ISTQB® Certified Tester

Foundation Level

Hivatalos magyar nyelvű tanterv

Alapszintű képesítés

Verzió: 3.02, 2012.02.28.

(Official ISTQB CTFL Syllabus – Hungarian)

(ISTQB CTFL Syllabus version: 2010)

© HTB – Hungarian Testing Board
Magyar Szoftvertesztelői Tanács Egyesület

Neumann János u.1/E
H-1117 Budapest, Hungary
Tel: +36 1 382 7297
Fax: +36 1 382 7298
info@hstqb.com

Eredeti mű címe:
Certified Tester Foundation Level Syllabus, Version 2010, International Software Testing Qualifications Board

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 tananyag változásainak, illetve a kifejezésgyűjtemény (glosszárium) változásainak átvezetése
ISTQB 2010 3.0	2011-05-04	A HTB által felülvizsgált változat. Szinkronban a *Foundation+Level+Syllabus+(2010).pdf* angol nyelvű tananyaggal, illetve a *Szoftvertesztelés egységesített kifejezéseinek gyűjteménye 3.1* verzióval
ISTQB 2010, HTB 3.01	2011-11-13	Kisebb javítások
ISTQB 2010, 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 téveszméje -> 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 obejct) -> teszt tárgya

Tartalomjegyzék

Köszönetnyilvánítás

Magyar változat elkészítésében közreműködtek: Kapros Gábor, Csonka Béla, Kusper Gábor, Beszédes Árpád, Kovács Attila.

Bevezetés

A dokumentum célja

Jelen tananyag 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 tananyag a Magyar Szoftvertesztelői Tanács Egyesület (Hungarian Testing Board – HTB) által hivatalosan kiadott, az eredeti angol nyelvű tananyag magyar nyelvű fordítása a hivatalosan elfogadott magyar kifejezésgyűjtemény *(HTB Szoftvertesztelés egységesített kifejezéseinek gyűjteménye, 3.2 verzió, 2011-02-05)* alapján. A tananyagot 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 tananyag pedig segíti a tanulókat a vizsgára való felkészülésben.

Az Alapszintű Tesztelő Tanúsítvány a szoftvertesztelésben

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

Tanulási célok/tudásszint

Jelen tananyag minden fejezetéhez egy megértési szintet rendelünk:

o K1: emlékezés, ismeret, felidézés;
o K2: megértés, kifejtés, indoklás (érvelés), összehasonlítás, osztályozás, besorolás, példák használata, összefoglalás;
o K3: alkalmazás, használat
o K4: elemzés

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

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

A vizsga

Az Alapszintű Tanúsítványhoz kapcsolódó vizsga az itt közölt tananyagra épül. A vizsgakérdések megválaszolásához a tananyag több fejezetének ismeretére is szükség lehet. A tananyag bármely fejezete vizsga tárgyát képezheti. A vizsga feleletválasztós feladatokat tartalmaz. A vizsga letehető akkreditált képzés részeként vagy egyénileg. Magyarországon az Alapszintű Bizonyítvány 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.

Akkreditáció

Egy, az ISTQB által elismert nemzeti bizottság jogosult akkreditálni azokat a képzési szolgáltatókat, amelyek tanterve követi a jelen tananyagot. Az akkreditáció irányelveit az akkreditációt végrehajtó bizottságtól vagy testülettől lehet beszerezni. Az akkreditált tanfolyamokat elismerik, mint a jelen tananyagnak megfelelőt. További részletek a D mellékletben.

Részletesség

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

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

A tananyag nem nyújt teljes leírást a szoftvertesztelésről; csak az alapszintű képzések részletességével érinti a témát.

A tananyag felépítése

A tananyag hat fő részt tartalmaz. A felső címsor tartalmazza a rész által lefedett oktatási célok szintjeit, valamint az adott részre szánt időt. Például:

2. Tesztelés a szoftver életciklusán át (K2) 115 perc

Azt jelenti, hogy a 2. rész K1 (ha magasabb szint van megjelölve, az alsóbb szintek jelenlétét is feltételezzük) és K2 oktatási célokat tartalmaz (de K3-t már nem), és a rész anyagának oktatásához 115 perc szükséges. Minden rész több fejezetet tartalmaz. A fejezetekhez is fel vannak tüntetve a tanulási célok valamint a fejezet oktatásához szükséges idő. Azon alfejezetek, melyeknél nincs idő megadva, a fejezetre adott időbe tartoznak.

1 A tesztelés alapjai (K2) — 165 perc

A tesztelés alapjainak tanulási céljai

A Tanulási Célok (TC) összefoglalják, hogy az egyes fejezetek ismeretanyagának elsajátításának mik a céljai.

1.1 Miért szükséges a tesztelés? (K2)

TC-1.1.1 Annak bemutatása példákkal alátámasztva, hogyan okozhat egy programhiba kárt egy személynek, a környezetnek, vagy egy cégnek. (K2)
TC-1.1.2 Egy hiba kiváltó okának és a hatásainak megkülönböztetése. (K2)
TC-1.1.3 A tesztelés szükségességének indoklása, példákkal alátámasztva. (K2)
TC-1.1.4 Annak bemutatása, hogy a tesztelés miért a minőségbiztosítás része, és példákon keresztül szemléltetni a tesztelés szerepét a jobb minőség elérésében. (K2)
TC-1.1.5 A következő kifejezések kifejtése és összehasonlítása példákon keresztül: emberi eredetű hiba, programhiba, hiba, meghibásodás. (K2)

1.2 Mi a tesztelés? (K2)

TC-1.2.1 A tesztelés általános céljainak felidézése. (K1)
TC-1.2.2 Példák bemutatása a tesztelés céljával kapcsolatban a szoftver életciklus különböző fázisaiban. (K2)
TC-1.2.3 A tesztelés és a hibakeresés megkülönböztetése (K2)

1.3 Általános tesztelési alapelvek (K2)

TC-1.3.1 Az általános tesztelési alapelvek kifejtése. (K2)

1.4 A tesztelés alapvető folyamata (K1)

TC-1.4.1 Az öt alapvető teszttevékenység felidézése a teszttervezésétől a teszt lezárásáig, az egyes teszttevékenységek fő feladatainak leírása. (K1)

1.5 A tesztelés pszichológiája (K2)

TC-1.5.1 Annak felidézése, hogyan befolyásolják pszichológiai tényezők a tesztelés sikerét. (K1):
TC-1.5.2 A tesztelők és a fejlesztők gondolkodásmódjának szembeállítása. (K2)

1.6 Etikai kódex (K2)

1.1 Miért szükséges a tesztelés? (K2)	*20 perc*

Kifejezések
Programhiba, emberi eredetű hiba, meghibásodás, minőség, kockázat.

1.1.1 Szoftverrendszerek környezete (K1)

A szoftverrendszerek egyre nagyobb szerepet játszanak életünkben, az üzleti alkalmazásoktól (mint például bankok (banki tranzakciók)) a fogyasztói termékekig (például autók). A legtöbb ember tapasztalt már olyat, hogy egy szoftver nem megfelelő módon működött. Egy helytelenül működő szoftver rengeteg problémát okozhat, beleértve a pénz– és időveszteséget, az üzleti jó hírnév elvesztését, és akár sérülést, vagy halált is eredményezhet.

1.1.2 A szoftverhibák okai (K2)

Az ember hibát követhet el, tévedhet, ami szoftverhibát, vagy dokumentációs hibát okozhat. Ha a hibás kódot lefuttatjuk, a rendszer nem hajtja végre azt, amit kellene (vagy végrehajt olyat, amit nem kellene), s ez meghibásodáshoz vezet. A szoftver, a rendszer vagy a dokumentumok hibái meghibásodásokat okozhatnak, azonban nem minden programhiba okoz meghibásodást.
Hibák azért lépnek fel, mert az emberek időnként tévednek, szoros határidőket kell betartaniuk, bonyolult kódokkal, komplex infrastruktúrában, változó technológiákkal és/vagy számos rendszer-kölcsönhatással kell dolgozniuk.
Meghibásodást okozhatnak a környezeti viszonyok is.Hibát okozhatnak, pl. a sugárzás, a mágnesesség, az elektromos mezők és a szennyezés, vagy a hardver feltételek megváltoztatása által befolyásolhatják a szoftver lefutását.

1.1.3 A tesztelés szerepe a szoftverfejlesztésben, karbantartásban és üzemeltetésben (K2)

A rendszerek és a dokumentáció szigorú tesztelése hozzájárulhat a használat során fellépő problémák kockázatának csökkentéséhez, valamint a szoftverrendszer minőségének javításához azáltal, hogy a talált hibákat a rendszer kibocsátása előtt kijavítják.
A szoftvertesztelésre szükség lehet akkor is, ha a szoftvernek szerződésben foglalt vagy jogi előírásoknak, ipari szabványoknak kell megfelelnie.

1.1.4 A tesztelés és a minőség (K2)

A tesztelés lehetőséget teremt a szoftver minőségének mérésére a talált hibák alapján, mind a funkcionális, mind a nem-funkcionális szoftverkövetelmények, illetve a jellemzők terén (például megbízhatóság, használhatóság, hatékonyság, karbantarthatóság és hordozhatóság). További információ a nem-funkcionális tesztről a 2. fejezetben található; a szoftverjellemzőkről bővebben pedig a Software Engineering – Software Product Quality ” (ISO 9126)-ban (magyar megfelelője: MSZ ISO/IEC 9126:2000 . Informatika. Szoftvertermékek értékelése. Minőségi jellemzők és használatuk irányelvei) olvashat.

A tesztelés növeli a bizalmat a szoftver minőségével szemben, ha a tesztelés során kevés hibát találunk, vagy egyáltalán nem találunk hibát. Ha egy helyesen tervezett teszt sikeresen lefut, az csökkenti a rendszer általános kockázati szintjét. Ha tesztelés során találnak hibát, akkor azok kiküszöbölésével javul a szoftverrendszer minősége.

A korábbi projektek tapasztalatait fel kell használni. A régebbi projektekben talált hibák kiváltó okainak megértésével a folyamatok javíthatók, így kiküszöbölhetjük ezen hibák újbóli megjelenését, ezáltal javul a jövőbeni rendszerek minősége. Ez a minőségbiztosítás egyik területe.

A tesztelés helye a minőségbiztosítási tevékenységek között van (a fejlesztési szabványokkal, képzéssel és hibaelemzéssel együtt).

1.1.5 Mennyi tesztelés elegendő? (K2)

Ahhoz, hogy eldönthessük, mennyi tesztelés elegendő, figyelembe kell vennünk a kockázati szintet, beleértve a technikai és vállalati termékek, projektek kockázatát valamint a projektek időre és költségvetésre vonatkozó megkötéseit. (A kockázatról bővebben az 5. részben.)

A tesztelésnek elegendő információt kell biztosítania a projekt résztvevői számára, hogy megalapozott döntést hozhassanak a tesztelt szoftver vagy rendszer kibocsátásával kapcsolatban, a következő fejlesztési szakaszról, vagy az ügyfelek számára történő átadásról.

1.2 Mi a tesztelés? (K2)	*30 perc*

Kifejezések

Hibakeresés, követelmény, felülvizsgálat, teszteset, tesztelés, tesztcél.

Háttér

Az általános felfogás szerint a tesztelés csak tesztek futtatásából áll, vagyis a szoftver használatát, futtatását jelenti. Ez része a tesztelésnek, de a tesztelés nem csak ebből áll.

Teszttevékenységek a tesztfuttatás előtt és után is léteznek, ilyenek: teszttervezés- és irányítás, tesztfeltételek meghatározása, tesztesetek tervezése és futtatása, az eredmények ellenőrzése, kilépési feltételek kiértékelése, jelentés készítése a tesztfolyamatról és a tesztelt rendszerről, lezárás vagy befejezés (egy tesztfázis végén). Továbbá a tesztelés magában foglalja a dokumentumok felülvizsgálatát (beleértve a forráskódot is) valamint a statikus elemzést.

A dinamikus teszt és a statikus teszt is alkalmazható hasonló célok elérésére; mindkettő nyújt információkat, amelyek felhasználhatók mind a tesztelendő rendszer, mind a fejlesztési és tesztfolyamatok javításához.

Különböző tesztcélok léteznek:
- megtalálni a hibákat ;
- növelni a minőséggel kapcsolatos megbízhatóságot, információt nyújtani;
- megelőzni a hibákat .

Az életciklus elején történő műszaki teszttervezés során végrehajtott folyamatok és tevékenységek (a tesztbázis ellenőrzése a tesztelés tervezésével) segíthetnek annak megelőzésében, hogy hibák kerüljenek a kódba. A dokumentumok (például a követelmények) felülvizsgálata is segít annak megelőzésében, hogy hibák kerüljenek a kódba.

A tesztelés különböző szempontok szerint történhet, melyek más-más célokra vonatkoznak. A fejlesztői teszt (például a komponens-, integrációs- és rendszerteszt) fő célja lehet például az, hogy a lehető legtöbb meghibásodást előhozza, ezzel a szoftverhibákat felismerhetik és javíthatják. Az átvételi teszt fő célja annak igazolása lehet, hogy a rendszer az elvárásoknak megfelelően működik, illetve megbizonyosodni arról, hogy a rendszer teljesíti-e a követelményeket. Bizonyos esetekben a tesztelés fő célja a szoftver minőségének elemzése lehet (hibák javítására irányuló szándék nélkül) azért, hogy a projekt résztvevői megtudják, hogy adott időpontban milyen kockázattal járna a rendszer kiadása. A karbantartási teszt során gyakran arra vonatkozó tesztet végeznek, hogy a változtatások során nem kerültek-e be újabb hibák a rendszerbe. A működési teszt fő célja a rendszer olyan jellemzőinek elemzése lehet, mint például a megbízhatóság vagy rendelkezésre állás.

A hibakeresés és a tesztelés két különböző fogalom. A dinamikus teszt kimutathatja a programhibák által okozott meghibásodásokat. A hibakeresés pedig egy fejlesztői tevékenység, mely során felderítik a hibák okát, kijavítják a kódot és ellenőrzik, hogy a hibát megfelelően orvosolták-e. Ezt követően egy tesztelő által végrehajtott újrateszteléssel bizonyosodnak meg arról, hogy a javítás eredményes volt. A feladatok más-más felelősségi körbe tartoznak: a tesztelést a tesztelők, a hibakeresést a fejlesztők végzik.

A tesztfolyamatot és az ahhoz tartozó tevékenységeket az 1.4. fejezet írja le.

1.3 Általános tesztelési alapelvek (K2)	*35 perc*

Kifejezések
Kimerítő teszt.

Alapelvek
Az elmúlt 40 évben számos tesztelési alapelvet dolgoztak ki, melyek általános iránymutatást nyújtanak az összes tesztelési formára.

1. alapelv – Hibák látszólagos hiánya
Bár 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 annak a bizonyítéka, hogy a rendszer tökéletes (értsd: valóban nincs benne hiba).

2. alapelv – Nem lehetséges kimerítő teszt
Kimerítő teszt – azaz mindenre (a bemenetek és előfeltételek minden kombinációjára) kiterjedő tesztelés – a triviális eseteket leszámítva – nem lehetséges. A kimerítő teszt helyett kockázatelemzést és prioritásokat kell alkalmazni, ezáltal növelve a teszttevékenységek összpontosításának hatékonyságát.

3. alapelv – Korai tesztelés
A tesztelést a szoftver vagy rendszerfejlesztési életciklusban a lehető legkorábban el kell kezdeni, és előre meghatározott célokra kell összpontosítani.

4. alapelv – Hibafürtök megjelenése
A tesztelést a feltételezett, illetve a megtalált hibák eloszlásának megfelelően kell koncentrálni. A kiadást megelőző tesztelés során a megtalált hibák többsége néhány modulban van, vagy legalábbis ezen modulok felelősek a működési hibák többségéért.

5. alapelv – A féregirtó paradoxon
Ha mindig ugyanazokat a teszteket hajtjuk végre, akkor az azonos a tesztkészlet egy idő után nem fog új hibákat találni. A „féregirtó paradoxon” megjelenése ellen a teszteseteket rendszeresen felül kell vizsgálni, és új, eltérő teszteseteket kell írni annak érdekében, hogy a szoftver vagy a rendszer különböző részei kerüljenek futtatásra, és ezáltal további programhibákat találhassanak.

6. alapelv – A tesztelés függ a körülményektől (körülményfüggés)
A tesztelést különböző körülmények esetén különbözőképpen hajtják végre. Például egy olyan rendszert, ahol a biztonság kritikus szempont, másképp tesztelnek, mint egy e-kereskedelmi oldalt.

7. alapelv – A hibamentes rendszer téveszméje
A hibák megtalálása és javítása hasztalan, ha a kifejlesztett rendszer használhatatlan, és nem felel meg a felhasználók igényeinek, elvárásainak.

1.4 A tesztelés alapvető folyamata (K2)	*35 perc*

Kifejezések

Ellenőrző teszt, újratesztelés, kilépési feltétel , incidens , regressziós teszt, tesztbázis, tesztfeltétel, tesztlefedettség, tesztadatok, tesztvégrehajtás, tesztnapló, teszterv, teszteljárás, tesztelési irányelvek, tesztkészlet, összefoglaló tesztjelentés, tesztver

Háttér

A tesztelés leglátványosabb része a tesztek végrehajtása. A hatékonyság érdekében azonban a teszttervnek a teszterv elkészítéséhez a tesztesetek tervezéséhez, a végrehajtásra való előkészületekhez és az eredmények értékeléséhez szükséges időt is célszerű tartalmaznia.

A tesztelés alapvető folyamata a következő fő tevékenységekből áll:

- tervezés és irányítás;
- elemzés és műszaki tervezés;
- megvalósítás és végrehajtás;
- a kilépési feltételek értékelése és jelentés;
- teszt lezárása.

Bár logikailag egymás után következnek, a folyamat tevékenységei átfedhetik egymást, és párhuzamosan is folyhatnak. Általában szükséges ezen fő tevékenységeknek a projekten, illetve a rendszeren belüli testreszabása.

1.4.1 Teszttervezés- és irányítás (K1)

A teszttervezés során a teszt célját és a tevékenységeket definiálják annak érdekében, hogy elérjék a kitűzött célokat és elvégezzék a feladatokat.

A tesztirányítás egy folyamatosan végzett tevékenység, mely során összehasonlítják az aktuális folyamatot a tervvel, az adott állapotról jelentést készítenek, amely tartalmazza a tervtől való eltéréseket. A tesztirányítás során végrehajtják a projekt feladatainak és céljainak eléréséhez szükséges feladatokat. A megfelelő irányítás érdekében a teszttevékenységeket a teljes projekt során felügyelni kell. A teszttervezésénél figyelembe kell venni a felügyeletből és az irányításból származó visszajelzéseket.

A teszttervhez és az irányításhoz kapcsolódó feladatokat a tananyag 5. fejezete tartalmazza.

1.4.2 Tesztelemzés és terv (K1)

A tesztelemzés és tervezés az a tevékenység, mely során az általános tesztcélokból kézzelfogható tesztfeltételeket és teszteseteket alakítanak ki.

A tesztelemzéshez és tervezéshez a következő fő feladatok tartoznak:

o A tesztbázis (mint például követelmények, a szoftver integritás szintje[1] (kockázati szint) kockázatelemzési jelentés felépítés, műszaki teszterv, interfész specifikációk) átnézése.
o A tesztbázis és a tesztelés tárgyainak értékelése tesztelhetőség szempontjából.
o A tesztfeltételek meghatározása és priorizálása a tesztelemek, a specifikáció, a szoftver viselkedésének és struktúrájának elemzése alapján.
o Tesztesetek megtervezése és priorizálása.

[1] A szoftver mennyire felel meg, vagy mennyire kell megfelelnie egy, az érintettek által választott szoftvernek, és/vagy a szoftver-alapú karakterisztikáknak (pl. szoftver komplexitás, kockázatértékelés, biztonsági szint, a kívánt teljesítmény, megbízhatóság, vagy költség), amelyeket annak érdekében határoznak meg, hogy az érintetteket informálja a szoftver fontosságáról

o A tesztfeltételek és a tesztesetek támogatásához szükséges tesztadatok meghatározása.
o A tesztkörnyezet kialakításának megtervezése valamint a szükséges infrastruktúra és eszközök meghatározása.
o A tesztbázis és a tesztesetek közötti kétirányú nyomonkövethetőég létrehozása

1.4.3 Teszt megvalósítása és végrehajtása (K1)

A teszt megvalósítása és végrehajtása az a tevékenység, mely során meghatározzák a teszteljárásokat vagy szkripteket a tesztesetek adott sorrendű kombinálásával, és a tesztvégrehajtásához szükséges további információk felhasználásával kialakítják a tesztkörnyezetet, valamint futtatják a teszteket.

A teszt megvalósításának és végrehajtásának fő feladatai:

- Tesztesetek fejlesztése, megvalósítása és priorizálása (beleértve a tesztadatok meghatározását)
- Teszteljárások fejlesztése és priorizálása, tesztadatok létrehozása valamint opcionálisan a teszttámogató szoftverkörnyezet elkészítése és automatizált tesztszkriptek írása.
- Tesztkészletek létrehozása a teszteljárásokból a hatékony tesztvégrehajtás érdekében.
- Annak ellenőrzése, hogy a tesztkörnyezetet megfelelően kialakították.
- A teszteljárások a megtervezett sorrend szerinti végrehajtása manuálisan vagy tesztvégrehajtási eszközökkel.
- A tesztvégrehajtás eredményeinek naplózása és a tesztelt szoftver, a teszteszközök, valamint a a tesztver azonosítóinak és verzióinak feljegyzése.
- A kapott és az elvárt eredmények összehasonlítása.
- Az eltérések incidensként való jelentése és elemzése a kiváltó ok felderítése érdekében (például hiba volt a kódban, meghatározott tesztadatokban, a tesztdokumentumban, vagy a teszt végrehajtásában történt emberi eredetű hiba).
- A teszttevékenység ismétlése minden eltérésnél a lépések megtétele után. Például: az előzetesen sikertelen teszt újrafuttatása a javítás ellenőrzésére (ellenőrző teszt), a javított teszt végrehajtása és/vagy tesztek végrehajtása annak ellenőrzésére, hogy nem kerültek-e hibák a szoftver változatlanul hagyott területeibe, illetve hogy a hiba javításával nem jelentek-e meg további hibák (regressziós teszt).

1.4.4 A kilépési feltételek értékelése és jelentés (K1)

A kilépési feltételek értékelése az a tevékenység, mely során a teszt végrehajtását elemzik a meghatározott célokhoz viszonyítva. Ezt minden tesztszintre el kell végezni.

A kilépési feltétel értékelésének fő feladatai:

- A tesztnaplók és a teszttervben foglalt kilépési feltételek összehasonlítása.
- Annak megállapítása, hogy szükséges-e további tesztek futtatása, vagy a kilépési feltételek megváltoztatása.
- Tesztösszefoglaló jelentés elkészítése a projektben résztvevők számára.

1.4.5 Tesztlezárás (K1)

A teszt lezárása során adatokat kell gyűjteni a végrehajtott tesztekre vonatkozóan a tapasztalatok, a tesztver, a tények és számok véglegesítése / értelmezése céljából. A teszt lezárása projekt mérföldkőhöz kapcsolódik, mint például akkor, ha kiadnak egy szoftverrendszert, végrehajtanak (vagy törölnek) egy teszt projektet, eljutnak egy fontosabb szakaszig vagy végeznek egy karbantartó frissítéssel.

A tesztlezárás fő feladatai:

- Annak ellenőrzése, hogy a tervezett átadandókból mit sikerült ténylegesen átadni.
- Az incidens jelentések lezárása, vagy a változtatásokról szóló feljegyzések elkészítése a le nem zárt jelentésekhez.

- A rendszer átvételéről szóló dokumentáció elkészítése.
- A tesztver, és a teszt infrastruktúra véglegesítése és archiválása későbbi felhasználásra.
- A tesztver átadása a karbantartást végző szervezet részére.
- A tapasztalatok feldolgozása a jövőbeni termékekhez vagy projektekhez.
- Az információk hasznosítása a tesztfolyamat érettségének fejlesztéséhez.

1.5 A tesztelés pszichológiája (K2)	35 perc

Kifejezések
Hibasejtés, függetlenség.

Háttér
A tesztelés és a felülvizsgálat a szoftverfejlesztéstől eltérő gondolkodásmódot kíván. A megfelelő gondolkodásmód birtokában a fejlesztők képesek a saját kódjaik tesztelésére, általában azonban ezt a feladatot tesztelőre bízzák, hogy az erőforrások jobban összpontosuljanak. Ez további előnyöket is jelent, nevezetesen független, képzett, professzionális tesztelési erőforrásokat. Független tesztelést a tesztelés bármely szintjén lehet alkalmazni.

Bizonyos fokú függetlenség (a szerzői elfogultság kizárásával) gyakran hatékonyabbá teszi a tesztelőt a hibák és a meghibásodások megtalálásában. A függetlenség azonban nem helyettesíti a jártasságot, a fejlesztők saját kódjukban sok hibát hatékonyan megtalálnak. A függetlenség néhány szintje – alulról felfelé - a következőképpen definiálható:

- A tesztelt szoftver fejlesztője (fejlesztői) által tervezett tesztek (alacsony szintű függetlenség).
- Más(ok) által tervezett tesztek (például valaki a fejlesztői csapatból).
- Más szervezeti csoporthoz (például egy független tesztelő csapathoz) tartozó személy(ek) vagy tesztelési szakértők (például használhatósági vagy teljesítményteszt) által tervezett tesztek.
- Más szervezethez vagy céghez tartozó személy(ek) által tervezett tesztek (outsourcing vagy egy külső testület tanúsítványa).

Az embereket és a projekteket a célok viszik előre. Az emberek hajlamosak hozzáigazítani terveiket a menedzsment vagy más projektrésztvevők által meghatározott célokhoz, erre példa a hibák megtalálása vagy annak ellenőrzése, hogy a szoftver a céljának megfelelően működik. Ezért fontos a tesztcélok pontos, érthető meghatározása.

A teszt során tapasztalt meghibásodások feltárását gyakran a termék, vagy a szerző elleni kritikának fogják fel. Emiatt a tesztelést gyakran destruktív tevékenységnek tekintik, annak ellenére, hogy valójában konstruktív a termék kockázatának kezelésében. Egy rendszer programhibáinak kutatása kíváncsiságot, szakszerű pesszimizmust, kritikus szemléletet, a részletekre való odafigyelést, a fejlesztőkkel való jó kommunikációt és a hibasejtéseket megalapozó tapasztalatot kíván.

A hibák, programhibák konstruktív szemléletű közlésével elkerülhetők a tesztelők és elemzők, tervezők és fejlesztők közötti ellentétek. Ez érvényes a felülvizsgálat során talált hibákra is.

A tesztelőnek és a tesztvezetőnek a hibákkal, előrehaladással és kockázatokkal kapcsolatos tárgyilagos, konstruktív kommunikáció érdekében jó interperszonális képességekkel kell rendelkeznie. A hibákkal kapcsolatos információ a szoftver, vagy a dokumentum szerzőjének segíthet a képességei fejlesztésében. A teszt során megtalált és javított hibák a későbbiekben időt és pénzt takarítanak meg és csökkentik a kockázatokat.

Kommunikációs problémák előfordulhatnak, különösen akkor, ha a tesztelőket csak a hibákról szóló rossz hírek hozóinak tekintik. Léteznek viszont különböző módszerek arra, hogyan javítsuk a tesztelők és a többiek közti kommunikációt és kapcsolatot:
- Együttműködő szándékkal indítsunk hadakozás helyett – felhívni a figyelmet a közös célra: a minél jobb minőségű rendszerre.
- Semlegesen, tárgyilagosan kell előadni az eredményeket, a kidolgozó személy kritizálása nélkül; vagyis például objektív és tárgyilagos incidensjelentés, illetve felülvizsgálat eredmény jelentés készítése.
- Meg kell próbálni megérteni azt, hogy hogyan érez a másik fél, és miért olyanok a reakciói, amilyenek.

- Meg kell bizonyosodni arról, hogy a másik fél helyesen értette a közlendőnket – és viszont.

Irodalomjegyzék

1.1.5 Black, 2001, Kaner, 2002
1.2 Beizer, 1990, Black, 2001, Myers, 1979
1.3 Beizer, 1990, Hetzel, 1988, Myers, 1979
1.4 Hetzel, 1988
1.4.5 Black, 2001, Craig, 2002
1.5 Black, 2001, Hetzel, 1988

1.6 Etikai kódex (K2)	*10 perc*

A szoftver tesztje során a résztvevők bizalmas és kiváltságos információkhoz juthatnak hozzá. Az etikai kódexre többek között annak biztosítására van szükség, hogy az információkat nem használják fel illetéktelenül. Elismerve az ACM and IEEE mérnökökre vonatkozó etkai kódexet, az ISTQB® a következő etikai kódexet határozza meg:

KÖZÉRDEK – A képesített tesztelőknek következetesen a közérdeknek megfelelően kell tevékenykedniük

MEGRENDELŐ ÉS MUNKAADÓ – A képesített szoftvertesztelőknek úgy kell tevékenykedniük, hogy a megrendelőik, illetve munkaadóik igényeit legjobban kiszolgálják, ugyanakkor a közérdekkel ne kerüljenek szembe

TERMÉK – A képesített szoftvertesztelőknek biztosítaniuk kell, hogy az általuk tesztelt, átadásra kerülő termék, vagy rendszer megfelel a legmagasabb szakmai szabványoknak

VÉLEMÉNY – A képesített szoftvertesztelőknek feddhetetlennek és függetlennek kell maradniuk a szakmai véleményalkotáskor

MENEDZSMENT – A képesített szoftver tesztmenedzsereknek és vezetőknek azonosulniuk és támogatniuk kell az etikai kódexet a szoftvertesztelés menedzsmentje felé

SZAKMA – A képesített szoftvertesztelőknek elő kell segíteniük a szakma hírnevét és feddhetetlenségét a közérdeknek megfelelően

MUNKATÁRSAK – A képesített szoftvertesztelőknek korrektül és támogatóan kell fellépni a munkatársaikkal szemben és segíteniük kell a szoftverfejlesztőkkel való együttműködést

SZEMÉLYES - A képesített szoftvertesztelőknek életük végéig tanulniuk kell a szakmájukban és munkájuk végzése során figyelniük kell arra, hogy az etikai kódex a munkájuk részévé váljon

Hivatkozások

1.1.5 Black, 2001, Kaner, 2002
1.2 Beizer, 1990, Black, 2001, Myers, 1979
1.3 Beizer, 1990, Hetzel, 1988, Myers, 1979
1.4 Hetzel, 1988
1.4.5 Black, 2001, Craig, 2002
1.5 Black, 2001, Hetzel, 1988

2 Tesztelés a szoftver életciklusán át (K2) — 115 perc

Tanulási célok a Tesztelés a szoftver életciklusán át témában

A Tanulási Célok (TC) összefoglalják, hogy az egyes fejezetek ismeretanyagának elsajátításának mik a céljai.

2.1 Szoftverfejlesztési modellek (K2)

TC-2.1.1 A fejlesztés, a teszttevékenységek és a fejlesztési életciklus munkatermékei közötti viszony megértése, példák felhozása a projekt- és termékjellemzők és szempontok alapján (K2).

TC-2.1.2 Annak felismerése, hogy a szoftverfejlesztési modelleket igazítani kell a projektek szempontjaihoz és a termékek jellemzőihez. (K1)

TC-2.1.3 A helyes tesztelés jellemzőinek felidézése bármilyen életciklus modellben. (K1)

2.2 Tesztszintek (K2)

TC-2.2.1 A tesztelés különböző szintjeinek összehasonlítása: fő célok, a tesztelés tipikus eszközei, a tesztelés tipikus területei (pl. funkcionális vagy strukturális) és az ezekhez tartozó projekttermékek, a tesztet végző személyek, az azonosítandó hibák és meghibásodások. (K2)

2.3 Teszttípusok (K2)

TC-2.3.1 Négy szoftverteszt-típus összehasonlítása (funkcionális, nem-funkcionális, struktúra alapú és változáshoz kapcsolódó) példákon keresztül. (K2)

TC-2.3.2 Annak felismerése, hogy minden tesztszinten létezik funkcionális és struktúra alapú teszt. (K1)

TC-2.3.3 A nem-funkcionális követelményeken alapuló nem-funkcionális teszttípusok azonosítása és bemutatása. (K2)

TC-2.3.4 Egy szoftverrendszer strukturális vagy architekturális elemzésén alapuló teszttípusok azonosítása és bemutatása. (K2)

TC-2.3.5 Az ellenőrző teszt és a regressziós teszt céljának bemutatása. (K2)

2.4 Karbantartási teszt (K2)

TC-2.4.1 A karbantartási teszt (már létező rendszer tesztje) és az új alkalmazások tesztelésének összehasonlítása tekintettel a teszttípusokra, a tesztelést szükségessé tevő tényezőkre és a tesztek mennyiségére. (K2)

TC-2.4.2 A karbantartási teszt szükségességét indokoló tényezők meghatározása (módosítás, migráció és kivonás). (K1)

TC-2.4.3 A regressziós teszt és a hatáselemzés karbantartásban való szerepének bemutatása.. (K2)

2.1 Szoftverfejlesztési modellek (K2)	*20 perc*

Kifejezések
Dobozos szoftver (COTS), iteratív-inkrementális fejlesztési modell, validáció, verifikáció, V-modell.

Háttér
Mindentől elszigetelt tesztelés nem létezik, a teszttevékenységek a szoftverfejlesztési tevékenységekhez kapcsolódnak.
A különböző fejlesztési életciklus modelleknél különböző tesztelési megközelítésekre van szükség.

2.1.1 V-modell (szekvenciális fejlesztési modell) (K2)

Bár a V-modellnek többféle változata létezik, az általános V-modellnél négy tesztszintet alkalmaznak, melyek a négy fejlesztési szinthez tartoznak.

A jelen tananyagban a következő négy szintet használjuk:

- komponens (egység) teszt;
- integrációs teszt;
- rendszerteszt;
- átvételi teszt.

A gyakorlatban a projekttől és a szoftvertermékt̋ől függően a V-modellnek lehet ennél több, kevesebb, vagy eltérő fejlesztési és tesztszintje. Például a komponens teszt után következhet komponens integrációs teszt vagy a rendszerteszt után rendszer integrációs teszt.

A fejlesztés során létrehozott szoftvertermékek (mint az üzleti forgatókönyvek, vagy használati esetek, követelmény-specifikációk, műszaki teszterv dokumentumok és kód) gyakran képezik a teszt alapját egy vagy több tesztszinten. Az általános szoftvertermékekre referencia lehet az integrált Képesség-Érettség Modell (CMMI) vagy a szoftver életciklus folyamatokat leíró szabványban („Software life cycle processes” ,IEEE/IEC 12207). A verifikációt és a validációt (és a korai műszaki teszttervezést) a szoftvertermékek fejlesztése alatt lehet elvégezni.

2.1.2 Iteratív-inkrementális fejlesztési modellek (K2)

Az iteratív-inkrementális fejlesztés a követelmények meghatározásának, a tervezésnek, a rendszer felépítésének és tesztelésének folyamata, melyet több rövidebb fejlesztési ciklusban hajtanak végre. Példák: prototípus készítés, Gyors Alkalmazásfejlesztés (Rapid Agile Development, RAD), Rational Egységesített Folyamat (RUP) és az agilis fejlesztési modellek. Az iteráció során létrehozott rendszert minden egyes iterációban több tesztszinten lehet tesztelni. A már előzőleg kifejlesztett részekhez kiegészítés hozzáadásával növekvő részrendszer jön létre, amelyet szintén tesztelni kell. A regressziós teszt szerepe minden iterációnál egyre fontosabb. A verifikációt és validációt minden kiegészítés hozzáadásánál végre lehet hajtani.

2.1.3 Tesztelés egy életciklus modellen belül (K2)

Minden életciklus modellben megfogalmazhatók a megfelelő tesztelés jellemzői.
- Minden fejlesztési tevékenységhez tartozik egy teszttevékenység.
- Minden tesztszint rendelkezik az adott szintre jellemző tesztcéllal.
- A tesztek elemzését és tervezését az adott tesztszinthez tartozó fejlesztési tevékenység során kell megkezdeni.
- A tesztelőket be kell vonni a dokumentációk felülvizsgálatába, amint elkészültek az első munkaanyagok a fejlesztési életciklusban.

A projekt jellemzőitől és a rendszer felépítésétől függően a tesztszinteket kombinálhatjuk vagy átrendezhetjük. Egy dobozos szoftver valamely rendszerbe való integrációja esetén például a vevő végrehajthat rendszer-szintű integrációs tesztet (például integráció az infrastruktúrába és más rendszerekbe, vagy a rendszer bevezetése) vagy átvételi tesztet (funkcionális és/vagy nem-funkcionális, felhasználói és/vagy működési teszt).

2.2 Tesztszintek (K2)	*40 perc*

Kifejezések

Alfa teszt, béta teszt, komponens teszt, meghajtó, funkcionális követelmény, integráció, integrációs teszt, nem-funkcionális követelmény, robosztussági teszt, csonk, rendszerteszt, tesztszint, tesztvezérelt fejlesztés, tesztkörnyezet, felhasználói átvételi teszt.

Háttér

Minden tesztszint esetén a következőket kell tudni meghatározni: a teszt általános célja, a tesztesetek készítésekor meghivatkozott projekttermékek (pl.a tesztbázis) , a teszt tárgya (mit tesztelnek), a megtalálandó hibák és meghibásodások, a teszttámogató szoftverkörnyezet követelményei és az eszköztámogatás, a jellemző megközelítések és felelősségi körök.

2.2.1 Komponens teszt (K2)

Tesztbázis:

- Komponens követelmények
- Részletes műszaki teszterv
- Kód

A teszt jellemző tárgyai:

- Komponensek
- Programok
- Adatkonvertáló / migrációs programok
- Adatbázis modulok

A komponens teszt (nevezik egység(unit)-, illetve modultesztnek is, de programtesztnek is) hibákat keres az önállóan tesztelhető szoftverekben (pl. modulok, programok, objektumok, osztályok), és ellenőrzi működésüket. A fejlesztési életciklus körülményeitől és a rendszertől függően a rendszer többi részétől elkülönítve is végezhető. Használhatók csonkok, meghajtók és szimulátorok.

A komponens teszt magában foglalhatja a funkcionalis valamint meghatározott nem-funkcionális jellemzők tesztjét, mint például az erőforrásokkal kapcsolatos viselkedés (például memóriaszivárgás keresése) tesztje, robosztussági, illetve strukturális teszt (pl. elágazás lefedettség). A teszteseteket termékekből készítik, mint pl. a komponens specifikációja, a szoftverterv, vagy adatmodell-specifikáció.

A komponens teszt során általában hozzáférnek a tesztelt kódhoz és felhasználják a fejlesztési környezet által nyújtott támogatást, mint például egy egységteszt keretrendszer vagy hibakereső eszköz; és a gyakorlatban a komponens teszt során általában bevonják a fejlesztőt, aki a kódot írta. A hibákat általában már a megtaláláskor javítják anélkül, hogy a hibát formálisan kezelnék.

A komponens teszt egyik megközelítése, hogy a teszteseteket a kódolás előtt készítik el és automatizálják. Ezt nevezik „először a teszt” megközelítésnek vagy teszt-vezérelt fejlesztésnek . Ez a megközelítés rendkívül iteratív, és arra épül, hogy ciklikusan fejlesztik a teszteseteket, majd létrehozzák és integrálják a kód kisebb részeit, s addig folytatják a komponenstesztet, illetve javítják a hibákat, amíg azok sikeresen le nem futnak.

2.2.2 Integrációs teszt (K2)

Tesztbázis:

- A szoftver és a rendszer műszaki teszterve
- Architektúra

- Munkafolyamatok
- Használati esetek

A teszt jellemző tárgyai:

- Alrendszerek adatbázis implementálása
- Infrastruktúra
- Interfészek

Rendszerkonfiguráció:

- Konfigurációs adatok

Az integrációs teszt során a komponensek közötti interfészeket, egy rendszer más részeivel – mint az operációs rendszer, fájlrendszer, hardver – való kölcsönhatásait vagy a rendszerek közötti interfészeket tesztelik.

Az integrációs tesztnek több szintje lehet, és különböző méretű lehet a teszt tárgya is. Például:

1. A komponens integrációs teszttel a szoftverkomponensek közötti kölcsönhatásokat tesztelik; a komponens teszt után végzik;
2. A rendszer integrációs teszttel a különböző rendszerek, hardverek, vagy szoftverek közötti kölcsönhatásokat tesztelik; a rendszerteszt után végezhetik. Ebben az esetben előfordulhat, hogy a fejlesztő szervezet csak az interfész egyik oldalát kontrollálja, amelyet ebben az esetben kockázatként kell kezelni. A munkafolyamatként kivitelezett üzleti folyamatok rendszerek egész sorát foglalhatják magukba. A platformokat átölelő problémák nagy jelentőséggel bírhatnak.

Minél nagyobb az integráció mértéke, annál nehezebb egy adott komponenshez vagy rendszerhez visszavezetni a meghibásodásokat, ez pedig magasabb kockázati faktort jelenthet.

A szisztematikus integrációs stratégiák alapulhatnak a rendszer felépítésén (mint például felülről lefelé vagy lentről felfelé), a funkcionális feladatokon, tranzakció-feldolgozási sorrenden vagy a rendszer-, illetve a komponens egyéb jellemzőin. Annak érdekében, hogy megkönnyítsék a hibák korai felismerését és elkülönítését, az integrációnak inkrementálisnak kell lennie, nem pedig a „nagy bumm teszt” (big bang) módszernek.

Egyes nem-funkcionális jellemzők tesztje (pl. teljesítmény) is része lehet az integrációs tesztnek, csakúgy, mint a funkcionális tesztek.

Az integráció minden fázisában a tesztelők kizárólag magára az integrációra koncentrálnak. Például A modul és B modul egyesítésénél a két modul közötti kommunikáció tesztjére figyelnek, nem pedig az egyes modulok működésére, mivel azt már a komponens teszt során megtörtént. A funkcionális és a strukturális megközelítés egyaránt alkalmazható.

Ideális esetben a tesztelők ismerik a rendszer felépítését, és befolyással bírnak az integráció tervezésére. Ha az integrációs teszteket még a komponensek vagy rendszerek kifejlesztése előtt megtervezik, akkor ezeket a leghatékonyabb tesztelésnek megfelelő sorrendben lehet kifejleszteni.

2.2.3 **Rendszerteszt (K2)**

Tesztbázis:

- Rendszer –és szoftverkövetelmény specifikáció
- Használati esetek
- Funkcionális specifikáció
- Kockázatelemzés jelentések

A teszt jellemző tárgyai:

- Rendszer-, felhasználói-, és működési kézikönyv

- Rendszer konfiguráció
- Konfigurációs adatok

A rendszerteszt egy teljes rendszer/termék viselkedésével foglalkozik, mely viselkedést a fejlesztési projektben vagy programban határozták meg.

A rendszertesztnél a tesztkörnyezetnek a lehető legjobban kell hasonlítania a végfelhasználási vagy termelési környezetre, hogy minél kisebb esély legyen arra, hogy a környezet-specifikus meghibásodásokat nem találja meg a teszt.

A rendszerteszt magában foglalhat a kockázatokon és/vagy követelmény-specifikációkon, vállalati folyamatokon, használati eseteken alapuló vagy a rendszer viselkedésére, az operációs rendszerrel való kölcsönhatásokra, rendszer-erőforrásokra vonatkozó további magas szintű szöveges leírásokon, illetve modelleken alapuló teszteket.

A rendszerteszt során a rendszer funkcionális és nem-funkcionális követelményeit, valamint az adatminőségi karakterisztikákat is vizsgálni kell. A tesztelőknek olyan követelményeket is figyelembe kell venniük, melyek csak részben, vagy egyáltalán nincsenek dokumentálva. A funkcionális követelmények rendszertesztje a tesztelendő rendszer szempontjából legmegfelelőbb specifikáció alapú (feketedoboz) technikák alkalmazásával kezdődik.
Például döntési tábla hozható létre a vállalati szabályokban leírt hatások kombinációira. Ezután strukturális technikák (fehérdoboz) alkalmazhatók a tesztelés alaposságának elemzésére egy strukturális elem, mint pl. a menüstruktúra vagy a weboldal-navigáció tekintetében. (Lásd: 4. rész)

Gyakori, hogy a rendszertesztet független tesztcsapat hajtja végre.

2.2.4 **Átvételi teszt (K2)**

Tesztbázis:

- Felhasználói követelmények
- Rendszerkövetelmények
- Használati esetek
- Üzleti folyamatok
- Kockázatelemzés jelentések

A teszt jellemző tárgyai:

- Üzleti folyamatok teljesen integrált rendszeren
- Működési és karbantartási folyamatok
- Felhasználói eljárások
- Sablonok
- Jelentések
- Konfigurációs adatok

Az átvételi teszt gyakran a vevők vagy egy rendszer használóinak feladata; ugyanakkor más projektrésztvevők is bevonhatók ezen tevékenységbe.

Az átvételi teszt célja a rendszerbe, a rendszer egyes részeibe vagy a rendszer adott nem-funkcionális jellemzőibe vetett bizalom megteremtése. Az átvételi teszt elsődlegesen nem a hibák megtalálására irányul. Az átvételi teszt célja a rendszer kibocsáthatóságának és használhatóságának az elemzése, de nem minden esetben ez a teszt utolsó szintje. Egy rendszer átvételi tesztje után következhet például egy nagyobb méretű rendszer-integrációs teszt.

Az átvételi teszt az életciklus különböző szintjein is megjelenhet, például:

o Egy dobozos szoftvertermék átvételi tesztje történhet az installálásakor vagy integrálásakor.
o Egy komponens használhatóságának átvételi tesztje történhet a komponens teszt folyamán.
o Új funkcionális kiegészítés átvételi tesztje megelőzheti a rendszertesztet.

Az átvételi teszt tipikus formái a következők:

Felhasználói átvételi teszt
Általában a rendszer (megrendelő) vállalati felhasználók általi használatra való alkalmasságát ellenőrzi.

Működési (átvételi) teszt
A rendszer elfogadása a rendszeradminisztrátorok részéről, ide tartozik:

o mentés/helyreállítás tesztje;
o összeomlás utáni visszaállítás;
o felhasználói menedzsment;
o karbantartás feladatai;
o biztonsági rések periodikus ellenőrzése.

Szerződésre és előírásokra vonatkozó átvételi teszt
Megrendelésre kifejlesztett szoftverek esetében a szerződésre vonatkozó átvételi tesztet a szerződés átvételi kritériumaira hajtják végre. Az átvételi kritériumokat a szerződés megkötésekor kell meghatározni. Az előírásokra vonatkozó átvételi tesztet végezhetik betartandó előírások alapján, ilyenek lehetnek például közigazgatási-, jogi- vagy biztonsági rendelkezések.

Alfa és béta teszt
A piacra dolgozó, vagy dobozos szoftvereket előállító fejlesztők gyakran szeretnének visszajelzéseket kapni a potenciális vagy meglévő piaci ügyfeleiktől, még mielőtt a szoftverterméket kereskedelmi forgalomba hoznák. Az alfa tesztet a fejlesztő szervezetnél végzik, de nem a fejlesztő csapat. A béta tesztet a felhasználók, vagy a potenciális felhasználók saját munkaállomásokon folytatják.
Az egyes szervezetek különböző megnevezéseket használhatnak, mint működési átvételi teszt és helyszíni átvételi teszt, azokra a rendszerekre, melyeket az ügyfélnek történő átadás előtt és után is tesztelnek.

2.3 Teszttípusok (K2)	*40 perc*

Kifejezések

Feketedoboz teszt, kód lefedettség, funkcionális teszt, együttműködőképességi teszt, terheléses teszt, karbantarthatósági teszt, teljesítményteszt, hordozhatósági teszt, megbízhatósági teszt, biztonsági teszt, specifikáció alapú teszt, stressz teszt, strukturális teszt, használhatósági teszt, fehérdoboz teszt.

Háttér

A teszttevékenységek egy csoportjának lehet az a célja, hogy meghatározott indokból vagy adott tesztcélalapján ellenőrizzék a szoftverrendszert (vagy a rendszer egy részét).

Egy teszttípus egy meghatározott tesztcélra összpontosít, ami lehet egy, a szoftver által végrehajtandó függvény tesztje; egy nem-funkcionális minőségi jellemző, mint pl. megbízhatóság vagy használhatóság, a szoftver vagy rendszer struktúrája, illetve felépítése; kapcsolódhat változtatásokhoz: annak ellenőrzése, hogy a hibákat kijavították (ellenőrző teszt), illetve nem szándékolt változtatások kutatása (regressziós teszt).

A szoftver modell fejleszthető és/vagy használható a strukturális (pl. vezérlési folyam, vagy menüstruktúra modell), nem-funkcionális (pl. teljesítmény modell, használhatósági modell, biztonsági szál modell) és funkcionális tesztben (folyamatlefutási modell, egy állapotátmenet modell vagy egy egyszerűen megfogalmazott specifikáció).

2.3.1 Funkció tesztje (funkcionális teszt) (K2)

Egy rendszer, alrendszer vagy komponens által végrehajtandó funkcionalitások különböző projekt-termékekben fogalmazhatók meg, mint pl. követelmény-specifikáció, használati eset vagy funkcionális specifikáció, de előfordulhat, hogy nincsenek is dokumentálva. A funkcionalitások alatt azt értjük, „amit” a rendszer tesz.

A funkcionális tesztek alapjai a funkciók és a jellemzők (melyeket vagy a dokumentumok tartalmazzák, vagy a tesztelők ismerik őket) valamint ezeknek az adott rendszerekkel való együttműködőképességük. A funkcionális tesztek minden tesztszinten végrehajthatók (pl. a komponensek tesztjeinek alapja lehet egy komponens specifikáció).

A specifikáció alapú technikák alkalmazhatók arra, hogy a szoftver vagy rendszer funkcionalitásából tesztfeltételeket és teszteseteket származtassanak. (Lásd: 4. rész) A funkcionális teszt a szoftver külső viselkedésével foglalkozik (feketedoboz teszt).

A funkcionális teszt egyik típusa, a biztonsági teszt a rosszindulatú külső féltől származó fenyegetettségek – mint vírusok – felderítésére vonatkozó funkciókat (pl. tűzfal) vizsgálja. A funkcionális teszt egy másik típusa, az együttműködőképességi teszt a szoftvertermék azon képességét értékeli, hogy mennyire képes együttműködni egy vagy több adott komponenssel vagy rendszerrel.

2.3.2 Nem-funkcionális szoftverjellemzők tesztje (nem-funkcionális teszt) (K2)

A nem-funkcionális teszt magában foglalja többek között a teljesítménytesztet, a terheléses tesztet, a stressz tesztet, a használhatósági tesztet, a karbantarthatósági tesztet, a megbízhatósági tesztet és a hordozhatósági tesztet. Azt teszteli, hogy a rendszer „hogyan” működik.

A nem-funkcionális teszt minden tesztszinten végrehajtható. A nem-funkcionális teszt kifejezés azokat a teszteket takarja, melyek a különböző mennyiségi mutatókkal leírható rendszer- és szoftverjellemzők méréséhez szükségesek, mint pl. válaszidő a teljesítménytesztnél. Ezek a tesztek

bizonyos minőségi modellhez kapcsolhatók, mint pl. a „Software Engineering – Szoftvertermékek Minősége” (ISO 9126) által definiált mutatók.

2.3.3 A szoftver struktúrájának/felépítésének tesztje (strukturális teszt) (K2)

A strukturális (fehérdoboz) teszt minden tesztszinten végrehajtható. A strukturális technikák legjobban a specifikáció alapú technikák után használhatók annak érdekében, hogy egy adott típusú struktúra lefedettségének elemzésével támogassák a teszt lefedettségének mérését.

A lefedettség azt mutatja meg, hogy egy tesztkészlet milyen mértékben hívott meg egy struktúrát, és ezt az értéket a lefedett elemek százalékában fejezik ki. Ha a lefedettség nem 100%, további teszteket tervezhetnek, hogy a kimaradt elemeket is teszteljék, ezzel növelve a lefedettséget. A lefedettségi technikákat a 4. rész tárgyalja.

A különböző elemek - mint pl. utasítások és döntések - kód lefedettségének mérésére minden tesztszinten alkalmazhatnak eszközöket, de a gyakorlatban elsősorban a komponens tesztnél és a komponens integrációs tesztnél teszik ezt. A strukturális teszt alapja lehet a rendszer felépítése, mint például a hívási hierarchia.

A strukturális teszt megközelítéseket alkalmazhatják a rendszer-, rendszer integrációs- vagy átvételi teszt szinteken (például vállalati modelleknél vagy menüstruktúráknál).

2.3.4 Változásokhoz kapcsolódó teszt, újratesztelés és regressziós teszt (K2)

Egy hiba felismerése és javítása után a szoftvert újra kell tesztelni, ezáltal meggyőződve arról, hogy a hiba eltűnt. Ezt hívják ellenőrző tesztnek. A hibakeresés nem tesztelési, hanem fejlesztési tevékenység.

A regressziós teszt egy már letesztelt program módosítása után történő ismételt tesztelése, a változtatás(ok) eredményeként bekerült vagy feltáratlan hibák megtalálásának érdekében. Ezek a hibák lehetnek a tesztelt szoftverben vagy egy másik, a tesztelt szoftverrel kapcsolatban lévő vagy attól független szoftverkomponensben. A regressziós tesztet akkor alkalmazzák, ha a szoftvert vagy környezetét megváltoztatják. A regressziós teszt mértéke azon alapul, hogy találnak-e hibákat az előzőleg jól működő szoftverben.

A teszteknek megismételhetőknek kell lenniük, ha ellenőrző tesztre vagy regressziós teszt támogatására kívánják őket alkalmazni.

A regressziós teszt minden tesztszinten végrehajtható, és alkalmazható a funkcionális, nem-funkcionális és strukturális tesztre. A regressziós tesztkészleteket sokszor futtatják le, és többnyire lassan fejlődnek ki, így a regressziós tesztnél fontos lehet az automatizálás.

2.4 Karbantartási teszt (K2)	15 perc

Kifejezések
Hatáselemzés, karbantartási teszt.

Háttér
Bevezetésüket követően a szoftverrendszereket gyakran évekig vagy évtizedekig használják. Ez idő alatt a rendszert vagy a környezetét gyakran javítják, megváltoztatják vagy bővítik. A karbantartási tesztet létező, működő rendszeren hajtják végre, a tesztelésre a szoftver vagy rendszer módosítása, migrációja, vagy kivonása ad okot.

A módosítások lehetnek tervezett kiegészítő változtatások (pl. a kiadáshoz kapcsolódóan), javító vagy vészhelyzeti változtatások, a környezet változása, mint pl. az operációs rendszer vagy adatbázis frissítése, vagy az operációs rendszer újonnan kialakult vagy nemrég felfedezett sebezhető pontjainak védelme.

A migrációra (pl. egy platformról egy másikra) vonatkozó karbantartási tesztnek tartalmaznia kell az új környezet, illetve a megváltoztatott szoftver működési tesztjeit.

A rendszer lecseréléséhez kapcsolódó karbantartási teszt magában foglalhatja az adatmigráció tesztjét, illetve, ha hosszú adat-megőrzési periódus szükséges, az archiválás tesztjét.

A karbantartási teszt , a változtatások tesztje mellett a javítások által nem érintett részek széles körű regressziós tesztjét is magában foglalja. A karbantartási teszt mértéke függ a változtatások okozta kockázattól, a meglévő rendszer méretétől és a változtatás mértékétől.
A változtatásoktól függően a karbantartási tesztet bármely, illetve minden tesztszinten, bármely és minden teszttípusra elvégezhetik.

Hatáselemzés alatt értjük annak meghatározását, hogy a változtatások milyen befolyással lesznek a már létező rendszerre; ez segít annak eldöntésében, hogy mennyi regressziós tesztet kell végezni.

A karbantartási tesztet jelentősen megnehezíti, ha a specifikációk elavultak, vagy egyáltalán nem állnak rendelkezésre.

Irodalomjegyzék
2.1.3 CMMI, Craig, 2002, Hetzel, 1988, IEEE 12207
2.2 Hetzel, 1988
2.2.4 Copeland, 2004, Myers, 1979
2.3.1 Beizer, 1990, Black, 2001, Copeland, 2004
2.3.2 Black, 2001, ISO 9126
2.3.3 Beizer, 1990, Copeland, 2004, Hetzel, 1988
2.3.4 Hetzel, 1988, IEEE 829
2.4 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE 829

3 Statikus technikák (K2) — 60 perc

A statikus technikák tanulási céljai

A Tanulási Célok (TC) összefoglalják, hogy az egyes fejezetek ismeretanyagának elsajátításának mi a célja. .

3.1 A statikus technikák és a tesztfolyamat (K2)

TC-3.1.1 Azon szoftver projekt-termékek felismerése, melyek a különböző statikus technikákkal vizsgálhatók. (K1)

TC-3.1.2 A statikus technikák jelentőségének leírása a szoftver projekttermékek elemzésében. (K2)

TC-3.1.3 A statikus és dinamikus technikák közötti különbség kifejtése, figyelembe véve a tesztcélt, a találandó hibák típusát és ezen módszerek szerepét a szoftver életciklusa során. (K2)

3.2 A felülvizsgálat folyamata (K2)

TC-3.2.1 Egy jellemző formális felülvizsgálat fázisai, továbbá a hozzá kapcsolódó szerepek és felelősségek felidézése. (K1)

TC-3.2.2 A különböző típusú felülvizsgálatok közötti különbségek kifejtése: informális felülvizsgálat, technikai felülvizsgálat, átvizsgálás és inspekció. (K2)

TC-3.2.3 A felülvizsgálat sikeres végrehajtását befolyásoló tényezők ismerete. (K2)

3.3 Statikus elemzés eszközökkel (K2)

TC-3.3.1 A statikus elemzés során azonosított típushibák felidézése, összehasonlítása a felülvizsgálatok és a dinamikus teszt során talált hibákkal. (K1)

TC-3.3.2 A statikus elemzés főbb előnyeinek felsorolása példákon keresztül. (K2)

TC-3.3.3 Azon tipikus kód- és tervezési hibák felsorolása, melyeket a statikus elemző eszközök felismerhetnek. (K1)

3.1 A statikus technikák és a tesztfolyamat (K2)	*15 perc*

Kifejezések
Dinamikus teszt, statikus teszt, statikus technika.

Háttér
A dinamikus teszttel ellentétben – melyhez a szoftver futtatása szükséges –a statikus teszttechnikák a kód- vagy a projekt dokumentációk manuális vizsgálatára (felülvizsgálatok) és automatizált elemzésére (statikus elemzés) támaszkodnak.

A felülvizsgálat a szoftver projekttermékek (beleértve a kódot) tesztjének egy módja, s jóval a dinamikus tesztek futtatása előtt végrehajtható. A felülvizsgálatok során, az életciklus korai szakaszában megtalált hibák (pl. a követelményekben talált hibák) javítása gyakran jóval gazdaságosabb, mint azoké, melyeket a tesztek futtatása során találnak meg (pl. hibák a követelményekben).

A felülvizsgálatot teljesen manuálisan is végre lehet hajtani, de léteznek a felülvizsgálatot támogató eszközök. A legfőbb manuális tevékenység a projekttermékek megvizsgálása és észrevételek készítése. Bármely szoftver projekttermék felülvizsgálható, úgy mint a követelmény specifikációk, teszterv specifikációk, kód, műszaki teszttervek, tesztspecifikációk, tesztesetek, tesztszkriptek, felhasználói útmutatók vagy weboldalak.

A felülvizsgálatok előnyei: hibák korai megtalálása és javítása, fejlesztési termelékenység javítása, fejlesztési időtartamok csökkenése, tesztköltségek és idő csökkenése, a teljes élettartam költségeinek csökkenése, kevesebb hiba, jobb kommunikáció. A felülvizsgálatokkal olyan hiányosságok találhatók meg, például a követelményekben, melyeket a dinamikus teszttel valószínűleg nem találnának meg.

A felülvizsgálatoknak, a statikus elemzésnek és a dinamikus tesztnek azonosak a céljaik: a hibák azonosítása. Ezek a módszerek kiegészítik egymást: különböző technikák különböző típusú hibák hatékony megtalálását segítik elő. A dinamikus teszttel összehasonlítva, a statikus technikák inkább a meghibásodások okát találják meg, nem magukat a meghibásodásokat.

Az alábbi típushibákat könnyebb felülvizsgálattal megtalálni, mint dinamikus teszttel: szabványoktól való eltérések, követelményekkel kapcsolatos hibák, műszaki teszttervezési hibák, karbantarthatóság hiánya és hibás interfész specifikációk.

3.2 A felülvizsgálat folyamata (K2)	*25 perc*

Kifejezések

Belépési feltétel, formális felülvizsgálat, informális felülvizsgálat, inspekció, metrika, moderátor, egyenrangú felülvizsgálat, felülvizsgáló, jegyzőkönyv vezető, technikai felülvizsgálat, átvizsgálás.

Háttér

A felülvizsgálatok típusai a nagyon informálistól (amely az írásos utasítások hiányával jellemezhetők) a nagyon formálisig (jól strukturált és szabályozott) terjednek. A felülvizsgálati folyamat formalitása összefügg az olyan tényezőkkel, mint a fejlesztési folyamat állapota, jogi, rendelkezési követelmények, illetve az audit nyomvonal szükségessége.

A felülvizsgálat végrehajtási módja a megegyezés szerinti célkitűzéstől függ (pl. hibák megtalálása, megismerése vagy megvitatása és döntések meghozása konszenzus alapján).

3.2.1 Formális felülvizsgálat fázisai (K1)

Egy tipikus formális felülvizsgálat fázisai:

1. Tervezés:
   - a felülvizsgálati feltételek meghatározása
   - a személyek kiválasztása
   - a szerepkörök meghatározása
2. A belépési és kilépési feltétel meghatározása formálisabb felülvizsgálat esetén (pl. inspekció)
   - a felülvizsgálandó dokumentumrészek kiválasztása.
3. Kezdő lépések
   - a dokumentumok kiosztása
   - a célok, a felülvizsgálati folyamat és a dokumentumok elmagyarázása a résztvevőknek;
4. Belépési feltétel ellenőrzése (formálisabb felülvizsgálatoknál).
5. Egyéni felkészülés
   - minden résztvevő felülvizsgálja a dokumentumot, ezáltal felkészül a felülvizsgálatra
6. A lehetséges hibák, kérdések, megjegyzések feljegyzése
7. Vizsgálat/értékelés/feljegyzés (felülvizsgálati megbeszélés)
   - megbeszélés, vagy naplózás, dokumentált eredményekkel vagy feljegyzésekkel (formálisabb felülvizsgálatoknál).
   - A hibák feljegyzése, javaslatok készítése és döntéshozatal a hibákkal kapcsolatban.
8. Vizsgálat/értékelés, illetve feljegyzés a fizikai találkozó alatt, vagy a csoportos elektronikus kommunikáció nyomonkövetése
9. Átdolgozás:
10. A talált hibák kijavítása (jellemzően a szerző végzi)
    - A hibák frissített állapotának feljegyzése (formális felülvizsgálatok során)
11. Utókövetés:
    - a hibák javításának ellenőrzése
    - metrikák gyűjtése
12. A kilépési feltételek ellenőrzése (formálisabb felülvizsgálatoknál).

3.2.2 Feladatok, felelősségi körök (K1)

Egy tipikus formális felülvizsgálat a következő feladatköröket tartalmazza:

- Menedzser: dönt a felülvizsgálatok végrehajtásával kapcsolatban, kijelöli az időpontokat a projekt ütemtervében, s dönt arról, hogy a felülvizsgálat célkitűzéseit teljesítették-e.

- Moderátor: vezeti a dokumentum(ok) felülvizsgálatát, ezzel együtt a felülvizsgálat tervezését, a megbeszélés lebonyolítását és a találkozó utáni visszaellenőrzést. A moderátor szükség esetén közvetít a különböző álláspontok között, és gyakran rajta múlik a felülvizsgálat sikere.
- Szerző: a felülvizsgálandó dokumentum(ok) írója, vagy fő felelőse.
- Felülvizsgálati résztvevők: meghatározott technikai vagy vállalati háttérrel rendelkező személyek (nevezik őket vizsgálónak, vagy felülvizsgálónak is), akik a szükséges felkészülés után meghatározzák és bemutatják a termék felülvizsgálatának eredményeit (pl. hibákat). A felülvizsgálókat úgy kell kiválasztani, hogy különböző szempontokat és szerepköröket képviseljenek; minden felülvizsgálati megbeszélésen részt kell venniük.
- Jegyzőkönyv vezető (vagy írnok): minden, a találkozó alatt felmerült ügyet, problémát és nyitott kérdést dokumentál.

A szoftverek és más projekttermékek felülvizsgálatai hatékonyabbá tehetők a dokumentumok különböző szempontok szerint történő vizsgálatával és ellenőrző listák használatával. Ilyen lehet például a felhasználói, karbantartói, tesztelői vagy a műveletekre vonatkozó ellenőrző lista; vagy a tipikus követelmény problémák ellenőrző listája.

3.2.3 Felülvizsgálatok típusai (K2)

Egy dokumentum több felülvizsgálat tárgya is lehet. Ebben az esetben a felülvizsgálatoknak különböző sorrendjük lehet. Például végezhetnek informális felülvizsgálatot egy technikai felülvizsgálat előtt, vagy az ügyfelekkel végzett átvizsgálás előtt inspekciót hajthatnak végre a követelmény specifikáción. Az általános felülvizsgálat típusok fő jellemzői, opciói és céljai a következők:

Informális felülvizsgálat
Legfőbb jellemzői:

- nincs formális eljárás;
- lehetséges páros programozás vagy technikai irányítás a tervek és a kód felülvizsgálatára;
- a dokumentáció opcionális;
- a hatékonyság a felülvizsgálatot végzőtől függően változó lehet;
- fő cél: költségkímélő módszerrel némi eredményt elérni.

Átvizsgálás
Legfőbb jellemzői:

- a találkozót a szerző vezeti;
- forgatókönyvek, képzeletbeli futtatások, egyenrangú csoport;
- nem zárt szakaszok;
- opcionális: a felülvizsgálatot végzők találkozó előtti felkészülése, felülvizsgálati jelentés, az eredmények listája, jegyzőkönyvvezető (a szerző nem lehet ez a személy);
- a gyakorlatban a formája a nagyon informálistól a nagyon formálisig terjedhet;
- fő célok: megismerés, megértés, hibák megtalálása.

Technikai felülvizsgálat
Legfőbb jellemzői:

- dokumentált, definiált hibakeresési folyamat, egyenrangú kollégák és technikai szakértők részvételével;
- egyenrangú felülvizsgálatként is végrehajtható, a menedzsment részvétele nélkül;
- ideális esetben képzett moderátor vezeti (nem a szerző);
- találkozó előtti felkészülés;
- opcionális: ellenőrző lista használata, felülvizsgálati jelentés, eredmények listája és a menedzsment részvétele;
- a gyakorlatban a formája a nagyon informálistól a nagyon formálisig terjedhet;
- fő célok: megvitatás, döntéshozatal, alternatívák értékelése, hibák megtalálása, technikai problémák megoldása és annak ellenőrzése, hogy a specifikációk megfelelnek-e a szabványoknak.

Inspekció
Legfőbb jellemzői:

- képzett moderátor vezeti (nem a szerző);
- általában egyenrangú vizsgálat;
- definiált szerepkörök;
- metrikákat is magában foglal;
- formális eljárás, a belépési és kilépési feltételekre vonatkozó szabályok és ellenőrző listák alapján;
- találkozó előtti felkészülés;
- inspekció jelentés, eredmények listája;
- formális utókövetési eljárás;
    - opcionálisan: folyamatfejlesztési komponensek
- opcionálisan: felolvasó;
- fő cél: hibák megkeresése.

Az átvizsgálást, technikai felülvizsgálatot és az inspekciót végre lehet hajtani egyenrangú csoporton belül – ugyanazon szervezeti szinthez tartozó kollégák részvételével. Ezt a típusú felülvizsgálatot nevezik „egyenrangú felülvizsgálatnak”.

3.2.4 Felülvizsgálatok sikerességi tényezői (K2)

A felülvizsgálatok sikerességi tényezői közé tartoznak a következők:

- Minden felülvizsgálatnak van világosan meghatározott célkitűzése.
- A felülvizsgálati célok eléréséhez a megfelelő személyeket kell bevonni.
- A tesztelők értékes felülvizsgálók, akik egyrészt hozzájárulnak a felülvizsgálathoz, másrészt tanulnak az eredményből, amely hozzásegíti őket, hogy hamarabb elkészítsék a teszteket.
- A hibák megtalálását eredménynek kell tekinteni, és tárgyilagosan kezelni őket.
- Foglalkozni kell az emberi és pszichológiai tényezőkkel (pl. a szerző számára pozitív tapasztalatként jelenjen meg a felülvizsgálat).
- A felülvizsgálatot bizalmi légkörben kell lebonyolítani, az eredményt nem szabad felhasználni a résztvevők értékeléséhez.
- Olyan felülvizsgálati technikák alkalmazása, melyek illeszkednek a szoftver projekt-termékek típusához és szintjéhez valamint a felülvizsgálatot végzőkhöz.
- Ha lehetséges, a hibák azonosításának hatékonyabbá tételéhez ellenőrző listák vagy szerepkörök alkalmazhatók.
- A felülvizsgálati technikákról képzést tartanak, különösen a formálisabb technikák esetében, mint pl. az inspekció.
- A menedzsment támogatja a felülvizsgálati eljárást (pl. a projekt ütemtervében megfelelő időt biztosít a felülvizsgálati tevékenységekre).
- Hangsúlyt fektetnek a tanulásra és a folyamat javítására.

3.3 Statikus elemzés eszközökkel (K2)	*20 perc*

Kifejezések
Fordítóprogram, komplexitás, vezérlési folyam, adatfolyam, statikus elemzés.

Háttér
A statikus elemzés célja megtalálni a hibákat a szoftver forráskódjában és a szoftvermodellekben. A statikus elemzést az eszköz által vizsgált szoftver futtatása nélkül végzik, míg a dinamikus tesztnél futtatják a szoftverkódot. A statikus elemzés felderíthet olyan hibákat, melyeket teszttel nehéz megtalálni. A felülvizsgálatokhoz hasonlóan a statikus elemzéssel inkább programhibákat, mint meghibásodásokat találhatunk. A statikus elemző eszközök a programkódot (pl. vezérlési folyam és adatfolyam) valamint a generált kimenetet (mint pl. HTML vagy XML) elemzik.

A statikus elemzés előnyei:
- A hibák korai felismerése a teszt végrehajtása előtt.
- Korai figyelmeztetések a kód vagy a terv gyanús tényezőiről a metrikák számításának segítségével, például a magas komplexitás mérésével.
- Olyan hibák azonosítása, melyeket dinamikus teszttel nehéz megtalálni.
- Függőségi viszonyok és összeférhetetlenségek felismerése a szoftvermodellekben, például linkek.
- A kód és a terv jobb karbantarthatósága.
- Hibák megelőzése fejlesztési tapasztalatok használatával.

A statikus elemző eszközök által felismert tipikus hibák:
- meghatározatlan értékű változóra való hivatkozás
- összeférhetetlen interfész modulok és komponensek között
- soha nem használt változók
- elérhetetlen kód
- hiányzó vagy hibás logika (potenciális végtelen ciklusok)
- túl komplikált struktúra
- programozási szabványok megsértése
- biztonsági rések
- szintaxis megsértése a kódban vagy a szoftvermodellekben.

A statikus elemző eszközöket általában a fejlesztők használják (előre meghatározott szabályoknak és programozási szabványoknak való megfelelés ellenőrzése céljából) a komponens- és integrációs teszt előtt és után, illetve a tervezők a szoftver-modellezés során. A statikus elemző eszközök számos figyelmeztető üzenetet generálhatnak, melyeket megfelelően kell kezelni az eszköz hatékony használata érdekében.

A fordítók a statikus elemzésben és a metrikák számításában is támogatást nyújthatnak.

Irodalomjegyzék
3.2 IEEE 1028
3.2.2 Gilb, 1993, van Veenendaal, 2004
3.2.4 Gilb, 1993, IEEE 1028
3.3 Van Veenendaal, 2004

4 Műszaki teszttervezési technikák (K4) — 285 perc

Tanulási célok a műszaki teszttervezési technikákhoz

A Tanulási Célok (TC) összefoglalják, hogy az egyes fejezetek ismeretanyagának elsajátításának mik a céljai.

4.1 A teszt fejlesztési folyamata (K2)

TC-4.1.1 A műszaki teszterv specifikáció, a teszteset specifikáció és a teszteljárás specifikáció közötti különbség meghatározása. (K2)

TC-4.1.2 A tesztfeltétel, teszteset, teszteljárás kifejezések összehasonlítása. (K2)

TC-4.1.3 A tesztesetek minőségének meghatározása. Annak megítélése, hogy:
- egyértelműen visszavezethetők-e a követelményekre;
- tartalmaznak-e elvárt eredményt. (K2)

TC-4.1.4 A tesztesetek**Error! Bookmark not defined.** átalakítása jól strukturált teszteljárás specifikációvá a tesztelők ismereteinek megfelelő szintű részletességgel. (K3)

4.2 A műszaki teszttervezési technikák kategóriái (K2)

TC-4.2.1 A specifikáció alapú (feketedoboz) és a struktúra alapú (fehérdoboz) teszteset tervezési megközelítés használhatóságának indoklása, s ezek leggyakoribb technikáinak felsorolása. (K1)

TC-4.2.2 A specifikáció alapú teszt, a struktúra alapú teszt és a tapasztalat alapú teszt jellemzőinek, valamint az ezek közötti különbségek bemutatása. (K2)

4.3 Specifikáció alapú, vagy feketedoboz technikák (K3)

TC-4.3.1 Tesztesetek írása meghatározott szoftvermodellekből a következő műszaki teszttervezési technikák alkalmazásával: (K3)
- ekvivalencia particionálás;
- határérték elemzés;
- döntési tábla teszt;
- állapotátmenet teszt.

TC-4.3.2 A négy tesztmódszer fő céljának ismerete, valamint annak ismerete, hogy milyen szintű és típusú tesztnél lehet alkalmazni az egyes technikákat, hogyan lehet mérni a lefedettséget. (K2)

TC-4.3.3 A használati eset teszt fogalmának és előnyeinek ismerete. (K2)

TC-4.3.4 Az utasítás és döntési lefedettség értékelése a kilépési feltétel figyelembe vételével (K4)

4.4 Struktúra alapú, vagy fehérdoboz technikák (K4)

TC-4.4.1 A kód lefedettség fogalmának és jelentőségének bemutatása. (K2)

TC-4.4.2 Az utasítás- és döntési lefedettség fogalmának, valamint annak bemutatása, hogy ezek nem csak a komponens teszt szintjén alkalmazhatók (pl. üzleti folyamatoknál a rendszer szintjén). (K2)

TC-4.4.3 Tesztesetek írása megadott vezérlési folyamokból a következő műszaki teszttervezési technikák alkalmazásával:
- utasítás szintű teszt;
- döntési teszt. (K3)

TC-4.4.4 Az utasítás- és döntési lefedettség teljes körű elemzése a kilépési feltételek figyelembevételével. (K4)

4.5 Tapasztalat alapú technikák (K2)

TC-4.5.1 Az intuíció, tapasztalat és gyakori hibák ismerete alapján való teszteset készítés okainak indoklása. (K1)

TC-4.5.2 A tapasztalat alapú technikák összehasonlítása a specifikáció alapú teszttechnikákkal. (K2)

4.6. Teszttechnikák kiválasztása (K2)

TC-4.6.1 A műszaki teszttervezési technikák értékelése a környezettől, illetve a tesztbázistól függően, figyelembe véve a modelleket és a szoftver karakterisztikákat. (K2)

4.1 A teszt fejlesztési folyamata (K2)	*15 perc*

Kifejezések
Teszteset specifikáció, műszaki teströterv, tesztvégrehajtási ütemterv, teszteljárás specifikáció, tesztszkript, nyomonkövethetőség.

Háttér
Az ebben a fejezetben bemutatott tesztkészítési folyamatot különböző módokon lehet végrehajtani, a kevéssé, vagy egyáltalán nem dokumentált, nagyon informális módtól a nagyon formálisig (mint azt lentebb tárgyaljuk). A formalitás szintje a tesztelés tényezőitől függ, ide tartozik a tesztelési és a fejlesztési folyamat érettsége, az időbeli megkötések és a résztvevő személyek.

A tesztelemzés során történik a tesztbázis dokumentáció elemzése annak érdekében, hogy meghatározzuk, hogy mit kell tesztelni, azaz megállapításra kerülnek a tesztfeltételek. A tesztfeltétel olyan elem, vagy esemény, melyet egy vagy több teszteset ellenőriz (pl. egy függvény, tranzakció, minőségi jellemző, vagy strukturális elem).

A nyomonkövethetőség kialakítása a tesztfeltételektől vissza a specifikációkig és követelményekig lehetővé teszi a hatékony hatáselemzést a követelmények változása esetén, valamint a teszthalmazra vonatkozó követelmény lefedettség meghatározását is. A tesztelemzés során a részletes tesztelési megközelítést kivitelezzük, hogy azután kiválaszthassuk az alkalmazandó műszaki teszttervezési technikákat, többek között a felismert kockázatok alapján (a kockázatelemzést bővebben az 5. fejezet tárgyalja).

A teszt műszaki tervezése során a tesztesetek és tesztadatok megalkotása és meghatározása történik. Egy teszteset a következőkből áll: bemeneti értékek halmaza, végrehajtási előfeltételek, elvárt eredmények és végrehajtás utáni feltételek, melyeket bizonyos tesztfeltétel(ek) lefedésére fejlesztenek. A „Szoftverteszt Dokumentáció Szabvány” (IEEE 829) tárgyalja a (tesztfeltételeket is tartalmazó) műszaki teströterv specifikációk és a teszteset specifikációk tartalmát.

Egy teszteset specifikációjának részeként meg kell határozni az elvárt eredményeket, melyeknek tartalmazniuk kell kimeneti értékeket, az adatok és állapotok változásait, valamint a teszt minden más következményét is. Ha az elvárt eredmények nincsenek meghatározva, akkor lehet, hogy egy kézenfekvő, de hibás eredmény jó eredményként lesz értelmezve. Az elvárt eredményeket lehetőleg a teszt végrehajtása előtt meg kell határozni.

A teszt megvalósítása alatt a tesztesetek kifejlesztése, megvalósítása, priorizálása és teszteljárás specifikációba rendezése történik (IEEE STD 829-1998). A teszteljárás (vagy a manuális tesztszkript) részletezi a tesztvégrehajtás műveleteinek sorrendjét. Ha a teszteket tesztvégrehajtási eszközzel futtatjuk, akkor a műveletek sorrendjét a tesztszkriptben kell meghatározni (ami egy automatizált teszteljárás).

A különböző teszteljárásokból és automatizált tesztszkriptekből ezután tesztvégrehajtási ütemterv készül, mely meghatározza a különböző teszteljárások és esetenként automatizált tesztszkriptek végrehajtási sorrendjét. A tesztvégrehajtási ütemtervnél figyelembe kell venni olyan tényezőket, mint a regressziós teszt, a priorizálás, a technikai és logikai függőségi viszonyok.

4.2 Műszaki teszttervezési technikák kategóriái (K2)	*15 perc*

Kifejezések

Feketedoboz teszttervezési technika , tapasztalat alapú teszttervezési technika, specifikáció alapú teszttervezési technika, struktúra alapú teszttervezési technika, fehérdoboz teszttervezési technika.

Háttér

A műszaki teszttervezési technikák célja a tesztfeltételek és a tesztesetek meghatározása.

A teszttechnikák hagyományos megkülönböztetése a feketedoboz és a fehérdoboz megnevezés. A feketedoboz technikák (más néven a specifikáció alapú módszerek) segítségével tesztfeltételek, tesztesetek vagy tesztadatok nyerhetők és választhatók ki a tesztbázis dokumentáció elemzése alapján. Ez lehet funkcionális vagy nemfunkctionális teszt is. A feketedoboz technika – definíció szerint - nem használ semmilyen, a tesztelendő komponens vagy rendszer belső felépítésére vonatkozó információt.. A fehérdoboz technikák (nevezik őket még strukturális vagy struktúra alapú technikáknak) a komponens vagy rendszer struktúrájának elemzésén alapulnak. Mind a feketedoboz, mind a fehérdoboz tesztek alapulhatnak a fejlesztők, tesztelők, illetve a felhasználók tapasztalatain, hogy meghatározzák, mit kellene tesztelni.

Egyes technikák egyértelműen besorolhatók az egyik kategóriába; másoknál több kategória elemei megtalálhatók.
A jelen tananyag a specifikáció alapú, vagy tapasztalat alapú megközelítést feketedoboz technikaként, míg a struktúra alapú műszaki teszttervezési technikát fehérdoboz technikaként említi.

A specifikáció alapú technikák közös jellemzői:

- Formális vagy informális modellek alkalmazása a szoftvert, vagy annak komponenseit érintő megoldandó probléma specifikációjára.
- Ezekből a modellekből módszeresen tesztesetek nyerhetők.

A struktúra alapú technikák közös jellemzői:

- A szoftver felépítésével kapcsolatos információk használatával történik a tesztesetek származtatása, például: kód és részletes műszaki teszterv alapján.
- A szoftver lefedettségének mértékét a meglévő teszteseteken lehet mérni, a lefedettség növelése érdekében módszeresen származtathatók további tesztesetek.

A tapasztalat alapú technikák közös jellemzői:

- Az emberek tudása és tapasztalata szolgál a tesztesetek létrehozásának alapjául.
- A tesztelők, fejlesztők, felhasználók és más projektrésztvevők ismerete a szoftverről, annak használatáról és környezetéről is információként szolgál.
- A valószínűsíthető hibákkal és azok eloszlásával kapcsolatos ismeretetek szintén információként szolgálnak.

4.3 Specifikáció alapú, vagy feketedoboz technikák (K3)	*150 perc*

Kifejezések
Határérték elemzés, döntési tábla teszt, ekvivalencia particionálás, állapotátmenet teszt, használati eset teszt.

4.3.1 Ekvivalencia particionálás (K3)

A szoftver, vagy a rendszer bemeneteit olyan csoportokra kell osztani, melyek valószínűleg hasonlóan fognak viselkedni, így bizonyára ugyanúgy kerülnek feldolgozásra. Ekvivalencia partíciók (vagy osztályok) léteznek az érvényes – pl. elfogadandó - adatokra és az érvénytelen – elutasítandó – adatokra is. A partíciók alkalmazhatók továbbá kimenetekre, belső értékekre, idővel kapcsolatos értékekre (pl. egy esemény előtt vagy után) valamint interfész paraméterekre (pl. tesztelendő integrált komponensekre az integrációs teszt során). A partíciók lefedésére tesztek tervezhetők Az ekvivalencia particionálás minden tesztszinten alkalmazható.

Az ekvivalencia particionálás, mint technika használható a bemeneti és kimeneti lefedettség elérésére. Alkalmazható a felhasználó által megadott bemenetekre, rendszerhez kapcsolódó interfészek bemenetére vagy az integrációs tesztnél interfész paraméterre.

4.3.2 Határérték elemzés (K3)

Az ekvivalencia partíciók határain kisebb az esély a helyes viselkedésre, mint a partíción belül, ezért ott a tesztek nagy eséllyel találnak hibákat. Egy partíció maximális és minimális értékei a határértékek. Az érvényes partíciók határértéke érvényes határérték; míg az érvénytelen partíciók határa az érvénytelen határérték. A tesztek tervezhetők úgy, hogy lefedjék az érvényes és az érvénytelen határértékeket is. A tesztesetek tervezésénél minden határértékhez választani kell egy tesztet.

A határérték elemzés minden tesztszinten alkalmazható. Viszonylag egyszerű az alkalmazása, hibatalálási képessége magas. A részletes specifikációk hasznosak az érdekes határértékek megállapításakor..

Ezt a technikát sokszor az ekvivalencia particionálás kiterjesztésének tekintik. Használható ekvivalencia osztályokra a képernyős felhasználói bevitelnél, időtartamokra (pl. időtúllépés, tranzakció sebességére vonatkozó követelménye) vagy táblázat kiterjedésére (pl. a táblázat mérete 256*256). .

4.3.3 Döntési tábla teszt (K3)

A döntési táblák jól használhatók logikai feltételeket tartalmazó rendszerkövetelmények felvételére és a belső rendszerfelépítés dokumentálására. Alkalmazhatók a rendszer által megvalósított komplex üzleti szabályok rögzítésére. Döntési táblák létrehozásakor elemzik a specifikációt, és meghatározzák a rendszer feltételeit és műveleteit. A bemeneti feltételek és műveletek leggyakrabban úgy vannak megadva, hogy csak igaz vagy hamis értékek lehetnek (Boolean). A döntési tábla tartalmaz kiváltó okokat, melyek gyakran az összes bemeneti feltétel igaz, illetve hamis értékeinek kombinációi, valamint a feltételek kombinációihoz tartozó eredményeket. A tábla minden oszlopa egy üzleti szabályhoz tartozik, amely megadja a feltételek egy egyedi kombinációját, ez pedig az ahhoz a szabályhoz tartozó műveletek végrehajtását eredményezi. A döntési tábla tesztnél leggyakrabban alkalmazott lefedettségi szabvány szerint legalább egy tesztnek kell lennie oszloponként, amely általában a kiváltó okok összes kombinációját lefedi.

A döntési tábla teszt legfőbb előnye az, hogy a feltételek olyan kombinációit hozza létre, melyeket a teszt során esetleg nem érintenének. Minden olyan helyzetben alkalmazható, ahol a szoftver által végrehajtott művelet különböző logikai döntéseken alapul.

4.3.4 **Állapotátmenet teszt (K3)**

A rendszer az adott jellemzőitől vagy a megelőző eseményektől (az állapottól) függően különböző válaszokat adhat. Ebben az esetben a rendszer helyzetét állapotátmenet diagrammal lehet bemutatni. Ennek segítségével a tesztelő vizsgálhatja a szoftvert a különböző állapotaiban, az állapotok közötti átmeneteket, az állapotváltozásokat (átmeneteket) kiváltó bemeneteket és eseményeket valamint a műveleteket, melyek az átmenetek következményei. A tesztelt rendszer vagy objektum állapotai elkülöníthetők, beazonosíthatók és véges számúak. Az állapottábla megmutatja az állapotok és a bemenetek közötti összefüggéseket, s ebben a lehetséges érvénytelen átmenetek külön megjelölhetők. A tesztek tervezhetők az állapotok egy tipikus sorrendjének lefedésére, minden állapot lefedésére, minden átmenet végrehajtására, az átmenetek adott sorrendjeinek végrehajtására vagy az érvénytelen átmenetek tesztjére.

Az állapotátmenet-tesztet gyakran használják a beágyazott szoftvereknél és általánosságban a műszaki automatizálásban. Ez a technika jól alkalmazható a meghatározott állapotokkal rendelkező üzleti objektumok modellezésére vagy képernyő-dialógus folyamok (pl. internetes alkalmazások vagy üzleti forgatókönyvek) tesztelésére.

4.3.5 **Használati eset teszt (K2)**

A teszteket használati esetekből kiindulva határozhatják meg. Egy használati eset a szereplők – ide tartoznak a felhasználók és a rendszer – közötti kölcsönhatásokat írja le, melyek eredményeként egy a rendszer felhasználója számára értékes eredmény jön létre. Minden használati eset rendelkezik előfeltételekkel, melyeknek teljesülniük kell a használati eset sikeres működéséhez. Minden használati eset végrehajtás utáni feltételekkel ér véget, ezek a használati eset lezárása után megfigyelhető eredményeket és a rendszer végső állapotát tartalmazzák. A használati esetnek általában van egy fő (vagyis legvalószínűbb) forgatókönyve, valamint esetenként alternatív mellékágai.

A használati esetek leírják az „eljárási folyamatot” a rendszer valószínűsíthető használata alapján, így a használati esetekből származtatott tesztesetek a leghasznosabbak a rendszer valós használata folyamán fellépő hibák felderítésére. A használati esetek (gyakran forgatókönyvekként említik őket) nagyon hasznosak átvételi tesztek tervezésére, amelyekben az ügyfél/felhasználó részt vesz. Alkalmazhatók a különböző komponensek kölcsönhatása és interferenciája által okozott integrációs hibák felderítésére, melyeket az egyes komponens teszt nem találnának meg. A használati esetek alapján tervezett teszteset tervezést ötvözni lehet más, specifikáció alapú technikákkal.

4.4 Struktúra alapú, vagy fehérdoboz technikák (K4)	*60 perc*

Kifejezések

Kód lefedettség, döntési lefedettség, utasítás lefedettség, struktúra alapú teszt.

Háttér

A struktúra alapú – más néven fehérdoboz - teszt a szoftver vagy rendszer ismert struktúráján alapul, mint ahogy a következő példákban is látható:

- Komponens szint: a struktúra maga a kód, vagyis utasítások, döntések, elágazások.
- Integrációs szint: a struktúra lehet egy hívási fa (egy diagram, melyben modulok hívnak más modulokat).
- Rendszerszint: a struktúra lehet egy menüstruktúra, egy üzleti folyamat vagy egy weboldal struktúra.

Ez a fejezet két, kódhoz kapcsolódó, kód lefedettségre irányuló, utasításokra és döntésekre alapuló strukturális technikát mutat be. A döntési tesztnél vezérlési folyam diagramot lehet alkalmazni az egyes döntésekhez tartozó alternatívák szemléltetésére.

4.4.1 Utasítás szintű teszt és lefedettség (K4)

A komponens tesztnél az utasítás szintű lefedettség annak értékelése, hogy valamely teszteset készlet a futtatható utasítások hány százalékát érintette. Az utasítás szintű tesztnél a származtatott tesztesetek meghatározott utasításokat hajtanak végre, általában az utasítás lefedettség növelése érdekében.

4.4.2 Döntési teszt és lefedettség (K4)

A döntési lefedettség – amely összefüggésben áll az elágazás teszttel – annak értékelése, hogy valamely teszteset készlet a döntési eredmények (pl. egy If utasítás Igaz vagy Hamis lehetőségei) hány százalékát hajtotta végre. A döntési tesztnél a származtatott tesztesetek speciális döntési eredményeket hajtanak végre, általában a döntési lefedettség növelése érdekében.

A döntési lefedettséget a megtervezett, illetve végrehajtott döntési eredmények száma és a tesztelendő kódban található összes lehetséges döntési eredmény hányadosa határozza meg.

A döntési teszt a vezérlési folyam tesztelés egy formája, mivel a döntési pontokkal egy sajátos vezérlési folyamot hoz létre. A döntési lefedettség magasabb rendű az utasítás-lefedettségnél: 100%-os döntési lefedettség esetén garantált a 100%-os utasítás-lefedettség, aminek fordítottja nem igaz.

4.4.3 Egyéb struktúra alapú technikák (K1)

A strukturális lefedettségnek a döntési lefedettségnél magasabb rendű szintjei is vannak, mint például a feltétel lefedettség vagy a többszörös feltétel lefedettség.

A lefedettség fogalma alkalmazható más tesztszinteken is (pl. az integrációs szinten), ahol a modul-, komponens- vagy osztálylefedettség kifejezhető azzal, hogy valamely teszteset készlet a modulok, komponensek vagy osztályok hány százalékát érintette.

A kód strukturális tesztjében hasznos az eszköztámogatás.

4.5 Tapasztalat alapú technikák (K2)	*30 perc*

Kifejezések
Felderítő teszt, hibatámadás .

Háttér
A tapasztalat-alapú teszt esetén a tesztek a tesztelő szaktudásából és intuíciójából, valamint a hasonló alkalmazásokkal és technológiákkal kapcsolatos tapasztalataiból származnak. Ha a szisztematikus technikák kiegészítéseként alkalmazzák őket, akkor ezek a technikák hasznosak lehetnek olyan speciális tesztek meghatározásánál, melyeket formális technikákkal nehéz elérni, különösen akkor, ha formálisabb megközelítések után kerülnek alkalmazásra. Ezen technika hatékonysága azonban nagy eltéréseket mutathat, mivel függ a tesztelők tapasztalatától. Gyakran használt tapasztalat alapú technika a hibasejtés. A tesztelők általában a tapasztalat alapján előre sejtik a hibákat. A hibasejtés technika egyik strukturált megközelítése: elkészítik a lehetséges hibák listáját, s megtervezik az ezen hibákat előidéző teszteket. Ezt a szisztematikus megközelítést nevezik támadásnak. A programhibák és meghibásodások listája elkészíthető tapasztalat alapján, a hibákról és a meghibásodásokról rendelkezésre álló adatok alapján valamint a szoftver helytelen működésével kapcsolatos ismeretekből.

A felderítő teszt egy időben történő műszaki teszttervezést, tesztvégrehajtást, tesztnaplózást és tanulást jelent a tesztcélokat tartalmazó tesztelési elképzelések alapján és adott időkeretben végrehajtva. Ez a megközelítés akkor különösen hasznos, ha kevés, vagy hiányos specifikáció áll rendelkezésre, kevés az idő, vagy ha más, formálisabb tesztet szándékoznak bővíteni, kiegészíteni. Segítséget nyújthat a tesztfolyamat ellenőrzésében, hogy megbizonyosodjanak a legjelentősebb hibák megtalálásáról.

4.6 Teszttechnikák kiválasztása (K2)	*15 perc*

Kifejezések
Nincsenek jellemző kifejezések.

Háttér
A teszttechnika kiválasztásánál számos tényezőt vesznek figyelembe, ilyenek a rendszer típusa, a szabályzó rendelkezések, ügyfél követelményei, illetve a szerződésben foglalt követelmények, a kockázat szintje és típusa, a tesztcél, a rendelkezésre álló dokumentáció, a tesztelők ismeretei, az idő és a költségvetés, a fejlesztési életciklus, a használati eset modellek és a tapasztalat a talált hibák típusairól.

Egyes tesztek jobban használhatók adott szituációkban, illetve tesztszinteken; mások minden tesztszinten alkalmasak.

A tesztesetek létrehozásakor a tesztelők különböző teszttechnikákat használnak, amelyek mind folyamat-, mind szabály- vagy adatvezérelt technikák lehetnek, hogy biztosítsák a teszt tárgyának megfelelő lefedettségét.

Irodalomjegyzék
4.1 Craig, 2002, Hetzel, 1988, IEEE 829
4.2 Beizer, 1990, Copeland, 2004
4.3.1 Copeland, 2004, Myers, 1979
4.3.2 Copeland, 2004, Myers, 1979
4.3.3 Beizer, 1990, Copeland, 2004
4.3.4 Beizer, 1990, Copeland, 2004
4.3.5 Copeland, 2004
4.4.3 Beizer, 1990, Copeland, 2004
4.5 Kaner, 2002
4.6 Beizer, 1990, Copeland, 2004

5 Tesztmenedzsment (K3) — 170 perc

A tesztmenedzsment tanulási céljai

A Tanulási Célok (TC) összefoglalják, hogy az egyes fejezetek ismeretanyagának elsajátításának mik a céljai.

5.1 Tesztelő szervezet (K2)

TC-5.1.1 A független teszt jelentőségének felismerése. (K1)
TC-5.1.2 A szervezeten belüli független teszt előnyeinek és hátrányainak felsorolása. (K2)
TC-5.1.3 A tesztelő csapat létrehozásánál a különböző bevonandó tagok felismerése. (K1)
TC-5.1.4 A tipikus tesztvezetői és tesztelői feladatok felidézése. (K1)

5.2 Teszttervezés és becslés (K2)

TC-5.2.1 A teszttervezés különböző szintjeinek és célkitűzéseinek ismerete. (K1)
TC-5.2.2 A teszterv, a műszaki teszterv specifikáció és a teszteljárási dokumentumok céljainak és tartalmának összefoglalása a „Szoftverteszt Dokumentáció Szabvány” (IEEE 829) szerint. (K2)
TC-5.2.3 A fogalmilag eltérő tesztelési megközelítések megkülönböztetése, mint pl. analitikus, modell alapú, metodikus, folyamat/szabvány szerinti, dinamikus/heurisztikus, konzultatív és regressziós elkerülő. (K2)
TC-5.2.4 Egy rendszer teszttervének tárgya és a tesztvégrehajtás ütemtervének tárgya közötti különbség meghatározása. (K2)
TC-5.2.5 Egy tesztvégrehajtási ütemterv megírása adott tesztesethalmazra a priorizálás, technikai és logikai függőségi viszonyok figyelembe vételével. (K3)
TC-5.2.6 A teszterv elkészítése során figyelembe veendő előkészítési és végrehajtási tevékenységek felsorolása. (K1)
TC-5.2.7 A tesztelési erőforrást befolyásoló tényezők felidézése. Tipikus tényezők felidézése, melyek befolyásolják a teszt ráfordításait. (K1)
TC-5.2.8 Két fogalmilag különböző becslési megközelítés közötti különbség meghatározása: a metrika alapú megközelítés és a szakértő alapú megközelítés. (K2)
TC-5.2.9 Megfelelő belépési és kilépési feltételek ismerete/helyességének igazolása a meghatározott tesztszintekhez és teszteset csoportokhoz kapcsolódóan (pl. integrációs teszthez, átvételi teszthez vagy használhatósági teszt eseteihez). (K2)

5.3 A teszt előrehaladásának felügyelete és irányítása (K2)

TC-5.3.1 A tesztelőkészítés és végrehajtás nyomonkövetésére használt gyakori metrikák felidézése. (K1)
TC-5.3.2 A tesztjelentés és tesztirányítás metrikáinak megértése, bemutatása (pl. a talált és javított hibák száma, a sikeres és sikertelen tesztek). (K2)
TC-5.3.3 A tesztösszefoglaló jelentés céljainak és tartalmának összefoglalása a „Szoftverteszt Dokumentáció Szabvány” (IEEE 829) szerint. (K2)

5.4 Konfiguráció menedzsment (K2)

TC-5.4.1 Annak összefoglalása, hogyan támogatja a konfiguráció menedzsment a tesztelést. (K2)

5.5 Kockázat és tesztelés (K2)

TC-5.5.1 A kockázat bemutatása lehetséges problémaként, amely veszélyezteti egy vagy több projektrésztvevő célkitűzéseinek teljesülését. (K2)
TC-5.5.2 Annak tudatosítása, hogy a kockázatok szintjét (megtörténési) valószínűségük és hatásuk (a ténylegesen felmerült kockázati esemény által okozott kár) alapján határozzuk meg. (K1)
TC-5.5.3 A projekt- és termékkockázatok megkülönböztetése. (K2)
TC-5.5.4 Tipikus termék- és projektkockázatok ismerete. (K1)

TC-5.5.5 Annak bemutatása példákkal, hogyan használható a kockázatelemzés és a kockázatmenedzsment a teszttervezésben. (K2)

5.6 Incidensmenedzsment (K3)

TC-5.6.1 A „Szoftverteszt Dokumentáció Szabvány” (IEEE 829) szerinti incidensjelentés tartalmának ismerete. (K1)

TC-5.6.2 Incidensjelentés írása egy teszt során megfigyelt meghibásodásról. (K3)

5.1 Tesztelő szervezet (K2)	*30 perc*

Kifejezések
Tesztelő, tesztvezető, tesztmenedzser.

5.1.1 Tesztelő szervezet és a függetlenség (K2)

A tesztek és felülvizsgálatok útján való hibakeresés hatékonysága független tesztelők alkalmazásával növelhető. A függetlenség változatai:

- Nincsenek független tesztelők. A fejlesztők tesztelik saját kódjukat.
- Független tesztelők vannak a fejlesztői csapaton belül.
- Független tesztelő csapat vagy csoport van a szervezeten belül, amely a projektmenedzsment vagy a vállalati vezetés (menedzsment) felé tesz jelentést.
- Független tesztelők az üzleti szervezettől vagy a felhasználói körből.
- Független, különböző tesztcélokra szakosodott tesztelői szakértők, mint pl. használhatósági tesztelők, biztonsági tesztelők vagy tanúsítvány tesztelők (akik a szoftvertermék szabványoknak és előírásoknak való megfelelését tanúsítják).
- Kiszervezett vagy szervezeten kívüli független tesztelők.

A nagyméretű, komplex, vagy biztonságkritikus projekteknél általában a legjobb megoldás a többszintű teszt alkalmazása, ahol néhány vagy az összes tesztszintet független tesztelők hajtják végre. A fejlesztők részt vehetnek a tesztelésben, különösen az alacsonyabb szinteken, objektivitás híján azonban korlátozott a hatékonyságuk. A független tesztelőknek hatáskörébe tartozhat a tesztfolyamatok és szabályok igénylése és meghatározása, de az ezen folyamatokkal kapcsolatos feladatokat csak akkor végezhetik el, ha arra megbízásuk van a vezetőségtől.

A függetlenség előnyei közé tartoznak a következők:

- A független tesztelők más, különböző hibákat látnak meg, és elfogulatlanok.
- Egy független tesztelő ellenőrizni tudja azokat a feltételezéseket, amelyeket különböző szereplők tettek a rendszer specifikációja és megvalósítása során.

Hátrányok :

- Izoláció a fejlesztői csapattól (amennyiben teljesen független).
- A független tesztelők szűk keresztmetszetként jelenhetnek meg az utolsó ellenőrzési pontnál. A fejlesztők elveszthetik a minőség iránti felelősségérzetüket.
- A független tesztelőket a projekt szűk keresztmetszetének tekinthetik, vagy hibáztathatják őket a kiadások csúszásakor.

A tesztelés feladatait végezhetik tesztelő feladatkört betöltők vagy más beosztásúak is, mint pl. egy projekt-menedzser, minőségbiztosító, fejlesztő, vállalati szakértő vagy az üzleti terület szakértője, infrastrukturális vagy IT műveletek szakembere.

5.1.2 A tesztvezető és a tesztelő feladatai (K1)

Jelen tananyag két tesztelői munkakört tárgyal, a tesztvezetőét és a tesztelőét. A két feladatkörhöz tartozó tevékenységek és feladatok a projekt és a termék jellemzőitől, a beosztásokat ellátó személyektől és a szervezettől függnek.

A tesztvezetőt esetenként tesztmenedzsernek vagy teszt koordinátornak nevezik. A tesztvezető szerepét betöltheti egy projektmenedzser, fejlesztési menedzser, minőségbiztosítási menedzser vagy a tesztelő csoport menedzsere. Nagyobb projekteknél két beosztás is lehet: tesztvezető és

tesztmenedzser. Általában a tesztvezető tervezi, felügyeli és irányítja a teszttevékenységeket, amelyek az 1.4. fejezetben kerültek meghatározásra.

Tipikus tesztvezetői feladatok lehetnek:

- A tesztstratégia és terv koordinálása a projektmenedzserekkel és másokkal.
- A projektekhez tartozó tesztstratégia, illetve a szervezeti szintű tesztelési irányelvek elkészítése vagy felülvizsgálata.
- A tesztelési perspektívákat alkalmazza más projekttevékenységekre, mint pl. az integrációs tervezésre.
- A teszt tervezése – a tesztelési szempontok figyelembevételével és a tesztcélok, kockázatok ismeretében – ide tartozik a tesztelési megközelítések kiválasztása, a tesztelés időtartamának, ráfordításainak és költségeinek becslése, erőforrások beszerzése, tesztszintek és ciklusok meghatározása, incidens menedzsment tervezése.
- A tesztek specifikációjának, előkészítésének, kivitelezésének és végrehajtásának kezdeményezése, a teszteredmények monitorozása és a kilépési feltételek ellenőrzése.
- A tervezés átalakítása a teszteredmények és a tesztelés előrehaladása alapján (amit gyakran állapotjelentésekben dokumentálnak), a problémák kiküszöböléséhez szükséges lépések megtétele.
- A tesztver megfelelő konfiguráció menedzsmentjének létrehozása a nyomonkövethetőség érdekében.
- Megfelelő metrikák bevezetése a teszt-előrehaladás mérésére, valamint a tesztelés és a termék minőségének értékelésére.
- Annak eldöntése, hogy mit, milyen mértékig és hogyan kell automatizálni.
- A teszttámogató eszközök kiválasztása és az eszközök használatával kapcsolatos képzés megszervezése a tesztelők részére.
- Döntéshozatal a tesztkörnyezet kialakításáról.
- Összefoglaló tesztjelentés készítése a teszt során gyűjtött információk alapján.

Tipikus tesztelői feladatok lehetnek:

- Teszttervek felülvizsgálata és részvétel a kidolgozásukban.
- Felhasználói követelmények, specifikációk és tesztelhetőségi modellek elemzése és felülvizsgálata.
- Tesztspecifikációk készítése.
- Tesztkörnyezet kialakítása (gyakran együttműködésben a rendszer adminisztrációval és a hálózat menedzsmenttel).
- Tesztadatok előkészítése, felvétele.
- Tesztek kidolgozása minden tesztszinten, tesztek végrehajtása és naplózása, eredmények értékelése, az elvárt eredményektől való eltérések dokumentálása.
- Szükség esetén tesztelési adminisztrációs vagy menedzsment eszközök, valamint tesztfelügyeleti eszközök használata.
- Tesztek automatizálása (ebben támogatást nyújthat egy fejlesztő vagy egy tesztautomatizálási szakértő).
- Komponensek és rendszerek teljesítményének mérése (ha lehetséges).
- Mások által kifejlesztett tesztek felülvizsgálata.

A tesztelemzéssel, műszaki teszttervezéssel, speciális teszttípusokkal, illetve tesztautomatizálással foglalkozó személyek általában szakértők a saját területükön. A tesztszinttől és a termékkel kapcsolatos kockázatoktól függően különböző személyek vehetik át a tesztelői feladatokat, bizonyos fokú függetlenség megtartásával.
Általában a komponens- és integrációs szintű tesztelők a fejlesztők, az átvételi tesztelők üzleti szakértők és felhasználók, a működési átvételi tesztelők pedig operátorok lehetnek.

5.2 Teszttervezés és becslés (K2)	*40 perc*

Kifejezések
Tesztelési megközelítés, tesztstratégia

5.2.1 Teszttervezés (K2)

Ez a fejezet a teszttervezés céljait tárgyalja a projektek fejlesztésénél, implementálásánál, illetve a karbantartási tevékenységeknél. A tervezést dokumentálhatják fő tesztterv-ben, vagy az egyes tesztszintekhez – pl. rendszerteszt, átvételi teszt - tartozó tesztterv-ekben. A teszterv vázát a „Szoftverteszt Dokumentáció Szabvány” (IEEE 829) tartalmazza.

A tervezést befolyásolja a szervezet tesztstratégiája, 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 teszttervezést folyamatosan végzik, az életciklus minden fázisában és minden tevékenység során. A tesztelési tevékenységekből kapott visszajelzések alapján felismerhető a kockázatok változása, és ennek megfelelően alakítható a tervezés.

5.2.2 Teszttervezési tevékenységek (K2)

A teszttervezési tevékenységek a következők lehetnek:

- A teszt tárgya, kockázatok és célok meghatározása.
- A tesztelés általános megközelítésének definiálása, ezzel együtt a tesztszintek, valamint a bemeneti és kilépési feltételek meghatározása.
- A tesztelési tevékenységek beépítése a szoftver életciklusába.
- Beszerzés, beszállítás, fejlesztés, működtetés és karbantartás.
- Döntéshozatal arról, hogy mit tesztelünk, mely szerepkörbe tartozók hajtják végre a tesztelési tevékenységeket, és ezen tevékenységeket hogyan kell végrehajtani, hogy fogják kiértékelni a teszteredményeket.
- A tesztelemzési és a -tervezési tevékenységek ütemterve.
- A tesztmegvalósítás, -végrehajtás és -értékelés ütemterve.
- Erőforrások hozzárendelése a definiált tevékenységekhez.
- A tesztdokumentáció mennyiségének, részletességének, struktúrájának meghatározása, minták megadása.
- Metrikák kiválasztása a tesztelőkészítés és -végrehajtás felügyeletéhez és irányításához, a hibák és a kockázatok kezeléséhez.
- A teszteljárások részletességének meghatározása annak érdekében, hogy elegendő információ álljon rendelkezésre a tesztek ismételt előkészítéséhez és végrehajtásához

5.2.3 Belépési feltételek (K2)

A belépési feltételek meghatározása a tesztelés kezdetén, pl. egy tesztszint elején történhet, vagy akkor, ha egy adott tesztkészlet készen áll a végrehajtásra.

Tipikus belépési feltételek lehetnek:
- A tesztkörnyezet rendelkezésre állása.
- A teszteszközök rendelkezésre állása a tesztkörnyezetben.
- A tesztelhető kód rendelkezésre állása.
- A tesztadatok rendelkezésre állása.

5.2.4 **Kilépési feltétel (K2)**

A kilépési feltételek rendeltetése, hogy meghatározzák, mikor kell leállítani a tesztelést, például egy tesztszint végén, vagy ha egy teszthalmaznak meghatározott célja van.

A tipikus kilépési feltételek általában a következőkből tevődnek össze:

- Alapossági mérés, mint kód-, funkcionalitás- vagy kockázat-lefedettség.
- Hibasűrűség vagy megbízhatóság becslése.
- Költség.
- A fennmaradó kockázatok, mint például a javítatlan hibák, tesztlefedettség hiánya bizonyos területeken.
- Ütemtervek, melyek lehetnek például a piacra kerüléssel kapcsolatosak.

5.2.5 **A tesztbecslés (K2)**

Két megközelítés létezik a tesztbecslésre:

- A metrika alapú megközelítés: a tesztelési ráfordítás becslése régebbi, vagy hasonló projektek metrikái alapján, vagy tipikus értékek alapján.
- A szakértő alapú megközelítés: a feladatok becslése az azokat ismerő személy vagy szakértők becslése alapján történik.

A teszteléshez szükséges ráfordítások becslése után következik a szükséges erőforrások és az ütemterv meghatározása.
A tesztelés ráfordításai több tényezőtől függnek:

- A termék jellemzői: a specifikáció és más, a tesztelési modelleknél felhasznált információk (pl.: tesztbázis) minősége, a termék mérete, a problémakör komplexitása, megbízhatósági és biztonsági követelmények, dokumentációra vonatkozó követelmények.
- A fejlesztési folyamat jellemzői: a szervezet stabilitása, az alkalmazott eszközök, a tesztfolyamat, a résztvevő személyek szaktudása és az időtényező.
- A teszt eredménye: a hibák száma és a szükséges átdolgozás mértéke.

5.2.6 **Tesztelési megközelítések, tesztstratégiák (K2)**

A tesztelési megközelítés a tesztstratégia adott projektre történő implementálása. A tesztelési megközelítés meghatározása és finomítása a teszttervben és a műszaki teszttervben történik. Tipikusan a (teszt) projekt célján és kockázatértékelésen alapuló döntéseket tartalmaz. Ez a tesztfolyamat tervezésének, a műszaki teszttervezési technika kiválasztásának és az alkalmazandó teszttípus kiválasztásának a kiindulópontja, továbbá meghatározza a belépési és kilépési feltételeket. A kiválasztott megközelítés a környezettől függ és figyelembe veheti a kockázatokat és biztonsági előírásokat, a rendelkezésre álló erőforrásokat és képességeket, a technológiát, a rendszer természetét (pl. egyedi vagy kereskedelmi szoftver), a tesztcélt és a különböző előírásokat.

A tipikus megközelítések, vagy stratégiák a következőket foglalják magukba:

- Analitikus megközelítések esetén, mint amilyen például a kockázat alapú teszt, a teszt a nagyobb kockázatú területekre irányul.
- Modell alapú megközelítéseknél, mint például a sztochasztikus teszt, a meghibásodási rátákra (megbízhatóság-növekedés modellek) vagy a használatra (működési profilok) vonatkozó statisztikai információkat alkalmaznak.
- Módszeres megközelítések, mint például a meghibásodás alapú (ide tartozik a hibasejtés és a támadás), tapasztalat alapú, ellenőrző lista alapú, minőségi jellemző alapú.
- Folyamat- vagy szabvány szerinti megközelítések, mint például az iparhoz kapcsolódó szabványok, vagy a különböző agilis módszertanok által meghatározott megközelítések.
- Dinamikus és heurisztikus megközelítések, mint például a felderítő teszt, ahol a teszt nincs eltervezve, inkább eseményekre reagál, s a végrehajtás és az értékelés egyszerre történik.

- Tanácsadói megközelítések esetén a tesztlefedettséget elsősorban a technológia és/vagy a tesztelő csapaton kívüli vállalati szakértők iránymutatása viszi előre.
- Regressziós elkerülő megközelítéseknél újra alkalmazzák a meglévő tesztanyagot, széles körben automatizálják a funkcionális regressziós teszteket és a standard tesztkészleteket.

5.3 A teszt előrehaladásának felügyelete és irányítása (K2)	*20 perc*

Kifejezések
Hibasűrűség, meghibásodási ráta, tesztirányítás, tesztfelügyelet, összefoglaló tesztjelentés.

5.3.1 A teszt előrehaladásának felügyelete (K1)

A teszt felügyeletének célja, hogy visszajelzéseket adjon a teszttevékenységekről. A felügyelet alá vont információ manuálisan vagy automatizáltan gyűjthető, és alkalmazható a kilépési feltételek, például a lefedettség mérésére. A tervezett ütemtervhez és költségvetéshez képest történt előrehaladás értékelésére metrikákat is alkalmazhatnak. A leggyakoribb tesztmetrikák a következők:

- A tesztesetek előkészítésében végzett munka százalékosan (vagy hány százaléka készült el a tervezett teszteseteknek).
- A tesztkörnyezet előkészítésében végzett munka százalékosan.
- Teszteset végrehajtás (pl. futtatott/nem futtatott tesztesetek száma, sikeres/sikertelen tesztesetek).
- Információ a hibákról (pl. hibasűrűség, megtalált és javított hibák, meghibásodásr ráta, újratesztelési eredmények).
- A követelmények, a kockázatok vagy a kód tesztlefedettsége.
- A tesztelők szubjektív véleménye a termékről.
- A tesztelés mérföldköveinek dátumai.
- A tesztelés költségei, ahol számítandó a következő hiba megtalálásának vagy a következő teszt futtatásából származó nyereség aránya a befektetett költséghez képest.

5.3.2 Tesztjelentés (K2)

A tesztjelentés összefoglalja a teszttel kapcsolatos információkat úgy, mint:

- Mi történt a teszt adott szakaszában, mint például a kilépési feltétel teljesülésének időpontja.
- Feldolgozott információk és metrikák az elkövetkező lépésekkel kapcsolatos ajánlások és döntések elősegítésére, mint például elemzés a fennmaradó hibákról, a további teszt gazdasági előnyei, jelentősebb kockázatok, a tesztelt szoftver megbízhatósági szintje.

A tesztösszefoglaló jelentés fő pontjait a „Szoftverteszt Dokumentáció Szabvány” (IEEE 829) tartalmazza.

A metrikákat gyűjteni kell az adott tesztszint folyamán és végén, a következők elemzése érdekében:

- Megfelelőek-e az adott tesztszint célkitűzései.
- Megfelelőek-e az alkalmazott tesztelési megközelítések.
- A teszt hatékonysága a célkitűzésekre való tekintettel.

5.3.3 Tesztirányítás (K2)

A tesztirányítás a begyűjtött és bejelentett információk, metrikák alapján foganatosított korrekciós lépéseket jelenti. Ezek a műveletek bármely teszttevékenységre vonatkozhatnak és a szoftver életciklusának bármely tevékenységére vagy feladatára hathatnak.

Példák tesztirányításra:

- Döntéshozatal a tesztfelügyelet során nyert információk alapján.
- Tesztek prioritásának megváltoztatása, ha egy ismert kockázat bekövetkezik (pl. a szoftver késői kiszállítása).
- A teszt ütemezésének megváltoztatása a tesztkörnyezet felhasználhatóságának változása miatt.
- Belépési feltétel meghatározása, amelyhez javítások végzése és azok fejlesztő általi újratesztelése (ellenőrző teszttel) szükséges, mielőtt a szoftver build-be kerülne.

5.4 Konfiguráció menedzsment (K2)	*10 perc*

Kifejezések

Konfiguráció menedzsment, verziókövetés.

Háttér

A konfiguráció menedzsment célja a szoftver- vagy rendszerrészek (komponensek, adatok és dokumentáció) integritásának kialakítása és fenntartása a projekt és a termék teljes életciklusa alatt.

A tesztnél a konfiguráció menedzsment a következők biztosítását jelentheti:

- A tesztver minden eleme pontosan meghatározott, verziókövetés alá vont, a benne történt változások nyomonkövethetők, az elemek kapcsolódnak egymáshoz és a fejlesztési elemekhez (teszt tárgyaihoz), hogy a nyomonkövethetőség fenntartható legyen a teszt folyamán.
- A tesztdokumentációban egyértelmű hivatkozás található minden létező dokumentumra és szoftverelemre.

A konfiguráció menedzsment segíti a tesztelőt a tesztelt elemek, a tesztdokumentumok, a tesztek és a teszttámogató szoftverkörnyezet egyértelmű meghatározásában (és reprodukálásában).

A teszttervezés során kell megválasztani, dokumentálni és megvalósítani a konfiguráció menedzsment eljárásokat és infrastruktúrát (eszközöket).

5.5 Kockázat és tesztelés (K2)	*30 perc*

Kifejezések
Termékkockázat, projektkockázat, kockázat, kockázat alapú teszt.

Háttér
A kockázat úgy definiálható, mint egy esemény, veszély, fenyegetés vagy szituáció fellépésének esélye és nemkívánatos következményei, azaz egy potenciális probléma. A kockázat szintjét a káros esemény bekövetkezésének valószínűsége és hatása (az esemény okozta kár) határozza meg.

5.5.1 Projektkockázatok (K2)
A projektkockázatok olyan kockázatok, melyek a projekt céljainak elérését befolyásolják úgy, mint:
- Szervezeti tényezők:
  - szaktudás és munkaerő hiánya;
  - személyi kérdések;
  - szervezeti működés problémái, mint pl.
    - a tesztelők nem jól kommunikálják az igényeiket és a teszteredményeket;
    - nem alkalmazzák a teszt és a felülvizsgálat alkalmával szerzett információkat (pl. nem javítják a fejlesztési és teszteljárás-folyamatokat).
  - nem megfelelő hozzáállás vagy rossz elvárások a teszteléssel kapcsolatban (pl. nem értékelik a teszteléssel talált hibák jelentőségét).
- Technikai problémák:
  - a megfelelő követelmények meghatározásával kapcsolatos problémák;
  - a követelmények teljesíthetőségének mértéke a már létező megkötések figyelembevételével;
  - a terv, a kód és a tesztek minősége.
- Beszállítói problémák:
  - a harmadik fél nem teljesít;
  - szerződésbeli problémák.

Ezen kockázatok elemzésekor, kezelésekor és mérséklésekor a tesztmenedzsernek követnie kell a jól bevált projektmenedzsment-alapelveket. A „Szoftverteszt Dokumentáció Szabvány” (IEEE 829)-ban található teszttervezési vázlathoz meg kell állapítani a kockázatokat és a mérséklésükre hozott intézkedéseket.

5.5.2 Termékkockázatok (K2)
A szoftver vagy rendszer lehetséges hibás területeit (kedvezőtlen jövőbeli események, vagy veszélyek) termékkockázatoknak nevezik, mivel kockázatot jelentenek a termék minőségére vonatkozóan. Ilyenek például:

- Hibára hajlamos szoftver kiszállítása.
- Annak lehetősége, hogy a szoftver/hardver károkat okozhat egy személynek vagy egy cégnek.
- Gyenge szoftverjellemzők (pl. funkcionalitás, megbízhatóság, használhatóság és teljesítmény).
- Gyenge adatintegritás és adatminőség (pl. adatkonverziós, migrációs vagy adattovábbítási problémák, az adatszabványok megsértése).
- A szoftver nem teljesíti az elvárt funkciókat.

A kockázatokat annak eldöntésére használják, hogy hol kezdjék a tesztelést, és hol teszteljenek többet. A tesztelést egy ártalmas hatás fellépési kockázatának csökkentésére vagy egy ártalmas hatás által okozott kár csökkentésére használják.

A termékkockázatok a projekt sikerét veszélyeztető különleges kockázatok. A tesztelés, mint kockázatcsökkentő tevékenység, visszajelzéseket nyújt a fennmaradó kockázatokról úgy, hogy méri a kritikus hibák kiküszöbölésének és az előre nem látható veszélyekre vonatkozó tervek hatékonyságát.

A tesztelés kockázat alapú megközelítése proaktív lehetőségeket teremt a termékkockázat csökkentésére, már a projekt kezdeti szakasza itól kezdve. Ide tartozik a termékkockázatok azonosítása, és azok figyelembe vétele a teszttervezésben és - irányításban, a specifikáció valamint a tesztelőkészítés és -végrehajtás során. A kockázat alapú megközelítésnél a felismert kockázatok a következőkre használhatók:

- Az alkalmazandó teszttechnikák meghatározása.
- A végrehajtandó tesztelés mértékének meghatározása.
- A tesztelés priorizálása úgy, hogy a kritikus hibák a lehető leghamarabb felszínre kerüljenek.
- Nem teszteléshez kapcsolódó kockázatcsökkentő tevékenységek esetleges bevezetése (pl. képzés a tapasztalattal nem rendelkező tervezők részére).

A kockázat alapú teszt igénybe veszi a projektben résztvevők minden ismeretét és tudását a kockázatok és az azok kezeléséhez szükséges tesztszintek meghatározásához.

Hogy a termék sikertelenségének esélye minimális legyen, a kockázatmenedzsmentnek szigorú megközelítéseket kell alkalmazni a következők terén:

- Annak elemzése (és rendszeres újraelemzése), hogy milyen probléma léphet fel (kockázatok).
- Annak meghatározása, hogy mely kockázatok kezelése a legfontosabb.
- Lépések kidolgozása ezen kockázatok kezelésére.

Ezeken felül a tesztelés támogathatja az új kockázatok azonosítását, és annak meghatározását, hogy mely kockázatokat kell csökkenteni, valamint csökkentheti a kockázatokkal kapcsolatos bizonytalanságot.

5.6 Incidensmenedzsment (K3)	40 perc

Kifejezések
Incidensnaplózás, incidensmenedzsment.

Háttér
Mivel a tesztelés egyik célja a hibák megtalálása, ezért a valós és az elvárt eredmények közötti eltéréseket naplózni kell, mint incidenseket. Az incidenseket meg kell vizsgálni, és lehet, hogy hibák lesznek belőlük. Megfelelő lépéseket kell meghatározni az incidensek és a hibák kezelésére. Az incidenseket és a hibákat nyomon kell követni a felfedezéstől az osztályozáson át a javításig, illetve a megoldás ellenőrzéséig. A szervezetnek osztályozási eljárást és szabályokat kell kidolgoznia annak érdekében, hogy minden incidensnél teljes körű menedzsment történjen.

Incidensek előjöhetnek a szoftvertermék fejlesztése, felülvizsgálata, tesztelése, illetve használata során. Jelenthetik a kóddal vagy a működő rendszerrel kapcsolatos problémákat, vagy a dokumentációban levő problémákat, mint például követelményekben, fejlesztési dokumentumokban, tesztdokumentumokban, felhasználói információkban, - pl. a súgóban, vagy a telepítési útmutatóban.

Az incidensjelentések céljai a következők:

- Visszajelzést nyújtani a problémáról a fejlesztők és egyéb érintettek részére, hogy el lehessen végezni a szükség szerinti azonosítást, izolálást, javítást.
- Biztosítani, hogy a tesztvezetők nyomon követhessék a tesztelt rendszer minőségét és a teszt előrehaladását a teszt folyamán.
- A tesztfolyamat javítására vonatkozó elképzeléseket kidolgozni.

Az incidensjelentés a következőket tartalmazhatja:

- Keltezés, készítő szervezet, szerző.
- Elvárt és valós eredmények.
- A tesztelem (konfigurációs elem) és a környezet azonosítása.
- A szoftver- vagy rendszer életciklus folyamata, amiben az incidens fellépett.
- Az incidens leírása, hogy lehetséges legyen a megismétlése és a megoldása, felhasználva a naplókat, adatbázis mentéseket, screenshot-okat.
- Hatás az érintett felek érdekeire.
- A rendszerre való hatás mértéke.
- Javítás sürgőssége/prioritása.
- Az incidens állapota (pl. nyitott, elhalasztott, duplikált, javításra váró, javított és újratesztelésre váró, lezárt).
- Következtetések, javaslatok és jóváhagyások.
- Globális problémák, mint például további területek, melyekre az incidens okozta változások hatással lehetnek.
- A változtatások története: a projektcsapat tagjainak tevékenységei az incidens izolálására, javítására, javításának ellenőrzésére.
- Referenciák, melyekbe beletartozik a problémát felfedő teszteset-specifikáció azonosítása is.

Az incidensjelentés struktúráját a „Szoftverteszt Dokumentáció Szabvány” (IEEE 829) is tartalmazza.

Irodalomjegyzék
5.1.1 Black, 2001, Hetzel, 1988
5.1.2 Black, 2001, Hetzel, 1988
5.2.5 Black, 2001, Craig, 2002, IEEE 829, Kaner 2002
5.3.3 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE 829
5.4 Craig, 2002

5.5.2 Black, 2001 , IEEE 829
5.6 Black, 2001, IEEE 829

6 Eszköztámogatás a tesztelésben (K2)	80 perc

A tesztelés eszköztámogatásának tanulási céljai

A Tanulási Célok (TC) összefoglalják, hogy az egyes fejezetek ismeretanyagának elsajátításának mik a céljai.

6.1 Teszteszközök típusai (K2)

TC-6.1.1 A különböző teszteszközök osztályozása a tesztfolyamat tevékenységek szerint. (K2)

TC-6.1.3 A teszteszközök fogalmának meghatározása és a tesztelés eszköztámogatásának kifejtése (K2)

6.2 Eszközök hatékony használata: a lehetséges előnyök és kockázatok (K2)

TC-6.2.1 A tesztautomatizálással és a tesztelés eszköztámogatásával kapcsolatos lehetséges előnyök és kockázatok összefoglalása. (K2)

TC-6.2.2 A tesztvégrehajtási eszközök, a statikus analízis és a tesztmenedzsment eszközök speciális szempontjainak felidézése (K1)

6.3 Eszköz bevezetése egy szervezetnél (K1)

TC-6.3.1 Egy eszköz egy szervezetnél való bevezetésével kapcsolatos fő ek megfogalmazása. (K1)

TC-6.3.2 Az eszközzel kapcsolatos elképzelést igazoló/pilot fázis céljainak megfogalmazása. (K1)

TC-6.3.3 Annak felismerése, hogy a jó eszköztámogatáshoz több szükséges, mint egyszerűen egy eszköz beszerzése. (K1)

6.1 Teszteszközök típusai (K2)	*45 perc*

Kifejezések
Konfiguráció menedzsment eszköz, lefedettségi eszköz, hibakereső eszköz, dinamikus elemzés eszköz, incidens menedzsment eszköz, terheléses teszt eszköz, modellező eszköz, felügyeleti eszköz, teljesítményteszt eszköz, mérési mellékhatás, követelmény-menedzsment eszköz, felülvizsgáló eszköz, biztonsági eszköz, statikus elemző eszköz, stressz teszteszköz, teszt összehasonlító eszköz, tesztadat előkészítő eszköz, műszaki teszttervező eszköz, teszttámogató szoftverkörnyezet, tesztvégrehajtási eszköz, tesztmenedzsment eszköz, egységteszt keretrendszer eszköz

6.1.1 A tesztelés eszköztámogatásának célja (K2)

A teszteszközök akár többféle teszttevékenységet is támogathatnak:
1. Közvetlenül a teszt során használt eszközök, mint pl. a tesztvégrehajtó, tesztadat előállító, illetve eredményösszehasonlító eszközök.
2. A tesztfolyamat, a teszteredmények, adatok, követelmények, incidensek, hibák, stb. kezelését segítő, továbbá a tesztvégrehajtást felügyelő és a jelentéseket támogató eszközök.
3. A felderítő teszt során használt eszközök (pl. egy adott file műveleteit figyelő eszköz).
4. Bármilyen, a tesztelést segítő eszköz (egy táblázatkezelő ebben az értelemben szintén teszteszköz).

A teszteszközök akár több célt is szolgálhatnak:
- A teszttevékenységek hatékonyságának növelése az ismétlődő feladatok automatizálásával, vagy a kézi teszttevékenységek támogatása, mint a teszttervezést, a műszaki teszttervezést, a tesztjelentéseket, illetve a tesztfelügyeletet
- Jelentős kézi erőforrást igénylő teszttevékenységek automatizálása (pl. statikus teszt)
- 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)

A “keretrendszer” kifejezés szintén gyakran használt az iparban, legalább 3 értelemben:
- Újrahasználható és kiterjeszthető tesztkönyvtárak, amelyeket teszteszközök létrehozására lehet használni (nevezik ezeket teszttámogató szoftverkörnyezetnek is)
- A tesztautomatizálás műszaki tervezésének egy típusa (pl. adatvezérelt, vagy akciószó alapú
- Általános tesztvégrehajtási folyamat

Ezen tantervben a “keretrendszert” az első két értelemben használjuk, ld 6.1.6 fejezet.

6.1.2 Teszteszközök osztályozása (K2)
Számos eszköz létezik a tesztelés különböző szempontból való támogatására. Az eszközöket különböző szempontok szerint lehet osztályozni, úgymint céljuk, kereskedelmi/ingyenes/nyílt forráskódú/shareware programok. A tananyagban az eszközöket a támogatott teszttevékenység alapján osztályozzuk.

Egyes eszközök csak egy tevékenységet támogatnak, mások 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 időzítés eltérő lehet attól függően, hogy a különböző eszközökkel hogyan mérik; vagy különböző kód lefedettség értéket mérhetnek az alkalmazott lefedettségi eszköztől 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)" -vel jelöljük.

6.1.3 Eszköztámogatás a tesztelés és a tesztek menedzsmentjéhez (K1)

A menedzsment eszközök minden teszt tevékenységnél, a szoftver teljes életciklusán át alkalmazhatók.

Tesztmenedzsment eszközök

Ezen eszközök interfészt nyújtanak a tesztek végrehajtásához, a hibák nyomonkövetéséhez és a követelmények menedzseléséhez, továbbá támogatják a teszt tárgyának mennyiségi analízisét és ezekről történő jelentéskészítést. Támogatják tovább a teszt tárgyának a követelmény specifikációval szemben történő nyomonkövetését, képes lehet a független verziókövetésre, vagy lehet egy interfésze egy külső ilyen eszközhöz.

Követelmény menedzsment eszközök

A követelmény menedzsment eszközök képesek a követelmények, illetve a követelményekhez tartozó attribútumok tárolására (beleértve a prioritást). Egyedi azonosítókat kínálnak és támogatják a követelmények nyomonkövetését az egyes tesztekben. Ezek az eszközök segíthetnek az inkonzisztens, vagy hiányzó követelmények azonosításában.

Incidens menedzsment eszközök (Hibakövető eszközök)

Ezen eszközök tárolják és menedzselik az incidensjelentéseket, pl. hibák, meghibásodások és változáskezdeményezések, a talált problémák és rendellenességek kezelését. Támogatják továbbá az incidensek életciklusának menedzselését, ezen felül opcionálisan a statikus analízist.

Konfiguráció menedzsment eszközök

Bár nem kifejezetten teszteszközök, de szükségesek a tesztver és a kapcsolódó szoftverek tárolásához és verziókövetéséhez, különösen, ha a teszt egynél több hardver/szoftver környezetet (pl. operációs rendszert, fordítóprogramot, böngészőt, stb.) igényel.

6.1.4 A statikus teszt eszköztámogatása (K1)

A statikus teszteszközök lehetővé teszik a költséghatékony hibamegtalálást a fejlesztés korai fázisában.

Felülvizsgálati eszközök
Ezen eszközök támogatják a felülvizsgálati folyamatot, az ellenőrző listákat, a felülvizsgálati útmutatókat és gyakran használják a felülvizsgálati megjegyzések és jelentések, valamint a ráfordítási adatok tárolására és kommunikációjára. Ezen felül támogathatják az online felülvizsgálatokat a nagy, vagy földrajzilag szétosztott csapatok részére.

Statikus elemző eszközök (F)

A statikus elemző eszközök támogatják a fejlesztőket, a tesztelőket, hogy a hibákat a dinamikus teszt előtt megtalálják azáltal, hogy támogatást nyújtanak a kódolási irányelvek betartásának kikényszerítésében (beleértve a biztonsági kódolást), valamint a struktúrák és függőségek analízisében. Ezen felül segíthetik a tervezést és a kockázatelemzést is, azáltal, hogy kódmetrikákat szolgáltatnak (pl. a komplexitásra).

Modellező eszközök (F)
A modellező eszközökkel validálhatók a szoftver modelljei (pl. fizikai adatmodell egy relációs adatbázishoz) oly módon, hogy megmutatja az inkonzisztenciákat és megtalálja a hibákat. Ezen eszközöket gyakran használják modelleken alapuló teszteseteket készítésére.

6.1.5 A tesztspecifikáció eszköztámogatása (K1)

Műszaki teszttervező eszközök
A műszaki teszttervező eszközök tesztbemeneteket, vagy futtatható teszteket, és/vagy teszt-orákulumokat alakíthatnak ki a követelményekből, egy grafikus felhasználói interfészből, illetve teszt modellekből, vagy a kódból.

Tesztadat előkészítő eszközök
A tesztadat előkészítő eszközök adatbázisokat, fájlokat vagy adatátviteleket kezelnek a tesztek végrehajtásánál használandó tesztadatok létrehozásához, hogy az adatok anonimitásán keresztül biztosítsák a rendszer biztonságát

6.1.6 A tesztvégrehajtás és naplózás eszköztámogatása (K1)

Tesztvégrehajtási eszközök
A tesztvégrehajtási eszközök a tesztek automatikus vagy félautomatikus végrehajtását teszik lehetővé eltárolt bemenetek és elvárt eredmények segítségével, egy szkriptnyelv használatával és általában minden tesztfutást naplóznak. Ezen eszközökkel a teszteket fel is lehet venni és általában támogatják a szkriptkészítést, illetve az adatoknak az adott grafikus felhasználói felületnek megfelelő konfigurálását, vagy a tesztek más testreszabását.

Teszttámogató szoftverkörnyezet/egységteszt keretrendszer eszközök (F)
A teszttámogató szoftverkörnyezet elősegítheti a komponensek vagy egy rendszer részeinek tesztelését úgy, hogy szimulálja a környezetet, melyben a teszt tárgya futni fog. Mindezt helyettesítő objektumokkal teszik, úgy mint, csonkok és meghajtók.

Teszt összehasonlító eszközök
A teszt összehasonlító eszközök meghatározzák a fájlok, adatbázisok vagy teszteredmények közötti különbségeket. A teszt összehasonlító eszközök általában dinamikus összehasonlító eszközöket tartalmaznak, a végrehajtás utáni összehasonlítás elvégezhető külön összehasonlító eszközzel is. A teszt összehasonlító eszközök - különösen az automatizáltak - használhatnak teszt-orákulumot.

Lefedettség mérő eszközök (F)
A lefedettség mérő eszközök lehetnek beavatkozók, vagy nem beavatkozók. A kód lefedettség mérő eszközök egy tesztkészlet által adott típusú kódstruktúrák végrehajtását mérik százalékosan (pl. utasítások, elágazások vagy döntések, modul- vagy függvényhívások).

Biztonsági eszközök
A biztonsági eszközöket a szoftver biztonsági karakterisztikájának kiértékelésére használják. Ez magában foglalja a szoftver adatbiztonságát, adat-integritását, hitelesítését, engedélyezését, rendelkezésre állását és válaszmegtagadásait. A biztonsági eszközök általában a különböző technológiákra, platformokra és célokra koncentrálnak.

6.1.7 Teljesítmény és felügyelet eszköztámogatása (K1)

Dinamikus elemző eszközök (F)

A dinamikus elemző eszközök a csak a szoftver futása alatt nyilvánvalóvá váló hibákat találják meg, olyanokat, mint: időbeli függőségi viszonyok, vagy memóriaszivárgások. Általában a komponens, vagy komponens integrációs tesztnél használják, valamint köztes rétegű teszteknél.

Teljesítményteszt/terheléses teszt/stressz teszt eszközök
A teljesítményteszt eszközök felügyelik, hogy egy rendszer hogyan viselkedik különböző szimulált használati körülmények között – párhuzamos felhasználók, a tranzakciók gyakorisága és relatív százalékos aránya - majd erről jelentést készítenek. A terhelés szimulálása virtuális felhasználók létrehozásával érhető el, akik különböző tranzakciókat hajtanak végre, különböző tesztgépeken elosztva. Ezeket általában terhelés generátornak nevezik.

Felügyeleti eszközök
A felügyeleti eszközök folyamatosan elemzik, ellenőrzik a speciális rendszer erőforrásokat, továbbá jelentéseket készítenek róluk, illetve a lehetséges szervízelési problémáknál figyelmeztetéseket adnak.

6.1.8 Speciális tesztelői igények eszköztámogatása (K1)

Adatminőség értékelése
Több projekt fókuszában az adatok vannak, pl. az adatkonverziós/migrációs projektek, illetve az adatbázisokkal foglalkozó alkalmazások, és ezek attribútumai különbözhetnek mind a kritikusság, mind a méret tekintetében. Bizonyos körülmények között eszközök válhatnak szükségessé az adatminőség értékelésére, Meg kell vizsgálni, hogy a feldolgozott adatok helyesek-e, teljeskörűek-e, illetve megfelelnek-e az előre definiált szabványoknak.

További teszteszközök léteznek a használhatósági tesztekhez.

6.2 Eszközök hatékony használata: a lehetséges előnyök és kockázatok (K2)	*20 perc*

Kifejezések
Adatvezérelt teszt, akciószó alapú teszt, szkriptnyelv.

6.2.1 A tesztelés eszköztámogatásának lehetséges előnyei és kockázatai (minden eszközre) (K2)

Egy eszköz megvétele vagy bérlése nem garantálja az eszköz sikerét. Minden típusú eszköz esetében szükség lehet további erőfeszítésekre a valódi és tartós előnyök eléréséhez. A tesztelés eszköztámogatása esetleges előnyöket és lehetőségeket jelent, azonban kockázatokat is hordoz magában.

Az eszközök használatának lehetséges előnyei:

- Csökken az ismétlődő munka (pl. regressziós tesztek futtatása, ugyanazon tesztadatok ismételt bevitele, kódolási szabványok ellenőrzése).
- Jobb konzisztencia és megismételhetőség (pl. eszköz által végrehajtott tesztek, követelményekből származtatott tesztek).
- Objektív elemzés (pl. statikus mérések, lefedettség).
- Könnyű hozzáférni a tesztekkel, vagy teszteléssel kapcsolatos információkhoz (pl. a teszt-előrehaladást, az incidens-arányokat és teljesítményt mutató statisztikák és grafikonok).

Az eszközök használatának kockázatai:

- Irreális elvárások az eszközzel kapcsolatban (ilyen a funkcionalitás és a könnyű használat).
- Az eszköz bevezetésére szánt idő, költségek és erőfeszítések alábecslése (a képzés és a külső szaktudás is ilyen).
- Az eszköz által nyert jelentős, szignifikáns és folyamatos előnyök eléréséhez szükséges idő és erőfeszítés alábecslése (ide tartozik a tesztfolyamat átalakításának szükségessége és az eszköz használati módjának folyamatos javítása).
- Az eszköz által előállított teszteszközök karbantartásához szükséges erőforrások alábecslése.
- Az eszközbe vetett túl nagy bizalom (a műszaki teszterv specifikáció helyettesítése, illetve eszközhasználat ott, ahol a manuális teszt jobb lenne).
- Az eszközzel kapcsolatos verziókövetés elhanyagolása.
- A kritikus eszközök, mint pl. a követelmény menedzsment, verziókövető, incidens menedzsment, hibakövető, illetve a különböző eszközforgalmazóktól származó eszközök kapcsolatából, illetve együttműködéséből származó problémák elhanyagolása.
- Az eszköz forgalmazójának üzleti kockázata, mint pl. tönkremegy, nem támogatja tovább az eszközt, vagy eladja egy másik félnek.
- A forgalmazó gyenge minőségű support, upgrade, illetve hibajavító szolgáltatása
- Előre nem látható problémák, mint például egy új platform támogatásának hiánya

6.2.2 Különleges tényezők egyes eszköz-típusoknál (K1)

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

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 a műveleteket. Ez a típusú szkript nem várt incidensek fellépésekor instabil lehet.

Az adatvezérelt megközelítés elkülöníti a tesztbemeneteket (az adatokat), általában egy adatbázisba, és egy általánosabb szkriptet használ, amely képes tesztadatokat beolvasni és eltérő adatokkal végrehajtani ugyanazokat a teszteket. Azok a tesztelők, akik nem ismerik a szkriptnyelvet, bevihetik az adatokat ezeknél az előre meghatározott szkripteknél.

Vannak más adatvezérelt technikák is, ahol az adatbázisokban előre rögzített adatkombinációk helyett konfigurálható paramétereken alapuló algoritmusok segítségével futási időben generálják az adatokat, amelyeket az alkalmazás tesztjeihez használnak. Például egy eszköz használhat egy algoritmust, amely egy véletlenszerű felhasználói azonosítót generál, és a megismételhetőség érdekében az eszköz biztosítja a véletlenszerűség irányítását.
Az akciószó alapú tesztelési megközelítésnél az adatbázis akciószavakat (kulcsszavakat) tartalmaz, amelyek leírják, milyen adatokkal és milyen akciót kell végrehajtani. A tesztelők – még akkor is, ha nem ismerik a szkriptnyelvet – az akciószavak alapján teszteket definiálhatnak, amelyeket a tesztelendő alkalmazásra testreszabhatnak.
Technikai szaktudás a szkriptnyelvhez minden esetben szükséges (vagy a tesztelő, vagy a teszt automatizálási specialista részéről).
Függetlenül a használt szkript technikától, minden teszthez tárolni kell az elvárt eredményt későbbi összehasonlítás céljából.

Statikus elemző eszközök
A forráskódra alkalmazott statikus elemző eszközökkel be lehet tartatni a kódolási szabványokat, ha azonban már meglévő kódon alkalmazzák őket, sok üzenetet hozhatnak létre. A figyelmeztető üzenetek nem állítják le a kód futtatható programmá való fordítását, de érdemes kezelni őket, így a jövőben könnyebb lesz a kód karbantartása. Hatékony megközelítés lehet a fokozatos bevezetés, kezdetben szűrők használata bizonyos üzenetek kizárásához.

Tesztmenedzsment eszközök
A tesztmenedzsment eszközöknek kapcsolódniuk kell más eszközökhöz, vagy adatbázisokhoz, hogy a szervezet aktuális igényeinek megfelelő formátumban biztosíthassa az információkat.

6.3 Eszköz bevezetése egy szervezetnél (K1)	*15 perc*

Kifejezések
Nincsenek jellemző kifejezések.

Háttér
Egy szervezet a következő főbb szempontok alapján választ ki egy eszközt:

- Mennyire érett a szervezet egy eszköz bevezetésére; az eszköz erős és gyenge pontjainak elemzése; az eszközök által támogatott, továbbfejlesztett tesztfolyamatok alkalmazási lehetőségének felmérése.
- Világos követelmények és objektív feltételek kiértékelése.
- A kiértékelési fázisban meg kell győződni arról, hogy az eszköz hatékonyan működik-e együtt a tesztelendő szoftverrel az adott infrastruktúrában, illetve az infrastruktúrában szükséges változtatásokat meg kell határozni az eszköz hatékony használata érdekében.
- Az eszköz forgalmazójának értékelése (beleértve az oktatásokat, support-ot, illetve kereskedelmi szempontokat), illetve a kereskedelemben nem kapható eszközöknél a szervíz support szolgáltatást
- A belső követelmények meghatározása az eszköz használatával kapcsolatos coach, illetve mentorálási tevékenységek tekintetében
- Az oktatási igények kiértékelése a jelenlegi tesztcsapat automatizálási képességeinek tekintetében
- A konkrét üzleti eseten alapuló költség-ráfordítás becslés készítése

A kiválasztott eszköz szervezetnél való bevezetése egy pilot projekttel kezdődik, melynek céljai a következők:
- Az eszköz részletesebb megismerése.
- Annak értékelése, hogy az eszköz hogyan illeszkedik a meglévő folyamatokba és gyakorlatba; a szükséges változtatások meghatározása.
- Az eszköz és a teszteléssel kapcsolatos elemek szabványos használati, menedzselési, tárolási és karbantartási módjának meghatározása (pl. fájlok és tesztek elnevezési szabályai, könyvtárak létrehozása, tesztkészletek modularitásának definiálása).
- Annak értékelése, hogy az előnyöket elfogadható kiadásokkal érik-e el.

Az eszköz adott szervezetnél való sikeres bevezetésének tényezői:

- Az eszköz folyamatos bevezetése a szervezet további egységeiben.
- Az eszköz használatához illeszkedő folyamatok átvétele, illetve fejlesztése.
- Képzés és betanítás/tanácsadás biztosítása az új felhasználók részére.
- Használati irányelvek kidolgozása.
- Módszer kidolgozása az eszköz használatával kapcsolatos tapasztalatok feldolgozására.
- Az eszköz használatának és előnyeinek felügyelete.
- A tesztcsapat támogatása az eszközzel kapcsolatban
- A tanulságok begyűjtése minden csapattól

Irodalomjegyzék
6.2.2 Buwalda, 2001, Fewster, 1999
6.3 Fewster, 1999

7 Irodalomjegyzék

Szabványok

HTB Szoftvertesztelés egységesített kifejezéseinek gyűjteménye, 2.0 verzió, 2009-10-10 (hivatalos magyar glosszárium)

ISTQB Glossary of terms used in Software Testing Version 2.0

[CMMI] Chrissis, M.B., Konrad, M. and Shrum, S. (2004) CMMI, Guidelines for Process Integration *and Product Improvement*, Addison Wesley: Reading, MA
2.1 fejezet

[IEEE 829] IEEE Std 829™ (1998/2007) IEEE Standard for Software Test Documentation (currently under revision)
2.3, 2.4, 4.1, 5.2, 5.3, 5.5, 5.6 fejezetek

[IEEE 1028] IEEE Std 1028™ (1997) IEEE Standard for Software Reviews
3.2 fejezet

[IEEE 12207] IEEE 12207/ISO/IEC 12207-1996, Software life cycle processes
2.1 fejezet

[ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality
2.3 fejezet

Könyvek

[Beizer, 1990] Beizer, B. (1990*) Software Testing Techniques* (2nd edition), Van Nostrand Reinhold: Boston
1.2, 1.3, 2.3, 4.2, 4.3, 4.4, 4.6 fejezetek

[Black, 2001] Black, R. (2001) *Managing the Testing Process* (2nd edition), John Wiley & Sons: New York
1.1, 1.2, 1.4, 1.5, 2.3, 2.4, 5.1, 5.2, 5.3, 5.5, 5.6 fejezetek

[Buwalda, 2001] Buwalda, H. et al. (2001) *Integrated Test Design and Automation*, Addison Wesley: Reading, MA
6.2 fejezet

[Copeland, 2004] Copeland, L. (2004) *A Practitioner's Guide to Software Test Design*, Artech House: Norwood, MA
2.2, 2.3, 4.2, 4.3, 4.4, 4.6 fejezetek

[Craig, 2002] Craig, Rick D. and Jaskiel, Stefan P. (2002) *Systematic Software Testing*, Artech House: Norwood, MA
1.4.5, 2.1.3, 2.4, 4.1, 5.2.5, 5.3, 5.4 fejezetek

[Fewster, 1999] Fewster, M. and Graham, D. (1999) *Software Test Automation*, Addison Wesley: Reading, MA
6.2, 6.3 fejezetek

[Gilb, 1993]: Gilb, Tom and Graham, Dorothy (1993) *Software Inspection*, Addison Wesley: Reading, MA
3.2.2, 3.2.4 fejezetek

[Hetzel, 1988] Hetzel, W. (1988) *Complete Guide to Software Testing*, QED: Wellesley, MA
1.3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 4.1, 5.1, 5.3 fejezetek

[Kaner, 2002] Kaner, C., Bach, J. and Pettticord, B. (2002) *Lessons Learned in Software Testing*,

John Wiley & Sons: New York
1.1, 4.5, 5.2 fejezetek

[Myers 1979] Myers, Glenford J. (1979) *The Art of Software Testing*, John Wiley & Sons: New York
1.2, 1.3, 2.2, 4.3 fejezetek

[van Veenendaal, 2004] van Veenendaal, E. (ed.) (2004) *The Testing Practitioner* (Chapters 6, 8, 10), UTN Publishers: The Netherlands
3.2, 3.3 fejezetek

8 „A” függelék – a tananyag háttere

A dokumentum háttere

Az ezen dokumentum alapjául szolgáló eredeti angol nyelvű tananyagot az International Software Testing Qualifications Board (ISTQB) tagjaiból álló munkacsoport 2004 és 2007 között készítette. A belső felülvizsgálat után a szoftvertesztelés különböző területeinek képviselői is elvégezték a felülvizsgálatot. A dokumentum létrehozásának szabályait a „C” függelék tartalmazza.
Ez a tananyag az alapja az ISTQB által elismert az Alapszintű Tesztelői Tanúsítvány (Certified Tester - Foundation Level) nemzetközi minősítő vizsgának.

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 lehetvé 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 Tanúsítvány céljai (a Sollentunában, 2001 novemberében tartott ISTQB közgyűlés alapján)

- 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 ország-specifikus megközelítésnek
- A kifejezésgyűjtemény és a tananyag 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
- Azáltal, 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, illetve felhasználói átvételi tesztelői, vagy szoftverfejlesztői gyakorlattal rendelkezzen
- Részt vegyen az ISQTB, vagy az ISTQB nemzeti tagszervezete (ez Magyarországon a HTB) által akkreditált tanfolyamon.

Az Alapszintű 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 Németorszgában az ASQF kezdett foglalkozni egy tesztelői minősítési rendszerrel. A tananyag az ISEB és az ASFQ újraszerkesztett és frissített tananyagain alapul, kiegészítve a tartalmát, a tesztelők gyakorlati munkáját leginkább segítő témákat hangsúlyozva.

A Nemzetközi Tanúsítvány megjelenése előtt megszerezett Alapszintű tanúsítványok (melyeket pl. Az ISEB-nél, az ASQF-nél, illetve az ISTQB nemzeti tagszervezeteinél 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 nemzeti tagszervezete felügyeli. A nemzeti 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 tananyag tanulási céljait mutatja be. Minden egyes témát a tanulási cél szem előtt tartásával kell tanulmányozni.

1. szint: Felidézés (K1)

A jelölt képes felismerni és felidézni a fogalmat.

Kulcsszavak*: emlékezet, ismeret, felidézés, felismerés*

Példa*:*
a „meghibásodás” fogalmának felidézése, mint pl:
„amikor nem szállítunk ki egy szolgáltatást a végfelhasználónak, vagy bármely más érintett félnek”, 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, minősíteni, csoportosítani és az elméleti állítást példákkal alátámasztani
Kulcsszavak: összefoglalás, általánosítás, elvonatkoztatás, osztályozás, összehasonlítás, hozzárendelés, szembeállítás, szemléltetés, értelmezés, kifejezés, következtetés, csoportosítás, modell alkotás

Példák:

Miért kell a teszteket olyan korán megtervezni, amennyire csak lehetséges?
- hogy akkor találjuk meg a hibákat, amikor még olcsóbb az eltávolításuk
- hogy először a legfontosabb hibákat találjuk meg

Az integrációs teszt és a rendszerteszt hasonlóságainak és különbségeinek kifejtése
- hasonlóságok: egynél több komponens tesztje, nem-funkcionális aspektusokat is lehet tesztelni
- különbségek: az integrációs teszt az interfészekre és a kölcsönhatásokra koncentrál, a rendszerteszt az egész rendszerre, mint pl. a teljes rendszeren átívelő munkafolyamatokra

3. szint: Alkalmazás (K3)

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

Kulcsszavak: alkalmazás, végrehajtás, használat, eljárás követése, eljárás alkalmazása

Példák:

- a jelölt képes határértékekeket meghatározni érvényes és érvénytelen partíciókhoz

- a jelölt képes egy állapotátmenet diagram alapján olyan tesztesetek készíteni, amelyek az összes lehetséges átmenetet lefedik

4. szint: Elemzés (K4)

A jelölt képes egy eljáráshoz, vagy technikához kapcsolódó információkat a jobb megértés érdekében kisebb részekre bontani és képes a tények és a következtetések megkülönböztetésére. Jellemző alkalmazási terület lehet a dokumentációk, a szoftver, vagy a projekt helyzetének elemzése és javaslattétel megfelelő lépésekre a probléma, vagy feladat megoldása érdekében.

Kulcsszavak: elemzés, szervezés, összefüggések megtalálása, integrálás, vázlat, struktúra, attribútum, elemekre bontás, megkülönböztetés, fókusz, kiválasztás

Példák:

- a termékkockázatok kezelése, megelőző és javító kockázatcsökkentő lépések kezdeményezése
- annak leírása, hogy az incidensjelentés mely részei tartalmaznak tényeket, és melyek következnek az eredményekből

Könyvek

(a tanulási célok kognitív szintjeihez)
Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

10 „C” függelék – az ISTQB alkalmazott szabályai

Alapszintű tananyag

A tananyag elkészítése és felülvizsgálata során az alábbi szabályokat alkalmazták (a kifejezés után zárójelben a szabály rövidítése)

Általános Szabályok (ÁSz):

ÁSz1: A 0-6 hónap tesztelői gyakorlattal rendelkező tesztelők számára is érthetőnek és elsajátíthatónak kell lennie a tananyagnak (6 HÓNAP)

ÁSz2: a tananyag inkább gyakorlati, mint elméleti legyen (GYAKORLATIAS)

ÁSz3: a tananyagnak egyértelműnek és világosnak kell lennie a megcélzott olvasói számára (EGYÉRTELMŰ)

ÁSz4: a tananyagnak érthetőnek kell lennie a különböző anyanyelvű olvasók számára és könnyen le kell tudni fordítni más nyelvre (LEFORDÍTHATÓ)

ÁSz5: a tananyag amerikai angolt használ (AMERIKAI ANGOL)

Jelenlegi tartalom (JT):

JT1: a tananyagnak az általánosan elfogadott aktuális tesztkoncepciókat és bevált tesztelési gyakorlatokat kell tartalmaznia. A tananyagot 3-5 évenként újra felül kell vizsgálni (AKTUÁLIS)

JT2: a tananyagnak minimalizálnia kell a hosszútávra szóló, így könnyen aktualitását vesztő részeket – például a jelenlegi piaci körülmények befolyásoló hatását - hogy 3-5 éven keresztül aktuális maradhasson (AKTUALITÁS)

Tanulási Célok (TC):

TC1: a tanulási céloknál különbséget kell tudni tenni a következő elemek között: felismerendő/felidézendő (1. kognitív szint), koncepcionálisan megértendő (2. kognitív szint), gyakorlatban használandó (3. kognitív szint), a dokumentum, a szoftver, a projekt helyzetének használata során egy bizonyos összefüggésben használandó (4. kognitív szint) (ISMERETSZINT)

TC2: a tananyag tartalma összefüggésben kell, hogy legyen a tanulási céllal (TANULÁSI CÉL - KONZISZTENCIA)

TC3: a tanulási célok jobb megértése érdekében minden egyes főbb fejezethez mintakérdéseket célszerű bemutatni (TANULÁSI CÉL – VIZSGA)

Általános Struktúra (ÁS):

ÁS1: a tananyag struktúrájának világosnak kell lennie, kereszthivatkozásokat kell tartalmaznia. Mind a vizsgakérdéseknek, mind más dokumentumoknak meg kell hivatkoznia a tananyagot (KERESZTHIVATKOZÁS)

ÁS2: a tananyag fejezetei között minimalizálni kell az átfedéseket (ÁTFEDÉS)
ÁS3: a tananyag minden fejezetének ugyanolyan felépítésűnek kell lennie (STRUKTÚRA KONZISZTENCIA)
ÁS4: a tananyag minden oldalán szerepeljen a verziószám, a kiadás dátuma és az oldalszám (VERZIÓ)
ÁS5: a tananyagnak tartalmaznia kell egy iránymutatót, hogy az egyes fejezetekre mennyi időt kell szánni (ez a téma relatív fontosságára utal) (ELTÖLTÖTT IDŐ)

Hivatkozások (H):

H1: az oktató segítése érdekében a tananyagnak forrásokat és hivatkozásokat kell tartalmaznia, ahol az egyes témákhoz részletesebb információkat lehet találni (HIVATKOZÁSOK)
H2: amennyiben nincs elegendő forráshivatkozás, úgy a tanyanyagnak további részleteket kell tartalmaznia. Például a kifejezésgyűjtemény tartalmazza a definíciót, és csak a kifejezés van felsorolva a tananyagban. (NEM HIVATKOZOTT RÉSZLETEK)

Információforrás:

A tananyagban használt kifejezések megtalálhatók a „Szoftvertesztelés egységesített kifejezéseinek gyűjteménye” című, a HTB honlapjáról letölthető file-ban. Ennek az angol eredeti verziója (glossary) megtalálható és letölthető az ISTQB honlapjáról
A tananyaggal párhuzamosan az ajánlott könyvek listája szintén kiadásra kerül. A legfontosabb könyveket az „Irodalomjegyzék” szakasz is tartalmazza.

11 „D” függelék – megjegyzések az oktató cégek részére

Minden fejezet mellett megtalálható a hozzá rendelt oktatási idő mennyisége. Ez egyrészt azt szolgálja, hogy áttekintést nyújtson az akkreditált tanfolyamok során az egyes fejezetekhez rendelt oktatási idők részarányáról, másrészt megmutatja, mennyi az a minimális idő, amit ezen fejezet oktatására kell fordítani. Természetesen mind az oktatók, mind a hallgatók szánhatnak több időt az adott fejezetre. A tanfolyam anyagának sorrendje nem feltétlenül kell, hogy megegyezzen az itteni tananyag sorrendjével.
A tananyag meghivatkozza a szabványokat, amelyeket az oktatási anyag előkészítése során kell használni. Minden egyes felhasznált szabvány esetében a tananyagban meghivatkozott verziójú szabványt kell használni. Egyéb, a tananyagban meghivatkozott publikációk, példák, vagy szabványok szintén használhatók és meghivatkozhatók, de a vizsgán nem szerepelhetnek.

A tananyag alábbi speciális területei valamilyen gyakorlati példát tesznek szükségessé:

4.3 Specifikáció alapú, vagy feketedoboz technikák

Gyakorlati példák szükségesek az alábbi 4 technika elsajátításához: ekvivalencia particionálás, határérték elemzés, döntési tábla teszt és állapotátmenet teszt. Célszerű, hogy ezen feladatok és gyakorlatok szerepeljenek az oktatás során az adott fejezethez nyújtott hivatkozási listában.

4.4 Specifikáció alapú, vagy feketedoboz technikák

Gyakorlati példák szükségesek, amelyeknél el kell dönteni, hogy a tesztsorozat eléri-e a 100% utasítás-, illetve 100 döntési lefedettséget, továbbá egy adott vezérlési folyamhoz tartozó tesztek műszaki megtervezéséhez.

5.6 Incidens menedzsment

Gyakorlati példák szükségesek az incidensjelentés írására, és/vagy értékelésére.

12 „E” függelék – a 2010-es tananyag kiadási megjegyzései

1. Több ponton megváltoztak a Tanulási Célok.
    - Stilisztikai változás a következő TC-knél: 1.2.2, 1.4.1, 2.1.1, 2.1.3, 4.6.1, 6.3.2
    - K4 Szint hozzáadása. Indoklás: néhány követelmény (TC 4.4.4, TC 5.6.2) a K4 szintnek megfelelően lett megfogalmazva, valamint a TC 4.6.1 kérdéseket egyszerűbb K4 szinten leírni és vizsgálni.
    - TC 1.1.5 átfogalmazásra került, és K2 szintre lett átállítva. A hibákkal kapcsolatos fogalmak összehasonlítása elvárható.
    - TC 1.2.3 – a hibakeresés és a tesztelés fogalmai közötti különbség kifejtése egy új TC. A tartalom benne volt az anyagban.
    - TC 3.1.3 – az összehasonlítás bekerült az anyagba.
    - TC 3.1.4 – törölve, részben redundáns a TC 3.1.3-mal.
    - TC 3.2.1 – tartalmi konzisztencia
    - TC 3.3.2 – K2 szintre átállítva, hogy konzisztens legyen a TC 3.1.2-vel
    - TC 6.1.2 – törölve, mivel ez a TC 6.1.3 része is, amely a K2 szint helytelen használata miatt átfogalmazásra került
2. A tesztelési megközelítés konzisztens használata a kifejezésgyűjteménynek megfelelően. A tesztstratégia, mint felidézendő kifejezés már nem szerepel.
3. Az 1.4 fejezetbe bekerült a tesztbázis és a tesztesetek közötti nyomonkövethetőség
4. A 2.x fejezetekbe bekerült a teszt tárgya és a tesztbázis
5. A tananyag az ellenőrző teszt kifejezés helyett – a kifejezésgyűjteménynek megfelelően - jellemzően az újratesztelés kifejezést használja
6. Az adatminőség és a tesztelés szempontja több helyen bekerült a tananyagba: adatminőség és kockázat a 2.2, 5.5 és 6.1.8 fejezetekben
7. 5.2.3: a Belépési Feltételek megjelentek, mint új alfejezet. Indoklás: a Kilépési Feltételekkel való konzisztencia (-> TC 5.2.9 Belépési Feltétel bekerült)
8. A tesztstratégia és a tesztelési megközelítés fogalmainak a kifejezésgyűjteménnyel konzisztens használata
9. A 6.1. fejezet lerövidült, mivel az eszközök leírása túlzottan hosszú volt egy 45 perces leckéhez
10. Kiadásra került az IEEE Std 829:2008. Ezen tananyag ezt még nem tartalmazza. Az 5.2. fejezet meghivatkozza a Fő Teszterv dokumentumot. A fő teszterv dokumentum tartalmát lefedi a teszterv azáltal, hogy a tervezés különböző szintjeit határozza meg: teszttervet létre lehet hozni az egyes tesztszintekre, valamint projekt szinten is, ezáltal több tesztszintet lefedve. Ez utóbbit a kifejezésgyűjtemény és a tananyag is Fő Teszterv-nek nevezi.
11. Az etikai kódex a CTAL szintről átkerült a CTFL szintre.

•

Tárgymutató