangsúlyozza a szoftverfejlesztési életciklusban
* Megköveteli a nagyon hangsúlyos vevői részvételt a tesztelésben is, amint a kód hozzáférhetővé válik
* A kódnak elég stabilnak kell lennie ahhoz, hogy lehetővé tegye a rendszertesztelést
* Fontos a regressziós tesztelés
* A kommunikáció a csapaton belül és a csapatok között kulcsfontosságú!

Tehát a hagyományos teszteléssel szemben itt nincs meg előre minden információ, és nem a fejlesztés végén vonjuk be a tesztelőket, sőt. **A tesztelés alapja / fókusza a user story**, és az iterációk során bekerülő elemeket fokozatosan és folyamatosan integrálják a kódbázisba és tesztelik le. Minden új kód integrálása során automatizált tesztek ellenőrzik, hogy a rendszerbe való illesztés során okozott e valamilyen hibát az új kódrészlet és ennek eredményeként a lehető leghamarabb visszajelzést ad az integráció eredményéről.

* Folyamatosan vannak rövid iterációk a tervezés, kódolás és tesztelés tevékenységekre
* A tesztelési tevékenységek is iteratív módon, folyamatosan kerülnek végrehajtásra
* Az agilis tesztelők
* Korai fázistól kezdve dolgoznak
* A projekt teljes idején be vannak vonva a tervezésbe

Egy XP (extreme programming) tesztelési megközelítés a TDD (Test Driven Development) melynek során először teszteket írnak meg és csak aztán a funkcionalitást.

# 25.

A TMMi modell. Ismertesse röviden a TMMi modell célját, szerkezetét, folyamatait.