

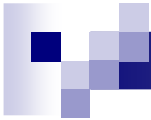


Szoftvertchnológia

6. Tesztelés (1)

BSc kurzus

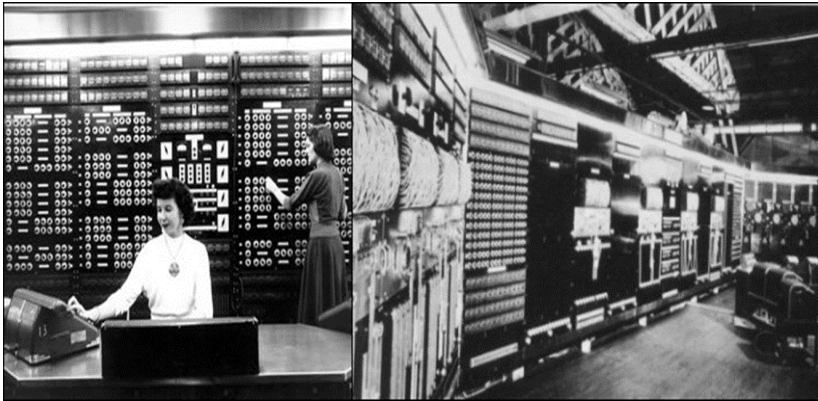
Dr. Balla Katalin



Tartalom

- Miért szükséges tesztelni?
- A tesztelés definíciója; hibák, programhibák, bukások
- A tesztelés és minőségbiztosítás közötti különbség
- A tesztelés 7 alapelve
- Az alapvető tesztelési folyamat
- A tesztelés alapidokumentumai
- A tesztelés pszichológiája
- A tesztek csoportosítása

Egy kis történelem...



<http://www.csfieldguide.org.nz/releases/1.9.9/teacher/SoftwareEngineering.html>

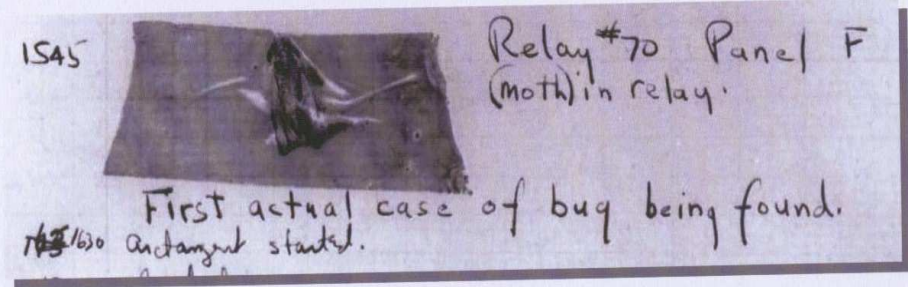


Wann fing alles an?

1947

The First Bug

A moth, accidentally entering the Mark II, causes the failure of relay number 70 in panel F. The first »bug« has been »born«. Mrs. Grace Murray Hoppers finds the poor thing and tapes it in her log book with an entry.



Prof. Dr. A. Spillner, SQM Conference, Düsseldorf, 2005

Miért szükséges tesztelni?

- Mert a szoftverben **hibák lehetnek**, és a tapasztalat azt mutatja, hogy **hibák vannak**.
- Ezek között súlyosak, életet veszélyeztetők is lehetnek.
- Megfelelő teszteléssel esélyünk van megtalálni a hibákat.
- A „Therac-25” esete.
 - Visszavonásáig legkevesebb öt páciens halt meg a túlzottan erős elektronbombázás miatt.
 - „Malfunction 54”. Hogy melyik hibakód mit jelent, az nem volt leírva sehol.
 - Rossz minőségű kód; sok „Enter” egymás után.
 - „X-ray” mód „Electron” mód, forgótárcsa.
 - A korábbi mechanikus védelmet kivették
 - A programot a gyártó egyáltalán nem tesztelte a Therac-25-ben.

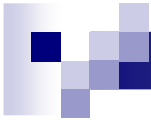




További források

- <https://www.mis-asia.com/tech/applications/top-software-failures-of-20152016/?page=5>
- http://hvg.hu/tudomany/20110213_bug_toplista
- http://lemil.blog.hu/2012/04/09/general_protection_fault_ii
- An Investigation of the Therac-25 Accidents. **Nancy Leveson, University of Washington. Clark S. Turner, University of California, Irvine.** Reprinted with permission, *IEEE Computer*, Vol. 26, No. 7, July 1993, pp. 18-41.
http://courses.cs.vt.edu/professionalism/Therac_25/Therac_1.html
- Bevezető előadásban említett „BKK esettanulmány”





Hibák a szoftverben

- A bevezető előadásban említettük, hogy a szoftverben hibák vannak, mert:
 - A szoftvert emberek írják, és “tévedni emberi”
 - Komplex feladatok elvégzésénél az emberek követnek el hibákat, ez elkerülhetetlen
- Tapasztalt programozók átlagban minden 7-10 forrássorban vétenek 1 hibát
- Ezen hibák felét a gépnyelvre történő fordításkor kijavítják
- A tesztelés során további hibák is kijavulnak, de a hibák 15%-a bent marad az ügyfélnek való átadáskor

(Watts Humphrey: „What if your life depended on software?” Presentation at EuroSPI2000 conference, Copenhagen, 2000. april)

Hibák a szoftverben



Hibák a szoftverben





Hibák a szoftverben

- És mi van, ha a rendszer kritikus? Akkor hány hiba lehet benne?
- Hogyan kezeljük ezeket a hibákat?
- Kézenfekvő válasz:

☐ **Teszteljünk!**

- Minél több ideig...
- Minél korábban...
- Minél több elemet...



Probléma...

- Sajnos, a gyakorlatban nem lehetséges mindent 100%-ban letesztelni...

Probléma...



Probléma...



50:50

Probléma...



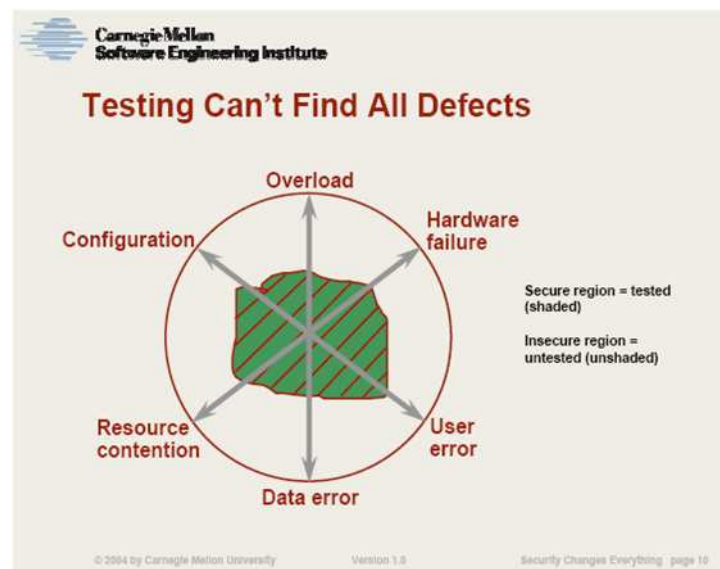


Probléma...

- A gyakorlatban nem lehetséges mindent 100%-ban letesztelni...
 - ...ezért a tesztelés **célját, tartalmát, mértékét, időtartamát...** és egyéb feltételeit **mindig az aktuális helyzet ismeretében kell meghatározni!**
- Ehhez kellene a készülő szoftverrel és a teszteléssel kapcsolatos ismeretek!

Tesztelés és szoftverminőség

- A tesztelés a szoftverminőség fontos eleme ...
 - De a szoftver jó minősége nem biztosítható csak teszteléssel!
 - További részletek a 10. előadásban (Minőségbiztosítás...)





Hogyan lehet a szoftver megfelelő minőségét biztosítani?

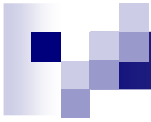
■ Folytonos odafigyeléssel

- ☐ Teszteléssel
- ☐ Megfelelő ellenőrzési, fejlesztési és támogató folyamatokkal, melyeket megfelelően határoztak meg és betartanak
 - Ezt sok minőségügyi modell, szabvány, módszer támogatja
It is supported by quality models, standards and methods
 - Ezekben a modellekben a tesztelés is egy folyamat
- ☐ Megfelelő számú, megfelelően képzett szakemberrel
- ☐ Egyéb, szükséges erőforrásokkal
- ☐ Folytonos szemlézésekkel, inspekciókkal
- ☐ ...



A tesztelés definíciója

- ...a rendszer próbafuttatása
- ...a valós működés szimulálása
- ...annak ellenőrzése, hogy jól értettük-e a követelményeket?
- ...annak ellenőrzése, hogy a követelmények mindegyikének van-e megfelelője a modellekben?
- ...a dokumentáció / kód átolvasása
- egy fejlesztéssel létrehozott / integrált / módosított / karbantartott ...szoftver rendszerrel kapcsolatos, tudatosan tervezett, definiált, végrehajtott, követett, mért, elemzett tevékenység-sorozat, amely arra irányul, hogy
 - a hibákat minél előbb megtaláljuk, kijavítsuk
 - a hibák okát felderítsük, újabb előfordulásukat megakadályozzuk.



Tesztelés \neq hibakeresés!
Tesztelés \neq tesztek futtatása!!!

Szoftvertesztelők Magyarországon és a nagyvilágban

- Egy sikeres nemzetközi szervezet: az ISTQB (International Software Testing and Qualifications Board)



www.istqb.org

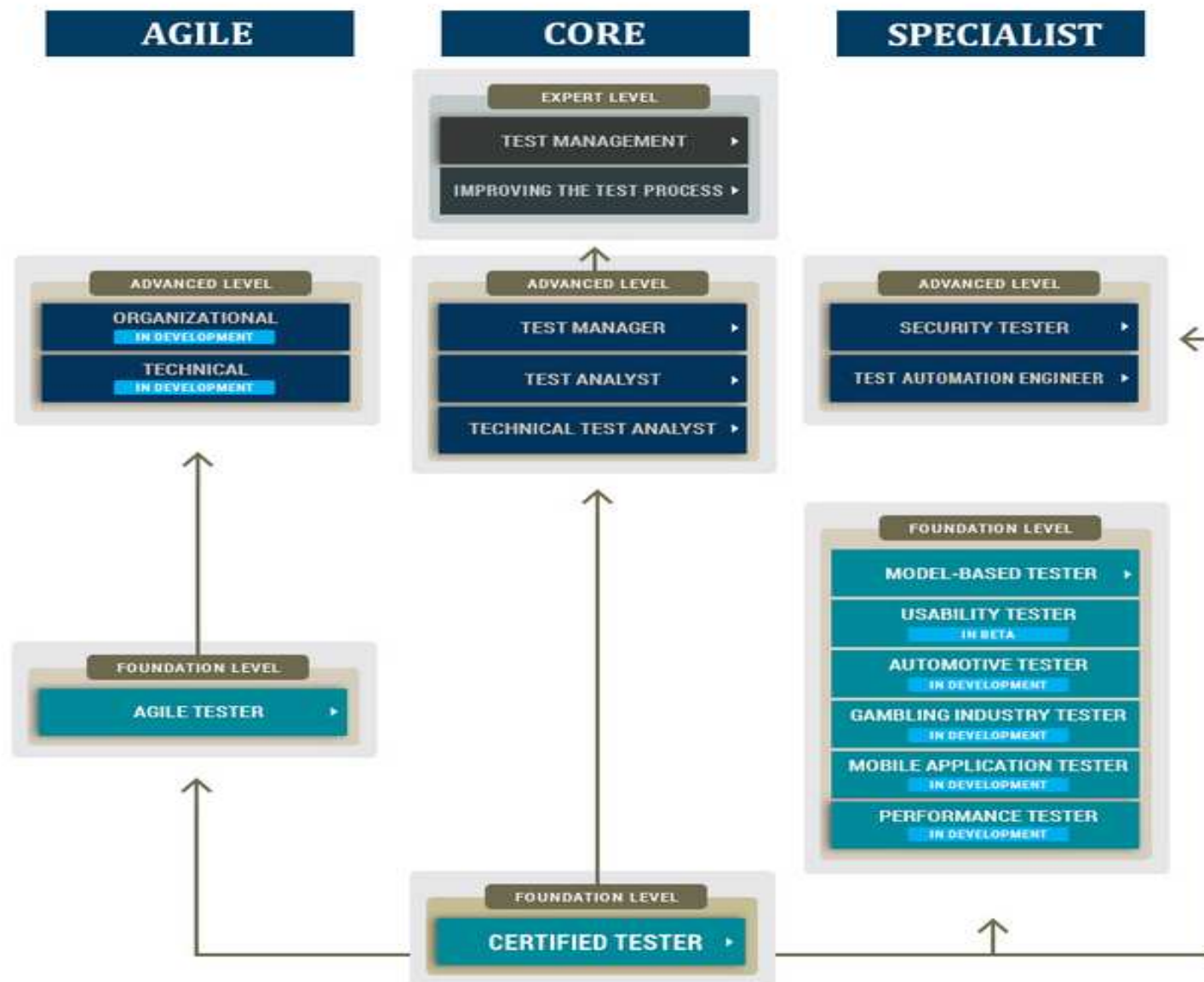


www.hstqb.com

- 2016 végén több, mint 700 000 szoftvertesztelői vizsgát szerveztek és több, mint 500 000 bizonyítványt adott ki az ISTQB® (és tagszervezetei), több, mint 117 országban.
- Szeptember 9-e a Tesztelők világnapja!

<http://www.worldwideweirdholidays.com/september-9-testers-day/#.WbOIBttJspg.facebook>

Tesztelők képzése



A szoftvertesztelés céljai

- Különböző tesztcélok léteznek:
 - Meggyőződni róla, hogy a tesztelt rendszer a követelményeknek megfelelően működik
 - Ellenőrizni, hogy minden követelményt teljesít a rendszer
 - megtalálni a hibákat ;
 - növelni a minőséggel kapcsolatos megbízhatóságot, információt nyújtani;
 - megelőzni a hibákat .
 - Elegendő információt nyújtani az érdekelt feleknek ahhoz, hogy megalapozott döntéseket hozhassanak

- Forrás: http://www.hstqb.com/images/d/d8/HTB-Glossary-3_2.pdf
- (Szoftvertesztelés egységesített kifejezéseinek gyűjteménye)
- http://www.hstqb.com/images/0/02/Syllabus_CTFL_HU-public-v10-20091006_3_02.pdf
- Hivatalos magyar nyelvű tanterv. Alapszintű képesítés
- Angolul:
- ISTQB Foundation Syllabus
- <http://www.istqb.org/downloads.html>





Definíciók

- **Programhiba:** a program olyan belső hibája, amely azt eredményezheti, hogy a szoftver nem tudja teljesíteni az elvárt viselkedését, azaz a program meghibásodásához vezethet. → *bug, defect, fault*
- **Meghibásodás:** a komponens, illetve a rendszer eltér az elvárt eredménytől, vagy szolgáltatástól. → *failure*
- **Emberi eredetű hiba:** emberi tevékenység, amely során helytelen eredmény jön létre. → *error, mistake*
- **Hibakeresés:** a szoftver meghibásodás okainak megtalálási, analízálási és eltávolítási folyamata. → *debugging*
- **Hibamaszkolás:** olyan állapot, amikor az egyik hiba megakadályozza a másik hiba megtalálását. → *defect masking, fault masking*
- **Előfeltétel:** környezeti vagy állapotbeli feltételek, amelyeket teljesíteni kell, mielőtt egy komponensen vagy rendszeren tesztet vagy tesztelési folyamatokat kezdenénk. → *precondition*

A szoftvertesztelés 7 alapelve (1)

- **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.



A szoftvertesztelés 7 alapelve (2)

- **5. alapelv – A féregirtó paradoxon**
- Ha mindig ugyanazokat a tesztekert 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.



A tesztelési folyamat elemei

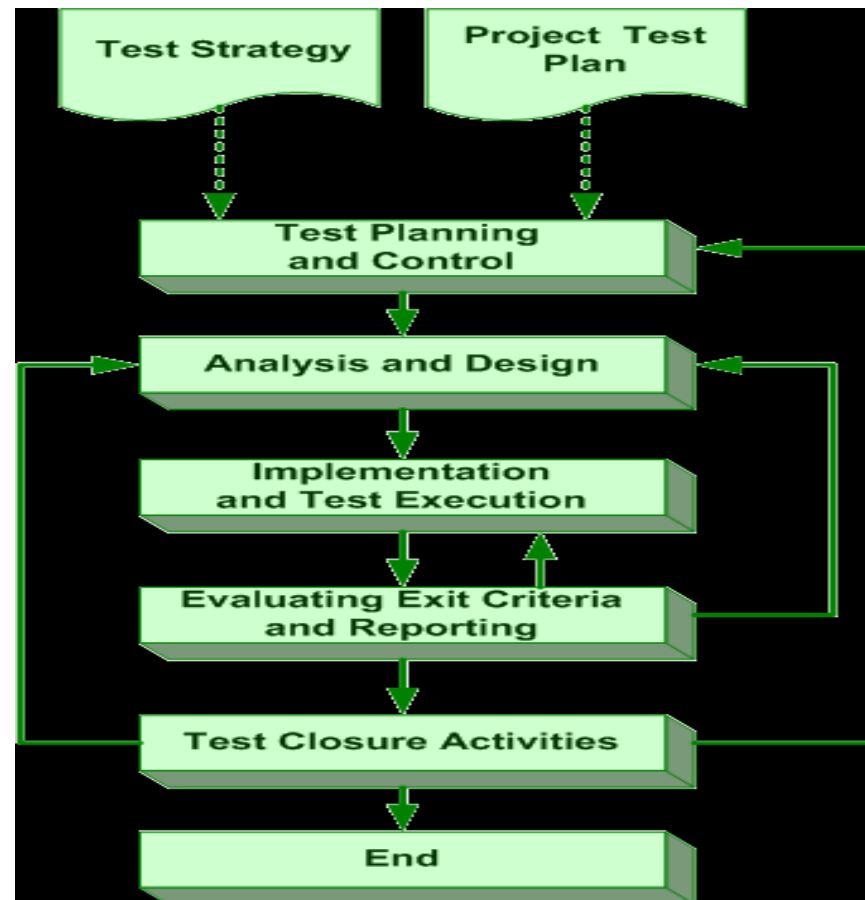
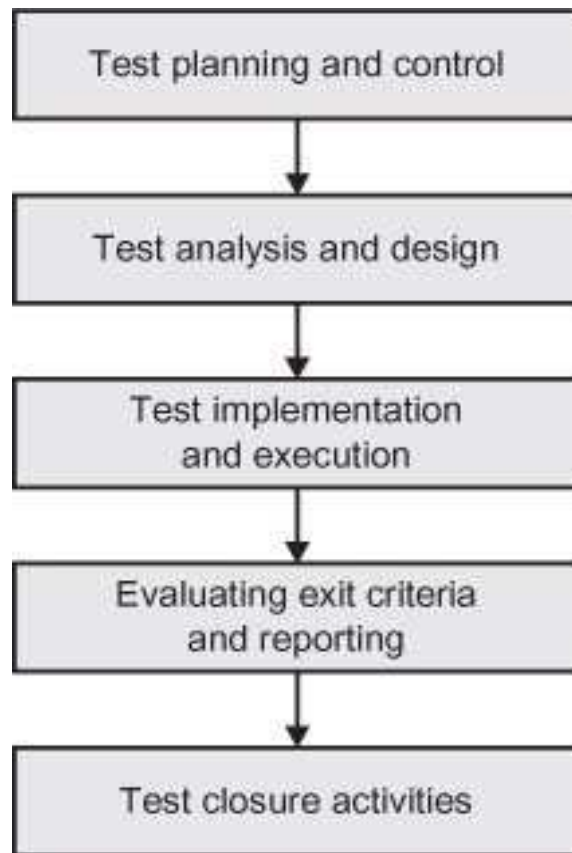
- A tesztelés alapvető folyamata a következő fő tevékenységekből áll:
 - ☐ tervezés és irányítás;
 - ☐ elemzés és műszaki tervezés;
 - ☐ megvalósítás és végrehajtás;
 - ☐ a kilépési feltételek értékelése és jelentés;
 - ☐ teszt lezárása.
- ☐ Ezek a szoftverfejlesztési életciklus modellbe integrálódnak. **Bármelyik** életciklus modellbe!
- ☐ 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.



A tesztelési folyamat elemei

- Ne feledjük:
- A tesztelés szoftverfejlesztési életciklus minden fázisában megjelenik
 - ☐ Követelmények fejlesztése
 - ☐ Tervezés
 - ☐ Kódolás
 - ☐ Tesztelés
 - ☐ Átadás
- A legjobb, ha a tesztelés a fejlesztési életciklus elején elkezdődik és a teljes életciklus alatt tart!

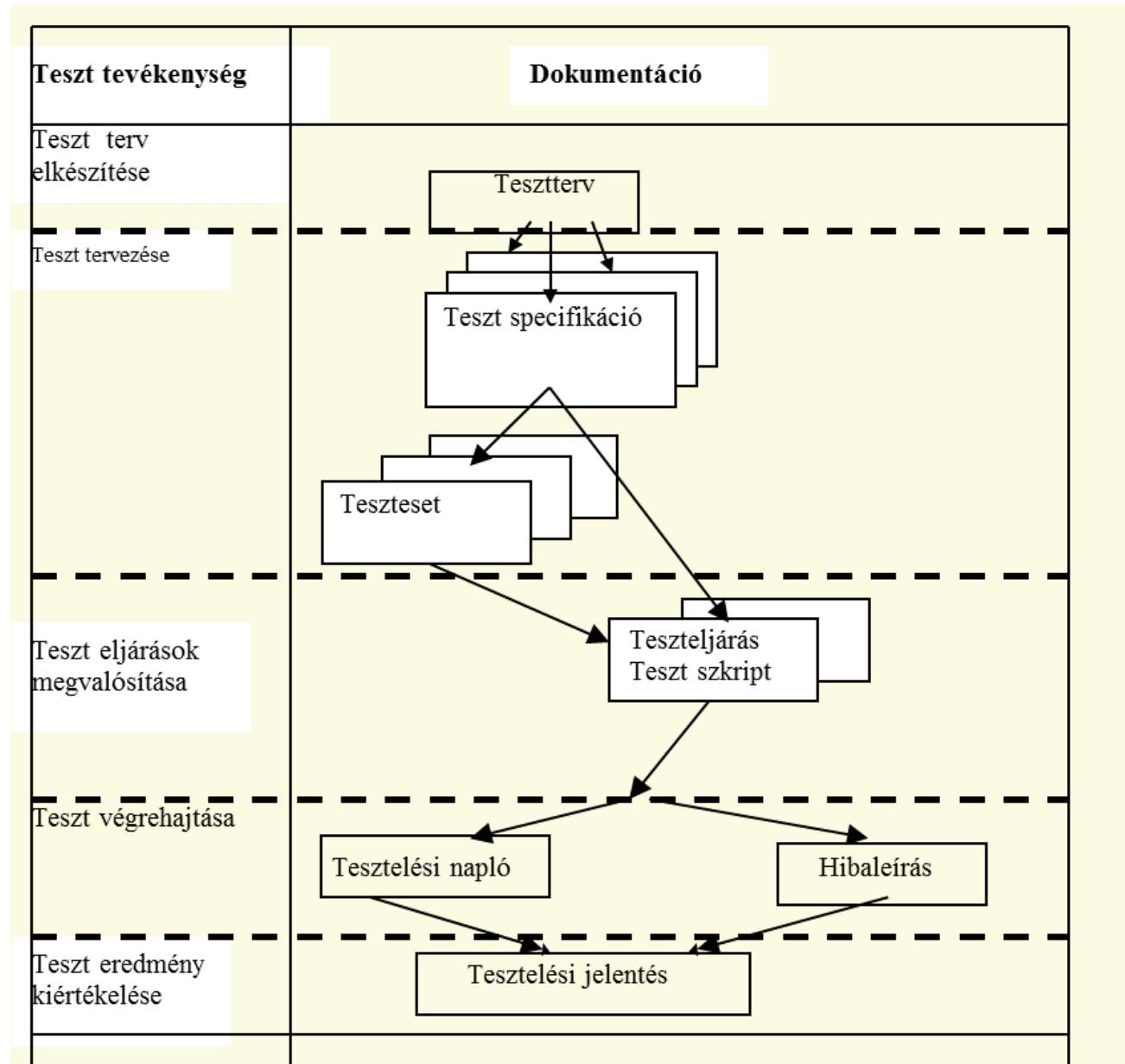
Az alapvető tesztelési folyamat



Az alapvető tesztelési folyamat

A tesztelés
tervezése,
végrehajtása,
kiértékelése
dokumentált
folyamat!

Példa, az IQSOFT –
nál kidolgozott modell
alapján



Tesztelés a CMMI-ben

- Nem egyetlen folyamat
 - ☐ VER
 - ☐ VAL
 - ☐ PI
- Mindegyik fejlesztési folyamat (Engineering), és ML3 szinten jelennek meg.
- Fontos, hogy a kódon kívül egyéb elemeket is tesztelni kell!

Tesztelés a CMMI-ben

- Verifikáció (VER): Célja annak biztosítása, hogy a munkatermékek teljesítsék a specifikált követelményeket.
- SG 1 Előkészítés verifikációra
 - SP 1.1 Munkatermékek kiválasztása verifikációra
 - SP 1.2 Verifikációs környezet létrehozása
 - SP 1.3 Verifikációs eljárások és kritériumok létrehozása
- SG 2 Egyenrangú szemlék végrehajtása
 - SP 2.1 Egyenrangú szemlék előkészítése
 - SP 2.2 Egyenrangú szemlék végrehajtása
 - SP 2.3 Egyenrangú szemlék adatainak elemzése
- SG 3 Kiválasztott munkatermékek verifikációja
 - SP 3.1 Verifikáció végrehajtása
 - SP 3.2 Verifikációs eredmények elemzése

Tesztelés a CMMI-ben

- Validáció (VAL): Megmutatja, hogy a termék vagy termék-komponens a tőle elvárt módon fog-e működni, amikor a célkörnyezetébe kerül.
- SG 1 Előkészítés validációra
 - SP 1.1 Érvényesítendő termékek kiválasztása
 - SP 1.2 Validációs környezet létrehozása
 - SP 1.3 Validációs eljárások és kritériumok létrehozása
- SG 2 Termék vagy termékkomponens validációja
 - SP 2.1 Validáció végrehajtása
 - SP 2.2 Validációs eredmények elemzése

Tesztelés a CMMI-ben

- Termék integráció (PI): Célja a termék összerakása a termék-komponensekből és annak biztosítása, hogy az integrált termék megfelelő módon működjön, valamint a késztermék átadása.
- SG 1 Termék integráció előkészítése
 - SP 1.1 Integrációs stratégia meghatározása
 - SP 1.2 Termékintegrációs környezet meghatározása
 - SP 1.3 Termékintegrációs folyamatok és kritériumok meghatározása
- SG 2 Interfész kompatibilitás biztosítása
 - SP 2.1 Interfész leírás szemléje a teljesség biztosítására
 - SP 2.2 Interfészek menedzselése
- SG 3 Termékkomponensek összeépítése és a termék átadása
 - SP 3.1 A termékkomponensek integrációra alkalmasságának igazolása
 - SP 3.2 Termékkomponensek összeépítése
 - SP 3.3 Összeépített termékkomponensek kiértékelése
 - SP 3.4 A termék vagy termék-komponens összeállítása és átadása

Testing in CMMI

- Verification
 - The purpose of Verification (VER) is to ensure that selected work products meet their specified requirements.
- SG 1 Prepare for Verification
 - SP 1.1 Select Work Products for Verification
 - SP 1.2 Establish the Verification Environment
 - SP 1.3 Establish Verification Procedures and Criteria
- SG 2 Perform Peer Reviews
 - SP 2.1 Prepare for Peer Reviews
 - SP 2.2 Conduct Peer Reviews
 - SP 2.3 Analyze Peer Review Data
- SG 3 Verify Selected Work Products
 - SP 3.1 Perform Verification
 - SP 3.2 Analyze Verification Results



Testing in CMMI



■ Validation

- ☐ The purpose of Validation (VAL) is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment

■ SG 1 Prepare for Validation

- ☐ SP 1.1 Select Products for Validation
- ☐ SP 1.2 Establish the Validation Environment
- ☐ SP 1.3 Establish Validation Procedures and Criteria

■ SG 2 Validate Product or Product Components

- ☐ SP 2.1 Perform Validation
- ☐ SP 2.2 Analyze Validation Results



Testing in CMMI

- Product Integration
 - The purpose of Product Integration (PI) is to assemble the product from the product components, ensure that the product, as integrated, behaves properly (i.e., possesses the required functionality and quality attributes), and deliver the product.
- SG 1 Prepare for Product Integration
 - SP 1.1 Establish an Integration Strategy
 - SP 1.2 Establish the Product Integration Environment
 - SP 1.3 Establish Product Integration Procedures and Criteria
- SG 2 Ensure Interface Compatibility
 - SP 2.1 Review Interface Descriptions for Completeness
 - SP 2.2 Manage Interfaces
- SG 3 Assemble Product Components and Deliver the Product
 - SP 3.1 Confirm Readiness of Product Components for Integration
 - SP3.2 Assemble Product Components
 - SP 3.3 Evaluate Assembled Product Components
 - SP 3.4 Package and Deliver the Product or Product Component



TMMi – 16 tesztelési folyamat

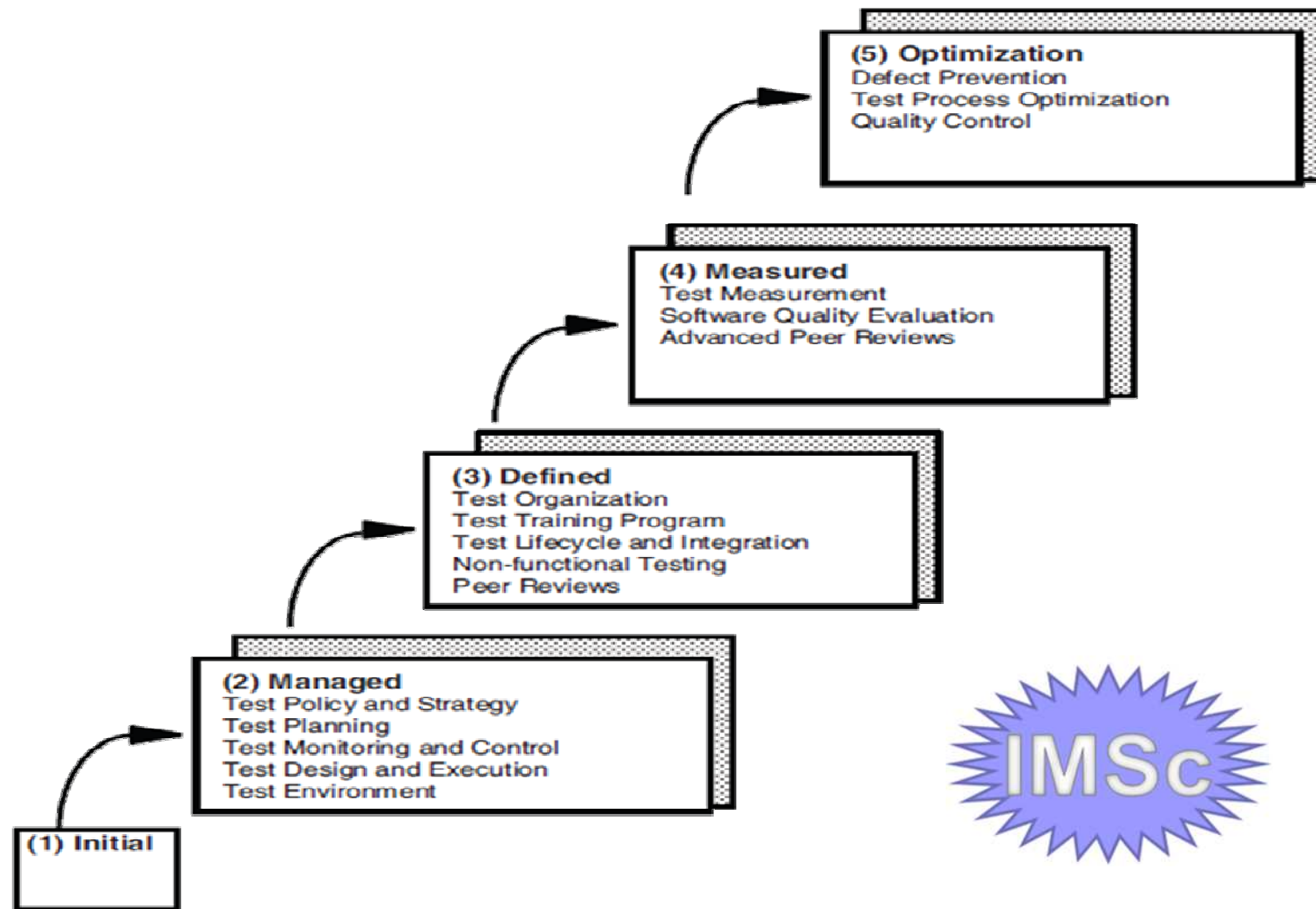
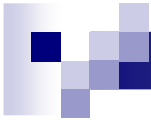


Figure 1: TMMi maturity levels and process areas



A tesztelés dokumentálása

- A tesztelés minden elemét / tevékenységét dokumentálni kell
- A felhasználók is kérnek tesztelést igazoló dokumentumokat
- Általában a következő dokumentumokat készítjük el a teszteléshez kapcsolódóan:
 - ☐ Tesztelési irányelvek
 - ☐ Teszt terv (és ennek részletezése)
 - ☐ Teszt scenárió / forgatókönyv
 - ☐ Teszt eset
 - ☐ Teszt szkript
 - ☐ Teszt adatok
 - ☐ Tesztelési jegyzőkönyvek
 - ☐ Hibalista, hibák leírása
 - ☐ Átadási –átvételi dokumentumok



Tesztelési irányelvek

- Magas szintű dokumentum, amely a szervezet elveit, megközelítésmódját, valamint céljait mutatja be a tesztelésre vonatkozóan. → *test policy*



Tesztterv

- A teszt hatáskörét, megközelítését, erőforrásait valamint a tevékenységek tervezett ütemezését tartalmazó dokumentum. Ezen kívül meghatározza a tesztelemekeket, a tesztelendő funkciókat, feladatokat, a tesztet végrehajtó személyek függetlenségét, a tesztkörnyezetet, a műszaki teszttervezési technikákat, a belépési és kilépési feltételeket, valamint kockázatokat. A teszttervezési folyamat meghatározó dokumentuma [IEEE 829 alapján]. → *test plan*

Példa tesztterv template (minta)

Sample Template for a Unit Test Plan

Sr.	Requirements	Typical Components	Detailed Description
1)	Introduction	a) Test Strategy and Approach	
		b) Test Scope	
		c) Test Assumptions	
2)	Walkthrough (Static Testing)	a) Defects Discovered and Corrected	
		b) Improvement Ideas	
		c) Structured Programming Compliance	
		d) Language Standards	
		e) Development Documentation Standards	
3)	Test Cases (Dynamic Testing)	a) Input Test Data	
		b) Initial Conditions	
		c) Expected Results	
		d) Test Log Status	
4)	Environment Requirements	a) Test Strategy and Approach	
		b) Platform	
		c) Libraries	
		d) Tools	
		e) Test Procedures	
		f) Status Reporting	

<http://www.softwaretestinggenius.com/unit-test-plan-and-its-sample-template>

11/5/2017

<http://www.softwaretestinggenius.com>



A részletek leírása

- **Teszteljárás specifikáció / forgatókönyv / szcenárió:** a teszt futtatásának tevékenységsorozatot rögzítő dokumentum. Teszt szkript illetve manuális teszt szkript néven is ismert. [IEEE alapján]. Lásd még: tesztspecifikáció. → *test procedure specification*
- **Teszteset specifikáció:** egy teszteleme-re vonatkozó, a teszteseteket leíró dokumentáció (cél, bemenetek, teszttevékenységek, elvárt eredmények, végrehajtás előfeltételei) [IEEE 829 alapján]. Lásd még: tesztspecifikáció. → *test case specification*

Teszt forgatókönyv / szcenárió

■ Példa: Tesztelendő követelmény: :

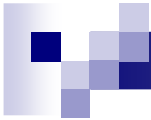
“Use case ID: UC0001:

Verify and validate the end to end functionality of an ecommerce web application. Any registered customer should be able to login with the valid username, password and place an order.”

■ Erre a követelményre vonatkozó teszt szcenárió :

Use case ID / Req.ID	Scenario ID	Test Scenario	No. of test cases
UC0001	TS001	Validate the login functionality of the application to ensure only the registered customers are allowed to login	4
UC0001	TS002	Validate the order placing functionality	6

■ A fenti példában az adott követelményre vonatkozóan 2 szcenáriót azonosítottunk, és ezek 4, illetve 6 teszt eset futtatását feltételezik.



Teszt eset

- Bemeneti értékek, végrehajtási előfeltételek, elvárt eredmények és végrehajtási utófeltételek halmaza, amelyeket egy konkrét célért vagy a tesztért fejlesztettek
 - (például egy program forgatókönyv végrehajtása, vagy egy követelménynek való megfelelés). [IEEE 610 alapján]. → *test case*

Tesztesetek azonosítása

■ Elvárás (általában):

- ☐ Teszt eset azonosító:
- ☐ Cél:
- ☐ Előfeltételek:
- ☐ Input adatok (értékek):
- ☐ Elvárt output (értékek):
- ☐ Utófeltételek:
- ☐ Teszt eset végrehajtásának története:
Dátum Eredmény Verzió Ki végezte

**A tesztelés
menedzselésében
hasznos**



Teszt eset -template példa

Your Company LOGO	Project Name:		Test Designed by:
	Module Name:		Test Designed date:
	Release Version:		Test Executed by:
			Test Execution date:
Pre-condition			
Dependencies:			
Test Priority			
Test Case#	Test Title	Test Summary	Test Steps

<http://www.softwaretestinghelp.com/test-case-template-examples/>

Teszteset példa

Teszteset példa az előbbi ID UC0001 szcenárióhoz:

	A	B	C	D	E	F	G	H	I	J
1	Scenario ID	Test case ID	Test Case Name	Test Case Description	Test Step No	Test Step description	Expected Result	Actual Result	Status	comments
2	TS0001	TC001	Valid credentials login	Test the login functionality of the ecommerce application to ensure that a registered user is allowed to login successfully by entering valid user name and password in the respective fields in the login page	Precondition	a. Ensure the application under test is available and stable to test b. Ensure the required test data for testing available and complete				
3					Step 1	Launch the ecommerce application with the given URL: <>	The ecommerce application should be launched successfully and all the menus should be loaded completely.	Application launched successfully	Pass	
4					Step 2	Navigate to the login page	The login page should be displayed with the user ID, password fields with submit button	Login page loaded successfully	Pass	
5					Step 3	Enter the valid user ID<> in the user ID field	The user ID field should be editable and should accept input	Input accepted	Pass	
6					Step 4	Enter the valid password <> in the password field	The password field should be editable and should accept input	Password field is disabled to enter input	Fail	
7					Step 5	Click on Login button	The user should be taken successfully to the user home page with a greetings message		Fail	

Teszt eset példa

Test Case Example1 (simple test)

Test Case #: 2.2

System: ATM

Designed by: ABC

Executed by:

Short Description: Test the ATM Change PIN service

Test Case Name: Change PIN

Subsystem: PIN

Design Date: 28/11/2004

Execution Date:

Page: 1 of 1

Pre-conditions

The user has a valid ATM card - The user has accessed the ATM by placing his ATM card in the machine

The current PIN is 1234

The system displays the main menu

Step	Action	Expected System Response	Pass/Fail	Comment
1	Click the 'Change PIN' button	The system displays a message asking the user to enter the new PIN		
2	Enter '5555'	The system displays a message asking the user to confirm (re-enter) the new PIN		
3	Re-enter '5555'	The system displays a message of successful operation The system asks the user if he wants to perform other operations		
4	Click 'YES' button	The system displays the main menu		
5	Check post-condition 1			

Post-conditions

1. The new PIN '5555' is saved in the database

<https://www.template.net/business/word-templates/test-case-template/>



Teszt szkript

- Legtöbbször teszteljárás specifikációra használt kifejezés, elsősorban automatizált teszt esetén. → *test script*
- Típusai:
 - Manuális tesztszkript– manuálisan létrehozott teszt esetek sorozata, több tesztadattal, mely segít a kevésbé hozzáértőknek is a teszt elvégzésében.
 - Automatizált tesztszkript– programozott teszt esetek , különböző tesztadat-kombinációkkal, melyeket eszközök végeznek
 - Ha tesztszkriptről hallunk, legtöbbször automatizált esetre gondolunk, amikor a tesztek végrehajtását valamilyen programnyelven vagy egy külön erre a célra létrehozott szkriptnyelvben írjuk le.

Teszt szkript példa

■ Példa manuális teszt szkriptre

□ Zölddel jelölték a beírandó adatokat

	A	B	C	D	E	F	G	H	I	J
1	Scenario ID	Test case ID	Test Case Name	Test Case Description	Test Step No	Test Step description	Expected Result	Actual Result	Status	comments
2	TS0001	TC001	Valid credentials login	Test the login functionality of the ecommerce application to ensure that a registered user is allowed to login successfully by entering valid user name and password in the respective fields in the login page	Precondition	a. Ensure the application under test is available and stable to test b. Ensure the required test data for testing available and complete				
3					Step 1	Launch the ecommerce application with the given URL: <http://sbc.com>	The ecommerce application should be launched successfully and all the menus should be loaded completely.	Application launched successfully	Pass	
4					Step 2	Navigate to the login page	The login page should be displayed with the user ID, password fields with submit button	Login page loaded successfully	Pass	
5					Step 3	Enter the valid user ID<adminuser> in the user ID field	The user ID field should be editable and should accept input	Input accepted	Pass	
6					Step 4	Enter the valid password <admin@123> in the password field	The password field should be editable and should accept input	Password field is disabled to enter input	Fail	
7					Step 5	Click on Login button	The user should be taken successfully to the user home page with a greetings message <Welcome home>		Fail	

Tesztzkript példa

■ Példa automatizált tesztzkriptre

```
nRowCount=DataTable.GlobalSheet.GetRowCount
systemutil.closeProcessByName "iexplore.exe"
For i = 1 to nRowCount
    Systemutil.Run "iexplore", "http://abc.com"
    Browser("title:=.*").Page("title:=.*").Sync
    DataTable.GlobalSheet.SetCurrentRow(i)
    sEmailid=DataTable("UserName",Global)
    sPassword=Datatable("Password",dtGlobalsheet)
    Browser("title:=.*").Page("title:=.*").WebEdit("name:=Email").Set sEmailid
    Browser("title:=.*").Page("title:=.*").WebEdit("name:=Passwd").Set sPassword
    Browser("title:=.*").Page("title:=.*").WebButton("name:=Sign in").Click
    Browser("title:=.*").Page("title:=.*").Sync
    wait(10)
    If left(browser("title:=.*").Page("title:=.*").getROProperty("title"),13) ="welcome home"Then
        msgbox("Signed in Successfully")
    Else
        msgbox("Login Unsuccessful")
    End If
    systemutil.closeProcessByName "iexplore.exe"
Next
```



Tesztadat

- Olyan adat, amely a teszt előtt is létezik (például egy adatbázisban) és amely kölcsönhatásban van a tesztelés alatt álló rendszerrel, vagy a rendszerkomponenssel. → *test data*
 - ☐ Input
 - ☐ Előfeltételek
 - ☐ Output
 - ☐ Utófeltételek

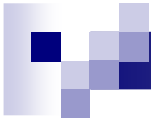
Tesztadatok - példák

	A	B	C	D	E	F	G	H	I
1									
2	First Name	Last Name	Gender	Birthdate	Email ID	Address	Phone number	Password	Confirm Password
3	Aparajita	Jain	F	24/8/90	aparajita@STC.com		1236547890	*****	*****
4	XYZ	moore	K	40/13/00		USA			
5	ABC	wilson	D	06/08/1941	Abc@wilson.com		659874123	Password	Password
6	Mohit	jain	M	08/11/1987		canada	sdfergrgrth		
7	Stc	website	M	01/04/2015	stc@website.ca	India	528	&&&&	
8									
9									

This document is a FIT input file that tests login conformance and deviance.

PetShop			
login	demo	password	
login	DEMO	password	
login	demo@bivio.biz	password	
login	Demo@Bivio.Biz	password	
login	demo	PASSWORD	does not match
login	demo		must supply a value
login		password	must supply a value
login	notuser	password	not found
login	demo'li	password	not found
login	%demo%	password	not found

<http://www.extremeperl.org/bk/acceptance-testing>



Tesztbázis

- Az összes olyan dokumentum, amelyből a komponensekre vagy rendszerekre vonatkozó követelmények származnak. Ezek azok a dokumentumok, amelyeken a tesztesetek alapulnak.
 - Ha egy ilyen dokumentumot csak formális változáskezelési folyamat során módosíthatnak, a tesztbázist ún. fagyasztott tesztbázisnak nevezik. [TMap alapján]. → *test basis*





A tesztelés bizonylatolása

- A végrehajtás során a következő információkat kell rögzíteni:
 - a tesztelést végző személy nevét
 - a tesztelt funkciót
 - a tesztelés végrehajtásának dátumát
 - a tesztelés sikeres vagy sikertelen voltát
 - milyen gépen történt a tesztelés



A tesztelés végrehajtásának dokumentálása

- **Tesztnapló:** A tesztvégrehajtáshoz kapcsolódó részletek időrendi rögzítése. → *test log, test record*
- **Hibajelentés:** olyan dokumentum, amely leírja a szoftver azon hibáit, amelyek a program elégtelen működéséhez vezethetnek. → *bug report, defect report*
- **Összefoglaló tesztjelentés:** a teszttevékenységeket és eredményeket tartalmazó dokumentum. Ebben a dokumentumban található a kilépési feltételeknek megfelelően ellenőrzött tesztelemelek kiértékelése is. → *test summary report*

Tesztnapló - példák

Test Details					
Test	Scenario	Start Time	Test Time	Outcome	Details
WebTestWithErrors	Scenario1	6/9/2009 7:24:16 AM	1.277	Failed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:16 AM	3.041	Failed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:16 AM	1.009	Passed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:16 AM	1.009	Passed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:17 AM	1.009	Passed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:17 AM	0.074	Failed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:17 AM	0.99	Passed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:18 AM	0.071	Failed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:18 AM	0.074	Failed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:18 AM	0.975	Passed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:18 AM	0.048	Failed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:18 AM	0.975	Passed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:18 AM	0.979	Passed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:18 AM	0.048	Failed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:18 AM	0.047	Failed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:18 AM	0.046	Failed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009 7:24:18 AM	1.016	Passed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:19 AM	0.048	Failed	-
WebTestWithErrors	Scenario1	6/9/2009 7:24:19 AM	0.051	Failed	-
UnitTestThatLogsAndPasses	Scenario1	6/9/2009			

Login Tests Test Log

Test Statistics

	Total Executions	Total n	Pass n	Fail n	Expected n	Pass/Fail
Current Tests	1	1	1	0	1	100%
All Tests	1	1	1	0	1	100%

Statistics by Test

Test Name	Total n	Pass n	Fail n	Expected n	Pass/Fail
UnitTestThatLogsAndPasses	1	1	0	1	100%
WebTestWithErrors	1	0	1	1	0%
UnitTestThatLogsAndPasses	1	1	0	1	100%
WebTestWithErrors	1	0	1	1	0%

Test Results Log

100% Login Tests

Full Name: Login Tests

Source: A:\source\src\Testing\testrunner\Login Tests

Start/Stop Expected: 6/9/2009 7:24:16 AM - 6/9/2009 7:24:19 AM

Result: 1 Passed, 0 Failed, 0 Expected

100% UnitTestThatLogsAndPasses

Full Name: UnitTestThatLogsAndPasses

Source: A:\source\src\Testing\testrunner\UnitTestThatLogsAndPasses

Start/Stop Expected: 6/9/2009 7:24:16 AM - 6/9/2009 7:24:17 AM

Result: 1 Passed, 0 Failed, 0 Expected

100% WebTestWithErrors

Full Name: WebTestWithErrors

Source: A:\source\src\Testing\testrunner\WebTestWithErrors

Start/Stop Expected: 6/9/2009 7:24:17 AM - 6/9/2009 7:24:18 AM

Result: 0 Passed, 1 Failed, 0 Expected

11/5/2017



A tesztek kiértékelése

- **Teszt-kiértékelési jelentés:** a tesztfolyamat végén készített dokumentáció, amely összegzi a összes teszttevékenységet és eredményt. Ezen kívül tartalmazza a tesztfolyamat kiértékelését és a teszt során szerzett tapasztalatokat. → *test evaluation report*
- E jelentés alapján születethet döntés a továbblépésről.
 - Előre meghatározott kritériumok szükségesek!



Tesztelési hibák

- A tesztelt szoftvernek a teszt során tapasztalt hibás viselkedését nem a tesztelt szoftver hibái okozzák.
- A tesztelési hiba lehet:
 - Teszt specifikációs vagy tesztadat hiba.
 - A hibát a vizsgálati pont minősítésének hibás specifikációja vagy a rosszul megválasztott tesztadatok okozták.
- A teszt hibás végrehajtása
 - A hibát a teszt nem megfelelő végrehajtása (pl. a végrehajtás lépéseinek helytelen sorrendje, ütemezési hibák) okozták.
- Hibás tesztkörnyezet
 - A hibát a tesztkörnyezet rossz megválasztása (pl. hibás tesztprogram) okozta.
 - Tesztelési hibák feltárásakor a hibát okozó elemet el kell hárítani (pl. javítani kell a teszt specifikációt, módosítani kell a teszt adatokat, a teszt környezetet, a teszt lépéseinek végrehajtási sorrendjét stb.), majd meg kell ismételni a tesztet.



Szoftver hibák

- A hibás viselkedést a tesztelt szoftver hibái okozzák. A szoftver hiba lehet:
 - Megvalósítási hiba: a szoftver nem teljesíti a rendszertervben előírtakat.
 - Tervezési hiba: a hiba a nem megfelelő rendszerterv következménye.
- Hiba felmerülése esetén a projekt vezetője dönt a további lépésekről. Elrendelheti újabb, a hiba keletkezésének körülményeit pontosabban feltáró tesztek elvégzését, visszaadhatja a tesztelt programot a hiba kijavításáért felelős személynek vagy apróbb hibák esetén azok kijavítását az átvevővel egyeztetve későbbre halaszthatja. Az ilyen döntéseket bizonylatolni kell (pl. bejegyzés a teszt naplóba, e-mail stb.)



Teszteset, ellenőrzés

- Az eredmények ellenőrzése: automatizáltan / manuálisan
- Valid és invalid eredményekre!
 - Pozitív teszt eset: valid funkcionalitás
 - Negatív teszt eset: invalid funkcionalitás (hibaüzenet...)
- **Téves hiba eredmény:** olyan teszteredmény, amely hibát jelez, bár az adott hiba valójában nem létezik a termékben. → *false-fail result, false-positive result*
- **Téves siker eredmény:** olyan teszteredmény, amely nem találja meg az adott, a termékben meglevő hibát. → *false-pass result, false-negative result*
- **Hibamaszkolás:** olyan állapot, amikor az egyik hiba megakadályozza a másik hiba megtalálását. → *defect masking, fault masking*



Hibák dokumentálása

- Az egyes hibákról a következő információkat kell rögzíteni :

- hiba azonosító
- hibajelenség leírása
- tesztelt program azonosítója
- teszteset azonosító
- a hiba felfedezésének dátuma
- a hibát felfedező személy
- a hiba súlyossága
- a hiba prioritása
- a hiba állapota (rögzített, javítás alatt, kijavított)
- a hiba javításáért felelős személy
- javítás dátuma.



Hibaelemzés

- A hiba elemzéshez általában négy fő paramétert használhatunk:
 - A hiba aktuális állapotának státusza (nyitott, kijavított, lezárt, stb.)
 - A hiba fontosságának prioritása.
 - A hiba súlyossága.
 - A forrás, ahol a hiba előfordult, és mi a hiba eredeti oka.
- A hiba elemzéskor az alábbi kimutatásokat szokás elkészíteni:
 - Hiba eloszlás riport: a hibák száma egy vagy két paraméter függvényében. (pl. a hibák száma prioritás vagy súlyosság szerint)
 - Hiba korosítás riport: milyen sokáig marad javítatlanul egy hiba.
 - Hiba tendencia riport: hibák száma státusz szerint az idő függvényében.
 - Teszt eredmény és fejlődés riport: a tesztelés végrehajtásának eredményét mutatja meg az iterációk számán és teszt ciklusokon keresztül.



Bug report template

November 11, 2005

Your name

<Print your full name>

System under test

<for example Bob and Mary's>

Date

<mm/dd/yy>

Bug

Severity of problem

<Fatal, Serious or Minor>

Problem and how to reproduce it

<Describe the problem>

Suggested fix (optional)

<describe any ideas or preferences about fixing>

Bug

Severity of problem

<Fatal, Serious or Minor>

Problem and how to reproduce it

<Describe the problem>

Suggested fix (optional)

<describe any ideas or preferences about fixing>

Bug

Severity of problem

<Fatal, Serious or Minor>

Problem and how to reproduce it

<Describe the problem>

Suggested fix (optional)

<describe any ideas or preferences about fixing>

Cut and paste bug description to document as much bugs as you need. For each error do its own description. You will probably find much more than three errors.

Resource:

"Testing Computer Software" by Cem Kaner, Jack Falk, Hung Q. Nguyen

Példa hiba bizonylatolására

<https://www.sampletemplates.com/business-templates/bug-report-template.html>

Példa hiba bizonylatolására

Microsoft Word interface showing an "Error Log - Word" document. The ribbon includes FILE, HOME, INSERT, DESIGN, PAGE LAYOUT, REFERENCES, MAILINGS, REVIEW, VIEW, and DEVELOPER. The user is Ivan Walsh.

[Project Name] [Document Name - Version Number]

Error Log Template

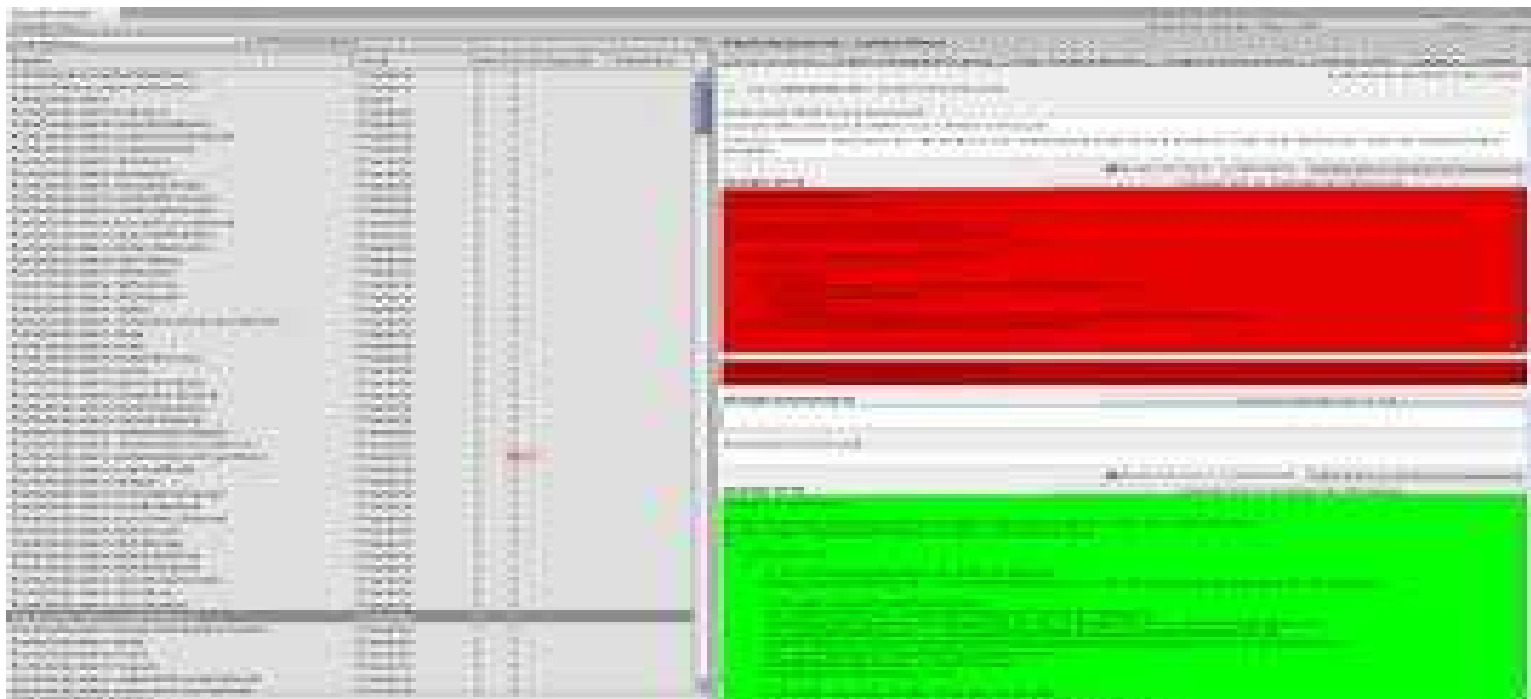
Use this form to record error details.

SCR #	Description of error/failure	Tester	Follow-up	Reviewed	Comments
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	Yes	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	No	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	Yes	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	No	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	Yes	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	No	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	Yes	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	No	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	Yes	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	No	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	Yes	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	No	<Add comments>
SCR #	Describe the nature of the error.	<Tester>	<Follow-up date>	Yes	<Add comments>

© [Name of Company] 1 | Page

PAGE 1 OF 2 425 WORDS ENGLISH (UNITED STATES)

Példa hiba bizonylatolására



- <https://www.google.hu/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwj91eeq45fWAhWDIIAKHYwPDjYQjhwIBQ&url=https%3A%2F%2Fblogs.unity3d.com%2F2017%2F08%2F18%2Fverifying-the-scripting-docs-fun-with-editor-tests%2F&psig=AFQjCNEYXerHmTfntpm2ZeRxVwAlt4jZVw&ust=1505035119897212>



Hibák rangsorolása

Hibák súlyosság szerinti csoportosítása , példa (Jorgensen, 2002)

1.	Enyhe	Elírt szó
2.	Mérsékelt	Félrevezető vagy redundáns információ
3.	Kellemetlen	Csonka nevek, 0,00\$-os számla
4.	Zavaró	Egyes tranzakciók nem hajtódnak végre
5.	Súlyos	Elvész egy tranzakció
6.	Nagyon súlyos	Hibás tranzakció végrehajtás
7.	Extrém	Gyakori „nagyon súlyos” hibák
8.	Elfogadhatatlan	Rossz adatbázis
9.	Katasztrofális	Rendszer leállás
10.	Fertőző	Leállás, amely máshová is áttérjed





Teszt-kiértékelési jelentés

- Tesztfolyamat végén készített dokumentáció, amely összegzi a összes teszttevékenységet és eredményt. Ezen kívül tartalmazza a tesztfolyamat kiértékelését és a teszt során szerzett tapasztalatokat. (*test evaluation report*)

Teszt-kiértékelési jelentés példa

Test Report

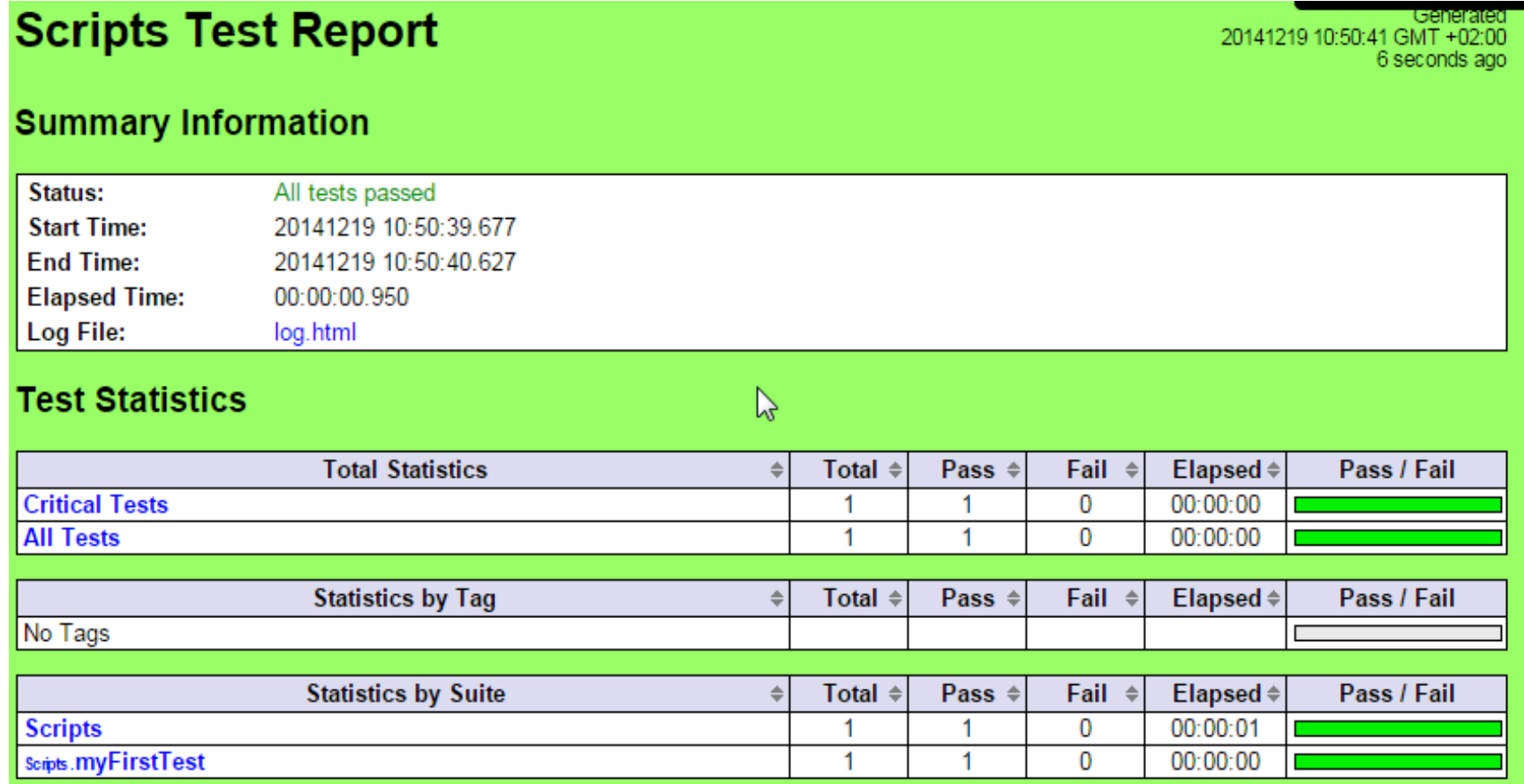
Test Cycle

System Test

EXECUTED	PASSED			130	130
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				
PENDING					0
IN PROGRESS					0
BLOCKED					0
(Sub-Total) TEST PLANNED					130
(PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)					

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	High	
Edit Customer	Check Customer can be edited	100%	100%	0	High	
New Account	Check New account is added	100%	100%	0	High	
Edit Account	Check Account is edit	100%	100%	0	High	
Delete Account	Verify Account is delete	100%	100%	0	High	
Delete customer	Verify Customer is Deleted	100%	100%	0	High	
Mini Statement	Verify Ministatement is generated	100%	100%	0	High	
Customized Statement	Check Customized Statement is generated	100%	100%	0	High	

Teszt-kiértékelési jelentés példa





Teszt lezárása

- A tesztlezárási fázisban gyűjtjük össze a tesztelés során előállított adatokat, hogy a tesztverből, számokból, tényekből és egyéb tapasztalatokból összegyűjtött adatokat konszolidáljuk. A tesztlezárási fázisban véglegesítjük és archiváljuk a tesztvert, értékeljük ki az eredményeket és készítjük elő az összefoglaló tesztjelentést. Lásd még: tesztelési folyamat. → *test closure*

Teszt lezárása

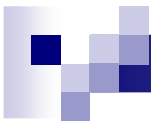
OPEN ISSUES	PRIORITY	ASSIGNED	TEST CASE ID	Reported DATE	STATUS	Updated Date
1	High	QA Team	34	09-Apr-15	In Progress	13-Apr-05
2	Low	Dev Team	98	09-Apr-15	Resolved	13-Apr-05
3	Important	BA Team	198	09-Apr-15	Unresolved	09-Apr-05
4	Critical	Mangement	NA	09-Apr-15	Closed	10-Apr-15
CLOSED ISSUES	TEST CASE ID	CLOSURE DATE				
1	87	13-Apr-05				
2	45	13-Apr-05				
3	NA	10-Apr-15				
COMING Week Priorities						
Project ID:		W10978				
Total test Cases to Cover:		109 to 450				
Scheduled date:		14-May-15				
Reason for extending:		New requirements from Business				
Upcoming Task						
Project:		W10978				
Scheduled Task:		New feature				
Date of release:		29-May-15				
BUGS/DEFECT SUMMARY						
ACTIVE DEFECTS		40				
CLOSED DEFECTS		8				
EXECUTED TEST CASE		298				
UN EXECUTED TEST CASE		498				

<http://www.softwaretestingclass.com/software-testing-weekly-status-report/>



Tesztelési dokumentáció – összefoglalás (1)

- Nincs egyetlen, mindenki által mindenütt elfogadott dokumentumhalmaz a tesztelésre vonatkozóan (sem nevükben, sem tartalmukban) , de:
- Jelenleg 5 szabványra szoktak gyakorta hivatkozni:
 - [ISO/IEC 29119-1: Concepts & Definitions \(published September 2013\)](#)
 - [ISO/IEC 29119-2: Test Processes \(published September 2013\)](#)
 - [ISO/IEC 29119-3: Test Documentation \(published September 2013\)](#)
 - [ISO/IEC 29119-4: Test Techniques \(at DIS stage, anticipating publication in late 2014\)](#)
 - [ISO/IEC 29119-5: Keyword Driven Testing \(at CD stage, anticipating publication in 2015\)](#)
- Plusz, az ISO/IEC/IEEE 29119-2- ben definiált folyamatok alapján:
 - [ISO/IEC 33063: Process Assessment Model \(at DIS stage\)](#)
- Megj: az ISO/IEC/IEEE 29119 szabványcsalád egy sor korábbi, szoftvertesztelésre, tesztelési dokumentumok elvárt tartalmára vonatkozó szabványt kivált / ki fog váltani:
 - [IEEE 829 Test Documentation](#) - az ISTQB Syllabus jelen verzióiban még szerepel
 - [IEEE 1008 Unit Testing](#)
 - [BS 7925-1 Vocabulary of Terms in Software Testing](#)
 - [BS 7925-2 Software Component Testing Standard](#))



Tesztelési dokumentáció – összefoglalás (2)

Az **ISO/IEC/IEEE 29119-3** –ban felsorolt, tesztelésre vonatkozó dokumentumok a következők:

Szervezeti teszt folyamat dokumentumok / Organizational Test Process Documentation:

- Test Policy
- Organizational Test Strategy

Teszt menedzsment dokumentumok / Test Management Process Documentation:

- Test Plan (including a Test Strategy)
- Test Status Report
- Test Completion Report

Dinamikus tesztelés dokumentumai / Dynamic Test Process Documentation:

- Test Design Specification
- Test Case Specification
- Test Procedure Specification
- Test Data Requirements
- Test Data Readiness Report
- Test Environment Requirements
- Test Environment Readiness Report
- Actual Results
- Test Result
- Test Execution Log
- Test Incident Report





A tesztelő legfontosabb feladata

- A megfelelő tesztelési technikák kiválasztása / kombinálása
 - Miért is nem létezik egyetlen olyan tesztelési megközelítés, amely mindig, minden helyzetre „jó”?



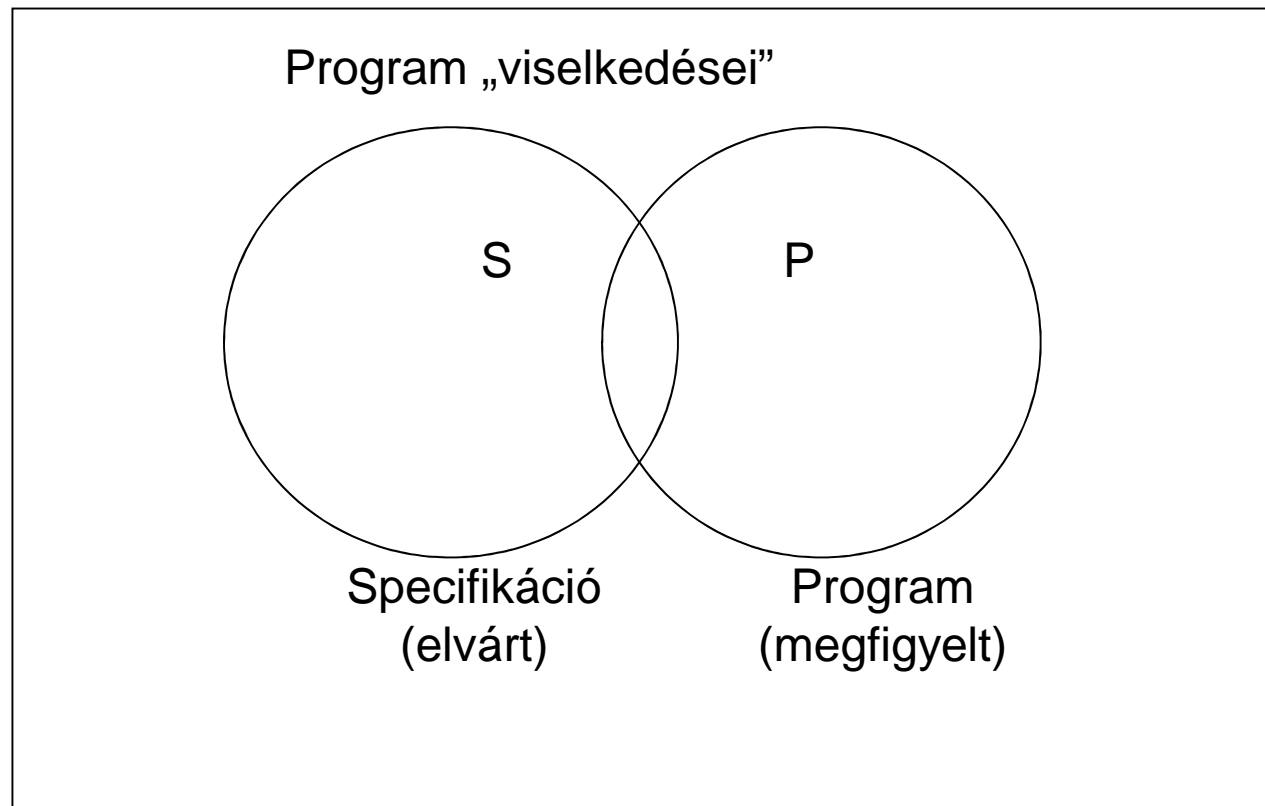


A tesztelés pszichológiája

- A tesztelés és a felülvizsgálat a szoftverfejlesztéstől eltérő gondolkodásmódot kíván.
- Jó, ha van független tesztelői csapat.
 - Ha van, általában az integrációs tesztől kezdődően.
 - A tesztelők hasznos szempontokat tudnak bevinni a fejlesztésbe, már a tervezés szakaszában. Vonjuk be őket!
- A tesztelő hibákat keres, közölnie kell, ha hibát talál. Ez konfliktusforrás lehet.
- A hibák, programhibák konstruktív szemléletű közlésével elkerülhetők a tesztelők és elemzők, tervezők és fejlesztők közötti ellentétek.
- Tárgyilagos megfogalmazás fontos. Állapítsuk meg és dokumentáljuk a hibát. Ne minősítsük a fejlesztőket!

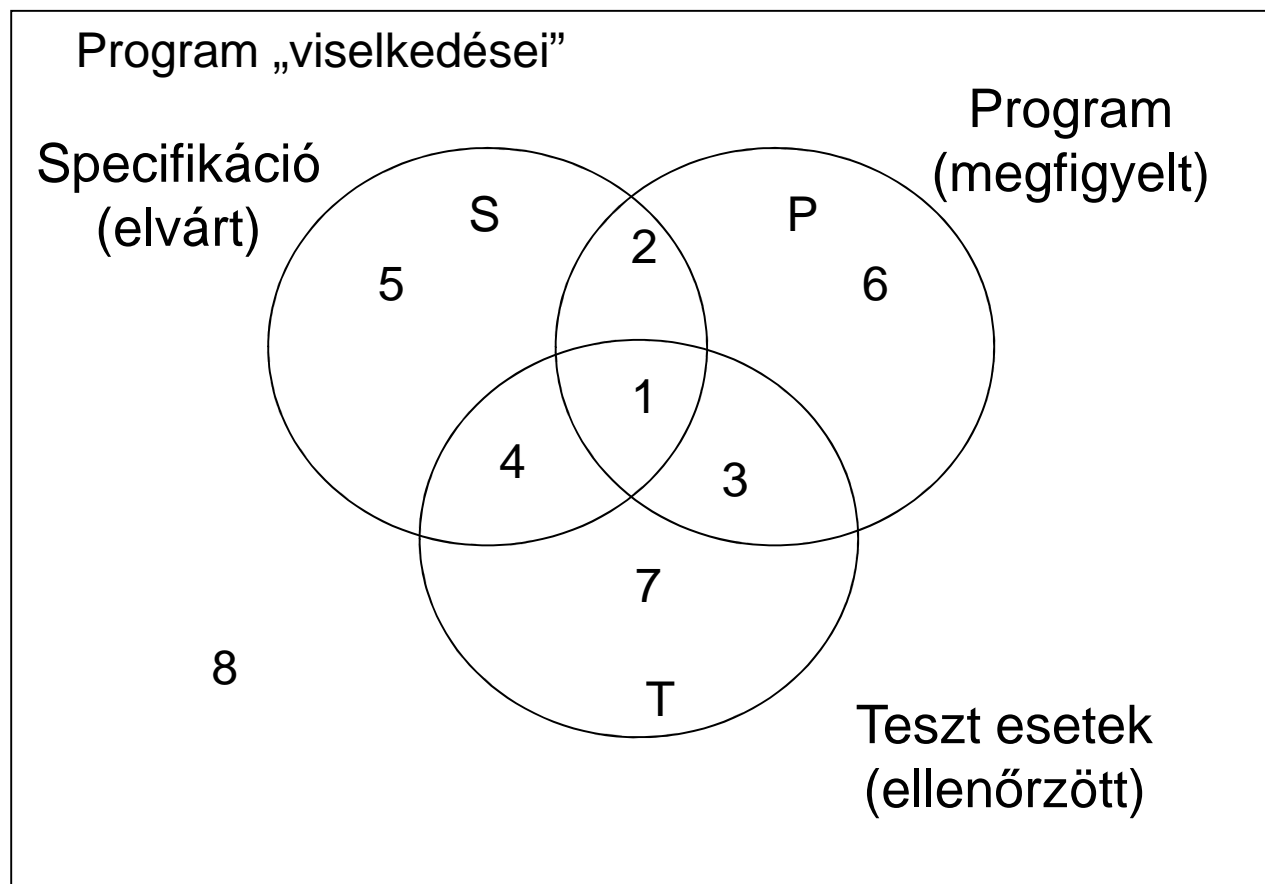


Egy Venn diagram olvasatai (1)



Tesztelés lehetséges definíciója:
annak meghatározása / eldöntése, hogy a program-viselkedés milyen aránya egyszerre specifikált és implementált.

Egy Venn diagram olvasatai (2)



Kérdés: mit jelent a „7” rész? Mire utalhat?



Tesztek típusai

- Sokféle csoportosítás
 - átfedések, ismételések
- Pl:
 - Tesztelt elem szerint
 - Az életciklusban elfoglalt hely szerint
 - Tesztelő személye szerint
 - A teszt eredményének felhasználása szerint
 - Alkalmazott technikák szerint...



A tesztek típusai


☐ Az tesztelt elem szerint

- ☐ Komponens (egység/ unit) -teszt
- ☐ Integrációs teszt
- ☐ Rendszerteszt



Teszt típusok a tesztelt elem szerint

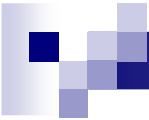
- Egységteszt / komponens teszt / unit test
 - Komponens: a legkisebb önállóan tesztelhető szoftver egység
 - Az egységeket / komponenseket teszteljük a specifikációval szemben. Egy egység: egy v. több eljárás, függvény
 - A forráskód egy adott egységének helyességét vizsgáló teszt-módszer
 - Alapötlet: készüljön teszt eset a modul minden nem triviális függvényére, funkciójára vagy metódusára, úgy, hogy minden teszt eset önálló, a többitől különválasztható legyen



Teszt típusok a tesztelt elem szerint

■ Integrációs teszt


- Az egységeket integráljuk, és az integrált részek interfészeit teszteljük a specifikációval szemben.
- Általában az egységtesztet követi és a rendszertesztet előzi meg
- Az egységteszten sikeresen „átesett” modulokat veszi alapul
- Célja, hogy a nagyobb elemekkel szemben támasztott, funkcionalitásra, teljesítményre, megbízhatóságra vonatkozó követelményeket ellenőrizze
- A modulok közötti interfészek kerülnek kipróbálásra
- A modulok / alrendszerek közötti együttműködést szimulálva történik a tesztelés
- Alapötlet: mindig tesztelt, ellenőrzött alapelemekből építkezünk



Teszt típusok a tesztelt elem szerint

- Integrációs teszt

- **Nagy bumm teszt:** az integrációs tesztelés egyik fajtája, ahol a szoftver és a hardver elemeket akár egyetlen rendszerbe integrálva teszteljük (*big-bang testing*) (HTB)



Teszt típusok a tesztelt elem szerint

■ Rendszerteszt

- A teljes rendszert teszteljük a specifikációval szemben
- Input a teszthez: integrált, sikeresen tesztelt (al)rendszer(ek)
- A rendszertesztet a felhasználónak is értenie kell – gyakran szorosan kapcsolódik az átvételi teszthez



Teszt típusok a tesztelt elem szerint

- Teszt típusok egymásra épülése
 - Teszteljük az egységeket / komponenseket
 - Integráljuk a komponenseket alrendszerekbe, teszteljük ezeket
 - Folytassuk az integrációt, mígnem a teljes rendszert tesztelni tudjuk

Teszt típusok az élelciklusban elfoglalt hely szerint



- Kifejezik, hogy a készülő szoftver mennyire „van készen”, mennyire haladtunk előre a fejlesztési élelciklusban, mennyi van még hátra a fejlesztésből
- Alapesetek (formálisan meghatározhatók):
 - ☐ Félkész állapot (új elemek hozzáadása, de nem teljes) (pre-alfa állapot)
 - ☐ Új elemek hozzáadása (alfa állapot)
 - ☐ Aktív hibakeresés (beta állapot)
 - ☐ A lényeges hibákat javították (stabil állapot)
 - ☐ Átadás előtti állapot (release-candidate)(Microsoft-nál: „RTM” – release to manufacturing)
 - ☐ Átadott / átadandó állapot (release / gold)
 - ☐ ...
- Köztes állapotokat is azonosíthatunk ...

Teszt típusok az élelciklusban elfoglalt hely szerint



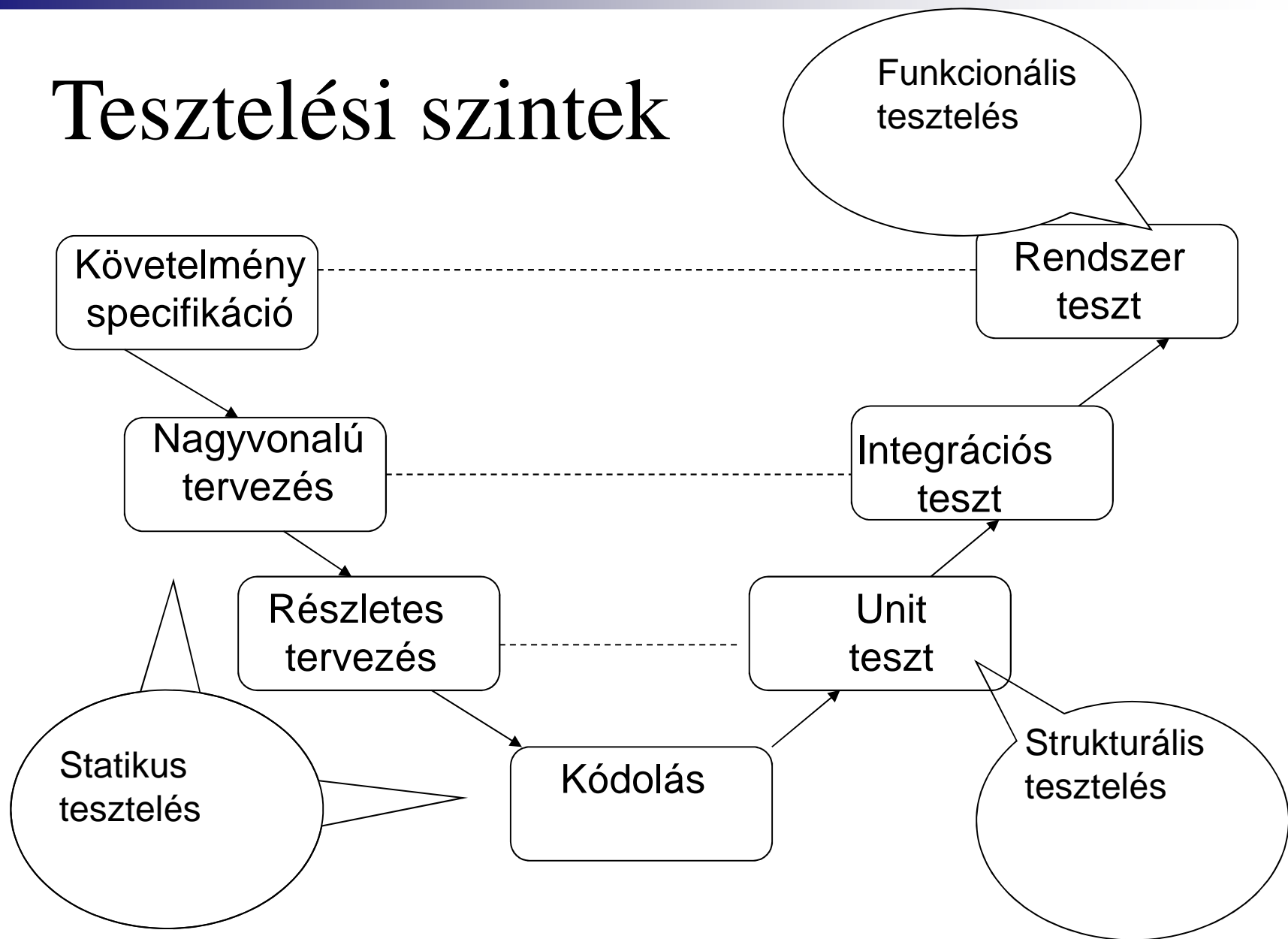
- Pre-alfa teszt: egyes funkciók (fejlesztői) tesztje
- Alfa teszt: a fejlesztő belső rendszertesztje
 - Szimulált, vagy tényleges tesztelés, amelyet potenciális felhasználók, vagy egy független tesztcsoport végez a fejlesztés helyszínén, de a fejlesztői szervezettől függetlenül. Gyakran használják dobozos szoftverek belső átvételi tesztjeihez (HTB)
- **Béta teszt:** korlátozott számú külső felhasználó teszteli a rendszert
 - a szoftvernek egy szűkebb felhasználói körben való külső tesztelése a végső kiadás előtt annak érdekében, hogy meghatározzuk, hogy a szoftver megfelel-e a felhasználók piaci igényeinek. Gyakran megfelel a dobozos (OTS) szoftverek külső átvételi tesztjének (HTB)
- (Gamma teszt, gamma verzió: vicces, gúnyos elnevezése egy hibákkal teli rendszernek)
- Gamma, delta, omega tesztek: egyre fejlettebb release-ek tesztje

Tesztelési szintek / tesztszintek

- A tesztelési szintek a szoftverfejlesztési életciklus szerinti csoportosításai a teszteknek. Az ISTQB a következő 4 szintet említi:
 - ☐ Komponens (Unit) teszt
 - ☐ Integrációs teszt
 - ☐ Rendszer teszt
 - ☐ Átvételi teszt
- 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 megtalálendő hibák és meghibásodások,
 - ☐ a tesztátmozgató szoftverkörnyezet követelményei és az eszköztámogatás,
 - ☐ a jellemző megközelítések és felelősségi körök.



Tesztelési szintek





Teszt típusok tesztelésben részt vevők szerint

- Tervezői teszt
- Fejlesztői teszt
- Felhasználói teszt
- Közös, több érdekelt fél által végzett teszt
 - Közös szemlék



Teszt típusok a teszt eredményének felhasználása szerint

- A tesztelés eredményeképpen valamilyen (hivatalos) döntés születik / lépés történik
- A tesztelést azért végzik, hogy a rendszert
 - ☐ Minősítsék – minősítési teszt
 - ☐ Átadják / átvegyék – átvételi teszt
 - ☐ Üzembe helyezzék – üzemi teszt
 - ☐ Installálják – installációs teszt
 - ☐ Együttesen átvizsgálják – együttes átvizsgálás
 - ☐ Felülvizsgálják - felülvizsgálat
 - ☐ ...



Teszt típusok a teszt eredményének felhasználása szerint

- Átvételi teszt: a felhasználó, vagy a megrendelő által a végterméken végzett feketedoboz teszt, amely azt hivatott eldönteni, hogy megfelel-e a termék a megfogalmazott (üzleti) elvárásoknak, illetve folyamatoknak. → acceptance testing



Teszt típusok a teszt eredményének felhasználása szerint

■ Üzemi tesztelés

- A szoftvertermék minden kiadásakor az üzemeltetőnek üzemi (operational) tesztet kell végrehajtania, és, ha az előírt kritériumok teljesülnek, a szoftverterméket üzemi használatra ki kell adni
- Az üzemeltetőnek biztosítani kell, hogy a szoftver kód és az adatbázis a tervben leírtak szerint induljon el, működjön és álljon le



Teszt típusok a teszt eredményének felhasználása szerint

■ Installációs tesztelés

- A fejlesztői környezeten kívül végzett tesztelés
- Egyes esetekben (pl. ha a rendszert egy már „élő” célkörnyezetben fogják használni) adatbázis-struktúra módosításra is szükség lehet – ilyenkor hasznos visszaállítási eljárásról is gondoskodni



Teszt típusok egyéb kritériumok szerint

- Az előbbieken kívül / azokat kombinálva is nagyon sokféle csoportosítás
 - Életciklusban elfoglalt hely + tesztelésben részt vevők + teszt eredmény szerint
 - Algoritmus szerint
 - ...



Teszt típusok egyéb kritériumok szerint

- „Kézi” tesztelés
- Automatizált tesztelés
 - Felhasználó által készített szkriptek alapján történő tesztelés
 - Tesztelési eszközt használó tesztelés



Teszt típusok egyéb kritériumok szerint

■ Terheléses teszt (Load testing)

- Többféle értelemben használják. Általában azt jelenti, hogy a szoftver várható alkalmazását úgy modellezzük, hogy az egy időben sok felhasználó jelenlétét szimulálja
- Az ilyen tesztelés a többfelhasználós rendszerekben hasznos
- Ha a terhelés a normálisan elvárt határ fölé emelkedik, akkor stressz tesztről beszélünk
 - Szokták még teljesítmény- teszt –értelemben is emlegetni



Teszt típusok egyéb kritériumok szerint

■ Teljesítmény teszt (Performance test)

- Annak megállapítására, hogy a rendszer bizonyos aspektusai milyen gyorsan „nyilvánulnak meg” egy bizonyos terheléskor
- Céljai lehetnek:
 - Kimutatni, hogy a rendszer megfelel a teljesítményre vonatkozó elvárásoknak
 - Két rendszer összehasonlítását teszi lehetővé („melyik a jobb teljesítményű”)
 - Megmérhetővé teszi, hogy a rendszer melyik része(i) okozza a gyenge teljesítményt



Teszt típusok egyéb kritériumok szerint

- Stressz teszt (Stress testing)
 - Egy rendszer vagy entitás stabilitásának megvizsgálásra használják
 - Normál terhelés fölötti terhelés („szakítópont”) hatásának vizsgálatára használják
 - A stressz teszt a terheléssel teszt egyik formája



Teszt típusok az alkalmazott technika szerint

- Különböző szinteken különböző technikákat használunk.
- Bármely szint esetében igaz, hogy az levárt eredményeket **többféle technika ötvözésével** lehet elérni.



Teszt típusok az alkalmazott technika szerint

- Statikus tesztelés
- Dinamikus tesztelés
 - Strukturális tesztelés
 - Funkcionális tesztelés
- Ezekről a következő előadásban lesz szó.



Miről volt szó...

- Miért szükséges tesztelni?
- A tesztelés definíciója; hibák, programhibák, bukások
- A tesztelés és minőségbiztosítás közötti különbség
- A tesztelés 7 alapelve
- Az alapvető tesztelési folyamat
- A tesztelés alapidokumentumai
- A tesztelés pszichológiája
- A tesztek csoportosítása