ISTQB® Certified Tester
Foundation Level
Alapszintű tanúsítvány
Hivatalos magyar nyelvű tanterv

Verzió: 1.10, 2019.12.31.

(Official ISTQB® CTFL Syllabus – Hungarian)

(ISTQB® CTFL Syllabus version: 2018 V3.1)

Magyar Szoftvertesztelői Tanács Egyesület

Gábor Dénes utca 2. Infopark D épület

H-1117 Budapest, Hungary

Tel: +36 1 382 7297

Fax: +36 1 382 7298

info@hstqb.com

Jelen hivatalos Tanterv az alábbi feltételek mellett használható fel:

1) Oktatást szervező személy vagy intézmény a Tantervet felhasználhatja az oktatási anyagának alapjául, de kizárólag abban az esetben, amennyiben a jelen Tanterv, mint forrás és annak szerzői jogvédelmi jelzései egyértelműen megjelölésre kerülnek mind az oktatási anyagban és minden hivatkozási helyen, mind a kurzusokra vonatkozó hirdetésekben, továbbá amennyiben az oktatást szervező és oktatási anyaga a Tantervre vonatkozó érvényes, hivatalos akkreditációval rendelkezik melyet egy ISTQB® tagbizottság állított ki.
2) Egyéb célra való felhasználás, úgymint cikkekben, könyvekben való hivatkozás, illetve részletek közlése a forrás és jelen szerzői jog feltüntetésével történhet.

Eredeti mű címe:

Certified Tester Foundation Level Syllabus, Version 2018 V3.1, International Software Testing Qualifications Board

Eredeti mű szerzői tulajdonjog védelmi jelölése:

Dokumentum verziókövetés

• Verzió	Dátum	Megjegyzés
HTB-2009-1.0	2009-10-06	Első hivatalosan publikált magyar változat az alábbi eredeti változat alapján. Teendők: függelékek megírása.
ISTQB® 2007	2009-05-01	Certified Tester Foundation Level Syllabus Maintenance Release
ISTQB® 2007 1.1	2009-11-15	Kisebb javítások, a Syllabus 2.0 változások átvezetése
ISTQB® 2007 1.2	2009-11-22	Kisebb javítások, a Syllabus 2.0 változások átvezetése
ISTQB® 2007 1.3	2010-02-26	A HTB felülvizsgálatának eredménye. A glosszárium 2.1 verziójával való összehangolás
ISTQB® 2007 2.0	2010-03-15	A HTB felülvizsgálatának eredménye. Szinkronizálás a Szoftvertesztelés egységesített kifejezéseinek gyűjteménye (glossary) 3.0 verzióval, amellyel együtt kerül kiadásra.
ISTQB® 2010 2.1	2011-02-14	A Foundation+Level+Syllabus+(2010).pdf angol nyelvű tanterv változásainak, illetve a kifejezésgyűjtemény (glosszárium) változásainak átvezetése
ISTQB® 2011 3.0	2011-05-04	A HTB által felülvizsgált változat. Szinkronban a Foundation+Level+Syllabus+(2011).pdf angol nyelvű tantervvel, illetve a Szoftvertesztelés egységesített kifejezéseinek gyűjteménye 3.1 verzióval
ISTQB® 2011, HTB 3.01	2011-11-13	Kisebb javítások
ISTQB® 2011, HTB 3.02	2012-02-28	rendszer teszt –> rendszerteszt halott kód –> elérhetetlen kód hibaarány –> meghibásodási ráta hibák csoportosulása -> hibafürtök megjelenése teljesítmény teszt -> teljesítményteszt hibajelenlét kimutatása -> hibák látszólagos hiánya a hibátlan rendszer téveszménye -> a hibamentes rendszer téveszméje tesztelési alapelv (test policy) –> tesztelési irányelvek a „tesztelés” szó több helyen „teszt”-re változtatva, pl: tesztelés becslése –> tesztbecslés tesztelési stratégia –> tesztstratégia tesztelési feltétel -> tesztfeltétel tesztelési folyamat -> tesztfolyamat tesztcél-> tesztcél tesztelési technika -> teszttechnika tesztelési szint -> tesztszint tesztelés tárgyköre (test object) -> teszt tárgya
ISTQB® 2011, HTB 3.03	2017-08-02	3.2.1 fejezet pontosítása (tördelés)
ISTQB® 2018, HTB 1.00	2019-04-19	Az angol nyelvű, Certified Tester Foundation Level Syllabus, Version 2018 Syllabus fordítása
ISTQB® 2018 V3.1, HTB 1.10	2019-12-31	Az ISTQB® által kiadott 2018 V3.1-es frissítés feldolgozása Az eddig beérkezett visszajelzések alapján kisebb javítások

Tartalomjegyzék

Köszönetnyilvánítás

A magyar változat elkészítésében közreműködtek: Albert Laura, Gillespie Raymond, Frei Dávid, Horváth Ágota, Horváth Tamás, Mátyás Márton, Medzihradszky Dénes, Stöckert Tamás.

A korábbi magyar változatok elkészítésében közreműködtek: Albert Laura, Csapó Ákos György, Gillespie Raymond, Frei Dávid, Horváth Ágota, Horváth Tamás, Mátyás Márton, Medzihradszky Dénes, Csonka Béla, Beszédes Árpád, Kapros Gábor, Kovács Attila, Kusper Gábor.

0. Bevezetés

0.1 A hivatalos magyar nyelvű tanterv célja

Jelen tanterv alapot nyújt az International Software Testing Qualifications Board (ISTQB®) által hivatalosan jóváhagyott Alapszintű Tesztelői Tanúsítvány (Certified Tester Foundation Level (CTFL)) alapszintű nemzetközi szoftvertesztelési szakember minősítéshez. Jelen tanterv a Magyar Szoftvertesztelői Tanács Egyesület (Hungarian Testing Board – HTB) által hivatalosan kiadott, az eredeti angol nyelvű tanterv magyar nyelvű fordítása, a hivatalosan elfogadott magyar kifejezésgyűjtemény (ISTQB® Glosszárium) alapján. A tanterv a HTB a nemzeti vizsgáztató szerv (HTB) részére vizsgakérdések saját nyelven történő kidolgozásához, valamint a képzési szolgáltatók akkreditációjához bocsájtja rendelkezésre. A képzési szolgáltatók biztosítják a tanfolyam anyagát, és meghatározzák a megfelelő oktatási módszereket az akkreditációhoz; a tanterv pedig segíti a tanulókat a vizsgára való felkészülésben.

0.2 Az Alapszintű Tesztelő Tanúsítvány a szoftvertesztelésben

Az Alapszintű Tanúsítványt bárki megszerezheti, aki szoftverteszteléssel foglalkozik. Ebbe a körbe tartoznak a tesztelők, tesztelemzők, tesztmérnökök, tesztelési tanácsadók, tesztmenedzserek, felhasználói elfogadási tesztelők és szoftverfejlesztők. Az Alapszintű Tanúsítvány azok számára is kívánatos lehet, akik a szoftvertesztelés alapjait szeretnék megismerni: terméktulajdonosok, projektmenedzserek, minőségirányítók, szoftverfejlesztő menedzserek, üzleti elemzők, IT vezetők és menedzsment tanácsadók. Az Alapszintű Tanúsítvánnyal rendelkezők számára lehetőség nyílik magasabb szintű szoftvertesztelési tanúsítvány megszerzésére.

Az ISTQB® Foundation Level Overview 2018 egy különálló angol nyelvű dokumentum, mely továbbra is érvényes a tanterv ezen verziójára. A dokumentum az alábbi információkat foglalja magába:

- A hivatalos tanterv által kínált üzleti kimenetelek
- Az üzleti kimenetelek és a tanulási célok nyomonkövethetőségi mátrixa.
- A hivatalos tanterv összefoglalója

0.3 A vizsga tárgyát képező Tanulási Célok és kognitív Tudásszintek

A tanulási célok támogatják az üzleti kimeneteleket, valamint ezek alapján készülnek az Alapszintű Tanúsítvány vizsga kérdés sorai.

Általánosságban elmondható, hogy jelen tanterv minden fejezete a vizsga tárgyát képezheti a K1 tudásszinten, kivéve a bevezetőt, illetve a mellékleteket. Ez azt jelenti, hogy a tanterv által, a hat fejezet bármelyikében említett kulcsszavakat vagy koncepciókat a jelöltnek fel kell ismernie, fel kell tudni idéznie és emlékeznie kell rájuk. A tanulási célokhoz rendelt tudásszintek minden fejezet elején felsorolásra kerülnek és az alábbiak szerint vannak osztályozva:

- K1: emlékezés
- K2: megértés
- K3: alkalmazás

A tanulási célokkal összefüggő további példák és tanulási célok a B mellékletben találhatók.

A fejezetek címei alatt lévő „Kulcsszavak” bekezdésekben felsorolt szakkifejezésekre emlékezni kell (K1) akkor is, ha a tanulási célok ezt külön nem említik meg.

0.4 Az Alapszintű Tanúsítvány vizsga

Az Alapszintű Tanúsítványhoz kapcsolódó vizsga az itt közölt tantervre épül. A vizsgakérdések megválaszolásához a tanterv több fejezetének ismeretére is szükség lehet. A tanterv bármely fejezete vizsga tárgyát képezheti, kivéve a bevezetőt, illetve a mellékleteket. Szabványok, szakkönyvek és egyéb ISTQB®

tantervek hivatkozásként szerepelnek, de azok tartalma nem képezheti a vizsga tárgyát azon túlmenően, amit a jelen tanterv összefoglal az érintett szabványokat, szakkönyveket, illetve egyéb ISTQB® tanterveket illetően.

A vizsga feleletválasztós feladatokat tartalmaz. Egy vizsga 40 kérdésből áll. A sikeres vizsgához legalább a kérdések 65%-át (26 kérdés) helyesen kell megválaszolni.

A vizsga letehető akkreditált képzés részeként vagy egyénileg. Magyarországon az Alapszintű Tanúsítvány megszerzéséhez szükséges vizsgát a HTB vizsgaközpontjainál lehet letenni. Bármilyen más vizsgaközpontban való vizsgatételhez a HTB engedélye szükséges. Akkreditált tanfolyam elvégzése nem előfeltétele a vizsga letételének.

0.5 Akkreditáció

Egy, az ISTQB® által elismert nemzeti bizottság jogosult akkreditálni azokat a képzési szolgáltatókat, amelyek tanterve követi a jelen tantervet. Az akkreditáció irányelveit az akkreditációt végrehajtó bizottságtól vagy testülettől lehet beszerezni. Az akkreditált tanfolyamokat elismerik, mint a jelen tantervnek megfelelőt.

0.6 Részletesség

A tanterv részletessége nemzetközi szinten egységes oktatást és vizsgáztatást tesz lehetővé. Ennek érdekében a tanterv a következőkből áll:

- Általános képzési célok, melyek az alapszint célkitűzéseit tartalmazzák.
- Kifejezések listája, melyeket a hallgatóknak képesnek kell lenniük felidézni.
- Az egyes területek oktatási céljai, melyek az elérendő kognitív tanulási eredményt és tudást írják le.
- Az oktatandó alapfogalmak leírása, köztük források például a felhasználható irodalom, illetve szabványok jegyzéke.

A tanterv nem nyújt teljes leírást a szoftvertesztelésről; csak az alapszintű képzések részletességével érinti a témát. Azon tesztfogalmakra és technikákra koncentrál, melyek miden projektre alkalmazhatók, beleértve az Agilis projekteket is. Jelen tanterv nem tartalmaz semmilyen szoftverfejlesztési életciklusra vagy módszerre vonatkozó specifikus Tanulási Célt, de tárgyalja, hogy ezen fogalmak miként értelmezhetők Agilis projektek, illetve iteratív, inkrementális vagy szekvenciális életciklusmodellek esetén.

0.7 A tanterv felépítése

A tanterv hat, a vizsga tárgyát is képező fejezetet tartalmaz. A címsor tartalmazza az adott fejezetre szánt időt. A fejezetszint alatt nincsenek meghatározott időkorlátok. Az akkreditált tanfolyamok számára jelen tanterv minimum 16,75 órás oktatási tervet vár el a hat fejezetre az alábbiak szerint elosztva:

- 1 Fejezet: 175 perc A tesztelés alapjai
- 2 Fejezet: 100 perc Tesztelés a szoftver életciklusán át
- 3 Fejezet: 135 perc Statikus tesztelés
- 4 Fejezet: 330 perc Teszttechnikák
- 5 Fejezet: 225 perc Tesztmenedzsment
- 6 Fejezet: 40 perc Eszköztámogatás a tesztelésben

1. A tesztelés alapjai 175 perc

Kulcsszavak

emberi eredetű hiba, hiba, hibakeresés, kiváltó ok, lefedettség, meghibásodás, minőség, minőségbiztosítás, műszaki teszttervezés, nyomonkövethetőség, tesztadat, tesztbázis, tesztcél, tesztelemzés, tesztelés, teszteljárás, teszteset, tesztfeltétel, tesztfelügyelet, tesztirányítás, tesztkészlet, tesztlezárás, teszt megvalósítása, tesztorákulum, teszt tárgya, teszttervezés, tesztvégrehajtás, tesztfolyamat, tesztver, validáció, verifikáció

Tanulási célok A tesztelés alapjai fejezethez

1.1 Mi a tesztelés?

AK-1.1.1 (K1) Azonosítsa a tesztelés tipikus céljait

AK-1.1.2 (K2) Különböztesse meg a tesztelést és a hibakeresést

1.2 Miért szükséges a tesztelés?

AK-1.2.1 (K2) Adjon példákat a tesztelés szükségességére

AK-1.2.2 (K2) Ismertesse a tesztelés és a minőségbiztosítás közötti kapcsolatot és adjon példákat arra, hogy a tesztelés hogyan járul hozzá a jobb minőség eléréséhez

AK-1.2.3 (K2) Tegyen különbséget az emberi eredetű hiba, hiba és meghibásodás között

AK-1.2.4 (K2) Tegyen különbséget egy hiba kiváltó oka és hatásai között

1.3 Általános tesztelési alapelvek (Hét tesztelési alapelv)

AK-1.3.1 (K2) Ismertesse az általános tesztelési alapelveket (A hét tesztelési alapelv)

1.4 Tesztfolyamat

AK-1.4.1 (K2) Ismertesse a kontextus tesztfolyamatra gyakorolt hatását

AK-1.4.2 (K2) Ismertesse a tesztfolyamaton belüli teszttevékenységeket és a kapcsolódó feladatokat

AK-1.4.3 (K2) Tegyen különbséget a tesztfolyamatot támogató munkatermékek között

AK-1.4.4 (K2) Ismertesse a tesztbázis és a tesztmunkatermékek közötti nyomonkövethetőség fenntartásának fontosságát

1.5 A tesztelés pszichológiája

AK-1.5.1 (K1) Azonosítsa a tesztelés sikerességét befolyásoló pszichológiai tényezőket

AK-1.5.2 (K2) Ismertesse a teszttevékenységekhez, illetve a fejlesztői tevékenységekhez szükséges gondolkodásmód közötti különbségeket

1.1 Mi a tesztelés?

A szoftverrendszerek az élet szerves részét képezik, az üzleti alkalmazásoktól (pl. banki rendszerek) a fogyasztói termékekig (pl. autók). A legtöbb embernek volt már tapasztalata olyan szoftverrel, ami nem az elvártaknak megfelelően működött. A helytelenül működő szoftver sok problémához vezethet, beleértve a pénz- és időveszteséget, az üzleti hírnév elvesztését, illetve akár sérülést vagy halált is eredményezhet. A szoftvertesztelés lehetőséget teremt a szoftverminőség kiértékelésére, illetve a szoftver működés közbeni meghibásodási kockázatának csökkentésére.

Egy gyakori félreértés szerint a tesztelés csak tesztek futtatásából áll, tehát a szoftver végrehajtásából és az eredmények ellenőrzéséből. Az 1.4-es fejezetben leírtak alapján, a szoftvertesztelés egy folyamat, amely számos különböző tevékenységet foglal magában: a tesztvégrehajtás (beleértve az eredmények ellenőrzését) csak egy ezen tevékenységek közül. Ezen felül a tesztfolyamat részét képezik az olyan tevékenységek is, mint a teszttervezés, elemzés, műszaki teszttervezés, tesztek megvalósítása, az eredmények és az előrehaladás jelentése, illetve a teszt tárgyának minőségre vonatkozó kiértékelése.

Néhány tesztelés magában foglalja a tesztelt komponens vagy rendszer végrehajtását is; ezt a fajta tesztelést dinamikus tesztelésnek nevezzük. Más esetben a tesztelt komponens vagy rendszer futtatására nem terjed ki a tesztelés; ezt statikus tesztelésnek hívjuk. Ennek megfelelően, a munkatermékek, mint például követelmények, felhasználói történetek, illetve forráskód felülvizsgálata is a tesztelés részét képezik.

Egy másik gyakori félreértés a teszteléssel kapcsolatban, hogy teljes mértékben a követelmények, felhasználói történetek vagy más specifikációk verifikációjára fókuszál. A tesztelés valóban kiterjed a specifikált követelményeknek való megfelelés ellenőrzésére, azonban tartalmazza a validációt is, amely azt ellenőrzi, hogy a rendszer megfelel-e a felhasználói, illetve más érintett fél igényeinek a működési környezetben.

A teszttevékenységek szervezése és kivitelezése másképpen történik a különböző életciklusokban (lásd a 2.1-es fejezetben).

1.1.1 A tesztelés tipikus céljai

Bármely projekt esetén a tesztelés céljai az alábbiak lehetnek:

- A munkatermékek, mint például a követelmények, felhasználói történetek, műszaki tervek és kód hibamegelőzés céljából történő kiértékelése
- Annak igazolása, hogy az összes meghatározott követelmény teljesült-e
- Annak ellenőrzése, hogy a teszt tárgya teljes mértékben implementálásra került, továbbá annak, validálása, hogy a felhasználók, illetve más érintett felek elvárásainak megfelelően működik.
- A teszt tárgyának minőségébe vetett bizalom kiépítése
- A meghibásodások és hibák megtalálása és ezáltal a nem megfelelő szoftverminőség kockázati szintjének csökkentése
- Megfelelő információ biztosítása az érintett feleknek, hogy ezáltal megalapozott döntést hozhassanak különösen a teszt tárgyának minőségi szintjére vonatkozóan
- A szerződésben foglalt, jogi vagy szabályozott követelményeknek, szabványoknak való megfelelés biztosítása, és/vagy igazolni a teszt tárgyának ezen követelményeknek, szabványoknak való megfelelőségét

A tesztelés céljai változhatnak a tesztelt komponens vagy rendszer, a tesztszint és a szoftverfejlesztési életciklusmodell kontextusától függően. Ezek a különbségek lehetnek például:

- Komponenstesztelés során egy cél lehet a lehető legtöbb meghibásodás előidézése annak érdekében, hogy az őket okozó hibákat azonosítsák és hamar kijavítsák. Egy másik cél lehet a komponenstesztek kód lefedettségének növelése.
- Elfogadási tesztelés során a cél lehet annak igazolása, hogy a rendszer az elvárásoknak megfelelően működik és teljesíti a követelményeket. Egy másik célja lehet az érintett felek informálása arról, hogy egy adott időpontban milyen kockázattal járna a rendszer kiadása.

1.1.2 Tesztelés és hibakeresés

A tesztelés és a hibakeresés két különböző fogalom. A tesztek futtatásával kimutathatók a programhibák által a szoftverben okozott meghibásodások. A hibakeresés az a fejlesztési tevékenység, amelynek során megtalálják, elemzik és kijavítják ezeket a hibákat. Ezt követően az ellenőrző tesztelés során ellenőrzik, hogy a javítások orvosolták-e a hibákat. Néhány esetben a tesztelők felelősek a kezdeti tesztért és a végleges ellenőrző tesztért, amíg a fejlesztők a hibakeresést és a kapcsolódó komponens- és komponensintegrációs tesztelést (folyamatos inegráció) végzik. Azonban Agilis fejlesztés és néhány más szoftverfejlesztési életciklus során a tesztelőket is bevonhatják a hibakeresésbe és a komponenstesztelésbe.

Az ISO szabvány (ISO/IEC/IEEE 29119-1) további információkkal szolgál a szoftvertesztelési fogalmakról.

1.2 Miért szükséges a tesztelés?

A komponensek és rendszerek, illetve a hozzájuk kapcsolódó dokumentációk szigorú tesztelése segíthet a működés közbeni meghibásodások kockázatának csökkentésében. A hibák detektálása és az ezt követő kijavítása hozzájárul a komponensek és rendszerek minőségéhez. Továbbá szoftvertesztelésre szükség lehet akkor is, ha a szoftvernek szerződésben foglalt vagy jogi előírásoknak, ipari szabványoknak kell megfelelnie.

1.2.1 A tesztelés hozzájárulása a sikerhez

A számítástechnika története során gyakoriak voltak az olyan esetek, amikor a szoftvereket és rendszereket használatba vették, és a hibák jelenléte miatt ezt követően meghibásodások léptek fel, vagy más módon nem elégítette ki a szoftver az érintett felek elvárásait. Azonban a megfelelő tesztelési technikák használatával csökkenthető az ehhez hasonló problémás átadások gyakorisága, amennyiben ezen technikákat a megfelelő szintű tesztelői szakértelemmel, a megfelelő tesztszinten, illetve a szoftverfejlesztési életciklus megfelelő pontján alkalmazzák. Például:

- A tesztelők a követelmények felülvizsgálatába vagy a felhasználói történetek finomhangolásába történő bevonásával észlelhetők lennének a hibák ezen munkatermékekben. A követelményekben lévő hibák azonosítása és eltávolítása csökkenti a helytelen vagy tesztelhetetlen feature-ök fejlesztésének kockázatát.
- Ha a tesztelők és a rendszertervezők együttműködnek a rendszer tervezése során, az növelheti mindkét fél megértését a rendszerről és annak teszteléséről. A magasabb fokú megértés csökkentheti az alapvető tervezési hibákat és lehetővé teszi, hogy a teszteket már a korai szakaszban meg tudják tervezni.
- Ha a tesztelők és a fejlesztők együttműködnek a kód fejlesztése során, az javíthatja a kód, illetve a tesztelésének megértését mindkét fél részéről. A magasabb fokú megértés csökkentheti a kódban és a tesztekben előforduló hibák kockázatát.
- Ha a tesztelők a kiadás előtt verifikálják és validálják a szoftvert, akkor olyan meghibásodásokat is észlelhetnek, amelyeket másképp nem vettek volna észre, illetve támogathatják a

meghibásodásokat előidéző hibák eltávolításának folyamatát (hibakeresés). Ez növeli annak a valószínűségét, hogy a szoftver teljesíti az érintett felek igényeit és kielégíti a követelményeket.

Ezen példákon felül a meghatározott tesztcélok elérése (lásd az 1.1.1-es fejezetet) hozzájárul a teljes szoftverfejlesztés és a karbantartás sikerességéhez.

1.2.2 Minőségbiztosítás és tesztelés

Bár az emberek a minőségbiztosítás (vagy QA) kifejezést a tesztelésre utalva használják, a két fogalom nem ugyanaz, de kapcsolódnak egymáshoz. Egy tágabb fogalom, a minőségmenedzsment fogja őket össze. A minőségmenedzsment magában foglal minden olyan tevékenységet, amely egy szervezetet irányít és szabályoz a minőséget tekintve. Más tevékenységek mellett a minőségmenedzsment részét képezi mind a minőségbiztosítás, mind a minőségirányítás. Előbbi tipikusan a megfelelő folyamatok követésére fókuszál annak érdekében, hogy bizalmat teremtsen a megfelelő minőségi szint elérésében. A folyamatok helyes végrehajtásának eredményeként magasabb minőségű munkatermékek jönnek létre, ami hozzájárul a hibák megelőzéséhez. Továbbá a kiváltó ok elemzés használata, a hibák okának megtalálása és azok eltávolítása érdekében, a visszatekintő megbeszélés megállapításainak megfelelő alkalmazása a folyamatok fejlesztése érdekében fontos szerepet játszanak a hatékony minőségbiztosítás elérésében.

A minőségirányítás sokféle tevékenységből áll, beleértve a teszttevékenységeket, amelyek a minőség megfelelő szintjének elérését támogatják. A teszttevékenységek a teljes szoftverfejlesztés vagy a karbantartási folyamat részeit képezik. Mivel a minőségbiztosítás a teljes folyamat megfelelő végrehajtásával foglalkozik, ezért támogatja a megfelelő tesztelést. Az 1.1.1-es, illetve az 1.2.1-es fejezetben leírtak szerint a tesztelés különböző módokon járul hozzá a minőség eléréséhez.

1.2.3 Emberi eredetű hibák, hibák és meghibásodások

Egy ember elkövethet hibát (tévedhet), amely a szoftverkódban vagy valamely kapcsolódó munkatermékben hiba megjelenését idézheti elő. Egy emberi eredetű hiba, ami adott munkatermékben hibát okoz, kiválthat egy olyan emberi eredetű hibát, ami pedig egy kapcsolódó munkatermékben egy másik hiba megjelenését eredményezi. Például egy téves követelményértelmezés követelményhibához vezethet, ami pedig olyan programozási hibát eredményez, ami hibát okoz a kódban.

Ha a hibás kódot lefuttatjuk, meghibásodást okozhat, de nem feltétlenül minden esetben. Néhány hiba például speciális bemeneteket vagy előfeltételeket igényel, hogy meghibásodást váltson ki, ami lehet, hogy csak nagyon ritkán vagy egyáltalán nem fordul elő.

Az emberi eredetű hibáknak sok oka lehet, például:

- Szoros határidők
- Emberi tévedhetőség
- Tapasztalatlan vagy nem megfelelően képzett projektrésztvevők
- Nem megfelelő kommunikáció a projektrésztvevők között, beleértve a követelményekkel és a tervekkel kapcsolatos kommunikációs hibákat
- A kód, a terv, az architektúra, a megoldandó probléma és/vagy a használt technológia komplexitása
- Félreértések a rendszerbeli és a rendszerközi interfészekkel kapcsolatban, különösen, amikor nagy a rendszerbeli, illetve a rendszerközi kölcsönhatások száma
- Új, ismeretlen technológiák

A meghibásodásokat a hibás kódon kívül előidézhetik környezeti viszonyok is. Például sugárzás, elektromágneses mezők és a szennyezés is okozhatnak hibát a firmware-ben vagy a hardver feltételek megváltoztatása által befolyásolhatják a szoftver végrehajtását.

Nem minden váratlan teszteredmény meghibásodás. Hamis pozitív eredmények létrejöhetnek a hibásan végrehajtott tesztek eredményeként, vagy a tesztadatokban, a tesztkörnyezetben, más tesztverben lévő hibák miatt, illetve egyéb okokból. Fordított helyzet is előfordulhat, amikor hasonló emberi eredetű hibák vagy hibák hamis negatív eredményekhez vezetnek. Ezek olyan tesztek, amik nem detektálják a hibákat, melyeket kellett volna; a hamis pozitívak pedig hibaként vannak számontartva, de igazából nem azok.

1.2.4 Hibák, kiváltó okok és hatások

A hibák kiváltó okai azok a legkorábbi tevékenységek vagy feltételek, amelyek hozzájárultak a hibák kialakulásához. A hibákat lehet elemezni a kiváltó okok azonosítása érdekében vagy azért, hogy csökkentsék a hasonló hibák kialakulását a jövőben. A legjelentősebb kiváltó okokra fókuszálva, a kiváltó ok elemzés olyan folyamatfejlesztéseket eredményezhet, amelyek jelentős számú, a jövőben előforduló hibák megjelenését előzhetik meg.

Példaként, tegyük fel, hogy egyetlen hibás kódsor miatti helytelen kamatfizetés vásárlói panaszt eredményez. A hibás kódot egy olyan kétértelmű felhasználói történet keretein belül írták, amely a terméktulajdonos kamatszámításra vonatkozó félreértéséből adódott. Ha a kamatszámításokban nagy arányban találhatók hibák, és ezen hibák kiváltó oka hasonló félreértéseken alapul, a terméktulajdonosok oktathatók a kamatfizetésről, hogy a jövőben csökkentsék az ilyen jellegű hibákat.

Ebben a példában a vásárlói panaszok a hatások. A hibás kamatfizetések a meghibásodások. A kódban szereplő nem megfelelő számítás a hiba és ezt az eredeti hiba, a felhasználói történet kétértelműsége eredményezte. Az eredeti hiba kiváltó oka a terméktulajdonos hiányos tudása volt, ami azt okozta, hogy hibát vétett a felhasználói történet írásakor. A kiváltó ok elemzés folyamata az ISTQB-CTEL-TM Syllabus-ban, illetve az ISTQB-CTEL-ITP Syllabus-ban kerül részletezésre.

1.3 Általános tesztelési alapelvek

Az elmúlt 50 évben számos tesztelési alapelvet javasoltak, amelyek általános, mindennemű tesztelésre vonatkozó irányelveket adnak.

1. **A tesztelés a hibák jelenlétét mutatja, nem a hiányukat**

A tesztelés kimutathatja a hibák jelenlétét, de azt nem képes igazolni, hogy nincsenek hibák. A teszteléssel csökken annak az esélye, hogy a szoftverben felfedezetlen hibák maradnak, de ha nem találnak hibát, az nem bizonyítja, hogy a rendszer helyes.

2. **Nem lehetséges kimerítő teszt**

Kimerítő teszt, azaz a mindenre (a bemenetek és előfeltételek minden kombinációjára) kiterjedő tesztelés a triviális eseteket leszámítva nem lehetséges. Ahelyett, hogy megkísérelnénk a kimerítő tesztelést, kockázatelemzés, tesztelési technikák használata és priorizálása szükséges a tesztelési erőforrások összpontosításához.

3. **A korai tesztelés időt és pénzt spórol**

A hibák korai megtalálásának érdekében a szoftverfejlesztési életciklus során mind a statikus, mind a dinamikus teszttevékenységeket a lehető legkorábban el kell kezdeni. A korai tesztelésre az angol szakirodalomban néha shift left-ként utalnak. A szoftverfejlesztési életciklus alatti korai tesztelés segít a költséges változtatások csökkentésében vagy eliminálásában (lásd: 3.1-es fejezet).

4. **Hibafürtök megjelenése**

A kiadást megelőző tesztelés során megtalált hibák többsége rendszerint néhány modulban van, vagy ezen modulok felelősek a működési meghibásodások többségéért. A megjósolt hibafürtök és a tesztelés vagy

működés során ténylegesen megfigyelt hibafürtök fontos bemenetként szolgálnak a tesztelési erőforrások összpontosítása érdekében alkalmazott kockázatelemzés számára (ahogy a 2. elvben említettük)

5. **Kísérd figyelemmel a féregirtó paradoxont**

Ha ugyanazokat a teszteket hajtjuk végre újra és újra, akkor ezen tesztek egy idő után nem fognak új hibákat találni. Ahhoz, hogy új hibákat találjunk, a létező teszteket és tesztadatokat módosítani kell, illetve új teszteket kell írni. (A tesztek egy idő után már nem hatékonyak a hibák megtalálásában csakúgy, ahogyan egy idő után a féregirtók sem hatékonyak a rovarok elpusztításában.) Néhány esetben például az automatikus regressziós tesztelés esetében a féregirtó paradoxonnak egy előnyös kimenete is van, ami a regressziós hibák viszonylag alacsony száma.

6. **A tesztelés függ a körülményektől**

A tesztelést különböző körülmények esetén különbözőképpen hajtják végre. Például egy biztonságkritikus ipari irányító szoftvert másképp tesztelnek, mint egy e-kereskedelmi mobilalkalmazást. Egy másik példában, a tesztelés egy Agilis projektben különbözik egy szekvenciális szoftverfejlesztési életciklust alkalmazó projekt tesztelésétől (lásd: 2.1-es fejezet).

7. **A hibamentesség egy téveszme**

Néhány szervezet azt várja el, hogy a tesztelők minden lehetséges tesztet le tudjanak futtatni és minden lehetséges hibát megtaláljanak, de az 1. és a 2. alapelv kimondja, hogy ez lehetetlen. Továbbá téveszme (téves meggyőződés) azt várni, hogy csupán azzal, hogy sok hibát találnak meg és javítanak ki, biztosítják a rendszer sikerességét. Például a meghatározott követelmények teljeskörű tesztelése és az összes talált hiba kijavítása még mindig eredményezhet egy nehezen használható rendszert, ami nem elégíti ki a felhasználók összes igényét és elvárását, vagy más hasonló rendszerekhez képest silányabb minőségű.

Ezen és más tesztelési alapelvekkel kapcsolatos példákért lásd: Myers 2011, Kaner 2002, Weinberg 2008 és Beizer 1990.

1.4 Tesztfolyamat

Nincsen egyetlen univerzális szoftvertesztelési folyamat, de léteznek gyakori teszttevékenységek, amik nélkül kevésbé valószínű, hogy a tesztelés eléri a kitűzött céljait. A teszttevékenységek ezen halmaza a tesztfolyamat. A megfelelő speciális szoftvertesztelési folyamat bármely adott szituációban számos tényezőtől függ. A szervezet tesztstratégiája tartalmazza, hogy a tesztfolyamat melyik teszttevékenységeket foglalja magában, hogyan implementálják ezeket, és mikor fordulnak elő.

1.4.1 Tesztfolyamat kontextusba helyezve

Egy szervezet tesztfolyamatát befolyásoló környezetfüggő tényezők tartalmazzák, de nem korlátozottak a következőkre:

- A használt szoftverfejlesztési életciklusmodell és projektmódszertanok
- A figyelembe vett tesztszintek és teszttípusok
- Termék- és projektkockázatok
- Üzleti tevékenységi kör
- Működési korlátozások, amik tartalmazzák, de nem korlátozottak a következőkre:
    - Költségvetések és erőforrások
    - Időskála
    - Komplexitás

    - Szerződésben foglalt és szabályozási követelmények
- Szervezeti vezérelvek és gyakorlatok
- Szükséges belső és külső szabványok

Az alábbi alfejezetek egy szervezet tesztfolyamatának általános aspektusait írják le, mint például:

- Teszttevékenységek és feladatok
- Tesztmunkatermékek
- Nyomonkövethetőség a tesztbázis és a tesztmunkatermékek között

Nagyon hasznos, ha a tesztbázis (bármely tervezett tesztelési szintre vagy teszttípusra) rendelkezik előre meghatározott, mérhető lefedettségi feltételekkel. A lefedettségi feltételek hatékonyan szerepelhetnek, mint kulcs teljesítmény indikátorok (angol szakirodalomban KPI), hogy ezek irányítsák azokat a tevékenységeket, amelyek a szoftvertesztelési célok elérését mutatják (lásd az 1.1.1 fejezetet).

Például egy mobilalkalmazás esetében a tesztbázis magába foglalhatja a követelmények listáját és egy listát a támogatott mobileszközökről. Mindegyik követelmény és minden támogatott eszköz a tesztbázis egy eleme. A lefedettségi feltételek megkövetelhetik, hogy a tesztbázis minden elemére legalább egy teszteset létezzen. Miután végrehajtották, ezen tesztek eredményei tájékoztatják az érintett feleket arról, hogy a meghatározott követelmények teljesültek-e, valamint arról, hogy találtak-e meghibásodásokat a támogatott eszközökben.

Az ISO szabvány (ISO/IEC/IEEE 29119-2) további információkat tartalmaz a tesztfolyamatokról.

1.4.2 Teszttevékenységek és feladatok

A teszfolyamat az alábbi fő tevékenységcsoportokból áll:

- Teszttervezés
- Tesztfelügyelet és -irányítás
- Tesztelemzés
- Műszaki teszttervezés
- Teszt megvalósítása
- Tesztvégrehajtás
- Tesztlezárás

Minden főbb tevékenységcsoport további résztevékenységekből áll, amelyeket a következő alfejezetekben tárgyalunk. Minden egyes tevékenység több, különálló feladatból áll, amelyek projektenként vagy kiadásonként eltérőek lehetnek.

Továbbá bár ezen főbb tevékenységcsoportok közül több logikailag szekvenciálisnak tűnhet, gyakran iteratív módon kerülnek megvalósításra. Az Agilis fejlesztés például a szoftvertervezés, megvalósítás és tesztelés kis iterációiból áll, melyek folyamatosan zajlanak, folyamatos tervezéssel támogatva. A teszttevékenységek ebben a szoftverfejlesztési megközelítésben tehát szintén iteratív módon, folyamatosan zajlanak. Még a szekvenciális szofverfejlesztésben is, a a főbb tevékenységi csoportok egymást követő sorozata is tartalmazhat átfedéseket, kombinációkat, egybeeséseket, kihagyásokat, ezért általában szükséges ezen főbb tevékenységi csoportokat a rendszer, valamint a projekt kontextusában testreszabni.

Teszttervezés

A teszttervezés azokat a tevékenységeket foglalja magában, amelyek definiálják a tesztelés céljait, illetve ezen célok teljesítésének módját a kontextus (pl. megfelelő teszttechnikák és feladatok specifikálása, teszt ütemtervének kialakítása a határidők betartása érdekében) által meghatározott korlátozásokat figyelembe véve. A tesztterv szükség esetén újragondolandók a felügyeleti és irányítási tevékenységek visszajelzései alapján. A teszttervezést bővebben az 5.2-es fejezetben tárgyaljuk.

Tesztfelügyelet és -irányítás

A tesztfelügyelet magában foglalja az aktuális előrehaladás és a tervezett előrehaladás folyamatos összehasonlítását, a teszttervben definiált felügyeleti metrikák felhasználásával. A tesztirányítás a teszttervben (ez időről-időre frissülhet) foglalt célok eléréséhez szükséges tevékenységek végrehajtását foglalja magában. A tesztfelügyelet és -irányítás tevékenységeit a kilépési feltételek kiértékelése támogatja, amire néhány szoftverfejlesztési életciklus modellben a „Kész“ definíciójaként utalnak (lásd az ISTQB-AT Foundation Level Agile Tester Extension Syllabus referenciát). Például egy adott tesztszinthez tartozó tesztvégrehajtáshoz a kilépési feltételek kiértékelése tartalmazhatja:

- Teszteredmények és naplók ellenőrzése meghatározott lefedettségi feltételek szerint
- A komponens vagy rendszer minőségi szintjének kiértékelése a teszteredmények és naplók alapján
- Annak meghatározása, hogy szükséges-e több teszt (pl. ha a tesztekkel eredetileg elérni kívánt termékkockázat-lefedettséget nem sikerült elérni, további tesztek írása és végrehajtása szükséges)

A tervhez mért előrehaladásról az érintett feleket tesztstátuszjelentésekben tájékoztatják, amelyek tartalmazzák az eltéréseket a tervhez képest, illetve információkat, amelyek a tesztelés leállításáról szóló döntés meghozatalát támogatják.

A tesztfelügyelet- és tervezés az 5.3-as fejezetben kerül bővebben kifejtésre.

Tesztelemzés

A tesztelemzés során a tesztbázist elemzik annak érdekében, hogy azonosítsák a tesztelhető funkcionalitásokat és meghatározzák a kapcsolódó tesztfeltételeket. Másként fogalmazva, a tesztelemzés meghatározza azt, hogy „mit teszteljünk“ mérhető lefedettségi feltételekkel megadva.

A tesztelemzés az alábbi fő tevékenységeket tartalmazza:

- Az aktuális tesztszintnek megfelelő tesztbázis elemzése, például:
    - követelményspecifikációk például üzleti követelmények, funkcionális követelmények, rendszerkövetelmények, felhasználói történetek, epic-ek, használati esetek vagy hasonló munkatermékek, amik a komponens vagy rendszer kívánt funkcionális és nemfunkcionális viselkedését specifikálják
    - tervezéssel vagy implementációval kapcsolatos információ például rendszer- vagy szoftverarchitektúra diagramok vagy dokumentumok, műszaki terv specifikációk, hívási folyamok, modellezési diagramok (pl. UML vagy egyed-kapcsolat diagramok), interfész specifikációk vagy hasonló munkatermékek, amelyek a komponens vagy a rendszer struktúráját specifikálják
    - magának a komponensnek vagy rendszernek az implementációja, beleértve a kódot, adatbázis metaadatokat és lekérdezéseket, valamint az interfészeket
    - kockázatelemzés-jelentések, melyek a komponens vagy a rendszer funkcionális, nemfunkcionális és strukturális aspektusaira is vonatkozhatnak

- A tesztbázis és a tesztelemek kiértékelése a különböző típusú hibák azonosítása érdekében, melyek lehetnek:
    - Kétértelműség
    - Kihagyások
    - Inkonzisztenciák
    - Pontatlanságok
    - Ellentmondások
    - Felesleges utasítások
    - Tesztelendő funkcionalitások és funkcióhalmazok meghatározása
    - Minden egyes funkcióhoz a tesztfeltételek definiálása és priorizálása a tesztbázis elemzése alapján, illetve funkcionális, nemfunkcionális és strukturális jellemzők, valamint más üzleti és technikai tényezők, kockázati szintek figyelembevétele
    - A tesztbázis minden eleme és a hozzá kapcsolódó tesztfeltételek közötti kétirányú nyomonkövethetőség meghatározása (lásd: 1.4.3-as és 1.4.4-es fejezetek)

A feketedoboz, fehérdoboz és a tapasztalat alapú teszttechnikák alkalmazása hasznos lehet a tesztelemzés folyamata során (lásd: 4-es fejezet) annak érdekében, hogy csökkentsük a fontos tesztfeltételek kihagyásának valószínűségét, illetve, hogy precízebb és pontosabb tesztfeltételeket tudjunk definiálni.

Néhány esetben a tesztelemzés olyan tesztfeltételeket eredményez, amelyeket a tesztvázlatokban tesztcélként lehet használni. A tesztvázlatok tipikus munkatermékei a tapasztalat alapú tesztelés (lásd: 4.4.2-es fejezet) néhány típusának. Amikor ezen tesztcélok nyomonkövethetők a tesztbázisig, az elért lefedettség mérhető ezen tapasztalat alapú tesztelések során.

A tesztelemzés során a hibák azonosítása egy fontos lehetséges előny, különösen, amikor más felülvizsgálati folyamatot nem alkalmaznak és/vagy amikor a teszt- és a felülvizsgálati folyamat szorosan kapcsolódnak egymáshoz. Ezek a tesztelemzés tevékenységek nemcsak a követelmények konzisztenciáját, megfelelő kifejezésmódját és teljességét verifikálják, hanem azt is validálják, hogy a követelmények megfelelnek az ügyfél, a felhasználó és más érintettek igényeinek. Például az olyan technikák, mint a viselkedésvezérelt fejlesztés (angol irodalomban BDD) és az elfogadási tesztvezérelt fejlesztés (ATDD), amelyek magukban foglalják a tesztfeltételek és tesztesetek felhasználói történetekből és elfogadási feltételekből való generálását a kódolás előtt. Ezen technikák azon túl, hogy verifikálnak, validálnak is, további hibákat is képesek találni a felhasználói történetekben és az elfogadási feltételekben (lásd az ISTQB© Foundation Level Agile Tester Extension Syllabus referenciát).

Műszaki teszttervezés

A műszaki teszttervezés során a tesztfeltételekből magasszintű teszteseteket, magasszintű teszteset-halmazokat és más tesztvereket alakítanak ki. Amíg a tesztelemzés arra a kérdésre ad választ, hogy “mit teszteljünk?”, addig a műszaki teszttervezés azt válaszolja meg, hogy “hogyan teszteljünk?”.

A műszaki teszttervezés az alábbi fő tevékenységekből áll:

- Tesztesetek és tesztesetek halmazainak tervezése és priorizálása
- A tesztfeltételek és tesztesetek támogatásához szükséges tesztadatok meghatározása
- A tesztkörnyezet megtervezése, valamint a szükséges infrastruktúra és eszközök azonosítása

- A tesztbázis, tesztfeltételek és a tesztesetek közötti kétirányú nyomonkövethetőség meghatározása (lásd: 1.4.4-es fejezet)

A műszaki teszttervezés során a tesztesetek és teszthalmazok tesztfeltételekből történő kidolgozása gyakran teszttechnikák alkalmazását teszi szükségessé (lásd: 4-es fejezet).

A tesztelemzéshez hasonlóan, a műszaki teszttervezés is eredményezheti a tesztbázisban előforduló hasonló típusú hibák megtalálását. Ahogyan a tesztelemzés során is, a műszaki teszttervezés alatt a hibák azonosítása fontos potenciális haszon.

Tesztmegvalósítás

A tesztmegvalósítás során létrehozzák és/vagy befejezik a tesztvégrehajtáshoz szükséges tesztvert, beleértve a tesztesetek teszteljárásokba történő szervezését. Amíg a műszaki teszttervezés arra a kérdésre ad választ, hogy “hogyan teszteljünk?”, addig a tesztmegvalósítás azt válaszolja meg, hogy “rendelkezésünkre áll minden, ami a tesztek futtatásához szükséges?”

A tesztmegvalósítás az alábbi fő tevékenységekből áll:

- Teszteljárások fejlesztése és priorizálása, valamint potenciálisan automatizált tesztszkriptek készítése
- Tesztkészletek létrehozása a teszteljárásokból és (ha van) az automatizált tesztszkriptekből
- A tesztkészletek tesztvégrehajtási ütemtervbe történő szervezése olyan módon, hogy hatékony tesztvégrehajtást eredményezzen (lásd: 5.2.4-es fejezet)
- Tesztkörnyezet (beleértve potenciálisan a teszttámogató szoftverkörnyezetet, szolgáltatás virtualizációt, szimulátorokat és más infrastruktúra elemeket) kiépítése és annak ellenőrzése, hogy megfelelően lett beállítva minden, ami szükséges
- Tesztadatok előkészítése és annak biztosítása, hogy a tesztkörnyezetbe megfelelően vannak betöltve
- A tesztbázis, tesztfeltételek, tesztesetek, teszteljárások és tesztkészletek közötti kétirányú nyomonkövethetőség verifikálása és frissítése (lásd: 1.4.4-es fejezet)

A műszaki teszttervezés és a tesztmegvalósítás feladatait gyakran kombinálják.

A felderítő tesztelésnél és a tapasztalat alapú tesztelés más típusainál a műszaki teszttervezés, illetve az implementálás kivitelezése és dokumentálása történhet a tesztvégrehajtás részeként. A felderítő tesztelés alapulhat a tesztvázlatokon (ami a tesztelemzés során jött létre), valamint a felderítő teszteket a tervezés és implementálás után rögtön végrehajtják (lásd: 4.4.2-es fejezet).

Tesztvégrehajtás

A tesztvégrehajtás során a tesztkészleteket a tesztvégrehajtási ütemtervben meghatározott sorrendben futtatják.

A tesztvégrehajtás az alábbi fő tevékenységekből áll:

- A tesztelem(ek) vagy a teszt tárgyának, a teszteszköz(ök) és a tesztver azonosítójának, illetve verziójának rögzítése
- A tesztek manuálisan vagy tesztvégrehajtási eszközök segítségével történő végrehajtása
- A várt és a tényleges eredmények összehasonlítása

- A rendellenességek elemzése a valószínű okuk meghatározása céljából (pl. a meghibásodásokat okozhatják a kódban levő hibák, de hamis pozitívak is megjelenhetnek (lásd: 1.2.3-as fejezet)
- Hibák jelentése a megfigyelt meghibásodások alapján (lásd: 5.6-os fejezet)
- A tesztvégrehajtás kimenetének naplózása (pl. sikeres, elbukott, blokkolt)
- Teszttevékenységek ismétlése vagy egy rendellenesség esetén végzett tevékenységek eredményeként, vagy a tervezett tesztelés részeként (pl. egy kijavított teszt végrehajtása, ellenőrző tesztelés és/vagy regressziós tesztelés)
- A tesztbázis, tesztfeltételek, tesztesetek, teszteljárások és teszteredmények közötti kétirányú nyomonkövethetőség verifikálása és frissítése

Tesztlezárás

A teszlezárás tevékenységei során a tapasztalatok, a tesztver és az egyéb releváns információk véglegesítése érdekében adatokat gyűjtenek a befejezett teszttevékenységekből. A tesztlezáráshoz kapcsolódó tevékenységeket projektmérföldkövek elérésekor hajtják végre, például amikor kiadnak egy szoftverrendszert, befejeznek (vagy visszavonnak) egy tesztprojektet, egy Agilis projektiterációt, egy tesztszinten való tesztelést, vagy egy karbantartó frissítést.

A tesztlezárás az alábbi fő tevékenységekből áll:

- Annak ellenőrzése, hogy minden hibajelentést lezártak, változtatáskérések vagy termék-teendőlista elemek felvétele a tesztvégrehajtás végén megoldatlan hibák kezelésére
- Összefoglaló tesztjelentés készítése az érintett felek tájékoztatása céljából
- A tesztkörnyezet, a tesztadatok, a teszt infrastruktúra és más tesztver véglegesítése és archiválása későbbi felhasználásra
- A tesztver átadása a karbantartást végző csapatoknak, más projektcsapatoknak és/vagy más érintetteknek, akiknek előnyére válhat a használata
- A befejezett teszttevékenységek tanulságainak elemzése a jövőbeli iterációkhoz, kiadásokhoz és projektekhez szükséges változtatások meghatározására
- A gyűjtött információ felhasználása a tesztfolyamat érettségének fejlesztésére

1.4.3 Tesztmunkatermékek

A tesztmunkatermékeket a teszfolyamat részeként hozzák létre. Ahogy a tesztfolyamat megvalósítási módja is jelentősen eltér a különböző szervezetek esetében, abban is nagy különbségek vannak, ahogy a tesztfolyamat során a munkatermékeket szervezik és kezelik, illetve ahogy a munkatermékeket elnevezik. Ez a tanterv a fent leírt tesztfolyamathoz, illetve az itt és az ISTQB® glosszáriumban leírt munkatermékekhez igazodik. Az ISO szabvány (ISO/IEC/IEEE 29119-3) szintén alapul szolgálhat a munkatermékekhez.

Számos, ebben a fejezetben tárgyalt munkatermék kezelhető tesztmenedzsment-, valamint hibamenedzsment eszközök segítségével (lásd: 6-os fejezet).

Teszttervezési munkatermékek

A teszttervezési munkatermékek tipikusan egy vagy több tesztterv foglalnak magukban. A teszterv információt tartalmaz a tesztbázisról, amihez más munkatermékek kapcsolódhatnak a nyomonkövetési információkon keresztül (lásd: lejjebb, illetve az 1.4.4-es fejezetben), illetve kilépési feltételeket (vagy a

"Kész" definícióját), amelyet a tesztfelügyelet és -irányítás során fognak alkalmazni. A teszttervet az 5.2-es fejezetben tárgyaljuk.

Tesztfelügyelet és -irányítás munkatermékek

A tesztfelügyelet és -irányítás munkatermékek tipikusan különböző típusú tesztjelentéseket foglalnak magukban, többek között tesztstátuszjelentéseket folyamatosan, vagy rendszeres alapon létrehozva és az összefoglaló tesztjelentéseket, amelyeket a különböző mérföldkövek elérésénél hoznak létre. Az összes tesztjelentésnek az olvasókra szabott részleteket kell megjelenítenie, mint például a jelentés dátuma, valamint a tesztvégrehajtási eredmények összefoglalóját, amint azok rendelkezésre állnak.

A tesztfelügyeleti és -irányítási munkatermékeknek foglalkozniuk kell a projektmenedzsment kérdéseivel is, mint például a feladatok teljesítése, az erőforrások hozzárendelése és használata, valamint a ráfordítások.

A tesztfelügyelet és -irányítás kérdéseivel és az ezen tevékenységek során létrehozott munkatermékekkel a jelen tanterv 5.3 fejezetében foglalkozunk majd részletesen.

Tesztelemzés munkatermékek

A tesztelemzés munkatermékek meghatározott és priorizált tesztfeltételeket tartalmaznak, ideális esetben mindegyikükhöz létezik nyomonkövethető kétirányú kapcsolat az általuk lefedett tesztbázis elemhez. Felderítő tesztelés esetén a tesztelemzés magában foglalhatja a tesztvázlatok készítését is. A tesztelemzés a tesztbázis hibáinak felfedezését és jelentését is eredményezheti.

Műszaki teszttervezés munkatermékek

A műszaki teszttervezés a tesztelemzés során létrehozott tesztfeltételeket végrehajtó teszteseteknek és a tesztesetek halmazainak létrehozását eredményezi. Ez gyakran jó gyakorlat magasszintű tesztesetek létrehozása, a bemeneti adatok és az elvárt eredmények konkrét értékeinek meghatározása nélkül. Ezen magasszintű tesztesetek újrahasznosíthatók több tesztciklusban keresztül különböző konkrét adatokat használva, mialatt a teszteset hatóköre még mindig megfelelően dokumentálható. Ideális esetben minden teszteset és az általa lefedett tesztfeltétel(ek) között nyomonkövethető a kétirányú kapcsolat.

A műszaki teszttervezés további eredményei:

- a szükséges tesztadatok tervezése és/vagy azonosítása,
- a tesztkörnyezet tervezése,
- az infrastruktúra és az eszközök azonosítása.

Bár ezen eredmények dokumentálásának mértéke jelentősen eltérő lehet.

A tesztelemzés eredményeként meghatározott tesztfeltételek tovább finomíthatók a műszaki teszttervezés során.

Tesztmegvalósítás munkatermékek

Tesztmegvalósítás munkatermékek lehetnek:

- Teszteljárások, valamint ezen eljárások sorozatai
- Tesztkészletek
- Tesztvégrehajtási ütemterv

Ideális esetben, amint teljes lesz a tesztmegvalósítás, a teszttervben meghatározott lefedettségi feltételek teljesülése demonstrálható a teszteljárások és a tesztbázis meghatározott elemei közötti kétirányú nyomonkövethetőséggel a teszteseteken és a tesztfeltételeken keresztül.

Néhány esetben a teszt megvalósítása magában foglalja a különböző eszközöket használó vagy ezen eszközök által használt munkatermékeket is, például szolgáltatás virtualizáció vagy automatizált tesztszkriptek.

A teszt megvalósítása ezen felül a tesztadatok, illetve a tesztkörnyezet létrehozását és verifikálását is eredményezheti. A tesztadat és/vagy tesztkörnyezet verifikálásának eredményeit tartalmazó dokumentációk teljessége jelentősen eltérő lehet.

A tesztadatok arra szolgálnak, hogy konkrét értékeket rendeljenek a tesztesetek bemeneteihez és az elvárt eredményeikhez. Ezen konkrét értékek a használatukhoz tartozó explicit utasításokkal együtt a magasszintű teszteseteket alacsonyszintű, végrehajtható tesztesetekké alakítják. Ugyanazon magasszintű teszteset más-más tesztadatokat használhat a teszt tárgyának különböző kiadott verzióin való végrehajtáskor. A konkrét tesztadatokhoz tartozó konkrét elvárt eredményeket tesztorákulum alkalmazásával határozzák meg.

Felderítő tesztelés esetében, néhány műszaki teszttervezési- és megvalósítási munkatermék a tesztvégrehajtás során készülhet el, bár a felderítő tesztek dokumentálásának mértéke (illetve a tesztbázis egyes elemeihez tartozó nyomonkövethetőségük) jelentősen különbözhet.

A tesztelemzés során definiált tesztfeltételek tovább finomíthatók a teszt megvalósítása közben.

A tesztvégrehajtás munkatermékei

A tesztvégrehajtás munkatermékei lehetnek:

- Az egyes tesztesetek vagy teszteljárások állapotainak (pl. futásra kész, sikeres, elbukott, blokkolt, szándékosan kihagyott stb.) dokumentációi
- Hibajelentések (lásd: 5.6-os fejezet)
- Dokumentáció arról, hogy melyik tesztelem(ek)et, teszt tárgy(ak)at, teszteszközöket és tesztvereket használták a tesztelés során

Ideális esetben, amint kész a tesztvégrehajtás, a tesztbázis minden elemének státusza meghatározható és arról jelentés készíthető a kapcsolódó teszteljárásokkal való kétirányú nyomonkövethetőségen keresztül. Meg tudjuk mondani például hogy melyik követelményekhez tartozó tervezett tesztek futottak le mind sikeresen, mely követelményekhez tartoznak bukott tesztek és/vagy észlelt hibák, illetve melyik követelményekhez tartoznak olyan tesztek, amelyek még futásra várnak. Ez lehetővé teszi a lefedettségi feltételek teljesülésének verifikálását, illetve a teszteredmények érthető lejelentését az érintettek felé.

A tesztlezárás munkatermékei

A tesztlezárás munkatermékei lehetnek összefoglaló tesztjelentések, a következő projektek vagy iterációk fejlesztéséhez megfogalmazott tevékenységek, változtatáskérés vagy termék-teendőlista elemek, valamint a véglegesített tesztver.

1.4.4 A tesztbázis és a tesztmunkatermékek közötti nyomonkövethetőség

Ahogy az 1.4.3-as fejezetben említettük, a tesztmunkatermékek, illetve ezen munkatermékek nevei jelentősen eltérhetnek egymástól. Ezen különbségektől függetlenül, ahhoz, hogy hatásosan implementáljuk a teszfelügyelet és -irányítás tevékenységeit, fontos, hogy a tesztfolyamat egészén keresztül a tesztbázis összes eleme és az ezekhez kapcsolódó tesztmunkatermékek között létrehozzuk és fenntartsuk a nyomonkövethetőséget, a fent leírtak szerint. A teszt lefedettség kiértékelésén kívül a jó nyomonkövethetőség az alábbiakat támogatja:

- A változások hatásainak elemzése
- A tesztelés ellenőrizhetővé tétele

- Az IT irányelvek megvalósítása
- A tesztstátuszjelentések és az összefoglaló tesztjelentések megérthetőségének javítása, ami magában foglalja a tesztbázis elemeinek státuszát (például követelmények, amelyek tesztjei sikeresek voltak, követelmények, ahol a tesztek elbuktak, és olyan követelmények, amelyek tesztvégrehajtása még függőben van)
- A tesztelés technikai aspektusainak közlése az érintett személyek felé, számukra érthető módon
- Információ biztosítása a termék minőségének, a folyamat lehetőségeinek, és a projekt előrehaladásának az üzleti célokkal összevetett kiértékeléséről

Néhány tesztmenedzsmenteszköz tesztmunkatermék modelleket biztosít, amelyek az ebben a fejezetben körvonalazott munkatermékek egy részére vagy egészére illeszkednek. Néhány szervezet saját menedzsment rendszereket épít ki a munkatermékek szervezéséhez, illetve azért, hogy biztosítsák az információk számukra szükséges mértékű nyomonkövethetőségét.

1.5 A tesztelés pszichológiája

A szoftverfejlesztés, beleértve a szoftvertesztelést, valós személyeket igényel. Az emberi pszichológiának ezért fontos ráhatásai vannak a szoftvertesztelésre.

1.5.1 Pszichológia és tesztelés

A statikus tesztelés, mint például a követelmények felülvizsgálata vagy a felhasználói történet tökéletesítése során talált hibákat, vagy a dinamikus tesztelés alatt azonosított meghibásodásokat a termék és annak szerzője elleni kritikának érezhetjük. A pszichológia egy ismert eleme, a megerősítési torzítás megnehezítheti a jelenlegi meggyőződésnek ellentmondó információk elfogadását. Példaként, mivel a fejlesztők helyesnek vélik a kódjukat, a megerősítési torzítás miatt nehéz elfogadniuk a kód helytelenségét. A megerősítési torzítás mellett más kognitív torzítások is nehezíthetik, hogy az emberek megértsék vagy elfogadják a tesztelésből nyert információkat. Továbbá egy gyakori emberi jellemvonás, hogy a rossz hír hozóját okolják, és a tesztelés által szolgáltatott információ gyakran tartalmaz rossz hírt.

Ezen pszichológiai tényezők eredményeként néhányan a tesztelést destruktív tevékenységként fogják fel annak ellenére, hogy nagyban hozzájárul a projekt előrehaladásához és a termékminőséghez (lásd: 1.1-es és 1.2-es fejezetek). Ahhoz, hogy elkerüljük az ilyen jellegű hozzáállást, a hibákról és meghibásodásokról szóló információkat konstruktív módon kell kommunikálni. A tesztelők és az elemzők, terméktulajdonosok, tervezők és fejlesztők közötti feszültségek ily módon csökkenthetők. Ez a statikus és a dinamikus tesztelésre is vonatkozik.

A hibákról, meghibásodásokról, teszteredményekről, teszt előrehaladásáról és kockázatokról szóló hatékony kommunikáció, és a kollégákkal való pozitív kapcsolatok kiépítése érdekében a tesztelőknek és a tesztmenedzsereknek jó interperszonális képességekkel kell rendelkezniük A jó kommunikáció módjai magukban foglalják a következőket:

- Együttműködéssel kell indítani hadakozás helyett. Fel kell hívni a figyelmet a közös célra: a minél jobb minőségű rendszerre.
- A tesztelés előnyeinek kihangsúlyozása. A szerzők számára például a hibákról szóló információ segíthet a munkatermékeik és készségeik fejlesztésében. A szervezet számára a tesztelés során megtalált és kijavított hibák időt és pénzt spórolnak és csökkentik a termékminőség általános kockázatát.

- Semlegesen, tárgyilagosan kell kommunikálni az eredményeket és az egyéb találatokat, a hibás elem megalkotójának kritizálása nélkül. Objektív és tényszerű hibajelentéseket és felülvizsgálati eredmény jelentéseket kell írni.
- Meg kell próbálni megérteni a másik személy érzéseit és az okokat, amiért negatívan reagálhatnak az információkra.
- Meg kell győződni afelől, hogy a másik személy megértette az elmondottakat és fordítva.

A tipikus tesztcélokat már tárgyaltuk korábban (lást: 1.1-es fejezet). A tesztcélok megfelelő halmaza világos meghatározásának fontos pszichológiai következtetései vannak. A legtöbb ember hajlamos a saját terveiket és viselkedésüket a csapat, menedzsment és más érintett felek által kitűzött célokhoz igazítani. Az is fontos, hogy a tesztelők ezeket a célokat minimális személyes elfogultsággal kövessék.

1.5.2 A tesztelők és fejlesztők gondolkodásmódja

A fejlesztők és a tesztelők gyakran különbözőképpen gondolkodnak. A fejlesztés elsődleges célja egy termék tervezése és felépítése. Ahogy korábban tárgyaltuk, a tesztelés céljai magukban foglalják a termék verifikálását és validációját, a hibák kiadás előtti megtalálását, és így tovább. Ezek különböző célok, melyek különböző gondolkodást igényelnek. Ezen gondolkodásmódok összehangolása segít egy magasabb szintű termékminőség elérésében.

A szemléletmód tükrözi az egyén feltételezéseit és a döntéshozáshoz, problémamegoldáshoz alkalmazott preferált módszereit. Egy tesztelő gondolkodásmódjának magában kell foglalnia a kíváncsiságot, professzionális pesszimizmust, kritikus szemléletet, részletekre való odafigyelést, valamint a jó és pozitív kommunikáció és kapcsolatok iránti motivációt. A tesztelő gondolkodásmódja a tapasztalat szerzésével egyre növekszik és érettebbé válik.

A fejlesztői szemléletmód tartalmazhatja a tesztelői szemléletmód néhány elemét, de a sikeres fejlesztők gyakran jobban érdeklődnek megoldások tervezése és kidolgozása iránt, mint az iránt, hogy az ezen megoldásokban rejlő hibákat keressék. Ezen felül a megerősítési torzítás megnehezíti számukra, hogy felismerjék a saját maguk által elkövetett hibákat.

A megfelelő szemléletmóddal a fejlesztők képesek tesztelni a saját kódjukat. A különböző szoftverfejlesztési életciklusmodellek gyakran eltérő módon szervezik meg a tesztelőket és a teszttevékenységeket. Ha a teszttevékenységek egy részét független tesztelők hajtják végre, akkor nő a hibaészrevétel hatékonysága, ami különösen fontos nagy, komplex vagy biztonságkritikus rendszerek esetén. A független tesztelők egy, a munkatermék szerzőinek (pl. üzleti elemzők, projekttulajdonosok, tervezők és fejlesztők) szemléletétől különböző szemléletet hoznak, mivel a szerzőkétől különböző kognitív torzítással rendelkeznek.

2. Tesztelés a szoftver életciklusán át 100 perc

Kulcsszavak

alfatesztelés, bétatesztelés, elfogadási tesztelés, ellenőrző tesztelés, fehérdoboz tesztelés, felhasználói elfogadási tesztelés, funkcionális tesztelés, hatáselemzés, integrációs tesztelés, karbantartási tesztelés, kereskedelmi dobozos szoftver (COTS), komponens-integrációstesztelés, komponenstesztelés, nemfunkcionális tesztelés, regressziós tesztelés, rendszer integrációs tesztelés, rendszertesztelés, szabály alapú elfogadási tesztelés, szekvenciális fejlesztési modell, szerződéses elfogadási tesztelés, teszt tárgya, tesztbázis, tesztcél, teszteset, tesztkörnyezet, tesztszint, teszttípus, üzemeltetési elfogadási tesztelés

Tanulási célok a Tesztelés a szoftverfejlesztés életciklusán át fejezethez

2.1 Szoftverfejlesztési életciklusmodellek

AK-2.1.1 (K2) Ismertesse a szoftverfejlesztési életciklus során történő szoftverfejlesztési tevékenységek és teszttevékenységek közötti kapcsolatokat

AK-2.1.2 (K1) Tárja fel azon okokat, melyek indokolják, hogy a szoftverfejlesztési életciklusmodelleket a projekt szempontjaihoz és a termékek jellemzőihez kell igazítani

2.2 Tesztszintek

AK-2.2.1 (K2) Hasonlítsa össze a tesztelés különböző szintjeit az alábbi szempontok szerint: mi a tesztelés célja, a tesztbázis, a teszt tárgya, a tipikus hibák és meghibásodások, a tesztstratégia, megközelítések és felelősségek

2.3 Teszttípusok

AK-2.3.1 (K2) Hasonlítsa össze a funkcionális tesztelést, a nemfunkcionális tesztelést és a fehérdoboz tesztelést

AK-2.3.2 (K1) Ismerje fel, hogy minden tesztszinten létezik funkcionális tesztelés, nemfunkcionális tesztelés és fehérdoboz tesztelés

AK-2.3.3 (K2) Hasonlítsa össze az ellenőrző tesztelés és a regressziós tesztelés céljait

2.4 Karbantartási teszt

AK-2.4.1 (K2) Foglalja össze a karbantartási tesztelést szükségessé tevő tényezőket

AK-2.4.2 (K2) Mutassa be a hatáselemzés karbantartási tesztelésben való szerepét

2.1 Szoftverfejlesztési életciklusmodellek

A szoftverfejlesztési életciklusmodell leírja a szoftverfejlesztési projekt összes szakaszában végrehajtott tevékenységtípust és azt, hogy a tevékenységek hogyan kapcsolódnak egymáshoz logikailag és időrendileg. Számos különböző szoftverfejlesztési életciklusmodell létezik, amely mindegyike különböző tesztelési megközelítést igényel.

2.1.1 Szoftverfejlesztés és szoftvertesztelés

Tesztelőként fontos, hogy ismerjük az általános szoftverfejlesztési életciklusmodelleket, hogy a megfelelő teszttevékenységeket hajthassuk végre.

A megfelelő tesztelésnek számos jellemzője van minden szoftverfejlesztési életciklusmodellben:

- Minden fejlesztési tevékenységhez tartozik egy teszttevékenység
- Minden tesztszint rendelkezik az adott szintre jellemző tesztcéllal
- A tesztek elemzése és tervezése az adott tesztszinthez tartozó fejlesztési tevékenység során kezdődik
- A tesztelőket be kell vonni a követelmények és a tervezés kialakításába és finomhangolásába, valamint a dokumentációk felülvizsgálatába is minél előbb, vagyis amint a fejlesztési életciklusban elkészültek az első munkatermékek (pl. követelmények, műszaki tervek, felhasználói történetek).

Függetlenül attól, hogy melyik szoftverfejlesztési életciklusmodellt választjuk, a teszttevékenységeket az életciklus korai szakaszában kell kezdeni, a korai tesztelés elvének betartásával.

Ez a tananyag a következő szoftverfejlesztési életciklus-modelleket kategorizálja:

- Szekvenciális fejlesztési modellek
- Iteratív és inkrementális fejlesztési modellek

A szekvenciális fejlesztési modell a szoftverfejlesztési folyamatot a tevékenységek lineáris, szekvenciális folyamataként írja le. Ez azt jelenti, hogy a fejlesztési folyamat bármely fázisa az előző fázis befejezése után kezdődik. Elméletileg nincsenek átfedések a fázisok között, de a gyakorlatban előnyös, ha a következő fázisból hamar visszajelzést kapunk.

A vízesés modellben a fejlesztési tevékenységek (pl. követelményelemzés, tervezés, kódolás, tesztelés) egymás után fejeződnek be. Ebben a modellben a teszttevékenységek csak akkor kezdődnek el, ha minden fejlesztési tevékenység befejeződött.

A vízesés modelltől eltérően a V-modell integrálja a tesztfolyamatot a fejlesztési folyamattal, megvalósítva a korai tesztelés elvét. Továbbá a V-modell magában foglalja az egyes megfelelő fejlesztési fázisokhoz tartozó tesztszinteket, amelyek tovább támogatják a korai tesztelést (lásd a 2.2. fejezetet a tesztszintekről). Ebben a modellben az egyes tesztszintekhez kapcsolódó tesztek végrehajtása egymás után történik, de bizonyos esetekben átfedés történhet.

A szekvenciális fejlesztési modellek olyan szoftvert biztosítanak, amely tartalmazza ugyan a teljes funkcionalitást, de cserében rendszerint hónapokat, vagy éveket igényel az érintettek és felhasználók számára történő szállítás.

Az inkrementális fejlesztésbe beleértjük a követelmények meghatározását, a rendszer több részletben történő tervezését, felépítését és több részletben történő tesztelését, ami azt jelenti, hogy a szoftver funkcionalitása fokozatosan bővül. A funkciók bővülésének a mértéke változó, néhány módszernél nagyobb

változtatások, néhánynál pedig kisebbek kerülnek kiadásra. Az inkremensek lehetnek egészen aprók is, mint pl. egy új képernyő a felhasználói felületen, vagy új lekérdezési opció.

Iteratív fejlesztésről akkor beszélünk, amikor ciklusokban történik meg - gyakran előre meghatározott idő alatt - a funkciók specifikálása, tervezése, implementálása és tesztelése. Az iterációk magukban foglalhatják a korábbi iterációkban fejlesztett funkciók változásait, valamint a projekt hatókörében történt változásokat is. Mindegyik iteráció olyan működő szoftvert szállít, amely a teljes funkciókészlet egyre növekvő részhalmaza mindaddig, amíg a végső szoftver el nem készül, vagy a fejlesztést le nem állítják.

Példák:

- Rational Unified Process: Minden iteráció viszonylag hosszú (pl. két-három hónap), és a funkcióbővülés ennek megfelelően nagy, például két vagy három kapcsolódó funkciócsoport
- Scrum: Minden iteráció viszonylag rövid (például órák, napok vagy néhány hét), és a funkcióbővülés ennek megfelelően kicsi, például néhány bővítés és/vagy két-három új funkció.
- Kanban: rögzített hosszúságú iterációval vagy anélkül végrehajtva, amely befejezésekor akár egyetlen bővítést, funkciót is kiadhatnak, vagy csoportosíthatják is a funkciókat, hogy azok egyszerre kerüljenek kiadásra
- Spirál: kísérleti inkremensek létrehozása, amelyek egy részét a fejlesztés során gyakran átdolgozzák, de később ezek el is hagyhatók.

Az ilyen módszerekkel fejlesztett komponensek vagy rendszerek gyakran átfedő és ismétlődő tesztszinteket tartalmaznak a fejlesztés során. Ideális esetben minden egyes funkciót több tesztelési szinten is tesztelnek, ahogy a kiadás felé haladunk. Bizonyos esetekben a csapatok folyamatos kiadást vagy folyamatos telepítést alkalmaznak, ezek gyakran több tesztszint jelentős automatizálását teszik szükségessé és a szállítás részét képezik.

Az ilyen módszereket alkalmazó számos fejlesztési erőfeszítés magában foglalja az önszerveződő csapatok fogalmát is, amelyek megváltoztathatják a tesztelési munka szervezésének módját, valamint a tesztelők és a fejlesztők közötti kapcsolatot.

Ezen módszerek olyan növekvő rendszert eredményeznek, melyeket a végfelhasználók számára kiadhatnak funkciónként, iterációnként vagy hagyományosabb módon, nagyobb kiadásokra bontva. Függetlenül attól, hogy a szoftver inkremenseket kiadták-e a végfelhasználóknak, a rendszer bővülésével a regressziós tesztelés fontossága egyre növekszik.

A szekvenciális modellekkel ellentétben az iteratív és inkrementális modelleket alkalmazva heteken vagy akár napokon belül is kiadhatók használható szoftverek, de hónapokig vagy akár évekig is eltarthat mire az összes követelménynek megfelelő, végleges termék kiadása megtörténik.

Szoftvertesztéléssel kapcsolatos további információk Agilis fejlesztés témakörében a következő oktatási anyagban találhatók: ISTQB-CTFL-AT Foundation Level Agile Tester Extension Syllabus, Black 2017, Crispin 2008, and Gregory.

2.1.2 Szoftverfejlesztési életciklusmodell kontextusba helyezve

A szoftverfejlesztési életciklusmodelleket a projekt és a termék jellemzőinek megfelelően kell kiválasztani és kialakítani. A megfelelő szoftverfejlesztési életciklusmodell kiválasztásánál figyelembe kell venni a projekt célját, a fejlesztendő termék típusát, az üzleti prioritásokat (például a piacra kerülés) és az azonosított termék- és projektkockázatokat. Például egy kisebb belső adminisztrációt megvalósító rendszer fejlesztése és tesztelése különbözik egy biztonságkritikus rendszer - például egy autó fékvezérlő rendszerének - fejlesztésétől és tesztelésétől. Egy másik példa, hogy egyes esetekben szervezeti és kulturális problémák gátolhatják a csapattagok közötti kommunikációt, ami akadályozhatja az iteratív fejlesztést.

A projekt jellemzőitől függően szükség lehet a tesztszintek és/vagy a tesztelési tevékenységek kombinálására vagy átszervezésére. Például egy dobozos szoftver (COTS) nagyobb rendszerbe történő integrációja esetén a vevő elvégezheti az együttműködőképességi tesztelést (például integráció az infrastruktúrába és más rendszerekbe) a rendszer integrációs tesztelés szinten, vagy az elfogadási tesztelés szinten is (funkcionális és/vagy nemfunkcionális tesztelés együtt történik az elfogadási és/vagy működési teszteléssel). A tesztszintekről a 2.2 fejezetben, a teszttípusokról a 2.3. fejezetben olvashatunk bővebben.

Magukat a szoftverfejlesztési életciklusmodelleket is kombinálhatjuk. Például V-modell használható a backend rendszer és integrációjának fejlesztésére és tesztelésére, míg Agilis fejlesztési modell alkalmazható a front-end felhasználói interfész, illetve a funkcionalitás (UI) fejlesztésére és tesztelésére.

A prototípus készítése a projekt korai szakaszában jellemző, a kísérleti szakasz befejezése után az inkrementális fejlesztési modell az elfogadott.

A dolgok internete (IoT) rendszerek, amelyekre jellemző, hogy több különböző objektumból épülnek fel, mint például eszközök, termékek és szolgáltatások, általában minden egyes objektumhoz külön szoftverfejlesztési életciklusmodellt alkalmaznak. Ez különös kihívást jelent az IoT rendszerek rendszerverzióinak fejlesztése során. Ezen túlmenően az ilyen objektumok szoftverfejlesztési életciklusa során nagyobb hangsúlyt fektetnek a szoftverfejlesztési életciklus későbbi fázisaira, vagyis miután azokat bevezették az operatív használatba (pl. működtetés, frissítés és szoftverkivezetés fázisai).

Annak okai hogy a szoftverfejlesztési modelleket miért kell adaptálni a projekt és a termék jellemzőihez lehetnek:

- Eltérés a rendszerek termékkockázatai között (komplex vagy egyszerű projekt)
- Több üzleti ágazat is részt vehet egy projektben/programban (a szekvenciális és agilis fejlesztés kombinációja)
- Rövid idő alatt át kell adni a terméket (tesztszintek összevonása és/vagy teszttípusok integrációja a tesztszinteken belül)

2.2 Tesztszintek

A tesztszintek teszttevékenységek olyan csoportjai, amelyeket együtt szerveznek és kezelnek.

Minden tesztszint a tesztfolyamat egy példánya, amely az 1.4. fejezetben leírt tevékenységekből áll és a szoftverfejlesztési folyamat adott szintjén kerül végrehajtásra, az egység vagy komponens fejlesztéstől kezdve a teljes rendszeren át, adott esetben a multirendszerekig. A tesztszintek a szoftverfejlesztési életcikluson belüli egyéb tevékenységekhez kapcsolódnak. Az ebben a tananyagban használt tesztszintek a következők:

- komponenstesztelés
- integrációs tesztelés
- rendszertesztelés
- elfogadási tesztelés

A tesztszinteket a következő tulajdonságok jellemzik:

- konkrét célok
- tesztbázis, melyből a tesztesetek származtathatók
- tesztelés tárgya (azaz, amit tesztelünk)

- tipikus hibák és meghibásodások
- konkrét megközelítések és felelősségek

Minden tesztszinthez megfelelő tesztkörnyezetre van szükség. Az elfogadási tesztelés során például egy éles környezettel egyező tesztkörnyezet az ideális, míg a komponenstesztelés során a fejlesztők jellemzően saját fejlesztői környezetüket használják.

2.2.1 Komponenstesztelés

A komponenstesztelés céljai

A komponenstesztelés (nevezik egység (unit)-, illetve modultesztelésnek is) a külön tesztelhető komponensekre összpontosít. Céljai:

- A kockázat csökkentése
- Annak verifikálása, hogy a komponens funkcionális és nemfunkcionális viselkedése megfelel-e a specifikációknak és a terveknek
- A komponens minősége iránti bizalom megteremtése
- A komponensben levő hibák megtalálása
- Annak megelőzése, hogy a hibák magasabb tesztszintre is bekerüljenek

Bizonyos esetekben, különösen az inkrementális és iteratív fejlesztési modellekben (pl. Agilis modell), ahol a kódváltozások folyamatosak, az automatizált komponens regressziós tesztek kulcsszerepet játszanak abban, hogy megbizonyosodjanak afelől, hogy a változtatások ne rontsák el a meglévő komponenseket.

A komponens tesztelése gyakran a rendszer többi részétől elkülönítve történik a szoftverfejlesztési életciklusmodelltől és a rendszertől függően, ami megkövetelheti mock objektumok, virtualizált szolgáltatások, teszttámogató szoftverkörnyezet, csonkok és meghajtók használatát. A komponenstesztelés kiterjedhet a funkcionalitás tesztelésére (például a számítások helyessége), a nemfunkcionális jellemzőkre (például memóriaszivárgás keresése) és a strukturális tulajdonságokra (például a döntési tesztelés).

Tesztbázis

A komponenstesztelésnél tesztbázisként használható munkatermékek például:

- Részletes tervezés
- Kód
- Adatmodell
- Komponens

Teszt tárgya:

Komponenstesztelésnél a tesztelés tárgyai tipikusan a következő lehetnek:

- Komponensek, egységek vagy modulok
- Kód és adatstruktúrák
- Osztályok
- Adatbázis modulok

Jellemző hibák és meghibásodások

A jellemző hibák és meghibásodások a komponenstesztelés során a következők:

- Nem megfelelő funkcionalitás (pl. nem a műszaki specifikációban leírtak szerinti működés)
- Adatfolyam problémák
- Helytelen kód és logika

A hibákat általában kijavítják, amint megtalálják őket, gyakran informális hibamenedzsmentet használva, ha azonban a fejlesztők hibákat jeleznek, ez fontos információkat szolgáltat a kiváltó okok elemzéshez és a folyamatok javításához.

Konkrét megközelítések és felelősségek

A komponens tesztelését általában a kódot író fejlesztő végzi, de minimum a tesztelt kódhoz való hozzáférés szükséges hozzá. A fejlesztők felváltva is végezhetik a fejlesztési feladataikat, illetve a hibák megtalálását, kijavítását. A fejlesztők gyakran írnak és hajtanak végre teszteket, miután megírták a komponens kódját, azonban különösen az Agilis fejlesztés esetében gyakori, hogy az automatizált komponenstesztesetek írása megelőzheti az alkalmazás kódjának megírását.

Vegyük például a tesztvezérelt fejlesztést (TDD). A tesztvezérelt fejlesztés rendkívül iteratív, az alábbi tevékenységek ciklusain alapul: automatizált tesztesetek fejlesztése, kis kódrészek implementálása és integrálása, a komponenstesztek végrehajtása, hibák javítása és a kód refaktorálása. Ez a folyamat addig ismétlődik, amíg az összes komponens el nem készül, és az összes komponensteszt sikeres nem lesz. A tesztvezérelt fejlesztés példa egy “teszt először” megközelítésre. A tesztvezérelt fejlesztés az eXtrém Programozáson (XP) alapul, de az Agilis fejlesztés egyéb formáiban és a szekvenciális életciklusokban is elterjedten alkalmazzák. (lásd ISTQB-CTFL-AT Foundation Level Agile Tester Extension Syllabus).

2.2.2 Integrációs tesztelés

Az integrációs tesztelés céljai

Az integrációs tesztelés a komponensek vagy rendszerek közötti kölcsönhatásokra összpontosít. Az integrációs tesztelés céljai lehetnek:

- Kockázatok csökkentése
- Annak verifikálása, hogy az interfészek funkcionális és nemfunkcionális viselkedése megfelel a tervekben és specifikációkban foglaltaknak
- Bizalom megteremtése az interfészek minőségével kapcsolatban
- Hibák megtalálása (amelyek magukban az interfészekben vagy a komponenseken, illetve rendszereken belül vannak)
- Megakadályozni, hogy a hibák egy magasabb tesztszintre kerüljenek

Ahogy a komponenstesztelésnél, bizonyos esetekben az automatizált integrációs regressziós tesztek bizonyosságot adnak arra vonatkozóan, hogy a változtatások nem rontották el a meglévő interfészeket, komponenseket vagy rendszereket.

Az integrációs tesztelésnek két különböző szintje kerül bemutatásra a tantervben, amely különböző méretű teszt tárgyakon hajtható végre:

- A komponens integrációs tesztelés az integrált komponensek közötti kölcsönhatásokra és interfészekre összpontosít. A komponens integrációs tesztelést a komponenstesztelés után hajtjuk

végre és általában automatizált. Az iteratív és inkrementális fejlesztésben a komponens integrációs tesztek általában egy folyamatos integráció folyamatának részét képezik.

- A rendszer integrációs tesztelés a rendszerek, csomagok és mikroszolgáltatások közötti kölcsönhatásokra és interfészekre összpontosít. A rendszer integrációs tesztelés kiterjedhet külső szereplők által biztosított szolgáltatásokkal (pl. webszolgáltatások) kapcsolatos kölcsönhatások és interfészek tesztelésére is. Ebben az esetben a fejlesztést végzők nem tartják ellenőrzésük alatt a külső szereplők által szállított interfészeket, melynek során különböző kihívások jelentkezhetnek (például annak biztosítása, hogy a külső szereplő kódjának tesztelést blokkoló hibái javításra kerüljenek, tesztkörnyezetek szervezése stb.). A rendszer integrációs tesztelést végezhetik a rendszertesztelés után vagy párhuzamosan a folyamatban lévő rendszer-teszttevékenységekkel (mind a szekvenciális fejlesztés, mind az iteratív és inkrementális fejlesztés során).

Tesztbázis

Az integrációs tesztelés tesztbázisaként használható munkatermékek lehetnek:

- Szoftver- és rendszerterv
- Szekvencia diagram
- Interfész- és kommunikációs protokoll specifikációk
- Használati esetek
- Komponens- vagy rendszerszintű architektúra
- Munkafolyamatok
- Külső interfész leírások

Teszt tárgya

Integrációs tesztelésnél a teszt tárgya tipikusan a következő lehet:

- Alrendszerek
- Adatbázisok
- Infrastruktúra
- Interfészek
- API-k
- Mikroszolgáltatások

Jellemző hibák és meghibásodások

A jellemző hibák és meghibásodások a komponens integrációs tesztelés során a következők lehetnek:

- Helytelen adatok, hiányzó adatok vagy helytelen adatkódolás
- Az interfészhívások helytelen sorrendezése vagy időzítése
- Interfész hibás illesztése
- A komponensek közötti kommunikáció meghibásodásai
- Nem kezelt vagy helytelenül kezelt kommunikációs meghibásodások a komponensek között

- Helytelen feltevések a komponensek között átadott adatok jelentéséről, mértékegységéről vagy a komponensek között átadott adatok határértékeiről

Tipikus hibák és meghibásodások a rendszer integrációs tesztelés során a következők lehetnek:

- A rendszerek közötti inkonzisztens üzenetstruktúrák
- Helytelen adatok, hiányzó adatok vagy helytelen adatkódolás
- Interfészek hibás illesztése
- Rendszerek közötti kommunikációs meghibásodás
- Nem kezelt vagy helytelenül kezelt kommunikációs meghibásodások a rendszerek között
- Helytelen feltevések a rendszerek között átadott adatok jelentéséről, mértékegységéről vagy a rendszerek között átadott adatok határértékeiről
- A kötelező biztonsági előírások be nem tartása

Konkrét megközelítések és felelősségek

A komponens integrációs teszteknek és a rendszer integrációs teszteknek magukra az integrációra kell összpontosítaniuk. Például ha az A modult a B modullal integráljuk, akkor a modulok közötti kommunikációra kell összpontosítani, nem pedig az egyes modulok funkcionalitására, mivel ezt a komponenstesztelés során már le kellett tesztelni. Ha az X rendszert az Y rendszerrel integráljuk, a teszteknek a rendszerek közötti kommunikációra kell összpontosítaniuk, nem pedig az egyes rendszerek funkcionalitására, mivel ezt a rendszertesztelés során már le kellet volna fedni. Funkcionális, nemfunkcionális és strukturális teszttípusok alkalmazhatók.

A komponens integrációs tesztelés leggyakrabban a fejlesztők felelőssége. A rendszer integrációs tesztelés általában a tesztelők felelőssége. Ideális esetben, a rendszer integrációs teszteket végző tesztelőknek érteniük kell a rendszer architektúráját és részt kellett venniük az integráció tervezésben.

Amennyiben az integrációs teszteket és az integrációs stratégiát a komponensek vagy rendszerek fejlesztése előtt tervezik, akkor ezeket a komponenseket vagy rendszereket teszteléshez szükséges leghatékonyabb sorrendben lehet kialakítani. A szisztematikus integrációs stratégiák alapulhatnak a rendszer arhitektúrális felépítésén (pl. felülről lefelé és alulról felfelé), funkcionális feladatokon, tranzakciós feldolgozási szekvenciákon, illetve a rendszer, vagy komponensek bármely más aspektusán.

A hiba-elkülönítés egyszerűsítése és a hibák korai felismerése érdekében az integrációnak általában inkrementálisnak kell lennie (tehát egyidejűleg kis számú hozzáadott komponens vagy rendszer), nem pedig „big bang” (vagyis az összes komponens vagy rendszer egyetlen lépésben történő integrálása).

A legösszetettebb interfészek kockázatelemzése segíthet az integrációs tesztelés fókuszának meghatározásában.

Minél nagyobb az integráció hatóköre, annál nehezebb lesz elkülöníteni a hibákat egy adott komponensre vagy rendszerre, ami fokozott kockázatot jelent és további időt vesz igénybe a hibaelhárításnál. Ez az egyik oka annak, hogy a folyamatos integráció, ahol a szoftvert komponensről komponensre integrálják (funkcionális integráció), általános gyakorlat lett. Az ilyen folyamatos integráció gyakran magában foglalja az automatizált regressziós tesztelést, ideális esetben több tesztszinten végrehajtva.

2.2.3 Rendszertesztelés

A rendszertesztelés céljai

A rendszertesztelés a teljes rendszer vagy termék viselkedésére és képességeire összpontosít, gyakran figyelembe véve a végpontok közötti feladatokat, amelyeket a rendszer képes végrehajtani, és a nemfunkcionális viselkedéseket, amelyeket ezen feladatok végrehajtása közben mutat.

A rendszertesztelés céljai a következők lehetnek:

- Kockázatcsökkentés
- Annak verifikálása, hogy az interfészek funkcionális és nemfunkcionális viselkedése megfelel a tervekben és specifikációkban foglaltaknak
- Annak validálása, hogy a rendszer teljes és az elvártaknak megfelelően működik
- A rendszer egészének minősége iránti bizalom megteremtése
- A hibák megtalálása
- Megakadályozni, hogy a hibák magasabb tesztszintekre vagy a termékbe kerüljenek

Bizonyos rendszerek esetében cél lehet az adatminőség verifikálása is. Ahogy a komponens- és az integrációs tesztelésnél, bizonyos esetekben az automatizált rendszer regressziós tesztek is bizonyosságot adnak arra vonatkozóan, hogy a változtatások nem rontották el a meglévő funkciókat vagy a végpontok közötti működést.

A rendszertesztelés gyakran szolgáltat olyan információkat az érintettek számára, amelyek alapján a kiadásokkal kapcsolatos döntéseiket meg tudják hozni. Szükséges lehet az is, hogy a rendszer megfeleljen jogszabályi követelményeknek vagy szabványoknak is.

A tesztkörnyezet ideális esetben megfelel a végső cél- vagy produktív (éles) környezetnek.

Tesztbázis

Példák a rendszertesztelés tesztbázisaként szolgáló munkatermékekre:

- Rendszer- és szoftverkövetelményspecifikáció (funkcionális és nemfunkcionális)
- Kockázatelemzés-jelentés
- Használati esetek
- Epic-ek és felhasználói történetek
- Rendszerviselkedési modellek
- Állapotdiagramok
- Rendszer- és felhasználói kézikönyvek

Tesztelés tárgya

Rendszertesztelésnél a tesztelés tárgya jellemzően a következő lehet:

- Alkalmazások
- Hardver/szoftver rendszerek

- Operációs rendszer
- Tesztelt rendszer (SUT)
- Rendszerkonfiguráció és konfigurációs adatok

Jellemző hibák és meghibásodások

A jellemző hibák és meghibásodások a rendszertesztelés során a következők lehetnek:

- Helytelen számítások
- A rendszer helytelen vagy váratlan funkcionális- vagy nemfunkcionális viselkedése
- Helytelen vezérlés és/vagy adatfolyam a rendszeren belül
- A rendszer nem képes megfelelően végrehajtani a végponttól végpontig terjedő funkcionális feladatokat
- Meghibásodás az adott rendszerkörnyezet(ek)ben
- A rendszer nem a rendszer- és felhasználói kézikönyvekben leírtak szerint működik

Jellemző megközelítések és felelősségek

A rendszertesztelésnek a rendszer teljes, végponttól végpontig terjedő viselkedésére kell összpontosítania, mind funkcionális, mind nemfunkcionális tekintetben. A rendszertesztelés során a vizsgált rendszer szempontjából legmegfelelőbb technikákat (lásd a 4. fejezetet) kell használni.

Például döntési tábla hozható létre annak ellenőrzésére, hogy a funkcionális viselkedés az üzleti szabályokban leírtak szerint történik.

A rendszertesztelést legtöbbször független tesztelők végzik, akik főként a specifikációkra támaszkodnak. A specifikációk hibái (például a hiányzó felhasználói történetek, a helytelenül megadott üzleti követelmények stb.) a várt rendszer viselkedésének megértését akadályozhatják vagy téves elvárásokat támaszthatnak vele szemben.

Ezek hamis pozitív és hamis negatív helyzeteket okozhatnak, amelyekkel időt pazarolunk és csökkentik a hibakeresési hatékonyságot. A tesztelők korai bevonása a felhasználói történet finomításába vagy a statikus tesztelésbe, mint például a felülvizsgálat, segít csökkenteni az ilyen helyzetek előfordulását.

2.2.4 Elfogadási tesztelés

Az elfogadási tesztelés céljai

Az elfogadási tesztelés, akárcsak a rendszertesztelés, általában egy egész rendszer vagy termék viselkedésére és képességeire összpontosít.

Az elfogadási tesztelés céljai a következők lehetnek:

- 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

Az elfogadási tesztek információt adnak a rendszer készültségi fokáról, hogy mikor lesz kiadható és használatba vehető. Hibákat találhatunk az elfogadási tesztelés során is, de itt többnyire már nem a hibák megtalálása a cél, valamint az elfogadási tesztelés során talált jelentős számú hiba bizonyos esetekben

jelentős projektkockázatnak tekinthető. Szükséges lehet az is, hogy az elfogadási tesztelés megfeleljen jogszabályi követelményeknek vagy szabványoknak is.

Az elfogadási tesztelés általános formái a következők:

- felhasználói elfogadási tesztelés
- üzemeltetési elfogadási tesztelés
- szerződéses és szabály alapú elfogadási tesztelés
- alfa és bétatesztelés

Mindegyiket a következő négy alfejezet írja le.

Felhasználói elfogadási tesztelés (UAT)

A felhasználók által a rendszeren végzett felhasználói elfogadási tesztek általában arra irányulnak, hogy valós vagy szimulált éles működési környezetben validálják azt, hogy a rendszer a leendő felhasználók számára alkalmas lesz-e a használatra.

A fő cél a bizalom megteremtése aziránt, hogy a felhasználók a rendszert az igényeiknek és a követelményeknek megfelelően tudják majd használni és az üzleti folyamatokat minimális nehézséggel, költséggel és kockázattal tudják végrehajtani.

Üzemeltetési elfogadási tesztelés (OAT)

A rendszeren - az üzemeltetés vagy rendszergazda által - végrehajtott elfogadási tesztelés rendszerint szimulált éles környezeten történik.

A tesztek üzemeltetési szempontokra összpontosítanak, és magukban foglalhatják:

- A biztonsági mentés és visszaállítás tesztelését
- Telepítést, eltávolítást és frissítést
- Katasztrófa utáni helyreállítást
- Felhasználókezelést
- Karbantartási feladatokat
- Adatbetöltési és migrációs feladatokat
- Sebezhetőségek ellenőrzését
- Teljesítménytesztelést

Az OAT fő célja a bizalom megteremtése aziránt, hogy az üzemeltetők vagy a rendszergazdák a rendszer működését a működési környezetben is biztosítani tudják a felhasználók számára, még kivételes vagy nehéz körülmények között is.

Szerződéses és szabály alapú elfogadási tesztelés

A szerződéses elfogadási tesztelés a szerződés elfogadási feltételek ellenében történik az egyedi fejlesztésű szoftverek előállítása során. Az elfogadási kritériumokat akkor kell meghatározni, amikor a felek elfogadják a szerződést. A szerződéses elfogadási teszteket gyakran a felhasználók vagy független tesztelők végzik.

Szabály alapú elfogadási tesztelést végzünk minden olyan előírás ellenében, amelyet be kell tartani, mint például a kormányzati, jogi vagy biztonsági előírások. A szabály alapú elfogadási tesztelést gyakran a

felhasználók vagy a független tesztelők végzik, néha a szabályalkotó végzi az eredmények tanúsítását vagy auditját.

A szerződéses és szabály alapú elfogadási tesztelés fő célja a bizalom megteremtése azzal kapcsolatban, hogy a rendszer a szerződéses vagy szabályozási előírásoknak megfelel.

Alfa és bétatesztelés

Az alfa- és bétatesztelést tipikusan a kereskedelmi dobozos szoftverek (COTS) fejlesztői használják, akik visszajelzést szeretnének kapni a potenciális vagy meglévő felhasználóktól, ügyfelektől és/vagy üzemeltetőktől, mielőtt a szoftverterméket forgalomba hozzák. Az alfatesztelés a fejlesztést végző szervezetnél történik, de a tesztelést nem a fejlesztők, hanem a potenciális vagy meglévő ügyfelek, és/vagy üzemeltetők vagy egy független tesztcsapat végzi. A béta tesztelést a potenciális vagy a meglévő ügyfelek és/vagy az üzemeltetők a saját telephelyükön végzik el. A bétatesztelés az alfatesztelés után vagy előzetes alfatesztelés nélkül is történhet.

Az alfa- és bétatesztelés egyik célja a potenciális vagy meglévő ügyfelek és/vagy az üzemeltetők bizalmának megteremtése aziránt, hogy normál, mindennapi körülmények között és éles működési környezeten a lehető legkisebb nehézséggel, költséggel és kockázattal működik majd a rendszer.

Egy másik cél lehet olyan hibák feltárása, amelyek a rendszer használatának körülményeivel és környezetével kapcsolatosak, különösen akkor, ha ezeket a feltételeket és környezetet a fejlesztői csapatnak nehéz reprodukálni.

Tesztbázis

Példák munkatermékekre, melyek bármely elfogadási tesztelés tesztbázisaként használhatók:

- Üzleti folyamatok
- Felhasználói vagy üzleti követelmények
- Szabályozások, jogi szerződések és szabványok
- Használati esetek és/vagy felhasználói történet teszt
- Rendszerkövetelmények
- Rendszer- vagy felhasználói dokumentáció
- Telepítési eljárások
- Kockázatelemzési jelentések

Ezen felül tesztbázisként az OAT-hoz szükséges tesztesetek elkészítéséhez, az alábbi munkatermékek közül egy vagy több is használható:

- Biztonsági mentési és visszaállítási eljárások
- Katasztrófa helyreállítási eljárások
- Nemfunkcionális követelmények
- Üzemeltetési dokumentáció
- Telepítési utasítások
- Teljesítménnyel kapcsolatos célkitűzések
- Adatbázis csomagok

- Biztonsági szabványok vagy szabályozások

Tesztelés tárgya

Az összes elfogadási tesztelés esetén a tesztelés tárgya lehet az alábbi:

- Tesztelt rendszer (SUT)
- Rendszerkonfiguráció és konfigurációs adatok
- A teljesen integrált rendszer üzleti folyamatai
- Helyreállítási rendszerek és hot site-ok (az üzletfolytonossági- és katasztrófa utáni helyreállítási tesztekhez)
- Üzemeltetési és karbantartási folyamatok
- Űrlapok
- Jelentések
- Létező és átalakított, az éles rendszerből származó adat

Jellemző hibák és meghibásodások

Az elfogadási tesztelés minden formája esetén tipikus hibák lehetnek:

- A rendszer munkafolyamatai nem felelnek meg az üzleti- vagy felhasználói követelményeknek
- Az üzleti szabályokat nem megfelelően valósították meg
- A rendszer nem felel meg a szerződéses vagy szabály alapú követelményeknek
- Nemfunkcionális meghibásodások, mint például biztonsági sebezhetőség, nem megfelelő teljesítményhatékonyság a nagy terhelések esetén, vagy nem megfelelő működés egy támogatott platformon

Jellemző megközelítések és felelősségek

Az elfogadási tesztelés gyakran a vevők, az üzleti felhasználók, a terméktulajdonosok vagy a rendszer üzemeltetőinek a felelőssége, és más érintettek is részt vehetnek benne. Az elfogadási tesztelést gyakran úgy tekintik, mint az utolsó tesztszintet a szekvenciális fejlesztési életciklusban, de máskor is előfordulhatnak, például:

- Kereskedelmi dobozos szoftvernél (COTS) amikor telepítik vagy integrálják
- Elfogadási tesztelés az új funkcionális bővítményekre még a rendszertesztelés előtt

Az iteratív fejlesztéseknél a projektcsapat minden egyes iteráció alatt és minden egyes iteráció végén is alkalmazhat különböző elfogadási tesztelést, például olyanokat, amelyek az új feature-ök verifikációjára irányulnak szemben az elfogadási kritériumokkal, amelyek validációja során arra fókuszálnak, hogy ezek az új feature-ök megfelelnek-e a felhasználók igényeinek. Emellett alfatesztek és bétatesztek is végrehajthatók bármelyik iteráció végén, az egyes iterációk befejezése után, vagy ismétlődő iterációk után is. A felhasználói elfogadási tesztek, a működési elfogadási tesztek, a szerződéses és szabály alapú elfogadási tesztek is előfordulhatnak az egyes iterációk végén, az egyes iterációk befejezése után, vagy ismétlődő iterációkat követően.

2.3 Teszttípusok

A teszttípus a teszttevékenységek egy csoportja amelynek célja a szoftverrendszer vagy a rendszer bizonyos részeinek meghatározott tesztcélok alapján történő tesztelése. Ilyen célok lehetnek:

- A funkcionális minőségi jellemzők kiértékelése, mint például a teljesség, a helyesség és a megfelelőség
- A nemfunkcionális minőségi jellemzők értékelése, mint például a megbízhatóság, a teljesítmény-hatékonyság, a biztonság, a kompatibilitás és a használhatóság
- Annak értékelése, hogy a komponens vagy a rendszer struktúrája, illetve architektúrája helyes, teljes és a specifikációnak megfelel
- A változások hatásainak értékelése, például annak megerősítése, hogy a hibákat javították (ellenőrző tesztelés), illetve a szoftver vagy a környezet változásaiból eredő nem kívánt hatások keresése (regressziós tesztelés)

2.3.1 Funkcionális tesztelés

A rendszer funkcionális tesztelése olyan teszteket foglal magában, melyek kiértékelik a rendszer által végzendő funkciókat. A funkcionális követelmények leírhatók a munkatermékekben, például az üzleti követelmények specifikációiban, az epic-ekben, a felhasználói történetekben, a használati esetekben vagy a funkcionális specifikációkban, vagy akár dokumentálatlanok is lehetnek. A funkció az, amit a rendszernek tennie kell.

A funkcionális teszteket minden tesztszinten el kell végezni (pl. az összetevőkre vonatkozó tesztek, melyek a komponens specifikáción alapulnak), bár a fókusz minden tesztszinten máson van (lásd a 2.2. fejezetet).

A funkcionális tesztelés a szoftver viselkedését veszi figyelembe, így a feketedoboz technikák felhasználhatók a komponens vagy rendszer funkcionalitására vonatkozó tesztfeltételek és tesztesetek származtatására (lásd 4.2 fejezet).

A funkcionális tesztelés alapossága a funkcionális lefedettséggel mérhető. A funkcionális lefedettség azt mutatja meg, hogy a funkcionalitást milyen mértékben teszteljük le, mindez az érintett elem típusának százalékos arányában kifejezve. Például a tesztek és a funkcionális követelmények közötti nyomonkövethetőséget alkalmazva kiszámítható a tesztelt követelmények aránya, ez potenciálisan azonosítja a lefedettségi réseket.

A funkcionális teszttervezés és -végrehajtás speciális készségeket vagy ismereteket igényelhet, mint például az adott üzleti problémát kezelő szoftveres megoldás ismerete (pl. az olaj- és gázipar geológiai modellezési szoftvere).

2.3.2 Nemfunkcionális tesztelés

A rendszer nemfunkcionális tesztelése során a rendszerek és szoftverek jellemzőit vizsgáljuk, például a használhatóságot, a teljesítmény-hatékonyságot vagy a biztonságot. A szoftver termékjellemzőinek osztályozását az ISO szabvány (ISO/IEC 25010) tartalmazza. A nemfunkcionális tesztelés során azt vizsgáljuk, hogy miként viselkedik a rendszer.

A szokásos tévhitekkel ellentétben a nemfunkcionális tesztelés minden tesztszinten elvégezhető és gyakran meg is kell valósítani és a lehető leghamarabb végre kell hajtani. A nemfunkcionális hibák késői felfedezése rendkívül veszélyes lehet a projekt sikerére nézve.

A feketedoboz technikákat (lásd a 4.2. fejezetet) használhatjuk a tesztfeltételek és a nemfunkcionális tesztelés teszteseteinek meghatározásához. Például a határérték-elemzés segítségével meghatározhatók a teljesítménytesztek stresszfeltételei.

A nemfunkcionális tesztek alapossága a nemfunkcionális lefedettséggel mérhető. Azt mutatja meg, hogy a nemfunkcionális elemek bizonyos típusait milyen mértékben teszteljük le, mindezt az érintett elem típusának százalékában kifejezve. Például a tesztek és a mobil alkalmazások támogatott eszközei közötti nyomonkövethetőség segítségével kiszámítható a kompatibilitási tesztek által kezelt eszközök százalékos aránya, amely potenciálisan azonosítja a lefedettségi réseket.

A nemfunkcionális teszttervezés és -végrehajtás speciális készségeket vagy ismereteket igényelhet, mint például a tervezés vagy a technológia hiányosságainak ismerete (pl. bizonyos programozási nyelvekkel kapcsolatos biztonsági sebezhetőségek) vagy az adott felhasználói bázis ismerete (pl. egészségügyi létesítmények irányítási rendszerei).

A nemfunkcionális tulajdonságok tesztelésével kapcsolatos további részletek az ISTQB-CTAL-TA Advanced Level Test Analyst Syllabus, ISTQB-CTAL-TTA Advanced Level Technical Test Analyst Syllabus, ISTQB-CTAL-SEC Advanced Level Security Tester Syllabus és egyéb ISTQB® modulok foglalkoznak.

2.3.3 Fehérdoboz tesztelés

A fehérdoboz tesztelés a rendszer belső szerkezete vagy az implementáció alapján származtatja a teszteseteket. A belső szerkezet tartalmazhat kódot, architektúrát, munkafolyamatokat és/vagy adatfolyamokat a rendszeren belül (lásd 4.3 fejezet).

A fehérdoboz tesztelés alaposságát struktúra szintű lefedettséggel lehet mérni. A struktúra szintű lefedettség azt jelenti, hogy milyen mértékben hajtották végre a teszteket valamelyik strukturális elemen, ezt a lefedett elem típusának százalékában fejezzük ki.

A komponensteszt szintjén a kódlefedettség a tesztelt komponens kód százalékos arányán alapul és mérhető a kód (lefedett elemek) különböző szempontjai szerint, például a komponensben tesztelt végrehajtható utasítások százalékos arányával, vagy a tesztelt döntési eredmények százalékos arányával. Az ilyen lefedettséget együttesen kódlefedettségnek nevezzük. A komponens integrációs tesztszinten a fehérdoboz tesztelés alapulhat a rendszer architektúráján, például a komponensek közötti interfészeken, a strukturális lefedettséget a tesztelt interfészek százalékos arányában lehet mérni.

A fehérdoboz tesztelés teszttervezése és -végrehajtása olyan speciális készségeket vagy tudást is igényelhet, mint például a kódolás ismerete, az adatok tárolásának módja (pl. a lehetséges adatbázis lekérdezések kiértékelése) és annak ismerete, hogy hogyan kell használni ezeket az eszközöket és az általuk adott eredményeket miként kell értelmezni.

2.3.4 Változásokhoz kapcsolódó tesztelés

Ha egy rendszeren változtatásokat eszközölnek, pl. hibajavítás miatt, vagy hogy új funkciókat vezessenek be, esetleg a meglévőket módosítják, le kell tesztelni azt, hogy a változtatások valóban javították-e a hibát, vagy helyesen hajtották-e végre a funkció implementálását, és a változtatások nem jártak-e előre nem látható következményekkel.

- Ellenőrző tesztelés: A hiba javítása után a szoftvert tesztelhetjük az új szoftververzión az összes olyan meglévő tesztesetet végrehajtva, amely a hiba miatt korábban sikertelen volt. A szoftver új tesztekkel is bővíthető. A meghibásodást előidéző lépéseket a hiba reprodukálásához ismét le kell futtatni az új szoftververzióban. Az ellenőrző tesztelés célja annak megerősítése, hogy az eredeti hiba sikeresen javítva lett.

- Regressziós tesztelés: Lehetséges, hogy a kód egyik részében végrehajtott változtatás, függetlenül attól, hogy javítás vagy más típusú változtatás történt, véletlenül befolyásolhatja a kód más részeinek viselkedését, mindegy, hogy a változtatás ugyanazon komponensen belül van-e, vagy a rendszer többi komponensében vagy akár egy másik rendszerben. A változások magukban foglalhatják a környezet változásait, például az operációs rendszer vagy az adatbázis-kezelő rendszer új verzióját.

Az ilyen nem kívánt hatásokat nevezik regressziónak. A regressziós tesztelés magában foglalja a nem szándékolt hatások észlelésére szolgáló tesztek végrehajtását.

Az ellenőrző és a regressziós teszteket minden tesztszintnél végrehajtjuk.

Különösen az iteratív és inkrementális fejlesztési életciklusokban (pl. Agilis), az új funkciók, a meglévő funkciók változásai és a kódrefaktorálás gyakori kódváltozásokat eredményez, ami szintén változásokhoz kapcsolódó tesztelést igényel. A rendszer fejlődő jellege miatt az ellenőrző- és a regressziós tesztelés nagyon fontos. Ez különösen fontos a dolgok internete rendszerekben, ahol az egyes eszközöket gyakran frissítik vagy kicserélik.

A regressziós tesztkészleteket többször futtatják, és általában lassan fejlődnek, így a regressziós tesztek jól automatizálhatók. Ezeknek a teszteknek az automatizálása a projekt elején kezdődik (lásd a 6. fejezetet).

2.3.5 Teszttípusok és tesztszintek

Lehetőség van bármely teszttípus bármely tesztszinten történő elvégzésére. A szemléltetés érdekében a funkcionális, nemfunkcionális, fehérdoboz és a változásokhoz kapcsolódó tesztelésre mutatunk példákat minden tesztszintre, egy banki alkalmazás esetében, a funkcionális tesztekkel kezdve:

- A komponens teszteléséhez a tesztek tervezése azon alapszik, hogy egy komponensnek hogyan kell számítania a kamatos kamatot.
- A komponens integrációs tesztelés esetén a teszteket úgy tervezik meg, hogy a felhasználói felületen rögzített számlaadatok átadásra kerüljenek az üzleti logikának.
- A rendszertesztelés hez a teszteket úgy tervezik meg, hogy a számlatulajdonosok kérhessenek hitelkeretet a folyószámlájukon.
- A rendszer integrációs teszteléséhez a teszteket úgy tervezik meg, hogy a rendszer egy külső mikroszolgáltatást használjon a számlatulajdonos hitelminősítésének ellenőrzésére.
- Az elfogadási teszteléshez a tesztek tervezése azon alapszik, hogy a bankár hogyan kezeli egy hitelkérelem jóváhagyását vagy elutasítását.

Példák a nemfunkcionális tesztekre:

- Komponenstesztelés esetén a teljesítményteszteket egy komplex kamatos kamat számításhoz szükséges utasításciklusok számának kiértékelésére tervezik.
- Komponens integrációs tesztelés esetén a biztonsági tesztek a puffertúlcsordulási sebezhetőséget célozzák meg, mivel itt a felhasználói felület az üzleti logikának továbbít adatokat.
- A rendszertesztelésnél a hordozhatósági vizsgálatok célja annak ellenőrzése, hogy a prezentációs réteg minden támogatott böngészőn és mobil eszközön működik-e.
- A rendszer integrációs tesztek során a megbízhatósági teszteket úgy tervezzük, hogy értékeljék a rendszer robusztusságát, ha a hitelbírálati mikroszolgáltatás nem válaszol.
- Az elfogadási tesztelés esetén a használhatósági tesztelés célja, hogy értékelje a bankár hitelkezelési felületének hozzáférhetőségét a fogyatékkal élők számára.

Példák fehérdoboz tesztekre:

- Komponenstesztelés esetén a teszteket úgy tervezik, hogy teljes utasítás- és döntési lefedettséget érjenek el minden olyan komponens esetében, amely pénzügyi számításokat végez.

- Komponens integrációs tesztelésnél azt teszteljük, hogy a böngésző interfésze minden képernyőn átadja-e az adatokat a böngésző következő képernyőjének és az üzleti logikának.
- A rendszerteszteléshez a teszteket úgy tervezzük, hogy azok lefedjék azon weblapokat, amelyek a hitelkeret alkalmazás során megjelenhetnek.
- A rendszer integrációs tesztek során a teszteket úgy tervezzük meg, hogy a hitelbírálati mikroszolgáltatásra küldött összes lehetséges lekérdezési módot végrehajtsa.
- Az elfogadási teszteléshez a teszteket úgy tervezzük, hogy lefedjék az összes támogatott pénzügyi adatfájl struktúrát és a banki átutalások értéktartományait.

Végül példák a változásokhoz kapcsolódó tesztekre:

- A komponenstesztelés hez minden egyes komponensre automatizált regressziós tesztet készítünk és a folyamatos integráció keretrendszerébe illesztjük.
- Komponens integrációs tesztek esetén a teszteket az interfészhez kapcsolódó hibák javításának megerősítéséhez tervezzük, mivel a hibajavításokat a kódrepositoryban ellenőrzik.
- A rendszerteszteléshez az adott munkafolyamat minden tesztjét újra végrehajtják, ha a munkafolyamat bármely képernyője megváltozik.
- A rendszer integrációs tesztelés során a hitelbírálati mikroszolgáltatással kölcsönhatásban álló alkalmazás tesztjeit naponta újra végrehajtják a mikroszolgáltatás folyamatos telepítésének részeként.
- Elfogadási tesztelés esetén az összes korábban sikertelen tesztet újra végrehajtják, miután az elfogadási tesztben talált hiba javítása megtörtént.

Bár ez a fejezet minden egyes teszttípusra tartalmaz példát minden tesztszintre vonatkozóan, nem szükséges, hogy minden szoftver esetében minden teszttípust alkalmazzunk minden szinten. Fontos azonban, hogy minden megfelelő teszttípus futtásra kerüljön az egyes tesztszinteken, különös tekintettel az első olyan szintre, ahol az adott teszttípus végrehajtható.

2.4 Karbantartási tesztelés

Az éles környezetekbe való bevezetést követően a szoftvert és rendszereit karban kell tartani. A különböző típusú változtatások szinte elkerülhetetlenek a leszállított szoftverben és rendszereiben, például az üzemeltetés során felfedezett hibák javítása, új funkciók hozzáadása vagy a már leszállított funkcionalitás törlése, módosítása. A karbantartás feladata továbbá hogy megőrizze vagy javítsa a komponens vagy rendszer élettartalma során annak nemfunkcionális minőségi jellemzőit, különösen a teljesítmény hatékonyság, kompatibilitás, megbízhatóság, biztonság, és hordozhatóság tekintetében.

Amikor a karbantartás részeként bármilyen változtatás történik, akkor karbantartási tesztelést kell végezni annak érdekében, hogy kiértékeljék az eszközölt változtatások sikerességét, valamint, hogy ellenőrizzék a rendszer változatlanul maradt részeiben bekövetkező esetleges mellékhatásokat (pl. regresszió) (ami általában a rendszer legnagyobb része). A karbantartás a tervezett és nem tervezett kiadásokat is magában foglalhatja.

Egy karbantartás kiadás esetén a karbantartási tesztre több tesztszinten és különböző teszttípusokat felhasználva is szükség lehet a hatókör alapján. A karbantartási teszt hatóköre függ:

- A változtatás kockázatának mértékétől, például annak mértéke, hogy a szoftver megváltoztatott része milyen mértékben kommunikál más komponenssel vagy rendszerrel

- A meglévő rendszer méretétől
- A változtatás méretétől

2.4.1 Karbantartást kiváltó okok

Számos oka, hogy lehet miért történik szoftver karbantartás és így karbantartási teszt lehetséges mind tervezett és nem tervezett változások esetében.

A következőképpen csoportosíthatjuk a karbantartást kiváltó okokat:

- Módosítások, mint a betervezett bővítések (például: kiadás alapon), javító és szükség változtatások, éles környezetváltozások (mint például a betervezett operációs rendszer vagy adatbázis frissítések), a kereskedelmi dobozos szoftverek (COTS) frissítése, hibák javítása és sebezhetőségek
- Migráció, például egyik platformról a másikra váltás, ami mind az új környezeten, mind a megváltoztatott szoftveren üzemeltetési teszteket tesz szükségessé, vagy az adat konverziók tesztelése, amikor az adat másik alkalmazásból átmigrálásra kerül a karbantartott rendszerbe
    - Visszavonultatás, mint például mikor egy alkalmazás eléri a használatának végét. Amikor egy alkalmazás vagy rendszer visszavonultatásra kerül, akkor szükség lehet adat migrációs tesztre vagy archiválásra hosszú adatmegőrzési idő esetén.
    - Visszaállási / visszakeresési eljárások tesztelése lehet szükség hosszabb megőrzési időszakok archiválása esetén.
    - Regressziós tesztelésre lehet szükség a használatban maradt funkciók működésének biztosítása érdekében.

Amikor egy alkalmazás vagy rendszer visszavonultatásra kerül, akkor szükség lehet adat migrációs tesztre vagy archiválásra hosszú adatmegőrzési idő esetén. Szintén szükség lehet a visszaállási / visszakeresési eljárások tesztelésére hosszabb megőrzési időszakok archiválása esetén. Továbbá regressziós tesztelésre lehet szükség a használatban maradt funkciók működésének biztosítása érdekében.

A dolgok internete rendszerek esetében a karbantartás tesztelést a teljesen új vagy módosított dolgok, a rendszer egészébe történő bevezetése válthatja ki, mint például a hardver eszközök és szoftver szolgáltatások. A karbantartási tesztelés ezen rendszerek esetében külön hangsúlyt fektet a különböző szintek integrációs tesztelésére (például: hálózati szint, alkalmazás szint) és a biztonsági aspektusokra, különösen azokra, amelyek a személyes adatokhoz kötődnek.

2.4.2 Karbantartás hatáselemzés

A hatáselemzés kiértékeli azon változtatásokat, amelyek a karbantartás kiadás során mentek végbe annak érdekében, hogy azonosítsa a várható következményeit, valamint a változás várható és leetséges mellékhatásait, továbbá valamintm hogy azonosítsa azon részeit a rendszernek, amelyek érintettek a változás által. A hatáselemzés szintén segít azonosítani a változtatás kihatását a meglévő tesztekre. A regressziós kockázat miatt a változtatás mellékhatásait és érintett rendszer területeit tesztelni kell a regressziók miatt, várhatóan azt követően, hogy frissítésre kerültek a változtatás által érintett tesztek.

A változtatás bekövetkezte előtt is végre lehet hajtani hatáselemzést, hogy a rendszer más területein bekövetkező lehetséges következmények alapján segítséget nyújtson annak eldöntésében, hogy szükséges-e a változtatás.

A hatáselemzés nehéz lehet, ha:

- A specifikációk (üzleti követelmények, felhasználói történetek, architektúra specifikációk) elavultak vagy hiányosak
- A tesztesetek nem dokumentáltak vagy elavultak

- A tesztek és a tesztbázis közötti kétirányú visszakövethetőség nincs karbantartva
- Az eszköztámogatás gyenge vagy nem létezik
- A bevont embereknek nincs domén és/vagy rendszer tudása
- Nem volt elégséges figyelem fordítva a szoftver karbantarthatóságra a fejlesztés alatt

3. Statikus tesztelés 135 perc

Kulcsszavak

ad hoc felülvizsgálat, átvizsgálás, dinamikus tesztelés, ellenőrző lista alapú felülvizsgálat, felülvizsgálat, forgatókönyv alapú felülvizsgálat, formális felülvizsgálat, informális felülvizsgálat, inspekció, statikus elemzés, perspektíva alapú olvasat, statikus tesztelés, szerep alapú felülvizsgálat, technikai felülvizsgálat

Tanulási célok a Statikus tesztelés fejezethez

3.1. Statikus tesztelési alapok

AK-3.1.1. (K1) Ismerje fel azokat a szoftver munkatermék típusokat, amelyek vizsgálhatók a különböző statikus tesztelési technikákkal

AK-3.1.2. (K2) Ismertesse a statikus tesztelés előnyeit példákon keresztül

AK-3.1.3. (K2) Ismertesse a statikus és a dinamikus technikák közötti különbséget, figyelembe véve a tesztcélokat, a megtalálható hibák típusait, és ezen módszerek szerepét a szoftver életciklusa során

3.2. A felülvizsgálat folyamata

AK-3.2.1. (K2) Foglalja össze a munkatermék felülvizsgálati folyamatának tevékenységeit

AK-3.2.2. (K1) Ismerje fel a formális felülvizsgálat során a különböző szerepeket és felelősségeket

AK-3.2.3. (K2) Fejtse ki a különböző felülvizsgálati típusok (informális felülvizsgálat, átvizsgálás, technikai felülvizsgálat és inspekció) közötti különbségeket

AK-3.2.4. (K3) Alkalmazzon felülvizsgálati technikát egy adott munkatermékre, a hibák megtalálására

AK-3.2.5. (K2) Fejtse ki a sikeres felülvizsgálathoz hozzájáruló tényezőket

3.1 A statikus tesztelés alapjai

A dinamikus teszteléssel ellentétben – melyhez a tesztelés alatt álló szoftver futtatása szükséges – a statikus tesztelés a munkatermékek manuális vizsgálatára (azaz felülvizsgálatok), illetve a kód vagy más munkatermékek eszközvezérelt vizsgálatára (azaz statikus elemzésre) támaszkodik. A statikus tesztelés mindkét típusa úgy értékeli a tesztelés alatt lévő kódot vagy más munkaterméket, hogy ténylegesen nem hajtja végre a kódot vagy a munkaterméket.

A statikus elemzés fontos a biztonsági szempontból kritikus komputerrendszerek esetén (mint például repülési, orvosi, vagy nukleáris szoftverek esetén), de a statikus elemzés úgyszintén fontos és általános lett más körülmények esetén is. Így például a statikus elemzés fontos része a biztonsági tesztelésnek. A statikus elemzést ugyanakkor gyakran építik be automatikus szoftver fordító (compiler) és átadó rendszerekbe, mint például Agilis fejlesztés esetén a folyamatos átadási és telepítési rendszerekbe.

3.1.1 Statikus teszteléssel vizsgálható munkatermékek

Csaknem bármilyen munkatermék vizsgálható statikus tesztelés alkalmazásával (felülvizsgálatok, és/vagy statikus elemzés), mint például:

- Specifikációk, beleértve az üzleti követelményeket, funkcionális követelményeket, és biztonsági követelményeket
- Leírások, felhasználói történetek, és elfogadási feltételek
- Architekturális és tervezési specifikációk
- Kód
- Tesztver, beleértve teszttterveket, teszteseteket, teszteljárásokat és automatizált tesztszkripteket
- Felhasználói kézikönyvek
- Weboldalak
- Szerződések, projekt tervek, időbeosztások, és költségvetés tervezetek
- Konfiguráció terv és infrastruktúra terv
- Modellek, mint például aktivitási diagramok, amelyek alkalmazhatóak modell alapú tesztelés során (lásd például ISTQB-CTFL-MBT Foundation Level Model-Based Tester Extension Syllabus, valamint Kramer 2016)

Felülvizsgálatok alkalmazhatók bármely olyan munkatermék esetében, amelyről a résztvevők tudják, hogyan olvasandó és értelmezendő. Statikus elemzés hatékonyan alkalmazható bármely olyan munkatermék esetén, amely formális szerkezettel rendelkezik (tipikusan ez kódok és modellek esetén áll fenn) és amelyre megfelelő statikus elemzés eszköz létezik. Statikus elemzés még olyan eszközökkel is alkalmazható, amelyek természetes nyelven megírt munkatermékeket – mint például követelmények - képesek értékelni (például helyesírásra, nyelvtanra és olvashatóságra történő ellenőrzések).

3.1.2 A statikus tesztelés előnyei

A statikus tesztelési technikák különböző előnyöket nyújtanak. Amikor a szoftverfejlesztési életciklus korai szakaszában alkalmazzuk, a statikus tesztelés lehetővé teszi a hibák korai észlelését, még a dinamikus tesztelés végrehajtása előtt (például a követelmények, vagy a tervezési specifikációk felülvizsgálata során, teendőlista (backlog) pontosítása során, stb.). A korán megtalált hibák eltávolítása gyakran sokkal olcsóbb, mint az életciklus későbbi szakaszában megtalált hibáké, különös tekintettel azon hibákra, amelyeket csak a szoftver telepítése után és aktív használata során fedezünk fel. Statikus tesztelési technikák alkalmazása a

hibák megtalálására, és ezeknek a hibáknak az azonnali kijavítása csaknem mindig sokkal olcsóbb a szervezet számára, mint dinamikus tesztelés alkalmazása a hibák megtalálására a tesztobjektumban és az ezt követő javítás, különösen akkor, ha figyelembe vesszük a további költségeket, amelyek együtt járnak a többi munkatermék frissítésével, valamint ellenőrző és regressziós tesztelés végrehajtásával.

A statikus tesztelés további előnyei lehetnek a következők:

- Hatékonyabb hiba felismerés és javítás, még a dinamikus tesztelés végrehajtása előtt
- Olyan hibák azonosítása, amelyek nem találhatók meg könnyen dinamikus teszteléssel
- Hibák megelőzése a tervezésben vagy a kódolásban azzal, hogy felfedezzük az inkonzisztenciákat, nem egyértelmű meghatározásokat, ellentmondásokat, kihagyásokat, pontatlanságokat, és az ismétlődéseket a követelményekben
- A fejlesztés termelékenységének növelése (például a javított tervezésnek köszönhetően vagy a jobban karbantartható kód miatt)
- A fejlesztési költségek és időigény csökkentése
- A tesztelési költségek és időigény csökkentése
- A szoftver élettartama során a minőség teljes költségének csökkentése azáltal, hogy kevesebb meghibásodás kerül elő az életciklus későbbi szakaszában, vagy a működésbe állítás után
- A felülvizsgálatokban való részvétel javítja a csoporttagok közötti kommunikációt

3.1.3 A statikus és dinamikus tesztelés közötti különbségek

A statikus tesztelésnek és a dinamikus tesztelésnek lehetnek azonosak a céljai (lásd az 1.1.1 fejezetet), mint például a munkatermékek minőségének kiértékelése és a hibák lehető legkorábbi azonosítása. A statikus és a dinamikus tesztelés kiegészíti egymást azzal, hogy különböző típusú hibákat találnak meg.

Az egyik fő eltérés az, hogy a statikus tesztelés a munkatermékekben a hibákat közvetlenül találja meg, ellentétben azzal, hogy a hibák okozta meghibásodásokat azonosítjuk a szoftver futtatása során. A hiba hosszú ideig rejtőzhet egy munkatermékben anélkül, hogy meghibásodást váltana ki. Lehet, hogy az a végrehajtási útvonal, amelyen a hiba elhelyezkedik, csak ritkán hajtódik végre, vagy nehezen érhető el, így nem lesz könnyű olyan dinamikus tesztet létrehozni és végrehajtani, amely találkozik a hibával. A statikus tesztelés sokkal kevesebb erőfeszítés mellett is képes lehet a hiba megtalálására.

Egy másik különbség lehet az, hogy a statikus tesztelés felhasználható a munkatermékek konzisztenciájának és belső minőségének javítására, míg a dinamikus tesztelés tipikusan a kívülről látható viselkedésekre fókuszál.

A dinamikus teszteléssel összehasonlítva azok a tipikus hibák, amelyeket könnyebb és olcsóbb statikus teszteléssel megtalálni és javítani, a következőket foglalják magukba:

- Követelmény hibák (például következetlenségek, többértelműségek, ellentmondások, kihagyások, pontatlanságok és ismétlődések)
- Tervezési hibák (például nem hatékony algoritmusok vagy adatbázis struktúrák, erős kapcsolódás, kismértékű kohézió)
- Kódolási hibák (például változók értékadás nélkül, deklarált, de fel nem használt változók, elérhetetlen kód, ismétlődő kód)
- Szabványoktól való eltérés (nem követi például a kódolási konvenciókat)

- Helytelen interfész specifikációk (eltérő mértékegységek használata például a hívó rendszerben és a hívott rendszerben)
- Biztonsági sebezhetőségek (például a puffer túlcsordulás lehetőségei)
- Hiányok vagy pontatlanságok a tesztbázis nyomonkövethetőségében vagy a lefedettségben (például hiányzó tesztek egy elfogadási feltétel vizsgálatára)

A karbantarthatósági hibák legtöbb típusa továbbá csak statikus teszteléssel található meg (például a nem megfelelő modularizáció, a komponensek korlátozott újrafelhasználhatósága, nehezen elemezhető és új hibák létrehozása nélkül alig módosítható kód).

3.2 A felülvizsgálat folyamata

A felülvizsgálatok az informálistól a formálisig változhatnak. Az informális felülvizsgálatokra az jellemző, hogy nem követnek előre meghatározott folyamatot és nem rendelkeznek formális dokumentált kimenettel. A formális felülvizsgálatokra a csoportos részvétel, a felülvizsgálat dokumentált eredményei, és a felülvizsgálat végrehajtásához dokumentált eljárások felhasználása a jellemző. A felülvizsgálat folyamatának formalitása olyan tényezőktől függ, mint például a szoftverfejlesztés életciklusmodellje, a fejlesztési folyamat kidolgozottsága (érettsége), a felülvizsgálat alá vetett munkatermék komplexitása, bármely jogi vagy szabályozási követelmény, és/vagy egy audit szükségessége.

A felülvizsgálat fókusza függ a felülvizsgálat közös megegyezéssel kialakított céljaitól (például hibák megtalálása, ismeretszerzés és megértés, a résztvevők, mint például a tesztelők és az új csapattagok oktatása, vagy megbeszélés és megegyezéses döntéshozatal).

A vonatkozó ISO szabvány (ISO/IEC 20246) több, részletes leírást tartalmaz a munkatermékek felülvizsgálati folyamatáról, beleértve a szerepeket és a felülvizsgálati technikákat.

3.2.1 Munkatermék felülvizsgálati folyamata

A felülvizsgálati folyamat az alábbi főbb tevékenységekből áll:

Tervezés

- A hatókör meghatározása, amely magába foglalja a felülvizsgálat célját, azt, hogy milyen dokumentumok vagy dokumentumrészek kerülnek felülvizsgálatra, és az értékelendő minőségi jellemzőket
- A ráfordítás és az időkeret becslése
- A felülvizsgálati jellemzők meghatározása, mint például a felülvizsgálat típusa szerepekkel, tevékenységekkel és ellenőrzőlistákkal
- A felülvizsgálatban részt vevő személyek kiválasztása és a szerepek kiosztása
- A formálisabb felülvizsgálat típusokhoz (mint például inspekció) a belépési és kilépési feltételek meghatározása
- A belépési feltételek teljesülésének ellenőrzése (formálisabb felülvizsgálat típusok esetén)

A felülvizsgálat indítása

- A munkatermék vagy más anyagok kiosztása (fizikai vagy elektronikus úton), mint például jegyzőkönyv űrlapok, ellenőrzőlisták és kapcsolódó munkatermékek
- A hatókör, a célok, a felülvizsgálati folyamat, a szerepek, és munkatermékek ismertetése a résztvevők számára

- A felülvizsgálattal kapcsolatban a résztvevőkben felmerülő kérdések megválaszolása

Egyéni felülvizsgálat (azaz egyéni felkészülés)

- A munkatermék egészének vagy részének felülvizsgálata
- A lehetséges hibák, javaslatok és kérdések feljegyzése

Felmerülő kérdések ismertetése és elemzése

- Az azonosított lehetséges hibák ismertetése (például egy felülvizsgálati megbeszélés során)
- A lehetséges hibák elemzése, felelős kijelölése és állapotuk rögzítése
- A minőségi jellemzők értékelése és dokumentálása
- A felülvizsgálat eredményeinek értékelése a kilépési feltételekkel szemben, felülvizsgálati döntés meghozatala érdekében (elutasítás; nagyobb változtatásokra van szükség; elfogadás, esetleg kisebb változtatásokkal)

Hibajavítás és jelentéskészítés

- Hibajelentések létrehozása azon megtalált problémák esetében, amelyek egy munkatermék változtatását igénylik
- A megtalált hibák javítása a felülvizsgált munkatermékben (tipikusan a szerző végzi)
- A hibák közlése a megfelelő személy vagy csapat felé (amikor ilyet a felülvizsgált munkatermékkel kapcsolatban lévő munkatermékben találtak)
- A hibák frissített státuszának feljegyzése (formális felülvizsgálatok esetében), lehetőség szerint a megjegyzés eredeti szerzőjének beleegyezésével
- Metrikák gyűjtése (formálisabb felülvizsgálati típusok esetén)
- A kilépési feltétel elérésének ellenőrzése (formálisabb felülvizsgálati típusok esetén)
- A munkatermék elfogadása, amennyiben a kilépési feltétel teljesült

Egy munkatermék felülvizsgálatának eredményei változhatnak a felülvizsgálat típusának és formalitásának megfelelően, ahogy ez részletezésre kerül a 3.2.3 fejezetben.

3.2.2 Szerepek és felelősségek formális felülvizsgálat esetében

Egy tipikus formális felülvizsgálat az alábbi szerepeket foglalja magában:

Szerző

- Létrehozza a felülvizsgálat alá vett munkaterméket
- A felülvizsgálat alá vett munkatermék hibáit javítja (szükség esetén)

Menedzsment

- Felelős a felülvizsgálat megtervezéséért
- Meghatározza a felülvizsgálatok kivitelezését
- Személyeket, költségvetést és időkeretet rendel a felülvizsgálatokhoz
- Folyamatosan figyeli a költséghatékonyságot

- Kontroll döntéseket hajt végre amennyiben elégtelen kimeneteket tapasztal

Témavezető (vezető, gyakran moderátornak nevezzük)

- Biztosítja a felülvizsgálati értekezlet hatékony levezetését (az értekezlet megtartása során)
- Szükség esetén közvetít a különböző nézőpontok között
- Gyakran ő az a személy, akin a felülvizsgálat sikere múlik

Felülvizsgálat vezető

- Általános felelőssége van a felülvizsgálat végrehajtásáért
- Meghatározza a résztvevőket, valamint megszervezi, hol és mikor kerül végrehajtásra

Felülvizsgálók

- Lehetnek a témában járatos szakértők, olyan személyek, akik a projekten dolgoznak, a munkatermékben érintett személyek, és/vagy olyan személyek, akik speciális technikai vagy üzleti háttérrel rendelkeznek
- A felülvizsgálat alá vett munkatermék lehetséges hibáit képesek felismerni
- Képviselhetnek különböző nézőpontokat (mint például tesztelő, fejlesztő, felhasználó, üzemeltető, üzleti elemző, felhasználhatósági szakértő, stb.)

Írnok (vagy jegyzőkönyvvezető)

- Az egyes felülvizsgálati tevékenységek során megtalált lehetséges hibákat egyezteti
- Feljegyzi az új lehetséges hibákat, nyitott kérdéseket és döntéseket a megtartott felülvizsgálati értekezletek során

Néhány felülvizsgálat típus esetén adott személy több szerepet is betölthet, és az egyes szerepekhez társuló tevékenységek szintén változhatnak a felülvizsgálat típusa alapján. A felülvizsgálati folyamatot támogató eszközöknek köszönhetően gyakran nincs már szükség írnokra, különösképpen a hibák, nyitott kérdések és döntések rögzítésében.

További, jobban részletezett szerepek is lehetségesek, ahogy ez leírásra került a megfelelő ISO szabványban (ISO/IEC 20246).

3.2.3 Felülvizsgálat típusok

Bár a felülvizsgálatok különböző célokra használhatóak, az egyik legfontosabb céljuk a hibák felfedése. Mindegyik felülvizsgálati típus segítséget nyújthat a hibák megtalálásában, és a kiválasztott felülvizsgálati típust a projekt szükségletei, a rendelkezésre álló erőforrások, a termék típusa és a kockázatok, az üzleti terület és a szervezeti kultúra kell, hogy meghatározzák, egyéb további kiválasztási feltételek mellett.

Egyetlen munkatermék több felülvizsgálat típus tárgya is lehet. Amennyiben több, mint egy felülvizsgálatot végzünk, a sorrend változó lehet. Végezhetünk például informális felülvizsgálatot egy technikai felülvizsgálatot megelőzően, annak biztosítására, hogy a munkatermék alkalmas a technikai felülvizsgálatra.

A fentebb leírt felülvizsgálatok végrehajthatók mint egyenrangú felülvizsgálatok, azaz körülbelül hasonló szervezeti szinten álló kollégák részvételével.

Az adott felülvizsgálat során megtalált hibák típusa változó, elsősorban a felülvizsgálat alatt levő munkaterméktől függő módon. A 3.1.3 fejezetben találhatunk példákat olyan hibákra, amelyek különböző munkatermékek felülvizsgálata nyomán bukkantak fel, és a formális inspekciókról még további információk

is rendelkezésre állnak Gilb 1993. A felülvizsgálatok különböző sajátságaik mentén csoportosíthatóak. Az alább következőkben felsoroljuk a négy leggyakoribb felülvizsgálati típust és a hozzájuk társuló sajátosságokat.

Informális felülvizsgálat (mint például kollegiális ellenőrzés, páralkotás, páros felülvizsgálat)

- Fő célja: a lehetséges hibák kimutatása
- További lehetséges célok: új ötletek vagy megoldások megteremtése, apróbb problémák gyors megoldása
- Nem alapszik formális (dokumentált) folyamaton
- Nem szükséges felülvizsgálati megbeszélés összehívása
- Végrehajtható a szerző kollégájával (kollegiális ellenőrzés) vagy több személlyel
- Az eredmények dokumentálhatóak (de nem kötelező)
- Hasznossága a felülvizsgálóktól függő módon változó
- Ellenőrzőlisták használata opcionális
- Nagyon gyakran alkalmazzák Agilis fejlesztés során

Átvizsgálás

- Fő céljai: hibák megtalálása, a szoftver termék javítása, alternatív megvalósítások figyelembevétele, a szabványokkal és a specifikációkkal szembeni megfelelőség értékelése
- További lehetséges célok: véleménycsere a technikákról vagy a stílus változatokról, a résztvevők oktatása, egyetértés megteremtése
- A felülvizsgálati megbeszélést megelőző egyéni felkészülés opcionális
- A felülvizsgálati megbeszélést tipikusan a munkatermék szerzője vezeti
- Írnok kötelező
- Ellenőrzőlisták használata opcionális
- Végrehajtható forgatókönyvek, száraz tesztelések vagy szimulációk formájában történik
- A lehetséges hibák listái és felülvizsgálati jelentések jönnek létre
- A gyakorlatban az egészen informálistól az erősen formálisig változhat

Technikai felülvizsgálat

- Fő célok: egyetértés megteremtése, lehetséges hibák kimutatása
- Lehetséges további célok: minőség értékelése és bizalom felépítése a munkatermékkel szemben, tér biztosítása új ötleteknek, a szerzők motiválása és képessé tétele a jövőbeli munkatermékek minőségének javítására, alternatív megvalósítások figyelembevétele
- A felülvizsgálók a szerző technikai partnerei, és technikai szakértők legyenek, azonos vagy más területen
- A felülvizsgálati megbeszélés előtt egyéni felkészülés szükséges

- A felülvizsgálati megbeszélés opcionális, ideális esetben képzett témavezető irányítja (tipikusan ez a személy nem a szerző)
- Írnok kötelező, ideális esetben ez a személy nem a szerző
- Ellenőrzőlisták alkalmazása opcionális
- A lehetséges hibák listái és felülvizsgálati jelentések készülnek el

Inspekció

- Fő célok: lehetséges hibák kimutatása, minőség értékelése és bizalom felépítése a munkatermékkel szemben, a jövőbeli hasonló hibák megelőzése szerzői tanulási folyamaton és a kiváltó ok elemzésén keresztül
- Lehetséges további célok: a szerzők motiválása és képessé tétele a jövőbeli munkatermékek és a szoftverfejlesztési folyamat minőségének javítására, egyetértés elérése
- Meghatározott folyamatot követ formális dokumentált kimenetekkel, szabályok és ellenőrzőlisták alapján
- Világosan meghatározott szerepeket alkalmaz, mint például a 3.2.2 fejezetben meghatározott szerepek, amelyek alkalmazása kötelező, és kijelölt felolvasó is részt vehet benne (aki a munkatermék szövegét hangosan felolvassa, gyakran ki is fejti, azaz a saját szavaival írja le, a felülvizsgálati megbeszélés során)
- A felülvizsgálati megbeszélés előtt egyéni felkészülésre van szükség
- A felülvizsgálók a szerző egyenrangú partnerei, vagy szakértők más, a munkatermékhez kapcsolódó területeken
- Előre meghatározott belépési és kilépési feltételeket alkalmaznak
- Írnok részvétele kötelező
- A felülvizsgálati megbeszélést képzett témavezető irányítja (nem a szerző)
- A szerző nem lehet felülvizsgálat vezető, felolvasó vagy írnok
- Elkészül a lehetséges hibák listája és a felülvizsgálati jelentés
- Metrikákat gyűjtünk, és a teljes szoftverfejlesztési folyamat javítására használjuk fel, beleértve az inspekció folyamatát is

3.2.4 Felülvizsgálati technikák alkalmazása

Számos felülvizsgálati technika létezik, amelyek az egyéni felülvizsgálati tevékenységek (azaz egyéni felkészülés) során alkalmazhatóak a hibák feltárására. Ezek a technikák a fentebb ismertetett felülvizsgálati típusok mindegyike esetén felhasználhatóak. A technikák hatásossága eltérhet az alkalmazott felülvizsgálat típus függvényében. A változó felülvizsgálat típusok esetén a különböző egyéni felülvizsgálati technikákra az alábbiakban sorolunk fel példákat.

Ad hoc

Az ad hoc felülvizsgálat során a felülvizsgálók kevés irányítást kapnak, vagy egyáltalán nem kapnak irányítást arra vonatkozóan, hogyan kell a feladatot elvégezni. A felülvizsgálók a munkatermék részeit gyakran szekvenciálisan olvassák végig, és az egyes kérdések beazonosítását és dokumentálását akkor végzik, amikor azokkal találkoznak. Az ad hoc felülvizsgálat kevés előkészítést igénylő, gyakran alkalmazott

technika. Ez a módszer nagymértékben függ a felülvizsgáló képességeitől, és sok ismétlődő problémához vezethet, amelyeket a különböző felülvizsgálók jelentenek be.

Ellenőrzőlista alapú

Az ellenőrzőlista alapú felülvizsgálat egy szisztematikus technika, ahol a felülvizsgálók a problémákat a felülvizsgálat indításakor kiosztott (például a témavezető által szétosztott) ellenőrzőlisták alapján ismerik fel. Egy felülvizsgálati ellenőrzőlista a feltételezhető hibák alapján összeállított kérdéscsoportokat tartalmazza, ezek gyakran tapasztalat alapján lettek összeállítva. Az ellenőrzőlisták specifikusak legyenek a felülvizsgálat alá vont munkatermék típusra, és rendszeresen karbantartartásra szorulnak, hogy lefedjék azokat a problématípusokat, amelyek a korábbi felülvizsgálatokban esetleg kimaradtak. Az ellenőrzőlista alapú technika fő előnye az, hogy ez szisztematikus lefedése a gyakori hibatípusoknak. Gondot kell fordítani arra, hogy ne csak egyszerűen az ellenőrzőlistát kövessük az egyes felülvizsgálatok során, hanem figyeljünk az ellenőrzőlistáról kimaradt hibákra is.

Forgatókönyvek és száraz tesztelés

Egy forgatókönyv alapú felülvizsgálat során a felülvizsgálók strukturált iránymutatást kapnak arról, hogyan olvassák végig a munkaterméket. A forgatókönyv alapú felülvizsgálat támogatja a felülvizsgálókat abban, hogy száraz tesztelést (elméleti futtatást) végezzenek a dokumentumon, a munkatermék elvárt felhasználása alapján (amennyiben a munkatermék a megfelelő formában, például használati esetek formájában lett dokumentálva). Ezek a forgatókönyvek az egyszerű ellenőrzőlista elemekkel összehasonlítva jobb iránymutatást adnak a felülvizsgálóknak arra, hogyan azonosítsanak specifikus hibatípusokat. Ahogyan az ellenőrzőlista alapú felülvizsgálatokban is, a felülvizsgálók nem szorítkozhatnak kizárólag a dokumentált forgatókönyvekre annak érdekében, hogy ne tévesszék szem elől a további hibatípusokat (például hiányzó jellemzőket).

Perspektíva alapú olvasat

A perspektíva alapú olvasatban – egy szerep alapú felülvizsgálathoz hasonló módon – a felülvizsgálók a különböző érintett felek nézőpontjait veszik figyelembe az egyéni felülvizsgálat során. Az érintett felek nézőpontjai közé tipikusan a végfelhasználói, marketinges, tervezői, tesztelői vagy üzemeltetői nézőpontok tartoznak. Az eltérő érintett személyi nézőpontok alkalmazása mélyebb megértést jelent az egyéni felülvizsgálat során, kevesebb probléma ismétlődéssel az egyes felülvizsgálók között.

A perspektíva alapú olvasat ezen felül azt is megköveteli a felülvizsgálóktól, hogy kipróbálják a felülvizsgálat alá vont munkatermék használatát, ezzel előállítva azt a terméket, amelyet ebből származtatnának. Egy tesztelő megpróbálkozhat például azzal, hogy vázlatos elfogadási teszteket generál amikor perspektíva alapú olvasatot végez egy követelményspecifikáción, ezzel ellenőrizve, hogy az összes szükséges információ rendelkezésre áll az anyagban. A perspektíva alapú olvasat során az is elvárt, hogy ellenőrzőlistákat használjunk.

Kísérleti tanulmányok azt mutatják, hogy a perspektíva alapú olvasat a leghatékonyabb általános módszer a követelmények és a technikai munkatermékek felülvizsgálatára. A különböző érintett felek nézőpontjainak figyelembevétele és kockázat alapú súlyozása a technika sikerének kulcsfontosságú tényezője. Lásd még a Shul 2000 referenciát a perspektíva alapú olvasat részleteiért, és a Sauer 2000 referenciát a különböző felülvizsgálati technikák hatékonyságáról.

Szerep alapú

A szerep alapú felülvizsgálat olyan technika, ahol a felülvizsgálók a munkaterméket az egyes érintett személyi szerepek szempontjából értékelik. A tipikus szerepek közé tartoznak a specifikus végfelhasználó típusok (gyakorlott, gyakorlatlan, idős, gyerek, stb.) és a szervezetben specifikus szerepet viselők is (felhasználó, adminisztrátor, rendszer adminisztrátor, teljesítménytesztelő, stb.) Ugyanazon elvek vonatkoznak rá, mint a perspektíva alapú olvasatban, mivel a szerepek hasonlóak.

3.2.5 Felülvizsgálatok sikerességi tényezői

Annak érdekében, hogy sikeres felülvizsgálatot hajtsunk végre, meg kell határozni a megfelelő felülvizsgálat típust és a használandó technikát. Ezen felül számos további tényező van, amelyek befolyásolják a felülvizsgálat kimenetelét.

A felülvizsgálatok szervezeti sikerességi tényezői közé tartoznak a következők:

- Minden felülvizsgálatnak legyen világosan meghatározott célkitűzése, amelyet a felülvizsgálat tervezése során határoztak meg és mint mérhető kilépési feltételt használják fel
- Olyan felülvizsgálat típusokat használjanak, amelyek alkalmasak a célok elérésére és a szoftver munkatermékek típusához és szintjéhez jól illeszkednek, valamint a résztvevőknek is megfelelnek
- Bármely felülvizsgálati technikát is használjuk, mint például ellenőrzőlista alapú vagy szerep alapú felülvizsgálatot, alkalmas kell legyen a hatékony hibafeltárásra a felülvizsgálat alá vont munkatermékben
- A felhasznált ellenőrző listák mindegyike vegye figyelembe a főbb kockázatokat és legyen naprakész
- A terjedelmes dokumentumokat kisebb darabokban írják meg és vizsgálják felül, így a minőség-ellenőrzés jobban kivitelezhető, mert a szerzők korai és gyakori visszajelzést kapnak a hibákról
- A résztvevők elegendő időt kapjanak a felkészülésre
- A felülvizsgálatokat megfelelő előre bejelentéssel szervezzék meg
- A menedzsment támogassa a felülvizsgálat folyamatát (azaz a projekt ütemtervében kellő időt hagyjanak a felülvizsgálati tevékenységekre)
- A felülvizsgálatok épüljenek be a cég minőségbiztosítási és/vagy tesztelési irányvonalába

A felülvizsgálatok személyhez kötődő sikerességi tényezői közé tartoznak a következők:

- A felülvizsgálati célok eléréséhez a megfelelő személyeket kell bevonni, így például különböző készségekkel rendelkező, vagy eltérő nézőpontokat képviselő személyeket, akik a dokumentumot, mint munka bemenetet használhatják fel
- A tesztelőket értékes felülvizsgálóknak tekintsék, akik egyrészt hozzájárulnak a felülvizsgálathoz, másrészt megismerik a munkaterméket, ami hozzásegíti őket ahhoz, hogy hatékonyabb teszteket állítsanak össze, és ezeket a teszteket hamarabb el tudják készíteni
- A résztvevők elegendő időt szenteljenek a munkára és vegyék figyelembe a részleteket is
- A felülvizsgálatokat kisebb részletekben végezzék, így a felülvizsgálók figyelme nem lankad az egyéni felülvizsgálat során és/vagy a megtartott felülvizsgálati megbeszélés alatt
- A megtalált hibákat eredménynek tekintsék, értékeljék és tárgyilagosan kezeljék
- A felülvizsgálati megbeszélést hatékonyan kell lebonyolítani, hogy a résztvevők az megbeszélés időtartamát értékesen eltöltött időnek tekintsék
- A felülvizsgálatot bizalmi légkörben kell lebonyolítani, az eredményt nem szabad felhasználni a résztvevők értékeléséhez
- A résztvevőknek nem szabad olyan testbeszédet és viselkedést folytatni, amely a többi résztvevő számára unalmat, bosszúságot vagy ellenségességet jelezhet

- Megfelelő oktatást kell biztosítani, különösen a formálisabb felülvizsgálat típusok esetében, mint például az inspekció
- Hangsúlyt kell fektetni a tanulásra és elő kell segíteni a folyamat javítását

Lásd még a Gilb 1993, Wiegers 2002 és a van Veenendaal 2004 referenciákat a sikeres felülvizsgálatokról.

4. Teszttechnikák 330 perc

Kulcsszavak

állapotátmenet tesztelés, döntési lefedettség, döntési tábla tesztelés, ekvivalenciaparticionálás, ellenőrző lista alapú tesztelés, fehérdoboz teszttechnika, feketedoboz teszttechnika, felderítő tesztelés, használati eset tesztelés, határérték-elemzés, hibasejtés, lefedettségm, tapasztalat alapú teszttechnika, teszttechnika, utasítás lefedettség

Tanulási célok a Teszttechnikák fejezethez

4.1 A tesztelési technikák kategóriái

AK-4.1.1 (K2) Ismertesse a feketedoboz teszttechnikák, a fehérdoboz teszttechnikák és a tapasztalat alapú teszttechnikák jellemzőit, hasonlóságait, illetve különbségeit

4.2 Feketedoboz tesztelési technikák

AK-4.2.1 (K3) Alkalmazza az ekvivalenciaparticionálást adott követelményekből történő tesztesetek származtatására

AK-4.2.2 (K3) Alkalmazza a határérték-elemzést adott követelményekből történő tesztesetek származtatására

AK-4.2.3 (K3) Alkalmazza a döntési tábla tesztelést adott követelményekből történő tesztesetek származtatására

AK-4.2.4 (K3) Alkalmazza az állapotátmenet tesztelést adott követelményekből történő tesztesetek származtatására

AK-4.2.5 (K2) Ismertesse a használati esetből történő tesztesetek származtatási módját

4.3 Fehérdoboz tesztelési technikák

AK-4.3.1 (K2) Ismertesse az utasítás lefedettséget

AK-4.3.2 (K2) Ismertesse a döntési lefedettséget

AK-4.3.3 (K2) Ismertesse az utasítás- és a döntési lefedettség értékét

4.4 Tapasztalat alapú tesztelési technikák

AK-4.4.1 (K2) Ismertesse a hibasejtést

AK-4.4.2 (K2) Ismertesse a felderítő tesztelést

AK-4.4.3 (K2) Ismertesse az ellenőrzőlista alapú tesztelést

4.1 A teszttechnikák kategóriái

Egy teszttechnika (beleértve az ebben a fejezetben tárgyaltakat) alkalmazásának célja, hogy segítsen a tesztfeltételek, a tesztesetek és a tesztadatok meghatározásában.

Annak eldöntése, hogy melyik teszttechnikát alkalmazzuk számos tényezőtől függ, beleértve:

- A komponens vagy rendszer komplexitása
- Szabályozó rendelkezések
- Az ügyfél követelményei, illetve a szerződésben foglalt követelmények
- Kockázati szintek és típusok
- Rendelkezésre álló dokumentáció
- Tesztelő tudása és képességei
- Rendelkezésre álló eszközök
- Idő és költségvetés
- Szoftverfejlesztési életciklus modell
- A komponensben vagy rendszerben várt hibák típusai

Egyes technikák alkalmasabbak bizonyos szituációkban és tesztszinteken; mások minden tesztszinten használhatók. A tesztesetek létrehozásakor a tesztelők általában teszttechnikák kombinációit alkalmazzák, hogy a teszt ráfordítás mellett a lehető legjobb eredményt érjék el.

A tesztelemzés, műszaki teszttervezés és a teszt megvalósítása során használt teszttechnikák a nagyon informálistól (kevés vagy egyáltalán nem létező dokumentáció) a nagyon formálisig terjedhetnek. A formalitás megfelelő szintje a tesztelés kontextusától függ, beleértve a tesztelési és fejlesztési folyamatok érettségét, időbeli megkötéseket, biztonsági és szabályozási követelményeket, a résztvevők tudását, képességeit és a követett szoftverfejlesztési életciklusmodellt.

4.1.1 A teszttechnikák kategóriái és jellemzőik

Ezen tantervben a teszttechnikákat az alábbi osztályokra bontjuk: feketedoboz, fehérdoboz vagy tapasztalat alapú.

A feketedoboz teszttechnikák (viselkedési vagy viselkedés alapú technikáknak is nevezzük) a megfelelő tesztbázis (pl. formális követelménydokumentumok, specifikációk, használati esetek, felhasználói történetek vagy üzleti folyamatok) elemzésén alapulnak. Ezen technikák mind a funkcionális, mind a nemfunkcionális teszteléshez alkalmasak. A feketedoboz teszttechnikák a tesztelés tárgyának bemeneteire és kimeneteire koncentrálnak, a belső szerkezetre történő hivatkozás nélkül.

A fehérdoboz teszttechnikák (strukturális vagy struktúra alapú technikáknak is nevezzük) a teszt tárgyának architektúrája, részletes terve, belső struktúrája vagy a kódja elemzésén alapulnak. A feketedoboz teszttechnikával ellentétben, a fehérdoboz teszttechnikák a teszt tárgyán belüli struktúrára, feldolgozásra koncentrálnak.

A tapasztalat alapú teszttechnikák a fejlesztők, tesztelők és felhasználók tapasztalatait használják ki a tesztek tervezéséhez, megvalósításához és végrehajtásához. Ezen technikákat gyakran kombinálják feketedoboz és fehérdoboz teszttechnikákkal.

A feketedoboz teszttechnikák közös jellemzői többek között az alábbiak:

- A tesztfeltételeket, teszteseteket és tesztadatokat a tesztbázisból származtatják, ami lehet szoftverkövetelmény, specifikáció, használati eset és felhasználói történet
- A tesztesetek arra is használhatók, hogy azonosítsák a követelmények és a megvalósítás közötti réseket, illetve a követelményektől való eltéréseket
- A lefedettséget a tesztbázis tesztelt elemei és a tesztbázison alkalmazott technika alapján mérik

A fehérdoboz teszttechnikák közös jellemzői többek között:

- A tesztfeltételeket, teszteseteket és tesztadatokat a tesztbázisból származtatják, ami lehet kód, szoftverarchitektúra, részletes terv vagy bármi más, a szoftver struktúrájára vonatkozó információforrás
- A lefedettséget egy kiválasztott struktúrán (pl. kód vagy interfész) belüli tesztelt elemek, illetve a tesztbázison alkalmazott technika alapján számítjuk

A tapasztalat-alapú teszttechnikák közös jellemzői többek között:

- A tesztfeltételeket, teszteseteket és tesztadatokat a tesztbázisból származtatják, ami lehet a tesztelők, fejlesztők, felhasználók és más érintett fél tudása és tapasztalata

Ez a tudás és tapasztalat magában foglalja a szoftvernek és környezetének elvárt használatát, a valószínű hibákat, és ezen hibák eloszlását.

Az ISO szabvány (ISO/IEC/IEEE 29119-4) a teszttechnikákról és a megfelelő lefedettségi mértékekről tartalmaz leírásokat (további információkért a technikákról lásd a Craig 2002 és Copeland 2004 referenciát).

4.2 Feketedoboz teszttechnikák

4.2.1 Ekvivalenciaparticionálás

Az ekvivalenciaparticionálás az adatokat partíciókra (ekvivalencia osztályoknak is nevezzük) osztja úgy, hogy az egy partícióban szereplő elemek elvárt viselkedése ugyanolyan (lásd a Kaner 2013 és Jorgensen 2014 referenciát). Mind az érvényes, mind az érvénytelen értékekre léteznek ekvivalenciapartíciók.

- Érvényesek azok az értékek, amelyeket a komponensnek vagy rendszernek el kell fogadnia. Az érvényes értékeket tartalmazó ekvivalenciapartíciót érvényes ekvivalenciapartíciónak nevezzük.
- Érvénytelenek azok az értékek, amelyeket a komponensnek vagy rendszernek el kell utasítania. Az érvénytelen értékeket tartalmazó ekvivalenciapartíciót érvénytelen ekvivalenciapartíciónak nevezzük.
- Partíciókat minden olyan elemhez meghatározhatunk, amely a teszt tárgyához kapcsolódik, beleértve a bemeneteket, kimeneteket, belső értékeket, idővel kapcsolatos értékeket (pl. egy esemény előtt vagy után) és interfész paramétereket (pl. tesztelendő integrált komponenseket az integrációs teszt során)
- Minden partíció szükség esetén további, kisebb partíciókra osztható
- Minden érték egy és csak egy ekvivalenciapartícióhoz tartozhat
- Amikor egy tesztesetben érvénytelen ekvivalenciapartíciókat használnak, egyenként kell őket tesztelni, tehát nem lehet egymással kombinálni az érvénytelen partíciókat, ezzel biztosítva, hogy a meghibásodásokat nem fedjük el. A meghibásodásokat elfedhetjük, ha több, szimultán fellépő meghibásodás közül csak az egyik látható, ezáltal a többi észrevétlen marad.

Annak érdekében, hogy 100%-os lefedettséget érjünk el ezzel a technikával, a teszteseteknek le kell fedniük az összes azonosított partíciót (az érvénytelen partíciókat is beleértve), mindegyikből legalább egy értéket használva. A lefedettséget a következőképp mérjük: a legalább egy értékkel tesztelt ekvivalenciapartíciók

száma osztva az összes azonosított ekvivalenciapartíció számával, százalékban kifejezve. Az ekvivalenciaparticionálás minden tesztszinten alkalmazható.

4.2.2 Határérték-elemzés

A határérték-elemzés (angol rövidítése BVA) az ekvivalenciaparticionálás kiterjesztése, de csak numerikus vagy szekvenciális adatokat tartalmazó, rendezett partíciók esetén használható. A partíció minimum és maximum értékei (vagy első és utolsó értékei) a határértékek (lásd: Beizer 1990).

Példaként, tegyük fel, hogy egy bemeneti mező egyetlen egész értéket fogad el, számbillentyűzettel korlátozzák a bemeneteket, így nem egész értékek megadása nem lehetséges (csak 0-9 értékek vihetők be). Az érvényes tartomány az 1-től 5-ig terjedő zárt intervallum. Ekkor három ekvivalenciapartíció létezik: érvénytelen (túl kicsi érték); érvényes; érvénytelen (túl nagy érték). Az érvényes ekvivalenciapartíció határértékei: 1 és 5. A túl nagy értékeket tartalmazó érvénytelen partíció határértéke: 6. A túl kicsi értékeket tartalmazó ekvivalenciapartícióhoz csak egy határérték létezik, a 0, mivel ennek a partíciónak csak ez az egy eleme van.

A fenti példában határonként két határértéket határozunk meg. A túl kicsi érvénytelen értékek, illetve az érvényes értékek közötti határ a következő tesztértékeket adja: 0 és 1. Az érvényes és a túl nagy érvénytelen értékek közötti határ az 5-öt, illetve a 6-ot adja tesztértékként. Ezen technika néhány változata határonként három határértéket határoz meg: a határ előtti, a határon lévő és a határ utáni értéket. Az előző példában, a hárompontos határértékek használatával az alsó határon lévő tesztértékek a következők: 0, 1 és 2, a felső határon lévő tesztértékek pedig 4, 5 és 6 (lásd: Jorgensen 2014).

Az ekvivalenciapartíciók határain nagyobb valószínűséggel helytelen a viselkedés, mint a partíciók belsejében. Fontos megjegyezni, hogy mind a specifikált, mind a megvalósított határok elmozdulhatnak a szándékolt helyük alá vagy fölé, teljesen figyelmen kívül maradhatnak vagy kiegészülhetnek további, nem kívánt határokkal. A határérték-elemzés és tesztelés majdnem minden ilyen jellegű hibát felfed olyan módon, hogy a szoftvert rákényszeríti, hogy egy másik partíció viselkedését mutassa, mint, amihez a határértéknek tartoznia kellene.

A határérték-elemzés minden tesztszinten alkalmazható. Ezt a technikát általában olyan követelmények tesztelésére alkalmazzák, amelyek számok intervallumainak használatát igénylik (beleértve dátumokat és időpontokat). Egy partíció határérték-lefedettsége a következőképp adódik: a tesztelt határértékek száma osztva az összes azonosított határ tesztértékkel, százalékban kifejezve.

4.2.3 Döntési tábla tesztelés

A döntési táblák jól alkalmazhatók olyan komplex üzleti szabályok rögzítésére, amiket a rendszernek meg kell valósítania. Döntési táblák készítésekor a tesztelő meghatározza a rendszer követelményeit (gyakran bemenetek) és az eredményezett műveleteket (gyakran kimenetek). Ezek alkotják a táblázat sorait, általában a feltételek felül, a műveletek pedig alul helyezkednek el. Minden oszlop egy döntési szabálynak felel meg, ami a feltételek egy olyan egyedi kombinációját definiálja, ami az ehhez a szabályhoz rendelt műveletek végrehajtását eredményezi. A feltételeket és műveleteket általában Boole (igaz vagy hamis), vagy diszkrét (pl. piros, zöld, kék) értékekként ábrázolják, de lehetnek számok vagy tartományok is. Ezek a különböző típusú feltételek és műveletek együtt is megjelenhetnek ugyanabban a táblában.

A döntési táblák általános jelölései a következők:

Feltételek:

- Y: a feltétel igaz (további jelölések: T, 1)
- N: a feltétel hamis (további jelölések: F, 0)
- –: a feltétel értéke nem lényeges (további jelölés: N/A)

Műveletek:

- X: a műveletet végre kell hajtani (további jelölések: Y, T, 1)
- üres: a műveletet nem kell végrehajtani (további jelölések: N, F, 0)

Egy teljes döntési táblának elég oszlopa (tesztesete) van ahhoz, hogy a feltételek minden kombinációját lefedje. A kimenetet nem befolyásoló oszlopok törlésével a tesztesetek száma jelentősen csökkenthető. Például a feltételek lehetetlen kombinációinak eltávolításával. További információért a döntési táblák oszlopainak csökkentéséről lásd: ISTQB-CTAL-TA. Advanced Level Test Analyst Syllabus.

A döntési tábla tesztelés gyakori minimum lefedettségi szabványa szerint legalább egy tesztnek kell lennie döntési szabályonként. Ez tipikusan a feltételek összes kombinációját lefedi. A lefedettséget a következőképp mérjük: a legalább egy teszteset által tesztelt döntési szabályok száma osztva az összes döntési szabály számával, százalékban kifejezve.

A döntési tábla tesztelés ereje abban rejlik, hogy segít a feltételek fontos kombinációinak azonosításában, amelyek különben figyelmen kívül maradnának. Abban is segít, hogy a követelményekben lévő réseket megtalálják. Az összes olyan szituációban alkalmazható, amelyben a szoftver viselkedése a feltételek egy kombinációjától függ, bármely tesztszinten.

4.2.4 Állapotátmenet tesztelés

A komponensek vagy rendszerek különbözőképpen reagálhatnak az eseményekre a jelenlegi állapotok vagy előzmények függvényében (pl. események, amelyek a rendszer elindítása óta történtek). A megelőző események összefoglalhatók az állapotok fogalmának segítségével. Egy állapotátmenet diagram megmutatja a lehetséges szoftverállapotokat, illetve azt, hogy a szoftver hogyan lép be egy állapotba, hogyan lép ki onnan, valamint az állapotok közötti átmeneteket. Az átmenetet egy esemény váltja ki (pl. egy felhasználó beír egy értéket egy mezőbe). Ugyanazon esemény ugyanazon állapotból kettő vagy több különböző átmenetet is eredményezhet. Az állapotváltozás műveletvégzést a szoftverben (pl. egy számítás vagy hibaüzenet kiírása).

Az állapotátmeneti tábla az állapotok közötti összes érvényes és potenciálisan érvénytelen átmenetet, az eseményeket, illetve az érvényes átmenetekhez tartozó végrehajtandó műveleteket ábrázolja. Az állapotátmenet diagramok általában csak az érvényes átmeneteket mutatják és kizárják az érvényteleneket.

A tesztek tervezhetők az állapotok egy tipikus sorozatának lefedésére, minden állapot végrehajtására, minden átmenet végrehajtására, átmenetek meghatározott sorozatának végrehajtására vagy érvénytelen átmenetek tesztelésére.

Az állapotátmenet tesztelést menü alapú alkalmazások tesztelésére használják és a beágyazott szoftverek esetében is széles körben alkalmazott. A technika továbbá alkalmas speciális állapotokat tartalmazó üzleti szcenárió modellezésére vagy képernyőnavigáció tesztelésére. Az állapot egy absztrakt fogalom – jelenthet néhány sor kódot vagy akár egy egész üzleti folyamatot.

A lefedettséget általában a következőképpen határozzuk meg: a tesztelt azonosított állapotok vagy átmenetek száma osztva az összes azonosított állapot vagy átmenet számával, százalékban kifejezve. További információért az állapotátmenet tesztelés lefedettségi kritériumairól, lásd a ISTQB-CTAL-TA Advanced Level Test Analyst Syllabus referenciát.

4.2.5 Használati eset tesztelés

A tesztek származtathatók használati esetekből is, amik a szoftverelemekkel történő kölcsönhatások tervezésének speciális módjai. Egyesítik a szoftverfunkciókra vonatkozó követelményeket. A használati eseteket aktorokhoz (emberi felhasználók, külső hardver, másik komponens vagy rendszer) és tárgyakhoz (a komponens vagy rendszer, amelyre a használati esetet alkalmazzuk) kötjük.

Minden használati eset egy viselkedést határoz meg, amelyet a tárgy hajt végre együttműködésben egy vagy több aktorral (UML 2.5.1 2017). A használati eset az alábbiakkal írható le: kölcsönhatások és tevékenységek, elő- és utófeltételek, illetve adott esetben természetes nyelv. Az aktorok és a tárgyak közötti kölcsönhatások a tárgy állapotváltozását eredményezhetik. A kölcsönhatások grafikusan munkafolyamatok, aktivitás diagramok, vagy üzleti folyamatmodellek segítségével ábrázolhatók.

Egy használati eset részét képezhetik az alap viselkedés lehetséges variációi, beleértve a kivételes viselkedést és a hibakezelést (rendszerválasz és programozásból való helyreállítás, alkalmazási és kommunikációs hibák, pl. hibaüzenet kiváltása). A teszteket a meghatározott viselkedések végrehajtására tervezik (alap, kivételes vagy alternatív és hibakezelő). A lefedettséget a következőképpen mérjük: a tesztelt használati eset viselkedések száma osztva az összes használati eset viselkedés számával, százalékban kifejezve.

További információért a használati eset tesztelés lefedettségi kritériumairól lásd a ISTQB-CTAL-TA Advanced Level Test Analyst Syllabus referenciát.

4.3 Fehérdoboz teszttechnikák

A fehérdoboz tesztelés a teszt tárgyának belső struktúráján alapul. A fehérdoboz teszttechnikák minden szinten használhatók, de az ebben a fejezetben tárgyalt, két kódhoz kapcsolódó technika a leggyakrabban komponens szinten használt. Léteznek haladóbb szintű technikák, amelyeket néhány biztonságkritikus, küldetéskritikus vagy magas integritású környezetben használnak a teljesebb lefedettség elérése érdekében, de ezeket nem tárgyaljuk. További információkért lásd: ISTQB-CTAL-TTA Advanced Level Technical Test Analyst Syllabus.

4.3.1 Utasítástesztelés és -lefedettség

Az utasítástesztelés a kódban lévő potenciálisan végrehajtható utasításokat hajtja végre. A lefedettséget a teszt tárgyában a tesztek által végrehajtott utasítások és az összes végrehajtható utasítás számának hányadosaként értelmezzük, százalékban kifejezve.

4.3.2 Döntési tesztelés és lefedettség

A döntési tesztelés a kódban lévő döntéseket hajtja végre és a döntések kimeneteként végrehajtott kódot teszteli. Ehhez a tesztesetek az egy döntési pontból kiinduló vezérlési folyamokat követik (pl. egy IF utasítás esetén egyet a logikailag IGAZ és egyet a logikailag HAMIS kimenetre; egy CASE utasítás esetén minden lehetséges kimenethez szükségesek tesztesetek, beleértve az alapértelmezett kimenetet is).

A lefedettség a következőképp adódik: a teszt tárgyában tesztek által végrehajtott döntési kimenetek osztva az összes döntési kimenet számával, százalékban kifejezve

4.3.3 Az utasítástesztelés és a döntési tesztelés értéke

Ha az utasítás lefedettség elérte a 100%-ot, akkor ez biztosítja, hogy minden végrehajtható utasítás legalább egyszer le van tesztelve, de nem biztosítja, hogy minden döntési logika tesztelve van. E tantervben tárgyalt két fehérdoboz technika közül, az utasítástesztelés kisebb lefedettségi értéket érhető el, mint a döntési teszteléssel.

Ha a döntési lefedettség elérte a 100%-ot, akkor minden döntési kimenetet végrehajtunk, beleértve az igaz és a hamis kimenetek tesztelését is abban az esetben is, ha nincs explicit hamis utasítás (pl. egy IF utasítás ELSE nélkül). Az utasítás lefedettség segít a kódban lévő olyan hibák megtalálásában, amelyeket más tesztek nem hajtottak végre. A döntési lefedettség segít megtalálni a kódban azokat a hibákat, ahol más tesztek nem hajtották végre mind az igaz, mind a hamis ágat.

100%-os döntési lefedettség 100%-os utasítás lefedettséget garantál (ez fordítva nem igaz).

4.4 Tapasztalat alapú teszttechnikák

Tapasztalat alapú teszttechnika alkalmazásakor a tesztesetek a tesztelő képességeiből és intuíciójából, valamint a hasonló alkalmazásokkal és technológiákkal kapcsolatos tapasztalataiból származnak. Ezek a technikák segíthetnek azon tesztek meghatározásában, amelyeken más szisztematikus technikákkal nem egyszerű azonosítani. A tesztelő megközelítésmódjától és tapasztalatától függően, ezek a technikák egy széles skálán mozgó lefedettségi- és hatékonysági értékeket érhetnek el. A lefedettséget nehéz megállapítani és ezen technikákkal lehet, hogy nem is mérhető.

A gyakran használt tapasztalat alapú technikákat a következő fejezetekben tárgyaljuk.

4.4.1 Hibasejtés

A hibasejtés egy technika, melynek alkalmazása során megjósolják a hibák (defect), emberi eredetű hibák (error) és a meghibásodások (failure) előfordulását a tesztelő tudása alapján, beleértve:

- Hogyan működött az alkalmazás a múltban
- Milyen emberi eredeti hibák fordulnak elő várhatóan
- Más alkalmazásokban jelentkezett meghibásodások

A hibasejtés technika módszeres megközelítése, hogy létrehozunk egy listát a lehetséges emberi eredetű hibákról, hibákról, meghibásodásokról és olyan teszteket tervezünk, amelyek felfedik ezeket a meghibásodásokat és az őket okozó hibákat. Ezen listák készülhetnek tapasztalat alapján, hibákról és meghibásodásokról rendelkezésre álló adatok alapján, valamint a szoftver meghibásodásával kapcsolatos közös ismeretanyag alapján.

4.4.2 Felderítő tesztelés

A felderítő tesztelés során informális (előre nem definiált) teszteket terveznek, hajtanak végre, naplóznak és értékelnek ki a teszt végrehajtása közben dinamikusan. A teszteredményeket arra használják, hogy további ismereteket szerezzenek a komponensről vagy a rendszerről, illetve, hogy teszteket hozzanak létre a további tesztelést igénylő területekhez.

A felderítő tesztelést néha munkamenet alapú tesztelést használva hajtják végre a tevékenység strukturálása érdekében. A munkamenet alapú tesztelés során a felderítő tesztelést egy meghatározott időkereten belül végzik és a tesztelő egy tesztcélokat tartalmazó tesztvázlatot használ, ami irányvonalat ad a teszteléshez. A követett lépések és a felfedezések dokumentálására tesztszakasz jegyzőkönyv használható.

A felderítő tesztelés akkor a leghasznosabb, amikor kevés vagy nem megfelelő specifikáció áll csak rendelkezésre vagy szoros a határidő. A felderítő tesztelés más, formálisabb teszttechnikák kiegészítéseként is hasznos.

A felderítő tesztelés szorosan összefügg a reaktív tesztstratégiákkal (lásd: 5.2.2. fejezet). A felderítő tesztelés magában foglalhatja a feketedoboz-, fehérdoboz- és tapasztalat alapú technikák használatát.

4.4.3 Ellenőrzőlista alapú tesztelés

Ellenőrzőlista alapú tesztelés során a tesztelők egy ellenőrzőlistában található tesztfeltételek lefedésére tervezik, valósítják meg és hajtják végre a teszteket. Az elemzés részeként a tesztelők új ellenőrzőlistákat készítenek vagy meglévőket bővítenek ki, de módosítás nélkül is használhatják őket. Ilyen ellenőrzőlisták készíthetők tapasztalat alapján, a felhasználó számára fontos szempontokról meglévő tudás alapján vagy a szoftver meghibásodásának okairól és milyenségéről alkotott kép alapján.

Ellenőrzőlisták különböző teszttípusok támogatására készíthetők, beleértve a funkcionális, illetve a nemfunkcionális tesztelést is. Részletes tesztesetek hiányában, az ellenőrzőlista alapú tesztelés irányvonalat

és egy bizonyos fokú következetességet nyújt. Mivel ezek magasszintű listák, a tényleges tesztelésben előfordulhat némi változékonyság, potenciálisan nagyobb lefedettséget, de kisebb mértékű megismételhetőséget eredményezve.

5. Tesztmenedzsment 225 perc

Kulcsszavak

belépési feltétel, hibajelentés, hibamenedzsment, kilépési feltétel, kockázat alapú tesztelés, kockázat, kockázati szint, konfigurációmenedzsment, összefoglaló tesztjelentés, projektkockázat, termékkockázat, teszt megközelítés, tesztbecslés, tesztelő, tesztfelügyelet, tesztirányítás, tesztstátuszjelentés, tesztstratégia, teszterv, teszttervezés, tesztvezető

Tanulási célok a Tesztmenedzsment fejezethez

5.1. Tesztelő szervezet

AK-5.1.1 (K2) Magyarázza meg a független tesztelés előnyeit és hátrányait

AK-5.1.2 (K1) Azonosítsa a tesztvezető és a tesztelő feladatait

5.2. Teszttervezés és becslés

AK-5.2.1 (K2) Foglalja össze a teszterv céljait és tartalmát

AK-5.2.2 (K2) Különböztesse meg az eltérő tesztstratégiákat

AK-5.2.3 (K2) Adjon példákat lehetséges be- és kilépési feltételekre

AK-5.2.4 (K3) Alkalmazza tudását a sorrendiséggel, illetve a technikai és logikai függőségekkel kapcsolatosan tesztesetek egy adott halmazának elvégzéséhez

AK-5.2.5 (K1) Határozza meg a teszteléshez kapcsolódó ráfordításokat befolyásoló tényezőket

AK-5.2.6 (K2) Határozza meg a két becslési technika, a metrika alapú és a szakértő alapú technika közötti különbségeket

5.3. Tesztmonitorozás és kontroll

AK-5.3.1 (K1) Idézze fel a teszteléshez használt metrikákat

AK-5.3.2 (K2) Foglalja össze a tesztjelentések célját, tartalmát és célközönségét

5.4. Konfigurációmenedzsment

AK-5.4.1 (K2) Összegezze, hogyan támogatja a konfigurációmenedzsment a tesztelést

5.5. Kockázat és tesztelés

AK-5.5.1 (K1) Határozza meg a kockázat szintjét a valószínűség és a hatás használatával

AK-5.5.2 (K2) Különböztesse meg a projekt és termékkockázatokat

AK-5.5.3 (K2) Mutassa be példák használatával, hogy a termékkockázat-elemzés milyen hatással van a tesztelés mélységére és hatáskörére

5.6. Hibamenedzsment

AK-5.6.1 (K3) Írjon hibajelentést egy teszt során megfigyelt meghibásodásról

5.1 Tesztelő szervezet

5.1.1 Független tesztelés

Teszteléssel kapcsolatos feladatokat elláthat egy speciális tesztelői szerepkörben tevékenykedő személy, vagy egy bármely más szerepkörben tevékenykedő személy (például egy ügyfél). Köszönhetően a fejlesztő és a tesztelő eltérő felfogásának, a függetlenség egy bizonyos szintig növelheti a tesztelő hatékonyságát a hibák megtalálásában (lásd 1.5-ös fejezet). Ennek ellenére a függetlenség nem helyettesíti a rendszerismeretet, így saját kódjukban a fejlesztők számos hibát hatékonyabban találnak meg.

A tesztelési függetlenség fokát az alábbiak szerint állapíthatjuk meg (alacsonytól a magas függetlenségi szintig):

- Nincs független tesztelő; a tesztelés egyetlen formája, hogy a fejlesztők tesztelik saját kódjukat
- Független fejlesztők vagy tesztelők a fejlesztői vagy projektcsoportban; ilyen például ha a fejlesztők egymás termékét tesztelik
- Független tesztcsapat vagy csoport a szervezeten belül, akik a projektvezetőnek vagy a felsővezetésnek jelentenek
- Független tesztelők a szervezeten belül vagy a felhasználói közösségből; specializálódhatnak a különböző teszttípusokra, mint például a használhatóság, biztonság, teljesítmény, szabályok/szabályzatok vagy hordozhatóság
- Szervezeten kívüli független tesztelők, akik vagy a helyszínen (insource) vagy kihelyezetten (outsource) dolgoznak.

A legtöbb projekt esetében célszerű több tesztszintet alkalmazni, melyek közül néhányat független tesztelők végeznek. A fejlesztők is részt vesznek a folyamatban, különösen az alsóbb szinteken, így kontrollálva a saját munkájuk minőségét.

A szoftverfejlesztési életciklusmodell hatással van a tesztelési függetlenség kialakításának módjára. Például az Agilis fejlesztésben a tesztelők a fejlesztői csapat részét képezik. Néhány Agilis módszert alkalmazó szervezetnél a tesztelőket egy nagyobb, független tesztcsapat részének tekintik. Sőt, az ilyen szervezetekben a terméktulajdonos (product owner) is végezhet elfogadási tesztelést, hogy validálja a felhasználói történeteket (user story) az egyes iterációk végén.

A tesztelési függetlenség lehetséges előnyei:

- A független tesztelők valószínű, hogy más jellegű hibákat vesznek észre, mint a fejlesztők
- Egy független tesztelő ellenőrizheti, kétségbe vonhatja, megcáfolhatja az érdekeltek rendszer-specifikáció és implementáció közben kialakult feltevéseit
- A külső, független tesztelők pártatlanul és objektíven jelenthetnek a rendszerről, az őket alkalmazó cégen belüli (politikai) nyomástól mentesen

A tesztelési függetlenség lehetséges hátrányai:

- Elszigetelődés a fejlesztő csapattól, ami további hátrányokhoz vezethet: az együttműködés hiánya, a késleltetett visszajelzés a fejlesztői csapat számára, ellenséges kapcsolat a fejlesztői csapattal
- A fejlesztők elvesztik felelősségérzetüket a termék minősége iránt
- A független tesztelőkre szűk keresztmetszetként tekinthetnek

- Fontos információk (például a tesztelés tárgyával kapcsolatban) nem jutnak el a független tesztelőkhöz

Számos szervezet a hátrányok elkerülésével meg tudja valósítani a független tesztelés előnyeit.

5.1.2 A tesztmenedzser és a tesztelő feladatai

Jelen tananyag két tesztelői munkakört tárgyal, a tesztmenedzserét és a tesztelőét. A két feladatkörhöz tartozó tevékenységek és feladatok a projekt és a termék jellemzőitől, a beosztásokat ellátó személyek képességeitől és a szervezettől függenek.

A tesztmenedzser viseli a felelősséget a tesztfolyamatokért és a teszttevékenységek sikeres vezetéséért. A tesztmenedzser szerepét betöltheti egy szakavatott tesztmenedzser, egy projektmenedzser, egy fejlesztési menedzser, vagy egy minőségbiztosítási menedzser. Nagyobb projekteknél vagy szervezeteknél számos tesztelői csoport jelenthet egy tesztmenedzsernek, teszt tanácsadónak vagy teszt koordinátornak, míg minden csapatot egy tesztvezető vagy vezető tesztelő irányít.

Tipikus tesztmenedzseri feladatok lehetnek:

- A szervezeti tesztstratégia és teszterv kialakítása vagy felülvizsgálata
- A tesztelési tevékenységek tervezése a körülmények figyelembevételével és a tesztcélok, kockázatok megértése. Ide tartozhat a tesztelési megközelítések kiválasztása, a tesztelés időtartamának, ráfordításainak és költségeinek becslése, erőforrások beszerzése, tesztszintek és ciklusok meghatározása, a hibamenedzsment tervezése
- Teszterv(ek) készítése és frissítése
- Tesztterv koordinációja projekt menedzserekkel, termék tulajdonosokkal és másokkal
- A tesztelési perspektívák megosztása más projekttevékenységekkel, mint például az integráció tervezés
- A tesztek elemzésének, előkészítésének, kivitelezésének és végrehajtásának kezdeményezése, a tesztfolyamatok és eredmények monitorozása, a kilépési feltételek (vagy a „kész” definíció) állapotának ellenőrzése és a tesztelés elvégzésének segítése
- Az összegyűjtött információk alapján a tesztstátuszjelentés és az összefoglaló jelentés előkészítése és átadása
- A tervezés átalakítása a teszteredmények és a tesztelés előrehaladása alapján (amit gyakran tesztstátuszjelentésekben, vagy a projekt során már elvégzett tesztfeladatok összefoglaló jelentésében dokumentálnak), a tesztirányításhoz szükséges lépések megtétele.
- A hibamenedzsment rendszer kialakításának támogatása és a tesztver megfelelő konfiguráció menedzsmentjének létrehozása
- Megfelelő metrikák bevezetése a tesztelés állapotának mérésére, valamint a tesztelés és a termék minőségének értékelésére
- A tesztelési folyamatot segítő eszközök kiválasztásának és bevezetésének támogatása, javaslat az eszközkiválasztás költségvetésére (ide számítva megvásárlást és támogatást), bevezető projektek idő- és ráfordításszükségének megállapítása, az alkalmazás használatának folyamatos támogatása
- Döntéshozatal a tesztkörnyezet(ek) kialakításáról
- A tesztelők, a teszt csapat és a tesztelői hivatás segítése és támogatása a szervezeten belül

- A tesztelők képességeinek és karrierlehetőségeinek javítása (például oktatási tervek, teljesítményértékelés, coaching segítségével)

A szoftverfejlesztési életciklustól függően eltérhet a tesztmenedzseri szerep megvalósítása. Például az Agilis fejlesztésben néhány fentebb említett feladatot az Agilis csoport lát el, elsősorban azokat, melyek a csapat – gyakran csoporton belüli tesztelő – által végzett napi tesztelési feladatokhoz kötődnek. Néhány feladat több csoportot fog össze, sőt átívelhet az egész szervezeten is. A személyzeti kérdésekkel a fejlesztői csoporton kívüli tesztmenedzserek foglalkozhatnak, akiket gyakran teszt támogatónak hívunk (test coach). További információ található a tesztelési folyamat menedzseléséről R. Back Managing the Testing Process c. könyvében (Black, 2009).

Tipikus tesztelői feladatok lehetnek:

- Teszttervek felülvizsgálata és részvétel a kidolgozásukban
- Követelmények, felhasználói történetek és elfogadási kritériumok, specifikációk és tesztelhetőségi modellek elemzése, felülvizsgálata és kiértékelése
- Tesztfeltételek azonosítása és dokumentálása, nyomonkövethetőség biztosítása a tesztesetek, tesztfeltételek és a tesztbázis között
- Tesztkörnyezet(ek) tervezése, kialakítása és ellenőrzése, gyakran együttműködésben a rendszer adminisztrátorokkal és a hálózat menedzsmenttel
- Tesztesetek és eljárások tervezése és megvalósítása
- Tesztadatok előkészítése, felvétele
- Részletes teszt ütemezés elkészítése
- Tesztek végrehajtása, az eredmények értékelése, az elvárt eredményektől való eltérések dokumentálása
- Megfelelő teszteszközök használata a tesztfolyamat támogatásához
- Szükség esetén a tesztek automatizálása (ebben támogatást nyújthat egy fejlesztő vagy egy tesztautomatizálási szakértő).
- Nemfunkcionális jellemzők értékelése, mint teljesítményhatékonyság, megbízhatóság, használhatóság, biztonság, kompatibilitás és hordozhatóság
- Mások által kifejlesztett tesztek felülvizsgálata

A tesztelemzéssel, műszaki teszttervezéssel, speciális teszttípusokkal, illetve tesztautomatizálással foglalkozó személyek általában szakértők a saját területükön. A termékkel és a projekttel kapcsolatos kockázatoktól, valamint a kiválasztott szoftverfejlesztési életciklusmodelltől függően különböző személyek vehetik át a tesztelői feladatokat a különböző tesztelési szinteken. Például a komponens- és komponens integrációs szintű tesztelők szerepét gyakran a fejlesztők töltik be. Az elfogadási tesztek szintjén a tesztelők szerepét betölthetik üzleti szakértők, a témában járatos szakértők és felhasználók. A rendszerteszt és a rendszer integrációs teszt szintjén a tesztelők szerepét betöltheti egy független tesztelői csoport. Az üzemeltetési elfogadási tesztelők pedig gyakran az üzemeltetők és/vagy a rendszer adminisztrátorok lehetnek.

5.2 Teszttervezés és becslés

5.2.1 A teszterv célja és tartalma

A teszterv a fejlesztés és üzemeltetés során végzett tesztelési tevékenységeket vázolja fel. A tervezést befolyásolja a szervezet tesztstratégiája és tesztelés irányelvei, a fejlesztési életciklus és az alkalmazott módszerek (lásd 2.1-es fejezet), valamint a teszt tárgya, céljai, kockázatok, megkötések, kritikusság, tesztelhetőség és az elérhető erőforrások.

Minél előrehaladottabb fázisban van a projekt és a teszttervezés, annál több információ áll rendelkezésre és annál részletesebb lehet a terv. A teszttervezés folyamatos tevékenység, melyet az életciklus minden fázisában el kell végezni. (Ne felejtsük el, hogy a termék életciklusa nem mindig esik egybe a projekt hatáskörével, mivel magába foglalhatja az üzemeltetés időszakát is.) A teszttevékenységekből kapott visszajelzések alapján felmérhető a kockázatok változása, és ennek megfelelően alakítható a tervezés. A tervezést dokumentálhatják a fő teszttervben és az egyes tesztszintekhez tartozó teszttervekben, mint például a rendszertesztelés, elfogadási tesztelés, vagy a különböző teszttípusokhoz tartozó teszttervekben, mint például a használhatósági tesztelés és a teljesítménytesztelés. A teszttervezési tevékenységek részét képezheti és dokumentálható a teszttervben:

- A tesztelés tárgyának, a céloknak és a kockázatoknak a meghatározása
- A tesztelés általános megközelítésének definiálása
- A teszttevékenységek koordinálása és beépítése a szoftver életciklusába
- Döntéshozatal arról, hogy mit tesztelünk, mely személyek, illetve erőforrások szükségesek a teszttevékenységek végrehajtásához, és hogy ezen tevékenységeket hogyan kell végrehajtani
- A tesztelemzési, tervezési, megvalósítási, végrehajtási és értékelési tevékenységek ütemezése adott időpontra (például a szekvenciális fejlesztésben) vagy az egyes iterációkhoz kötődően (például az iteratív fejlesztésben)
- Metrikák kiválasztása a tesztfelügyelethez és irányításhoz
- Teszttevékenységek költségvetésének meghatározása
- A tesztdokumentáció részletességének és struktúrájának meghatározása (például sablonok vagy mintadokumentumok rendelkezésre bocsátása).

A teszttervek tartalma változó lehet, és túlnőhet a fent megadott témákon. Az ISO szabványban (ISO/IEC/IEEE 29119-3) megtalálható egy minta teszt terv struktúra és egy minta teszt terv.

5.2.2 Tesztstratégia és tesztelési megközelítés

A tesztstratégia általános módon írja le a tesztelési folyamatot, általában egy termék vagy szervezet szintjén. A tesztstratégiák általános típusai a következők:

- **Analitikus**: az ilyen típusú tesztstratégia egy bizonyos tényező elemzésén alapul (például követelmény vagy kockázat). A kockázat alapú tesztelés az analitikus megközelítés egy olyan példája, ahol a teszteket a kockázati szintek alapján tervezik és priorizálják.
- **Modell alapú**: Az ilyen típusú tesztstratégiában a teszteket néhány modell alapján tervezik, melyek a termék valamely szükséges jellemzőjére épülnek, mint például egy funkció, egy üzleti folyamat, a belső szerkezet, vagy egy nemfunkcionális jellemző (például megbízhatóság). Példaként szolgálhatnak az ilyen modellekre az üzleti folyamat modellek, állapot modellek és megbízhatósági növekedési modellek.

- **Módszeres**: Ez a fajta tesztstratégia szisztematikusan használja a tesztek és tesztfeltételek néhány előre meghatározott halmazát, mint például a gyakori vagy valószínű hibatípusok csoportjai, a fontos minőségi jellemzők listája, vagy a vállalati szintű megjelenési és érzelmi elvárások egy mobilalkalmazáshoz vagy weboldalhoz kapcsolódóan.
- **Folyamat szerinti** (vagy szabvány alapú): Az ilyen típusú tesztstratégia a tesztek olyan külső szabályokon és szabványokon alapuló elemzését, tervezését és megvalósítását jelenti, mint például az ágazatspecifikus szabványok, a folyamatdokumentáció, a tesztbázis szigorú megállapítása és használata, vagy bármely egyéb eljárás vagy szabvány, amelyet a szervezet maga határozott meg, vagy a szervezetre határoztak meg.
- **Irányított** (vagy konzultatív): Az ilyen típusú tesztstratégiát elsősorban az érdekelt felektől, az üzleti terület szakértőitől vagy a technológiai szakértőktől kapott tanácsadás, útmutatás vagy utasítás követése jellemzi – akik lehetnek a teszt csapaton, vagy akár a szervezeten kívüli személyek is.
- **Regresszió-kerülő**: Az ilyen típusú tesztstratégia hátterében az a vágy áll, hogy elkerüljük a meglévő képességek romlását. Ez a tesztstratégia magában foglalja a meglévő tesztverek (különösen a tesztesetek és tesztadatok) újra felhasználását, a regressziós tesztek széles körű automatizálását és a standard tesztkészleteket.
- **Reaktív**: Az előző stratégiákkal ellentétben az ilyen típusú tesztstratégiában a tesztelés az előre tervezettség helyett a komponens vagy a rendszer lényegére és a teszt végrehajtása során bekövetkezett eseményekre reagál. A teszteket úgy tervezik és hajtják végre, hogy azok azonnal, az előzetes vizsgálati eredményekből nyert tudás alapján futtathatók legyenek. A felderítő tesztelés (exploratory testing) gyakran alkalmazott technika a reaktív stratégiákban.

Általában a fenti típusok ötvözésével hozhatunk létre megfelelő tesztstratégiát. Például a kockázat alapú tesztelés (analitikus stratégia) kombinálható a felderítő teszteléssel (reaktív stratégia); kiegészítik egymást, és hatékonyabb tesztelést érhetünk el együtt alkalmazva őket.

Míg a tesztstratégia általános leírást ad a tesztelési folyamatról, a tesztelési megközelítés egy adott projekthez vagy kiadáshoz alakítja a tesztstratégiát. A tesztelési megközelítés a kiindulópontja a teszt technikák, a tesztelési szintek és a teszttípusok kiválasztásának, valamint a belépési és kilépési kritériumok (illetve az „előkészített” és a „kész” definíciójának) meghatározásának. A stratégia testre szabása olyan tényezőkön alapul, mint a projekt összetettsége és céljai, a fejlesztendő termék típusa és a termékkockázat elemzés. A kiválasztott megközelítés kontextusfüggő és figyelembe vehet olyan tényezőket is, mint a kockázatok, biztonság, rendelkezésre álló erőforrások és készségek, technológia, a rendszer jellege (például saját fejlesztésű vagy COTS), tesztcélok és szabályok.

5.2.3 Belépési és kilépési feltételek (a „kész” és az „előkészített” fogalma)

A szoftver és a tesztelés minőségének hatékony ellenőrzése érdekében tanácsos olyan kritériumokkal rendelkeznünk, amelyek meghatározzák, hogy mikor kell elindítani és mikor befejezni egy adott tesztelési tevékenységet. A belépési feltételek (melyeket az Agilis fejlesztésben tipikusan az „előkészített” definíciójának nevezünk) meghatározzák az adott teszttevékenység megkezdésének előfeltételeit. Ha a belépési kritériumok nem teljesülnek, valószínű, hogy a tevékenység nehezebb, időigényesebb, költségesebb és kockázatosabb lesz. A kilépési feltételek (melyeket az Agilis fejlesztésben tipikusan a „kész” definíciójának nevezünk) meghatározzák, hogy milyen feltételeket kell elérni annak érdekében, hogy egy tesztszintet vagy tesztkészletet készneknek nyilvánítsunk.

Minden tesztszintre és teszttípusra meg kell határozni a belépési és kilépési kritériumokat, melyek a tesztelési célok függvényében eltérők lesznek.

A tipikus belépési kritériumok a következők:

- A tesztelhető követelmények, felhasználói történetek és/vagy modellek (például modellalapú tesztelési stratégia esetén) elérhetősége
- Olyan tesztelemek rendelkezésre állása, amelyek megfeleltek az előzetes tesztszintek kilépési feltételeinek
- A tesztkörnyezet elérhetősége
- A szükséges teszteszközök rendelkezésre állása
- A tesztadatok és egyéb szükséges erőforrások rendelkezésre állása

A tipikus kilépési kritériumok a következők:

- A tervezett teszteket végrehajtottuk
- A lefedettség meghatározott szintjét elértük (például a követelmények, felhasználói történetek, elfogadási kritériumok, kockázatok, kódlefedettség)
- A megoldatlan hibák száma egy meghatározott határon belül van
- A becsült fennmaradó hibák száma kellően alacsony
- A becsült megbízhatósági, teljesítményhatékonysági, használhatósági, biztonsági szintek és más releváns minőségi jellemzők kielégítők

Még a kilépési kritériumok teljesítése nélkül is gyakori, hogy a vizsgálati tevékenységeket megrövidíti a költségkeret kimerülése, a tervezett idő lejárta és/vagy a termék piaci forgalomba hozatalának kényszere. Ilyen körülmények között elfogadható lehet a tesztelés befejezése, amennyiben a projekt érdekeltjei és a üzleti felek felülvizsgálták és elfogadták a kockázatot, hogy további vizsgálat nélkül kibocsássák a terméket.

5.2.4 Tesztvégrehajtási ütemterv

Miután a különböző tesztesetek és teszteljárások elkészültek (néhány teszteljárás lehetőség szerint automatizálva) és tesztcsomagokba szervezték őket, a tesztcsomagok egy tesztvégrehajtási ütemtervbe rendezhetők, amely meghatározza a futtatásuk sorrendjét. A tesztvégrehajtási ütemtervének figyelembe kell vennie olyan tényezőket, mint a prioritások, a függőségek, az ellenőrző tesztek, a regressziós tesztek és a tesztek végrehajtásának leghatékonyabb sorrendje.

Ideális esetben a teszteseteket a prioritási szintjük szerint rendezzük sorba, általában a legmagasabb prioritású teszteseteket hajtjuk végre először. Előfordulhat azonban, hogy ez a gyakorlat nem működik, ha a tesztesetek vagy a vizsgált funkciók függőségekkel rendelkeznek. Ha a magasabb prioritású teszteset egy alacsonyabb prioritású tesztesettől függ, akkor az alacsonyabb prioritású tesztesetet kell először végrehajtani. Hasonlóképpen, ha a tesztesetek között függőségek vannak, azokat megfelelően kell sorba rendezni függetlenül a relatív prioritásuktól. Az ellenőrző és regressziós teszteket is priorizálni kell, melynek alapja a gyors visszajelzés szükségessége, de függőségek itt is lehetnek.

Bizonyos esetekben különböző tesztsorozatok lehetségesek, amelyek különböző hatékonysági szinttel rendelkeznek. Ilyen esetekben kompromisszumot kell kötni a teszt végrehajtásának hatékonysága és a prioritások betartása között.

5.2.5 A teszteléshez szükséges ráfordításokat befolyásoló tényezők

A teszteléshez szükséges ráfordítások becslése magában foglalja a teszthez kapcsolódó munka mennyiségének előrejelzését, amely ahhoz szükséges, hogy elérjük egy adott projekt, a kiadás vagy az iteráció tesztelésének célkitűzéseit. A tesztelési ráfordítást befolyásoló tényezők közé tartozhatnak a termék

jellemzői, a fejlesztési folyamat jellemzői, az emberek jellemzői és a teszteredmények, amint az alábbiakban látható.

A termék jellemzői

- A termékkel kapcsolatos kockázatok
- A tesztbázis minősége
- A termék mérete
- A termék területének összetettsége
- A minőségi jellemzőkre vonatkozó követelmények (pl. biztonság, megbízhatóság)
- A tesztdokumentáció részletességének mértéke
- A jogi és szabályozási megfelelés követelményei

Fejlesztési folyamat jellemzői

- A szervezet stabilitása és érettsége
- A használt fejlesztési modell
- A teszt megközelítés
- Az alkalmazott eszközök
- A tesztelési folyamat
- Időbeli korlátozások

Az emberek jellemzői

- Az érintettek készségei és tapasztalatai, különösen hasonló projektekkel és termékekkel kapcsolatosan (pl. doménismeret)
- Csapat összetartás és vezetés

Vizsgálati eredmények

- A talált hibák száma és súlyossága
- A szükséges átdolgozások mennyisége

5.2.6 Tesztbecslési technikák

Számos becslési módszer létezik a megfelelő teszteléshez szükséges erőfeszítések meghatározásához. A leggyakrabban használt technikák közül kettő a következő:

- A metrikákon alapuló technika: korábbi hasonló projektek mérőszámain vagy tipikus értékeken alapul a tesztelési ráfordítás becslése
- A szakértő alapú technika: a tesztelési feladatok felelőseinek vagy szakértők tapasztalatán alapul a tesztelési ráfordítás becslése

Például az Agilis fejlesztés esetében jó példa a metrikákon alapuló megközelítésre a kivitelezési görbe (burndown chart), mely a fennmaradó ráfordításokat rögzíti és mutatja be, és melynek segítségével később a csapat sebességéhez igazítják a munka mennyiségét, melyet a csapat a következő iteráció során elvégezhet. A szakértői alapú megközelítésre példa a tervezési póker (planning poker), ahol a csapat tagjai

saját tapasztalataik alapján megbecsülik, hogy milyen ráfordítások szükségesek annak érdekében, hogy egy funkciót elkészítsenek (ISTQB-CTFL-AT Foundation Level Agile Tester Syllabus).

A szekvenciális projektek esetében a hibaeltávolító modellek szolgálhatnak példaként a metrikákon alapuló megközelítésre, ahol rögzítik és jelentik a hibákat és az eltávolításukra fordított időt. Ez hasonló természetű későbbi projektnél a becslés alapjául szolgálhat. Ezzel szemben a széleskörű Delphi eljárás (Wideband Delphi) esetében, mely egy példa a szakértői technikára, egy szakértői csoport a tapasztalatai alapján készíti a becsléseket (ISTQB-CTAL-TM Advanced Level Test Manager Syllabus).

5.3 Tesztfelügyelet és -irányítás

A tesztfelügyelet célja, hogy információt gyűjtsön, visszajelzést adjon és biztosítsa a teszttevékenységek átláthatóságát. Az ellenőrzendő információkat gyűjthetik manuálisan vagy automatizáltan, és használhatják a tesztelés előrehaladásának értékelésére, valamint a kilépési feltételek, vagy Agilis projekt „kész” definíciójával kapcsolatos tesztelési feladatok teljesülésének megítélésére, mint például a termékkockázatok, a követelmények vagy az elfogadási kritériumok lefedettségére vonatkozó célok teljesítése. A tesztirányítás olyan támogató vagy javító intézkedéseket foglal magába, melyeket az összegyűjtött és (valószínűleg) jelentett információk és metrikák eredményeként hajtunk végre. A műveletek kiterjedhetnek minden teszttevékenységre, és befolyásolhatják a szoftver egyéb életciklus-tevékenységeit is.

Példák a tesztirányítási műveletekre:

- A tesztek újra-priorizálása egy azonosított kockázat bekövetkezése esetén (például késedelem a szoftver elkészülésében)
- A tesztelési ütemterv módosítása a tesztkörnyezet vagy más erőforrások rendelkezésre állása vagy elérhetetlensége miatt
- Egy átdolgozás miatt a tesztelem újraértékelése a belépési vagy kilépési kritériumnak való megfelelés szempontjából

5.3.1 A tesztelésben használt metrikák

A tesztelési tevékenységek során és annak végén gyűjthetünk különböző mérőszámokat, metrikákat, hogy segítségükkel értékeljük például:

- A haladást a tervezett ütemtervhez és a költségvetéshez képest
- A tesztelés tárgyának aktuális minőségét
- A tesztelési megközelítés megfelelőségét
- A tesztelési tevékenységek hatékonyságát a célkitűzések tekintetében.

Gyakori tesztmetrikák:

- A tesztesetek előkészítésében végzett tervezett munka százalékosan (azaz hány százaléka készült el a tervezett teszteseteknek).
- A tesztkörnyezet előkészítésében végzett tervezett munka százalékosan.
- Teszteset végrehajtás (pl. futtatott/nem futtatott tesztesetek száma, sikeres/sikertelen tesztesetek vagy tesztfeltételek száma).
- Információ a hibákról (pl. hibasűrűség, megtalált és javított hibák, meghibásodási ráta, ellenőrző tesztelés eredményei).

- A követelmények, a felhasználói történetek, az elfogadási kritériumok, a kockázatok vagy a kód tesztlefedettsége.
- A feladatok befejezettsége, az erőforrás-allokáció és használat, valamint a ráfordítások.
- A tesztelés költségei, összevetve a következő hiba megtalálásának nyereségével vagy a következő teszt futtatásának nyereségével.

5.3.2 A tesztjelentés célja, tartalma és célközönsége

A tesztjelentés célja, hogy összefoglalja és átadja a tesztelési tevékenységre vonatkozó információkat a tesztelési tevékenység (például egy tesztszint) során és annak végén. A vizsgálati tevékenység során elkészített tesztjelentést más néven tesztelőrehaladási jelentésnek, míg a tesztelési tevékenység végén elkészített tesztjelentést összefoglaló tesztjelentésnek is nevezik.

A tesztfelügyelet és -irányítás során a tesztmenedzser rendszeresen készít tesztelőrehaladási jelentéseket az érdekeltek számára. A tesztelőrehaladási jelentések és az összefoglaló tesztjelentések általános tartalma mellett a tesztelőrehaladási jelentések tipikusan az alábbiakat tartalmazhatják:

- A teszttevékenységek állapota és előrehaladása a teszttervhez képest
- A haladást gátló tényezők
- A következő jelentési időszakra tervezett tesztelés
- A tesztobjektum minősége

Amikor a kilépési feltételek teljesülnek, a tesztmenedzser elkészíti az összefoglaló tesztjelentést. Ez tartalmazza az elvégzett tesztek összefoglalását, a legfrissebb tesztelőrehaladási jelentés és egyéb releváns információk alapján.

A tipikus összefoglaló tesztjelentések az alábbiakat tartalmazhatják:

- Az elvégzett tesztelés összefoglalása
- Tájékoztatás a tesztelési időszak alatt történtekről
- Eltérések a tervtől, beleértve a tesztelési tevékenységek ütemtervét, időtartamát vagy a ráfordításokat
- A tesztelés és a termékminőség állapota a kilépési kritériumok vagy a „kész” definíciójának tekintetében
- Azok a tényezők, amelyek blokkolták vagy továbbra is blokkolják a haladást
- A hibák, a tesztesetek, a teszt lefedettség, a tevékenység előrehaladása és az erőforrás-fogyasztás mérőszámai (például az 5.3.1. pontban leírtak szerint)
- A fennmaradó kockázatok (lásd 5.5. pont)
- Újrahasznosítható tesztelési munkatermékek

A tesztjelentés tartalma a projekttől, a szervezeti követelményektől és a szoftverfejlesztési életciklustól függően változik. Például egy számos érdekelt féllel rendelkező összetett projekt vagy egy irányított projekt részletesebb és szigorúbb jelentést igényelhet, mint a gyors szoftverfrissítés. Egy másik példa, hogy az Agilis fejlesztés során a tesztelőrehaladási jelentést beépíthetjük a feladattáblákba, a hibajelentésekbe és a kivitelezési görbékbe, amelyeket a napi stand-up találkozó során megvitathatunk (lásd az ISTQB-CTFL-AT Foundation Level Agile Tester Extension Syllabus referenciát).

A tesztjelentéseknek a projekt kontextusán alapuló testre szabása mellett a tesztjelentéseket a jelentés közönségének megfelelően kell kialakítani. A technikai közönségnek vagy a tesztcsoportnak szánt információ típusa és mennyisége eltérhet attól, amit egy felsővezetésnek címzett jelentés tartalmazna. Az előbbi esetben a hibatípusokra és trendekre vonatkozó részletes információk lehetnek fontosak. Utóbbi esetben viszont megfelelőbb lehet egy magas szintű jelentés (például a hibák csoportosítása prioritás szerint, a költségvetés, az ütemterv és a sikeres/sikertelen/nem tesztelt tesztfeltételek összefoglalása).

Az ISO szabvány (ISO/IEC/IEEE 29119-3) kétféle tesztjelentésre, a tesztelőrehaladási és a teszt befejező (jelen tananyagban összefoglaló teszt-) jelentésekre hivatkozik, és mindkét típushoz tartalmaz mintákat és példákat.

5.4 Konfigurációmenedzsment

A konfigurációmenedzsment célja a komponensek vagy a rendszer, a tesztver integritásának, illetve az ezek közötti kapcsolatok kialakítása és fenntartása a projekt és a termék teljes életciklusa alatt.

A tesztelés tényleges támogatásához a konfigurációmenedzsment a következők biztosítását jelentheti:

- Minden tesztelem pontosan meghatározott, verziókövetés alá vont, a benne történt változások nyomonkövethetők, az elemek kapcsolódnak egymáshoz
- A tesztver minden eleme pontosan meghatározott, verziókövetés alá vont, a benne történt változások nyomonkövethetők, az elemek kapcsolódnak egymáshoz és a tesztelem(ek)hez,

  ezáltal a nyomonkövethetőség fenntartható a teszt folyamán.
- A tesztdokumentációban egyértelmű hivatkozás található minden létező dokumentumra és szoftverelemre.

A teszttervezés során kell azonosítani és megvalósítani a konfigurációmenedzsment-eljárásokat és infrastruktúrát (eszközöket).

5.5 Kockázat és tesztelés

5.5.1 A kockázat fogalma

A kockázat magában foglalja egy jövőbeni esemény bekövetkezésének negatív következményekkel járó lehetőségét. A kockázat szintjét az esemény valószínűsége és az eseményből eredő hatás (kár) határozza meg.

5.5.2 Termék- és projektkockázatok

A termékkockázat magában foglalja annak a lehetőségét, hogy egy projekttermék (például egy specifikáció, komponens, rendszer vagy teszt) nem felel meg a felhasználók és/vagy az érdekelt felek jogos igényeinek. Ha a termék kockázatai a termék sajátos minőségi jellemzőivel (például funkcionális alkalmassággal, megbízhatósággal, teljesítményhatékonysággal, használhatósággal, biztonsággal, kompatibilitással, karbantarthatósággal és hordozhatósággal) kapcsolatosak, a termékkockázatokat minőségi kockázatoknak is nevezik.

Példák a termékkockázatokra:

- A szoftver nem a specifikációnak megfelelően hajtja végre a tervezett funkciókat
- A szoftver nem a felhasználó, az ügyfél és/vagy az érdekeltek igényeinek megfelelően hajtja végre a tervezett funkciókat
- A rendszer felépítése nem támogat megfelelően bizonyos nemfunkcionális követelményeket

- Bizonyos körülmények között egy adott számítás helytelenül kerül végrehajtásra
- Az iteratív szerkezet hibásan lett lekódolva
- A válaszidők nem megfelelőek egy nagy teljesítményű tranzakciófeldolgozó rendszerhez
- A visszajelzett felhasználói élmény (UX) nem felel meg a termék elvárásainak

A projektkockázat olyan helyzeteket foglal magában, amelyek esetlegesen negatív hatással lehetnek a projekt azon képességére, hogy elérje céljait.

Példák a projektkockázatoka:

- Projekt problémák:
    - Késedelmek fordulhatnak elő a szállítás, egy feladat befejezése vagy a kilépési feltételek, vagy a „kész” definíciójának kielégítése során
    - A pontatlan becslések, a források magasabb prioritású projektekhez való átcsoportosítása vagy az általános költségcsökkentés a szervezetben nem megfelelő finanszírozást eredményezhet
    - A késői változtatások jelentős átdolgozást indukálhatnak
- Szervezeti problémák:
    - A készségek, a tréningek és a személyzet nem elegendők
    - A személyzeti kérdések konfliktusokat és problémákat okozhatnak
    - Lehetséges, hogy a felhasználók, az üzleti személyek vagy az adott témában járatos szakértők az üzleti prioritások ütközése miatt nem elérhetőek
- Politikai problémák:
    - A tesztelők nem megfelelően kommunikálják igényeiket és/vagy a teszteredményeket
    - A fejlesztők és/vagy tesztelők nem követik nyomon a tesztelés és a felülvizsgálatok során található információkat (például nem javítják a fejlesztési és tesztelési gyakorlatokat)
    - Lehet, hogy nem megfelelő a teszteléshez való hozzáállás, vagy a teszteléssel szembeni elvárások (például nem ismerik el a hibakeresés értékét a tesztelés során)
- Technikai problémák:
    - Előfordulhat, hogy a követelmények nem elég jól definiáltak
    - A meglévő korlátok miatt előfordulhat, hogy a követelmények nem teljesülnek
    - Lehet, hogy a tesztkörnyezet nem készül el időben
    - Későn történik meg az adatátalakítás, a migrációs tervezés és ezek eszköztámogatása
    - A fejlesztési folyamat gyengeségei befolyásolhatják a projektmunkával kapcsolatos termékek, mint például a tervezés, a kód, a konfiguráció, a tesztadatok és a tesztesetek következetességét vagy minőségét.
    - A rossz hibakezelés és hasonló problémák felhalmozódó hibákat és egyéb technikai lemaradást eredményezhetnek

- Szállítói problémák:
    - Egy harmadik fél nem tud szállítani egy szükséges terméket vagy szolgáltatást, vagy csődbe megy
    - Szerződéssel kapcsolatos kérdések problémákat okozhatnak a projekt számára

A projektkockázatok befolyásolhatják mind a fejlesztési tevékenységet, mind a tesztelési tevékenységeket. Bizonyos esetekben a projektmenedzserek felelősek minden projektkockázat kezeléséért, de nem szokatlan, hogy a tesztvezetők felelősek a tesztekkel kapcsolatos projektkockázatokért.

5.5.3 Kockázat alapú tesztelés és termékminőség

A kockázatot arra használják, hogy a tesztelés során szükséges erőfeszítéseket összpontosítsák. Ezzel határozzák meg, hogy mikor kezdjék el és mikor fejezzék be a tesztelést, illetve ezzel azonosítják azokat a területeket, amelyekre nagyobb figyelmet kell fordítani. A tesztelést a nemkívánatos esemény előfordulási valószínűségének csökkentésére vagy a mellékhatások csökkentésére használják. A tesztelést kockázatcsökkentő tevékenységként használják, hogy információt adjanak az azonosított kockázatokról, valamint a fennmaradó (megoldatlan) kockázatokról.

A tesztelés kockázat alapú megközelítése proaktív lehetőségeket kínál a termékkockázat szintjének csökkentésére. Magába foglalja a termékkockázat-elemzést, amely a termékkockázatok azonosítását és az egyes kockázatok valószínűségének és hatásának értékelését jelenti. A kapott termékkockázati információkat a teszt tervezéséhez, a tesztesetek specifikálásához, előkészítéséhez és végrehajtásához, valamint a tesztellenőrzéshez és -vezérléshez használják. A termékkockázatok korai elemzése hozzájárul a projekt sikeréhez.

Kockázat alapú megközelítésben a termékkockázat elemzésének eredményeit az alábbi tevékenységek során használhatják:

- Az alkalmazandó vizsgálati technikák meghatározása
- Az elvégzendő tesztek megfelelő szintjeinek és típusainak meghatározása (például biztonsági tesztelés, hozzáférhetőségi tesztelés)
- Az elvégzendő tesztek mértékének azonosítása
- A tesztelés priorizálása a kritikus hibák minél hamarabb megtalálása érdekében
- Annak meghatározása, hogy a tesztelésen kívül tehetünk-e valamit a kockázat csökkentésére (például tréning tartása a tapasztalatlan tervezők számára)

A kockázat alapú tesztelés igénybe veszi a projektben résztvevők minden ismeretét és tudását a termékkockázat-elemzés elvégzéséhez. Ahhoz, hogy a termék meghibásodásának valószínűsége minimális legyen, a kockázatmenedzsmentnek szigorú megközelítéseket kell alkalmazni a következők terén:

- Annak értékelése (és rendszeres újraértékelése), hogy milyen probléma léphet fel (kockázatok)
- Annak meghatározása, hogy mely kockázatok kezelése a legfontosabb
- Lépések kidolgozása ezen kockázatok kezelésére
- Készenléti tervek készítése a kockázatok kezelésére, ha azok tényleges eseményekké válnak

Ezenkívül a tesztelés új kockázatokat azonosíthat, segít meghatározni, hogy mely kockázatokat kell enyhíteni, és mely kockázatok bizonytalanságát kell csökkenteni.

5.6 Hibamenedzsment

Mivel a tesztelés egyik célja, hogy hibákat találjon, a tesztelés során talált hibákat rögzíteni kell. A hibák rögzítésének módja eltérő lehet a vizsgált komponens vagy rendszer kontextusának, a tesztszintnek és a szoftverfejlesztési életciklusmodellnek függvényében. Minden azonosított hibát meg kell vizsgálni, és figyelemmel kísérni a felfedezéstől és a besoroláson át a megoldásukig (például a hibák kijavítása és a megoldás sikeres ellenőrző tesztelése, elhalasztás egy későbbi kiadásra, elfogadás, mint tartós termékkorlátozás stb.). Annak érdekében, hogy az összes hibára megoldást találjunk, a szervezetnek hibakezelési folyamatot kell létrehoznia, amely magában foglal egy munkafolyamatot és az osztályozási szabályokat. Ezt a folyamatot el kell fogadni mindenkinek, aki részt vesz a hibakezelésben, beleértve az architekteket, a tervezőket, fejlesztőket, tesztelőket és terméktulajdonosokat. Egyes szervezeteknél a hibarögzítés és követés nagyon informális lehet.

A hibakezelési folyamat során előfordulhat, hogy a jelentések némelyike tévesen hibás eredményt (false positive) tartalmaz, mely valójában nem létező hibát jelez. Például egy teszt megbukhat, ha a hálózati kapcsolat megszakad vagy időnként megszűnik. Ez a viselkedés nem a tesztobjektum hibájából ered, hanem anomália, amelyet meg kell vizsgálni. A tesztelőknek meg kell próbálniuk minimalizálni a tévesen hibás jelentések számát.

Hibákat jelenthetünk a kódolás, a statikus elemzés, a felülvizsgálatok, a dinamikus tesztelés vagy a szoftver termék használata során. Jelenthetünk hibákat a kódban, vagy futtatott rendszerekben, illetve bármilyen dokumentációban, beleértve a követelményeket, a felhasználói történeteket, az elfogadási kritériumokat, a fejlesztési dokumentumokat, a tesztdokumentumokat, a felhasználói kézikönyveket vagy a telepítési útmutatókat. A hatékony és hatásos hibakezelési folyamat érdekében a szervezetek meghatározhatják a hibák jellemzőinek, osztályozásának és munkafolyamatának szabványait.

A hibajelentések tipikus céljai:

- A fejlesztőknek és más feleknek tájékoztatás nyújtása minden olyan mellékhatásról, amely bekövetkezett, lehetővé téve számukra, hogy azonosítsák a konkrét hatásokat, izolálják a problémát egy minimális ismétlő teszteléssel, és szükség esetén javítsák a lehetséges hibákat, vagy más módon oldják meg a problémát.
- A tesztmenedzserek számára biztosítja a munkatermék minőségére és a tesztelésre gyakorolt hatások nyomonkövetésére szolgáló eszközt (például ha sok hibát jelentenek, a tesztelők sok időt töltöttek a tesztek helyett a jelentések megírásával, és több ellenőrző tesztelésre is szükség lesz)
- Ötleteket ad a fejlesztési és a tesztfolyamatok javítására

A dinamikus tesztelés során létrehozott hibajelentés általában a következőket tartalmazza:

- Azonosító
- A bejelentett hiba címe és rövid összefoglalása
- A hibajelentés dátuma, a kibocsátó szervezet és a szerző
- A tesztelem (az éppen tesztelt konfigurációs elem) és a környezet azonosítója
- A fejlesztési életciklus fázisa(i), amelyben a hibát észlelték
- A hiba reprodukálását és megoldását lehetővé tevő leírás, beleértve a naplófájlokat, az adatbázis-mentéseket, képernyőképeket vagy felvételeket (ha volt ilyen a teszt végrehajtása során)
- Elvárt és tényleges eredmények
- A hiba súlyosságának (severity) hatóköre vagy mértéke az érdekelt felek érdekeit tekintve

- A javítás sürgőssége/prioritása
- A hibajelentés állapota (pl. nyitott, elhalasztott, duplikált, megoldásra váró, ellenőrző tesztelésre váró, újranyitva, lezárt)
- Következtetések, ajánlások és jóváhagyások
- Globális problémák, például olyan területek, amelyekre a hibából eredő változás hatással lehet
- Változási előzmények, mint például a projektcsapat tagjai által a hibával kapcsolatos lépések, hogy elkülönítsék, javítsák és ellenőrizzék azt
- Hivatkozások, beleértve a problémát feltáró tesztesetet is

Hibakezelési eszközök használatával ezen információk egy része automatikusan rögzítésre és/vagy kezelésre kerülhet, például egy azonosító automatikus hozzárendelése, a hibajelentés állapotának hozzárendelése és frissítése a munkafolyamat alatt stb. A statikus tesztelés során, különösen a felülvizsgálatok esetében észlelt hibákat többnyire más módon dokumentáljuk, pl. a felülvizsgálati értekezlet naplójában.

A hibajelentés tartalmára az ISO szabványban (ISO/IEC/IEEE 29119-3) található egy példa (amely a hibajelentésre, mint incidensjelentésre hivatkozik).

6. Eszköztámogatás a tesztelésben 40 perc

Kulcsszavak

adatvezérelt tesztelés, kulcsszóvezérelt tesztelés, tesztautomatizálás, tesztmenedzsmenteszköz, tesztvégrehajtási eszköz

Tanulási célok az Eszköztámogatás a tesztelésben fejezethez

6.1. Teszteszközök szempontrendszerei

AK-6.1.1. (K2) Osztályozza a tesztelést támogató eszközöket azok céljai és a támogatott tesztelési folyamatok szerint

AK-6.1.2. (K1) Azonosítsa a tesztautomatizálás előnyeit és kockázatait

AK-6.1.3. (K1) Idézze fel a speciális szempontokat a tesztvégrehajtó és tesztmenedzsmenteszközökkel kapcsolatban

6.1. Hatékony eszközhasználat

AK-6.2.1. (K1) Azonosítsa a legfőbb elveket egy eszköz kiválasztásához

AK-6.2.2. (K1) Idézze fel a pilot projekt célkitűzéseit egy eszközbevezetéshez

AK-6.2.3. (K1) Azonosítsa azokat a tényezőket egy szervezetben, melyek a teszteszközök sikeres kiértékelése, megvalósítása, telepítése és folyamatos támogatása szempontjából elengedhetetlenek.

6.1 Teszteszközök szempontjai

Teszteszközöket használhatunk különböző tesztelési tevékenységek támogatására, ilyen eszközök lehetnek:

- Eszközök, melyeket közvetlenül a tesztelés folyamán használunk, ilyenek lehetnek a tesztvégrehajtó vagy a tesztadat-előkészítő eszközök
- Eszközök, melyek segítenek a követelmények, tesztesetek, teszteljárások, automatizált tesztszkriptek, teszteredmények, tesztadatok és hibák menedzselésében és segítik a tesztvégrehajtás nyomonkövetését és jelentések készítését
- Eszközök melyeket elemzések és kiértékelések során alkalmaznak
- Bármi más eszköz, ami a tesztelést támogatja (ebben az értelemben egy táblázat is teszteszköz)

6.1.1 Teszteszközök osztályozása

A teszteszközök akár több célt is szolgálhatnak az alábbiak közül:

- Teszttevékenységek hatékonyságának növelése az ismétlődő vagy a kézi végrehajtás esetén jelentősen erőforrás igényes feladatok (tesztvégrehajtás, regressziós teszt) automatizálásával
- Teszttevékenységek hatékonyságának növelése a kézi tesztelési tevékenységek támogatásával a teljes tesztfolyamat során
- A teszt minőségének javítása a tesztelés ismételhetőségének javításával és jobb illetve könnyebb hiba reprodukálhatósággal
- Kézzel végre nem hajtható tevékenységek automatizálása (pl. kliens-szerver alkalmazások széles skálájú teljesítménymérése)
- A teszt megbízhatóságának növelése (pl. nagy mennyiségű adat összehasonlítására, vagy szimulációs tevékenységekre)

Eszközöket sokféle szempont szerint lehet osztályozni úgymint cél, árazás, licenc típus (pl. kereskedelmi vagy nyílt forrás kódú), illetve alkalmazott technológia alapján. Ebben a tantervben a teszteszközöket az általuk támogatott teszttevékenység alapján osztályozzuk.

Néhány eszköz nyilvánvalóan egyetlen, vagy főként egy tevékenységet támogat, mások támogathatnak többet is, ezeket ahhoz a tevékenységhez sorolják, amelyhez leginkább köthetők. Egy adott szolgáltatótól származó eszközöket egy csomagban értékesíthetik, főleg akkor, ha együttes használatra tervezték őket.

A teszteszközök egyes típusai lehetnek beavatkozók olyan értelemben, hogy maga az eszköz befolyásolhatja a teszt aktuális eredményét. Például: a valós válaszidő eltérhet egy adott alkalmazásban abban az esetben, ha egy teljesítménytesztelő eszköz extra utasításokat von be a végrehajtásba, vagy a kód lefedettség értékek változhatnak az alkalmazott lefedettségi eszköz használatától függően. A beavatkozó eszközök használatának következményét mérési mellékhatásnak nevezik.

Egyes eszközök elsősorban a fejlesztők munkáját támogatják (pl. a komponens- vagy komponens integrációs teszt során). Ezeket az eszközöket a későbbi osztályozásnál „(F)” -fel jelöljük.

Eszközök tesztek és a tesztver menedzsmentjének támogatására

A menedzsment eszközök minden teszttevékenységnél a szoftver teljes életciklusán át alkalmazhatók. Ilyen eszközök, amik a tesztelés és a tesztver menedzselését támogatják lehetnek például:

- Tesztmenedzsmenteszközök és alkalmazás életciklus menedzselő eszközök (Application Lifecycle Managment – ALM)

- Követelménymenedzsment eszközök (pl. nyomonkövethetőség a tesztelés tárgya felé)
- Hibamenedzsment eszközök
- Konfigurációmenedzsment-eszközök
- Folyamatos integrációs (CI) eszközök (F)

Eszközök statikus tesztelés támogatására

Statikus tesztelési eszközök a 3. fejezetben leírt tevékenységekkel és előnyökkel hozhatók összefüggésbe. Példák ilyen eszközre:

- Statikus elemző eszközök (F)

Eszközök teszttervezés és tesztmegvalósítás támogatására

Teszttervező eszközök segítenek karbantartható projekttermékek előállításában a teszttervezés és megvalósítás során, ide értendők: teszteset, teszteljárás és tesztadat. Példák ilyen eszközökre:

- Modell alapú tesztelési eszközök
- Tesztadat-előkészítő eszközök

Néhány esetben a teszttervezést és megvalósítást támogató eszközök szintén segíthetik a tesztvégrehajtást és naplózást, vagy közvetlenül rendelkezésre bocsájtják a kimenetüket a végrehajtást és naplózást támogató eszközök számára.

Eszközök tesztvégrehajtás és naplózás támogatására

Rengeteg eszköz elérhető a tesztek végrehajtása és a végrehajtás során esedékes naplózás támogatásához, ezek lehetnek:

- Tesztvégrehajtó eszközök (pl. regressziós teszt végrehajtásához)
- Lefedettségi eszköz (pl. követelmény lefedettség, kód lefedettség (F))
- Teszttámogató szoftverkörnyezet (F)

Eszközök teljesítménymérés és dinamikus elemzés támogatására

Teljesítmény mérő és dinamikus analízis eszközök elengedhetetlenek a teljesítmény és terheléses tesztek támogatásához, ugyanis ezeket a tesztelési tevékenységeket kézzel nem lehet hatékonyan végrehajtani. Példák ilyen eszközökre:

- Teljesítményteszt eszközök
- Dinamikus elemző eszközök (F)

Eszközök speciális tesztelői igények támogatására

Az általános tesztelési folyamat támogatása mellett létezik számos eszköz, amik a nemfunkcionális jellemzők tesztelését támogatják.

6.1.2 Előnyök és kockázatok tesztautomatizálás során

Egy eszköz beszerzése nem garancia a sikerre. Bármely típusú eszköz esetében szükség lehet további erőfeszítésekre a valódi és tartós előnyök eléréséhez. Egy adott eszköz használata esetleges előnyöket és lehetőségeket jelent, azonban kockázatokat is hordoz magában. Ez kifejezetten igaz tesztvégrehajtó eszközökre (melyet gyakran tesztautomatizálásként említenek).

Lehetséges előnyök tesztvégrehajtást támogató eszközök használata esetén:

- Csökkenő ismétlődő munka (pl. regressziós tesztek futtatása, tesztkörnyezet felállítás és lebontás, ugyanazon tesztadatok ismételt bevitele, kódolási szabványok ellenőrzése) időmegtakarítást eredményez
- Magasabb fokú konzisztencia és ismételhetőség (pl. tesztadatok koherens előállítása, eszköz által végrehajtott tesztek azonos sorrendben és gyakorisággal, követelményekből következetesen származtatott tesztek)
- Tárgyilagosabb értékelés (pl. statikus mérések, lefedettség)
- Könnyebb hozzáférhetőség a teszteléssel kapcsolatos információkhoz (pl. a teszt-előrehaladást, az incidens-arányokat és teljesítményt mutató statisztikák és grafikonok)

Lehetséges kockázatok tesztvégrehajtást támogató eszközök használata esetén:

- Irreális elvárások az eszközzel kapcsolatban (ide tartozik a funkcionalitás és a könnyű használat is)
- Az eszköz bevezetésére szánt idő, költségek és ráfordítások esetleges alábecslése (ide tartozik a képzés és a külső szaktudás is)
- Az eszköz által nyert jelentős, szignifikáns és folyamatos előnyök eléréséhez szükséges idő és erőfeszítés esetleges alábecslése (ide tartozik a tesztfolyamat átalakításának szükségessége és az eszköz használati módjának folyamatos javítása is)
- Az eszköz által előállított tesztmunkatermékek karbantartásához szükséges erőforrások esetleges alábecslése
- Az eszköz iránti esetleges túlzott elkötelezettség (tesztttervezés vagy végrehajtás helyettesítése, illetve automatizált tesztelés használata ott, ahol a manuális teszt jobb lenne)
- A tesztmunkatermékek verzió követésének esetleges elhanyagolása
- Kritikus eszközök, mint pl. a követelménymenedzsment, konfigurációmenedzsment, hibamenedzsment, illetve különböző eszközforgalmazóktól származó eszközök kapcsolatából, illetve együttműködéséből származó problémák esetleges elhanyagolása
- Az eszköz forgalmazója tönkremehet, kivezetheti az eszközt a piacról vagy eladhatja azt egy másik forgalmazónak
- A forgalmazó esetleg nem megfelelően reagál támogatás, frissítés, illetve hibajavítás esetén
- Egy nyílt forráskódú projekt esetleges felfüggesztése
- Az eszköz esetleg nem támogat egy új platformot vagy technológiát
- Az eszköz tisztázatlan tulajdonosi jogköre (pl. mentorálás, frissítés stb.)

6.1.3 Különleges szempontok tesztvégrehajtó- és tesztmenedzsmenteszközök kapcsán

A sima és sikeres megvalósítás érdekében van néhány szempont, amit érdemes figyelembe venni egy tesztvégrehajtó, illetve tesztmenedzsmenteszköz kiválasztása és a szervezetbe történő bevezetése esetén.

Tesztvégrehajtó eszközök

A tesztvégrehajtási eszközök visszajátsszák az elektronikusan rögzített tesztek végrehajtására tervezett szkripteket. Ennél a típusnál gyakran jelentős erőfeszítések szükségesek a nagyobb előnyök eléréséhez**.**

Tesztrögzítés megközelítés: Vonzó lehetőség teszteket rögzíteni egy manuális tesztelő műveleteinek felvételével, ez a megközelítés azonban nem megfelelő nagyszámú automatizált teszt esetén. Egy felvett szkript lineárisan reprezentálja a meghatározott adatokat és műveleteket. Ez a típusú szkript nem várt incidensek fellépésekor instabil lehet, illetve továbbra is folyamatos karbantartási munkákat igényel, ahogy a felhasználói felület változik az idő során.

Adatvezérelt teszt megközelítés: Ez a teszt megközelítés elkülöníti a tesztbemeneteket, illetve az elvárt eredményeket, általában egy táblázatban tárolva, és egy általánosabb szkriptet használ, amely képes tesztadatokat beolvasni és eltérő adatokkal végrehajtani ugyanazt a tesztet.

Kulcsszó alapú teszt megközelítés: E tesz megközelítés esetében egy általános szkript kulcsszavakat (szintén hívják akciószavaknak is) dolgoz fel, amik a végrehajtandó műveleteket írják le, melyek azt követően meghívják a kulcsszó scripteket, hogy feldolgozzák a hozzárendelt tesztadatokat.

A fenti megközelítések megkövetelik a használótól a szkriptnyelv ismeretét (tesztelők, fejlesztők vagy tesztautomatizálási szakemberek). Adatvezérelt -, vagy kulcsszó alapú teszt megközelítés alkalmazásával a szkriptelési ismeretekkel nem rendelkező tesztelők is képesek az előre definiált szkriptekhez tesztadatokat és/vagy kulcsszavakat létrehozni. A használt szkript technikától függetlenül, az elvárt eredményeket minden tesztben az aktuális eredményekhez kell hasonlítanunk, akár dinamikusan (miközben a teszt fut) akár későbbi (futás utáni) összehasonlításhoz tároltan.

További részletek és példák az adatvezérelt – és a kulcsszó alapú teszt megközelítésekről itt találhatóak: ISTQB-CTAL-TAE, Fewster 1999 and Buwalda 2001.

A modell alapú tesztelési (MBT az angol szakirodalomban) eszközök lehetővé teszik a funkcionális specifikáció egy modell formájában való rögzítését úgymint egy tevékenység diagram. Ez a legtöbb esetben egy rendszertervező feladata. Az MBT eszköz értelmezi a modellt, hogy teszteset specifikációt készíthessen belőle, melyet azt követően egy tesztmenedzsmenteszközben tárolhat, és/vagy egy tesztvégrehajtó eszköz által lefuttathat (lásd az ISTQB-CTFL-MBT referenciát).

Tesztmenedzsmenteszközök

A tesztmenedzsmenteszközöknek gyakran szükséges kapcsolatot létesíteni más eszközökkel vagy táblázatkezelőkkel több okból kifolyólag is, ezek lehetnek:

- Hogy a szervezet számára megfelelő formátumban állítsanak elő hasznos információkat
- Hogy konzekvensen biztosítson nyomonkövethetőséget a követelmények felé a követelménymenedzsment eszközben
- Hogy a konfigurációmenedzsment-eszközben tárolt tesztobjektum verzióval összeköttetést biztosítson

Ez különösen fontos amennyiben integrált eszközt használunk (pl. Alkalmazás Életciklus-Menedzsment (ALM)), ami tartalmazza a tesztmenedzsmentmodult épp úgymint más modulokat (pl. projekt menetrend és költségvetési információk), amiket a szervezet különböző csoportjai használnak.

6.2 Eszközök hatékony használata

6.2.1 Az eszközválasztás legfőbb alapelvei

A fő megfontolások az eszköz választásánál egy szervezet részére a következőket foglalják magukban:

- A saját szervezet érettségének kiértékelése, illetve erősségeinek és gyengeségeinek ismerete
- A lehetőség felismerése egy eszközök által támogatott, továbbfejlesztett tesztelői folyamatra

- A teszt tárgya(i) által használt technológiák megismerése, annak érdekében, hogy egy azzal kompatibilis eszközt válasszunk
- A szervezet által már használatban lévő fejlesztési és folyamatos integrációs eszközök megismerése, a kompatibilitás és illeszkedés biztosítása érdekében
- Az eszköz kiértékelése egyértelmű követelmények és tárgyilagos feltételek alapján
- Az eszköz ingyenes próbaidőszakának (és annak hossza), vagy annak hiányának figyelembevétele
- Az eszköz forgalmazójának kiértékelése (beleértve a képzést, támogatást és kereskedelmi szempontokat), illetve az elérhető támogatás értékelése nem kereskedelmi (pl. nyílt forráskódú) eszközökhöz
- A belső követelmények meghatározása az eszköz használatával kapcsolatos coach, illetve mentorálási tevékenységek tekintetében
- Az oktatási igények kiértékelése, figyelembe véve a tesztelői (ill. tesztautomatizálási) szaktudását azoknak, akik a közvetlen használói lesznek az eszköznek
- A különböző licencelési minták előnyeinek és hátrányainak figyelembevétele (pl. kereskedelmi vagy nyílt forráskódú)
- A konkrét üzleti eseten alapuló költségráfordítás-becslés készítése (amennyiben szükséges)

Utolsó lépésként szükséges a kísérleti projekt kiértékelése annak érdekében, hogy az eszköz hatékonyan teljesít-e a tesztelendő szoftverrel és a jelenlegi infrastruktúrával; vagy ha nélkülözhetetlen, hogy a szükséges változtatásokat azonosítsuk az infrastruktúrában a hatékony eszközhasználat érdekében.

6.2.2 Pilot projekt a szervezeti eszközbevezetéshez

Az eszközválasztás lezárása és a sikeres kísérleti projekt után a kiválasztott eszköz bevezetése a szervezetbe általában egy kísérleti projekttel indul, aminek a céljai a következők:

- Az eszköz alapos ismerete, erősségeinek és gyengeségeinek megértése
- Az eszköz a meglévő folyamatokba és gyakorlatba való illeszkedésének értékelése; a szükséges változtatások meghatározása
- Az eszköz és a tesztmunkatermékek szabványos használati, menedzselési, tárolási és karbantartási módjának meghatározása (pl. fájlok és tesztek elnevezési szabályai, kódolási szabványok kiválasztása, könyvtárak létrehozása, tesztkészletek modularitásának definiálása)
- Annak értékelése, hogy az előnyöket elfogadható kiadásokkal érik-e el
- Azon metrikák megértése, amiket szeretnénk, hogy az eszköz gyűjtsön és jelentéseket készítsen róluk, illetve az eszköz beállítása, hogy ezeket a metrikákat mérni és jelenteni tudja

6.2.3 Sikertényezők eszközökhöz

Az eszközök értékelésének, megvalósításának, telepítésének és folyamatos támogatásának sikertényezői egy szervezeten belül magában foglalják:

- Az eszköz folyamatos bevezetése a szervezet további egységeiben
- Az eszköz használatához illeszkedő folyamatok átvétele, illetve fejlesztése
- Képzés és betanítás/tanácsadás biztosítása az eszköz felhasználói részére

- Használati irányelvek kidolgozása (pl. belső szabvány az automatizálásra)
- Módszer kidolgozása az eszköz használatával kapcsolatos tapasztalatok feldolgozására
- Az eszköz használatának és előnyeinek figyelemmel tartása
- Az adott eszköz felhasználóinak támogatása
- A tanulságok begyűjtése minden felhasználótól

Továbbá az is fontos, hogy mind műszaki, mind pedig szervezeti szempontból integrálódjon az eszköz a fejlesztési életciklusba, ami az üzemeltetésért felelős különböző szervezeteket és/vagy harmadik beszállító feleket is magában foglalhat.

Lásd még a Graham 2012 referenciát a gyakorlati tanácsokkal és a tesztvégrehajtási eszközök használatával kapcsolatban.

7. Irodalomjegyzék

Szabványok

ISO/IEC/IEEE 29119-1 (2013) Software and systems engineering - Software testing - Part 1: Concepts and definitions

ISO/IEC/IEEE 29119-2 (2013) Software and systems engineering - Software testing - Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2013) Software and systems engineering - Software testing - Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2015) Software and systems engineering - Software testing - Part 4: Test techniques

ISO/IEC 25010, (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models

ISO/IEC 20246: (2017) Software and systems engineering — Work product reviews

UML 2.5, Unified Modeling Language Reference Manual, http://www.omg.org/spec/UML/2.5.1/, 2017

ISTQB® dokumentumok

ISTQB® Glosszárium (glossary.istqb.org)

ISTQB® Foundation Level Overview 2018

ISTQB-CTFL-MBT Foundation Level Model-Based Tester Extension Syllabus

ISTQB-CTFL-AT Foundation Level Agile Tester Extension Syllabus

ISTQB-CTAL-TA Advanced Level Test Analyst Syllabus

ISTQB-CTAL-TTA Advanced Level Technical Test Analyst Syllabus

ISTQB-CTAL-TM Advanced Level Test Manager Syllabus

ISTQB-CTAL-SEC Advanced Level Security Tester Syllabus

ISTQB-CTAL-TAE Advanced Level Test Automation Engineer Syllabus

ISTQB-CTEL-TM Expert Level Test Management Syllabus

ISTQB-CTEL-ITP Expert Level Improving the Test Process Syllabus

Szakmai könyvek és cikkek

Beizer, B. (1990) Software Testing Techniques (2e), Van Nostrand Reinhold: Boston MA

Black, R. (2017) Agile Testing Foundations, BCS Learning & Development Ltd: Swindon UK

Black, R. (2009) Managing the Testing Process (3e), John Wiley & Sons: New York NY

Buwalda, H. et al. (2001) Integrated Test Design and Automation, Addison Wesley: Reading MA

Copeland, L. (2004) A Practitioner's Guide to Software Test Design, Artech House: Norwood MA

Craig, R. and Jaskiel, S. (2002) Systematic Software Testing, Artech House: Norwood MA

Crispin, L. and Gregory, J. (2008) Agile Testing, Pearson Education: Boston MA

Fewster, M. and Graham, D. (1999) Software Test Automation, Addison Wesley: Harlow UK

Gilb, T. and Graham, D. (1993) Software Inspection, Addison Wesley: Reading MA

Graham, D. and Fewster, M. (2012) Experiences of Test Automation, Pearson Education: Boston MA

Gregory, J. and Crispin, L. (2015) More Agile Testing, Pearson Education: Boston MA

Jorgensen, P. (2014) Software Testing, A Craftsman's Approach (4e), CRC Press: Boca Raton FL

Kaner, C., Bach, J. and Pettichord, B. (2002) Lessons Learned in Software Testing, John Wiley & Sons: New York NY

Kaner, C., Padmanabhan, S. and Hoffman, D. (2013) The Domain Testing Workbook, Context-Driven Press: New York NY

Kramer, A., Legeard, B. (2016) Model-Based Testing Essentials: Guide to the ISTQB® Certified ModelBased Tester: Foundation Level, John Wiley & Sons: New York NY

Myers, G. (2011) The Art of Software Testing, (3e), John Wiley & Sons: New York NY

Sauer, C. (2000) "The Effectiveness of Software Development Technical Reviews: A Behaviorally

Motivated Program of Research," IEEE Transactions on Software Engineering, Volume 26, Issue 1, pp 1-

Shull, F., Rus, I., Basili, V. July 2000. "How Perspective-Based Reading can Improve Requirement Inspections." IEEE Computer, Volume 33, Issue 7, pp 73-79

van Veenendaal, E. (ed.) (2004) The Testing Practitioner (Chapters 8 - 10), UTN Publishers: The Netherlands

Wiegers, K. (2002) Peer Reviews in Software, Pearson Education: Boston MA

Weinberg, G. (2008) Perfect Software and Other Illusions about Testing, Dorset House: New York NY

Egyéb nem közvetlenül meghivatkozott források

Black, R., van Veenendaal, E. and Graham, D. (2019) Foundations of Software Testing: ISTQB® Certification (4e), Cengage Learning: London UK

Hetzel, W. (1993) Complete Guide to Software Testing (2e), QED Information Sciences: Wellesley MA

8. „A” függelék – A tanterv háttere

A dokumentum háttere

Ezen dokumentum az ISTQB® Alapszintű Tesztelői Tanúsítvány Hivatalos Magyar Nyelvű Tanterve, mely az első szintű, az ISTQB® (www.istqb.org) által elismert nemzetközi képesítés.

Az ezen dokumentum alapjául szolgáló eredeti angol nyelvű tantervet az International Software Testing Qualifications Board (ISTQB®) tagjaiból álló munkacsoport 2019 június és agusztus között készítette. A módosítások a 2018-as tantervet használó ISTQB® tagbizottságaok felülvizsgálata után kerültek a jelen verzióba.

Az elöző verzió alapjául szolgáló eredeti angol nyelvű tantervet az International Software Testing Qualifications Board (ISTQB®) tagjaiból álló munkacsoport 2014 és 2018 között készítette. A 2018-as verzió felülvizsgálatában első körben az ISTQB® tagbizottságai vettek részt, majd a nemzetközi szoftvertesztelői közösség tagjai.

Az Alapszintű Tesztelői Tanúsítvány céljai

- A tesztelés, mint professzionális és nélkülözhetetlen szoftvermérnöki tevékenység elismertségének növelése
- Egységes keretet nyújt a tesztelők karrierfejlesztése érdekében
- A minősített tesztelők elismertségét növeli mind a munkaadó, mind az ügyfél szemében növeli a tesztelői profil értékét
- A szoftvertesztelés legtöbb területén konzisztens és jó tesztelői ötleteket, bevált gyakorlatokat nyújt
- Segít az ipar számára releváns és értékes tesztelői témakörök meghatározásában
- A szoftverfejlesztő cégek számára lehetővé teszi a minősített tesztelők felvételét, illetve a felvételiztetési gyakorlat hirdetésekben való megjelenítésével a versenytársaival szemben üzleti előnyre tehet szert
- Lehetőséget nyújt a tesztelők, illetve a tesztelés iránt érdeklődők számára, hogy egy nemzetközileg elismert tanúsítványt szerezzenek

A Nemzetközi Képesítés céljai

- A tesztelői képességek országhatároktól független összehasonlítása
- A tesztelők más országokban történő elhelyezkedésének megkönnyítése
- Könnyebben kialakítható közös tesztelői nyelv a több országban futó, nemzetközi projektek esetén
- A minősített tesztelők számának növekedése
- Egy nemzetközi kezdeményezésnek nagyobb értéke, illetve hatása van, mint egy országspecifikus megközelítésnek
- A kifejezésgyűjtemény és a tanterv révén a tesztelési ismeretek közös bázison nyugszanak, így a résztvevők magasabb szintű tesztelési ismeretekhez jutnak
- A tesztelés szakmaként történő elismertetése
- Lehetővé tenni a tesztelőknek, hogy anyanyelvükön szerezzenek egy elismert képesítést
- Az országok közötti tapasztalatok és erőforrások megosztása
- Annak következtében, hogy több országban is lehet ilyen képesítést szerezni, a tesztelők, illetve a képesítésük nemzetközi elismertséget kap

A minősítés belépési feltételei

Az ISTQB® Alapszintű Tanúsítvány megszerzésének alapvető feltétele, hogy a résztvevő érdeklődjön a tesztelés iránt. Emellett erősen ajánlott, hogy:

- Legalább minimális szoftverfejlesztő, vagy szoftvertesztelő háttérrel rendelkezzen, azaz minimum 6 hónap rendszertesztelői, vagy felhasználói elfogadási tesztelői, vagy szoftverfejlesztői gyakorlattal rendelkezzen
- Részt vegyen az ISTQB®, vagy az ISTQB® által elismert nemzeti bizottság (ez Magyarországon a HTB) által akkreditált tanfolyamon.

Az Alapszintű Szoftver Tesztelői Tanúsítvány történelmi háttere

A szoftvertesztelés független tanúsítását a British Computer Society's Information Systems Examination Board (ISEB) kezdte meg az Egyesült Királyságban, amikor felállította a Software Testing Board nevű szervezetet 1998-ban (www.bcs.org.uk/iseb). 2002-ben Németországban az ASQF kezdett foglalkozni egy tesztelői minősítési rendszerrel (www.asqf.de). Ezen tanterv az ISEB és az ASFQ újraszerkesztett és frissített tantervein alapul, kiegészítve a tartalmát, a tesztelők gyakorlati munkáját leginkább segítő témákat hangsúlyozva.

A Nemzetközi Tanúsítvány megjelenése előtt megszerezett Alapszintű Szoftver Tanúsítványok (melyeket pl. az ISEB-nél, az ASQF-nél, illetve az ISTQB® által elismert nemzeti bizottságainál lehetett megszerezni) egyenértékűek a Nemzetközi Tanúsítvánnyal. Az Alapszintű Tanúsítvány nem évül el és nem kell megújítani. A tanúsítvány megszerzésének dátuma szerepel a bizonyítványon.

Minden egyes résztvevő országban a nemzeti sajátosságokat az ISTQB® által elismert nemzeti vagy regionális bizottság felügyeli. A tagszervezetek kötelességeit az ISTQB® szabályozza, de ezeket az egyes országokban implementálják. Általában a fő tevékenységek az oktató cégek akkreditálása, illetve a vizsgák szervezése.

9. „B” függelék – Tanulási Célok/Kognitív Tudásszintek

Az alábbi rész a tanterv tanulási céljait mutatja be. Minden egyes téma a tantervben az adott témához tartozó tanulási célok szerint képezi a vizsga tárgyát.

1. szint: Felidézés (K1)

A jelölt képes felismerni és felidézni a fogalmat vagy koncepciót.

Kulcsszavak: azonosíts, emlékezz, elevenítsd fel, idézd fel, emlékezz vissza, ismerd fel

Példák:

A „meghibásodás” fogalmának felismerése, mint pl:

- „amikor nem szállítunk ki egy szolgáltatást a végfelhasználónak vagy bármely más érintettnek”, vagy
- „a komponens, illetve a rendszer eltér az elvárt eredménytől vagy szolgáltatástól”

2. szint: Megértés (K2)

A jelölt képes az egyes témákhoz tartozó állítások okát és magyarázatát kiválasztani, képes összegezni, összehasonlítani, osztályozni, csoportosítani és a tesztelési koncepciót példákkal alátámasztani.

Kulcsszavak: foglald össze, általánosíts, vonatkoztass el, osztályozd, hasonlítsd össze, rendeld hozzá, állítsd szembe, szemléltesd, értelmezd, ábrázold, mutasd be, következtess, csoportosítsd, alkoss modellt

Példák:

El tudja magyarázni az okát, miért kell a tesztanalízist és a műszaki teszttervezést olyan korán megkezdeni, amennyire csak lehetséges?

- Hogy akkor találjuk meg a hibákat, amikor még olcsóbb az eltávolításuk
- Hogy először a legfontosabb hibákat találjuk meg

El tudja magyarázni az integrációs tesztelés és a rendszertesztelés közötti hasonlóságokat és különbségeket:

- Hasonlóságok: a teszt tárgya mind az integrációs tesztelés mind pedig a rendszertesztelés esetén egynél több komponenst foglal magába, valamint mind az integrációs tesztelés, mind pedig a rendszertesztelés nemfunkcionális teszttípusokat is tartalmaz
- Különbségek: az integrációs tesztelés az interfészekre és a kölcsönhatásokra koncentrál, míg a rendszertesztelés az egész rendszerre, mint pl. a teljes rendszeren átívelő munkafolyamatokra koncentrál

3. szint: Alkalmazás (K3)

A jelölt képes a koncepció, vagy a technika kiválasztására, valamint helyes alkalmazására egy adott környezetben.

Kulcsszavak: valósítsd meg, hajtsd végre, használd, kövess egy eljárást, alkalmazz egy eljárást

Példák:

- Képes határértékeket meghatározni érvényes és érvénytelen partíciókhoz
- Képes egy állapotátmenet diagram alapján olyan teszteseteket készíteni, amelyek az összes lehetséges átmenetet lefedik

10. „C” függelék – Kiadási megjegyzések

Az eredeti angol nyelvű ISTQB® Foundation Syllabus 2018 V3.1 Syllabus egy kismértékű átdolgozása a 2018-as verziónak. Egy különálló dokumentumban az ISTQB® a fejezetenkénti változások összefoglalóját és a változtatások nyomon követését.

Az eredeti angol nyelvű ISTQB® Foundation Syllabus 2018 egy nagyfokú átdolgozása a Tanterv 2011-es verziójának. Ezen okból kifolyólag nem készültek részletes megjegyzések az egyes fejezetekhez. Ugyanakkor ez a függelék egy összefoglalót nyújt a fő változásokkal kapcsolatban. Emellett egy különálló dokumentumban az ISTQB® biztosítja a nyomonkövethetőséget a 2011-es verzió és a 2018-as verzió Tanulási Céljai között, mely alapján követhető az új Tanulási Célok megjelenése, illetve a korábbiak aktualizálása vagy törlése.

A 2017 eleji állapot szerint több, mint 550 000 fő tett alapszintű vizsgát összesen több, mint 100 országban és több, mint 500 000-en szereztek tanúsítványt világszerte. Feltételezve, hogy a sikeres vizsgájuk érdekében mindegyikük olvasta az Alapszintű Tantervet, az Alapszintű Tanterv a valaha készült legolvasottabb dokumentum a szoftvertesztelés témakörében.

Ez a nagyfokú átdolgozás ennek az örökségnek a tiszteletben tartásával készült, illetve, hogy növelje azt az értéket, melyet az ISTQB® nyújt majd a globális tesztelői közösségben a következő 500 000 fő számára.

Ebben a verzióban minden Tanulási Cél úgy lett megszerkesztve, hogy atomi, valamint világosan nyomonkövethető legyen, hogy az egyes Tanulási Célok mely tartalmi részekhez (illetve vizsga kérdésekhez) tartoznak, illetőleg az egyes tartalmi részek (illetve vizsga kérdések) szempontjából is nyomonkövethető legyen, hogy mely Tanulási Célokhoz tartoznak. Emellett az egyes fejezetekhez meghatározott időkorlátok is jobban megfelelnek a valóságnak, melyben más ISTQB® Tantervek Tanulási Céljainak kidolgozásánál alkalmazott és igazolt heurisztikák, illetve képletek segítettek.

Míg ez az Alapszintű Tanterv olyan jó gyakorlatokat és technikákat sorakoztat fel, melyek kiállták az idő próbáját, változtatásokat is eszközöltünk, hogy modernizáljuk a tananyag megjelenését, különösképp tekintettel a szoftverfejlesztési módszerekre (pl.: Scrum, folyamatos telepítés) és technológiákra (pl.: a dolgok internete). Aktualizáltuk a hivatkozott szabványokat az alábbiak szerint:

1. Az ISO/IEC/IEEE 29119 szabvány felváltja az IEEE 829 szabványt.
2. Az ISO/IEC 25010 szabvány felváltja az ISO 9126 szabványt.
3. Az ISO/IEC 20246 szabvány felváltja az IEEE 1028 szabványt.

Mindemellett, mivel az ISTQB® portfóliója dinamikusan növekedett az elmúlt évtizedben, ahol szükséges volt, létrehoztunk kereszthivatkozásokat az egyéb kapcsolódó ISTQB® tantervekhez, valamint alapos felülvizsgálatot végeztünk, hogy összehangoljuk a tanterveket, valamint az ISTQB® Glosszáriumot. A cél az volt, hogy ezen verzió könnyebben olvasható és megérthető legyen, fókuszálva a praktikusságra, illetve az egyensúlyra az elmélet és a gyakorlatiasság között.

A 2018-as kiadásban végzett változtatások részletes analízise megtalálható az angol nyelvű „ISTQB® Certified Tester Foundation Level Release Notes 2018” dokumentumban.

A magyar nyelvű kiadás az eredeti angol nyelvű tanterv magyar nyelvű fordítása a hivatalosan elfogadott magyar kifejezésgyűjtemény (ISTQB® Glosszárium) alapján.

11. Tárgymutató