A szoftvertesztelés típusai

Verifikációs tesztelés (hibatesztelés)

- Tesztek a rendszerhibák feltárására.
- A jó teszt feltárja a rendszerben lévő hibák jelenlétét: program és specifikáció közötti ellentmondás.

Validációs tesztelés

- Célja annak bizonyítása, hogy a szoftver megfelel a megrendelő igényeinek.
- A jó teszt megmutatja, hogy a rendszer teljesítménye és megbízhatósága valós körülmények között is megfelelő-e.

Hibatesztelés vs. hibakeresés

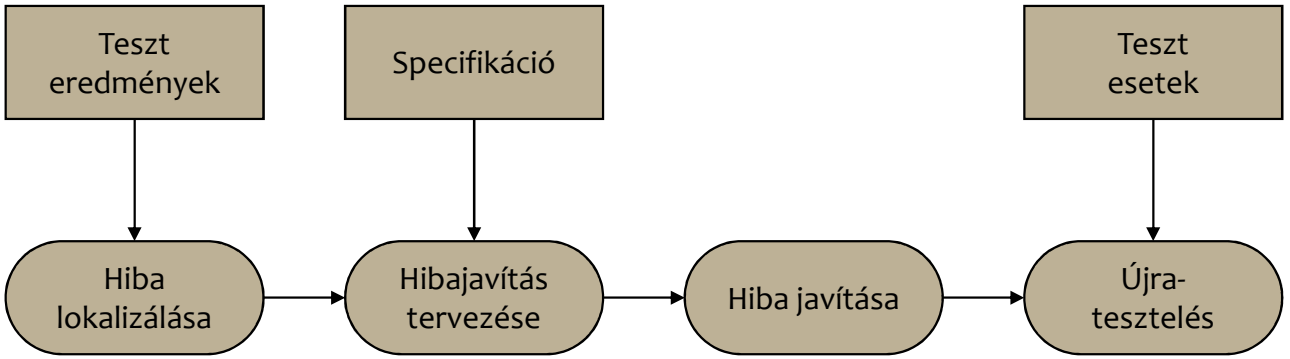
Hibatesztelés

- programhibák jelenlétének feltárása

Hibakeresés

- hibák lokalizálása és javítása

A hibakeresés folyamata



A hibakeresés során a program viselkedéséről hipotéziseket állítunk fel

- túlcímzések, nullával osztás, végtelen ciklusok, stb.

Komponens tesztelés

Komponens tesztelés

A komponens tesztelés az egyes komponensek izolált tesztelésének folyamata.

Jellemzően white-box tesztelés

Komponensek lehetnek

- egyedi függvények vagy objektumok metódusai
- objektum osztályok sok attribútummal és metódussal
- kompozit komponensek, amelyek szolgáltatásait interfészeken keresztül lehet elérni

Komponens tesztelés típusai

unit teszt

- metódusokat teszteli
- minden metódusra ismerjük a bemeneti paraméterekre adandó választ
- a tesztelés megvizsgálja, hogy a kapott érték megegyezik-e az elvárttal

modul test

- Nem a modul egy-egy funkcióját teszteli
- Hanem egy tulajdonságát
  - sebesség
  - szűk keresztmetszet
  - memóriaszivárgás
  - stb.

Interfész tesztelés

Interfész tesztelés

A cél az interfészek hibáinak, vagy az interfészekről alkotott hibás feltételezésekből eredő hibák felderítése.

Nagyon fontos objektum-orientált fejlesztés esetén, amikor is az objektumokat interfészeikkel definiáljuk.

Interfészek típusai

Paraméter interfészek

- Adat átadása egyik eljárásból a másikba.

Osztott memória interfészek

- Egy közös memóriarészt használ több eljárás vagy függvény.

Procedurális interfészek

- Egy alrendszer eljárásokat tartalmaz, amelyeket más alrendszerek hívhatnak.

Üzenettovábbításos interfészek

- Komponensek más komponensektől üzeneteken keresztül szolgáltatást kérnek.

Interfész hibák

Hibás interfész használat

- A hívó komponens egy másik komponenst akar használni, de rosszul használja annak interfészét (pl. rossz paraméter-sorrend).

Interfész félreértelmezés

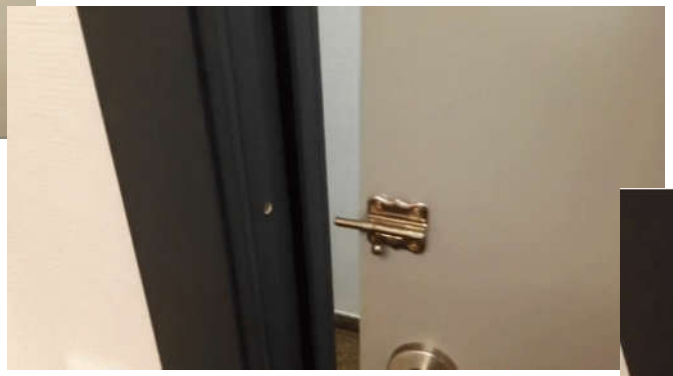
- A hívó komponens a hívott komponens viselkedéséről téves feltételezésekkel él.

Időzítési hibák

- A hívó és hívott komponensek más sebességgel működnek és így előfordulhat elavult adatok használata.

Integrációs tesztelés

Miért kell?



Integrációs tesztelés

A komponensek interakciójából eredő problémákkal foglalkozik.

A modulok összeillesztése során keletkező hibákat keresi.

- modulok eltérő forrásból

A hibalokalizálás megkönnyítése érdekében a rendszereket inkrementálisan célszerű integrálni

- no big-bang

Rendszertesztelés

Rendszertesztelés

A már komplett rendszert vizsgálja.

Elvárás, hogy a rendszer megfeleljen a specifikációnak és a rendszertervnek.

Jellemzően black-box teszt.

Nem a fejlesztő végzi.

Szcenárió-alapú rendszer tesztelés

Egy diák Amerikai történelmet tanul és éppen dolgozatot ír a polgárháborúról. Ehhez forrásokat keres különféle könyvtárakban. A LIBSYS rendszerbe bejelentkezve a kereső szolgáltatást használja eredeti dokumentumok keresésére. Talál is néhány forrást amerikai egyetemek könyvtáraiban és le is tölt onnan néhány másolatot. Az egyik dokumentumhoz azonban igazolásra van szüksége az egyetemétől, hogy valóban hallgató és a letöltés nem szolgál kereskedelmi célokat. Az igazolás kiállítását a LIBSYS rendszeren keresztül kéri. Ha az igazolást megkapja, akkor a dokumentumot letöltik a könyvtár szerverére és kinyomtatják. A diák egy e-mail üzenetet fog kapni, amelyben értesítik, hogy átveheti a dokumentumot.

Rendszertesztek

A bejelentkezési mechanizmus tesztelése helyes és helytelen azonosítókkal: annak ellenőrzése, hogy az érvényes azonosítókat elfogadja, az érvényteleneket visszautasítja.

A keresés tesztelése különféle kereső kifejezésekkel ismert forrásokra: ellenőrizhető, hogy a kereső valóban megtalálja-e a dokumentumokat.

A kijelzés tesztelése: a dokumentumokról szóló információ helyesen van-e kijelezve?

A letöltéshez engedélyt kérő mechanizmus tesztelése.

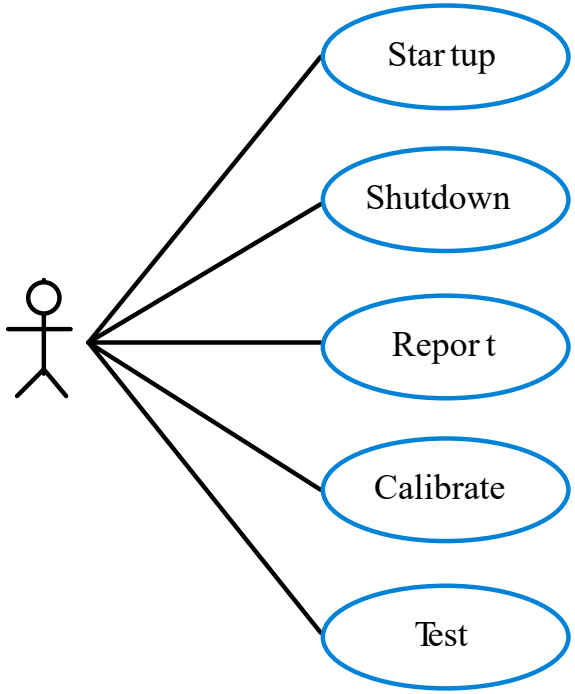
Az e-mail-es értesítő rendszer tesztelése: elküldi-e a levelet a dokumentum megérkezésekor.

Használati eset alapú rendszerteszt

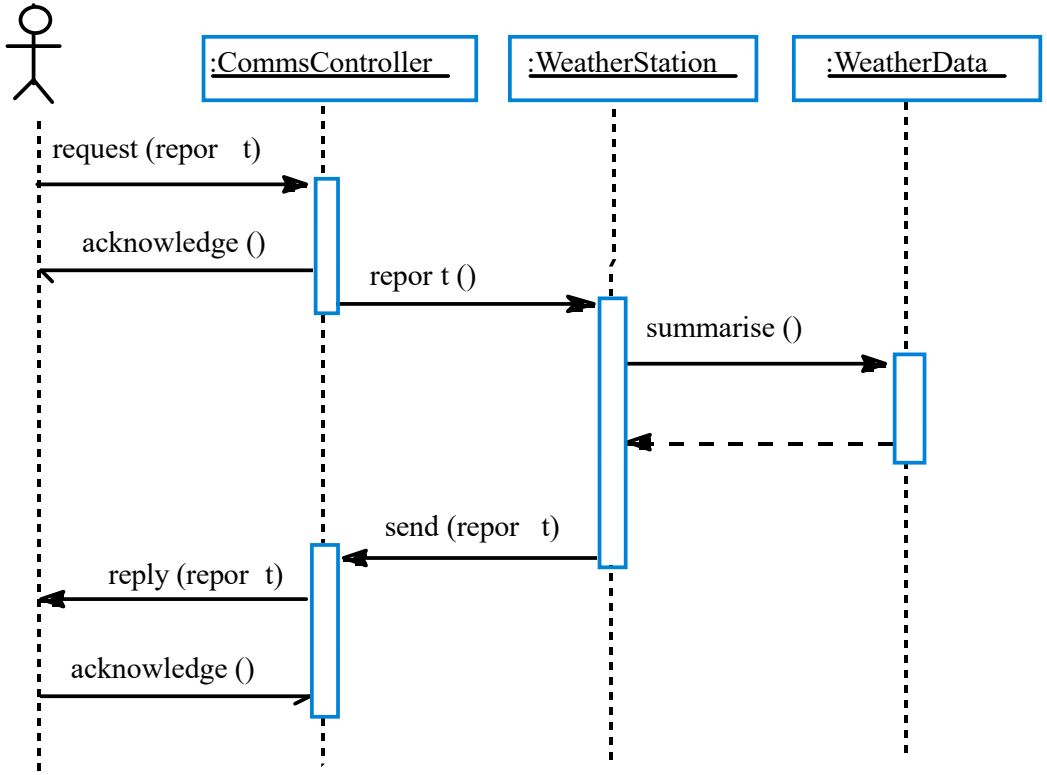
Használati esetek (use cases) alapján tesztek készíthetők. Segítenek a tesztelendő operációk kiválasztásánál és a szükséges teszt esetek megtervezésében.

A kapcsolódó szekvencia-diagramból a teszt számára a bemenetek és kimenetek meghatározhatók.

Egy meteorológiai állomás használati eset diagramja



A riport küldés szekvencia diagramja



Átvételi tesztelés

Átvételi tesztelés

Az egész rendszer átadáskor végzett tesztelésének folyamata

A végfelhasználóval közösen végzik

Szintjei

- alfa
- béta (zárt vagy nyílt)
- felhasználói átvétel
- üzemeltetői átvétel

Átvételi teszt – alfa és béta

Alfateszt

- cégen belül történik, de már nem a fejlesztő csapat által
- funkcionális tesztek mellett stabilitás, sebesség, megbízhatóság, használhatóság vizsgálata

Bétateszt

- publikus, de még nem éles használatra kiadott verzió
- több, egymást követő bétateszt is lehetséges, felhasználók köre folyamatosan bővül
- lehet hogy először csak más fejlesztők vagy csoportok kapják meg a béta verziókat

Átvételi teszt – felhasználói

A teljes célközönség használatba veszi.

A szoftver már teljes funkcionalitással rendelkezik.

A rendszer éles környezetben is használható.

Általában ekkor teszteli a legtöbb felhasználó, a legszélesebb hardver- és szoftverkörnyezetben.

A leggondosabban kiadott verziók esetében is előfordul, hogy új hibákat tárnak fel a felhasználók.

Átvételi teszt – üzemeltetői

A szoftvereknek csak egy olyan csoportját érinti, ahol a használó és az üzemeltető külön személy

A rendszergazdák ellenőrzik

- a különböző biztonsági funkciókat
- mentés és helyreállítás lehetőségét
- üzembiztonságot

A teljesítmény tesztelése

Az átvételi teszt egy része a rendszer eredő tulajdonságainak tesztelése, pl. teljesítmény, megbízhatóság.

A teljesítmény tesztelése általában egy teszt-sorozattal történik, ahol a terhelést fokozatosan növeljük, amíg a rendszer teljesítménye már elfogadhatatlanná válik.

Stressz tesztelés

A rendszert a tervezett értéknél jobban terheljük. A rendszer stresszelése gyakran fed fel hibákat.

A stresszelés a hibás működés közbeni viselkedését is teszteli. A rendszernek nem szabad katasztrofálisan összeomlania. Teszteli az elfogadhatatlan szolgáltatás-kiesést vagy adatvesztést.

A stressz teszt különösen fontos elosztott rendszereknél, ahol a rendszer súlyosan degradálódhat, ha a hálózat túlterhelődik.

Teszttervezés

Teszttervezés

A tesztelés során használt teszt esetek (bemenetek és kimentek) tervezésével foglalkozik.

A teszt esetek tervezésének célja hatékony tesztek készítése validációs és hibatesztelés céljára.

Tervezési módszerek (pl.)

- követelmény-alapú tesztelés
- partíciós tesztelés
- strukturális tesztelés

Teszterv

A teszterv leírja, hogy mit és hogyan kell tesztelni. Fontos szempont, hogy egy-egy tesztet mikor tekintünk sikeresnek. Jellemzően a rendszertervben, a minőségbiztosítás fejezetben található.

A következő fogalmak alapvetően szükségesek a teszterv elkészítéséhez

- a teszt tárgya – az adott teszt mely egységet érinti
- tesztbázis – a teszt tárgyára vonatkozó dokumentumok valamint követelmények
- tesztadat – a teszt során használt bemeneti, illetve kimeneti adatok
- kilépési feltétel – megadja, hogy egy teszt mikor tekinthető sikeresnek és/vagy lezárhatónak

Teszteset

Cél

- egy meghatározott vezérlési út végrehajtatása
- egy meghatározott követelmény teljesülésének ellenőrzése

Összetevői

- végrehajtási előfeltételek (preconditions)
- input értékek halmaza
- tesztelés lépései
- elvárt eredmény
- végrehajtási végfeltételek (postconditions)

Teszteset (példa)

Test Case: Login to application using correct username password			
Description: This test will ensure the working of logon procedure			
Data Requirements: Valid username and password & Connectivity to database is essential for this process			
Step #	**Step Description**	**Expected Results**	**Transaction Name**
1	Launch application by calling its url or address	Main application log on window should open	Launch application
2	Enter valid username and password	Password letters should be encrypted and not shown on screen	Enter logon information
3	Press logon button or press "ENTER" button on keyboard	User should log on and user main welcome window should open	Log on to application

Tesztnapló

A tesztelés során naplót kell vezetni

- milyen tesztlépéseket hajtottunk végre
- milyen eredményeket kaptunk
- a tesztelési folyamat megismételhetőségéhez kell
- hibás teszt esetén a hibajelentéshez kell

A tesztnapló alapján eldönthető kell legyen, hogy a teszt sikeres volt-e

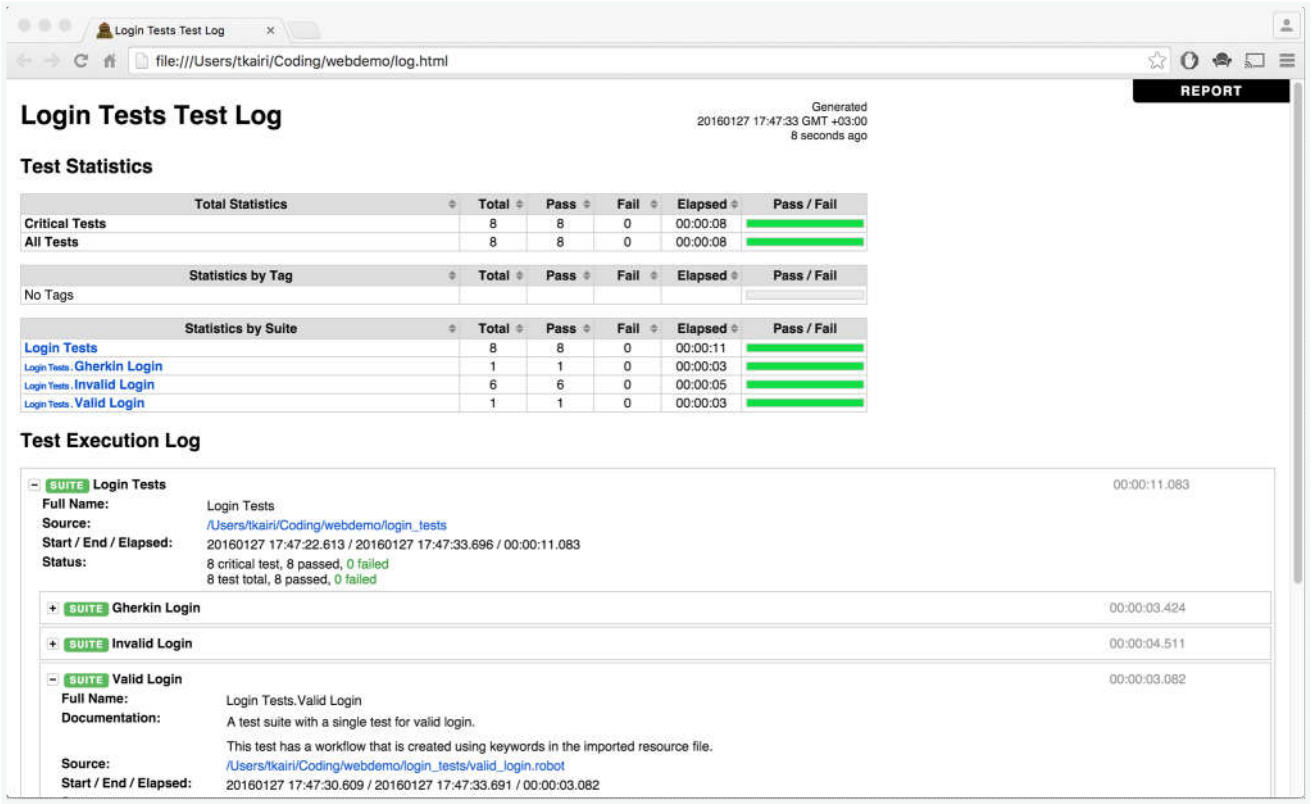
Tesztnapló (template)

Sample Template for Test Case Log

Test Name:	Describe the Name of the Test
Test Case Author:	Describe the Test Case Author Name
Tester Name:	Describe the Tester Name
Project ID / Name:	Describe the Name of the Project
Test Cycle ID:	Describe the Test Cycle ID
Date Tested:	Describe the Date on which Test Case Was Completed

Test Case ID	Test Objective ID	Category	Condition	Expected Results	Actual Results	Requirement ID
Describe the Test Case ID	Describe the ID from the Test Plan	Describe the Test Category e.g. (Edit, Numeric, Presentation etc.)	Describe the Specific Test Condition	Describe the Specific Results Expected on Executing the Condition	Record "Pass" or "Fail"	Describe the ID which Traces Back to the Specific Requirement

Tesztnapló (példa)



Tesztjelentés

A tesztelési ciklus végén, a tesztnapló alapján készül a tesztjelentés. Ez tartalmazza az adott ciklus eredményeit, a hibák javításához szükséges alapvető információkat: hol, mikor, milyen hiba volt, milyen bemeneti és kimeneti paraméterekkel.

Amennyiben a tesztelő és a fejlesztő személye különböző, fontos szempont, hogy ne ellenségként tekintsenek egymásra, hanem a közös célt, a magas minőségű, hatékony fejlesztést helyezzék előtérbe.

Tesztjelentés (példa)

Test Report

Test Cycle System Test

EXECUTED	PASSED			130	
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				**130**
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*	

Összefoglalás

A tesztelés felfedheti hibák jelenlétét a rendszerben, de nem tudja bizonyítani, hogy nem maradt több hiba.

A komponensek fejlesztői felelősek a komponens tesztelésért, a rendszertesztelés egy független csoport feladata.

Az integrációs tesztelés a rendszer növekményeinek tesztje, az átadási teszt pedig a megrendelőnek átadni kívánt rendszer tesztelésével foglalkozik.

A hibatesztelés tervezéséhez mind a tapasztalat, mind ökölszabályok használhatók.

Az interfész tesztelés feladata a kompozit komponensek interfészeiben levő hibák feltárása.