MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

Szoftvertechnológia gyakorlata

GEIAL316-B2

Szoftvertesztelés alapjai

Dr. Tompa Tamás
egyetemi adjunktus
Általános Informatikai Intézeti Tanszék

Miskolc, 2024

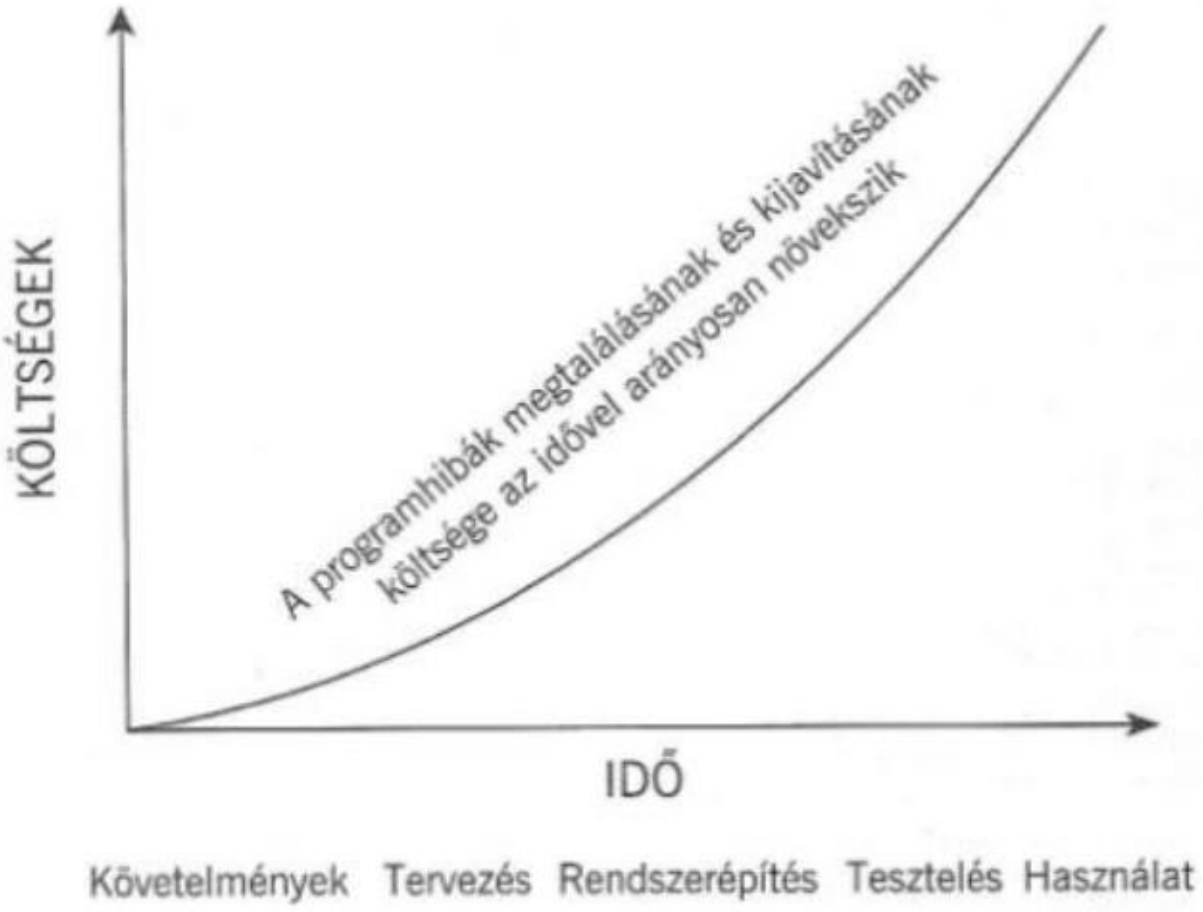
Szoftvertesztelés - Bevezetés

- **Mi a tesztelés, miért van rá szükség?**
  - a szoftver termékben meglévő **hibákat** még az üzembe helyezés előtt **megtaláljuk**, ezzel **növeljük a** termék **minőségét**, megbízhatóságát
  - Lesz-e hiba a szoftverben?
    - tuti :) -> emberek fejlesztik
    - forgalmi vizsga analógia
  - programok hibái
    - meghibásodás: a rendszer működése eltér az elvárt eredménytől (*failure*)
    - emberi hiba
      - helytelen használat miatt, helytelen tevékenység (*error, mistake*)
      - szoftverhiba hiba, belső hiba, nem az elvárt viselkedés (*bug, defect, fault*)

Szoftvertesztelés - Bevezetés

- Folyamat, nem egyszeri tevékenység
  - tesztelő végzi: szakma képviselője, aki azért dolgozik, hogy az ügyfél megfelelően működő szoftvert használjon, és azzal elégedett legyen
- Az összes szoftverfejlesztési életciklushoz kapcsolódik
  - bármely szinten előfordulhat programhiba
  - minél hamarabb találják meg annál olcsóbb javítani
- *A szoftvertesztelés a forráskód tesztelését jelenti?*

Szoftvertesztelés - Bevezetés



Forrás: Dorothy Graham, Erik van Veenendaal, Isabel Evans és Rex Black - A szoftvertesztelés alapjai

Szoftver minőségbiztosítás

- Adott a szoftver specifikációja
  - követelmények halamaza
- Cél
  - úgy implementálni a szoftvert, hogy az megfeleljen a specifikációnak
  - illetve a vele szemben támasztott követelményeknek
- Honnan tudjuk, hogy teljesíti a specifikációt?
  - Tesztelés
    - követelmények teljesülésének ellenőrzése
    - tesztelni nem csak forráskódot lehet

Definíció

- Az összes szoftverfejlesztési folyamathoz kapcsolódó, akár statikus, akár dinamikus folyamat, amely kapcsolatban áll a szoftvertermékek tervezésével, elkészítésével és kiértékelésével, hogy megállapítsa, a szoftvertermék teljesíti-e a meghatározott követelményeket és megfelel-e a célnak.

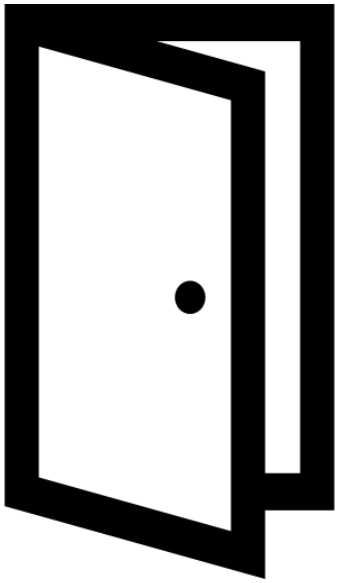
Dijkstra: „A tesztelés a hibák jelenlétét, és nem a hibamentességet tudja kimutatni.”

Definíció értelmezése

- Statikus, dinamikus folyamat: programhibák keresése úgy, hogy a futó teszt eredményeinek bemutatás céljából futtatjuk a szoftverkódot (dinamikus) és lehet úgy is, hogy nem (statikus).
  - a statikus magába foglalja a dokumentumok felülvizsgálatát beleértve a forráskódot is
- Tervezés: tesztelést menedzselni kell, meg kell tervezni, mit és hogyan szeretnénk
- Követelmény: olyan feltétel vagy képesség, amely a felhasználó számára egy probléma megoldásához vagy egy adott cél eléréséhez szükséges
- Nem csak kódot lehet tesztelni
  - követelményeket, specifikációkat, dokumentumokat

Ajtó példa

- A megrendelő a cégtől egy ajtót szeretne
- A szoftver, mint ajtó
  - a csapat minden tagja az ajtón dolgozik
  - milyen legyen az ajtó?
  - mekkora legyen a mérete?
  - milyen zár legyen rajta?
    - akasztós, tolós, újjlenyomat olvasós, retina szkenneres, dns elemzős
  - a csapat minden tagjának tudnia kell, hogy milyen egy ajtó?
    - nem -> menedzsment
  - amit tudnia kell
    - a kód amit ír az az ajtó része és az **működjön**
    - honnan tudni, hogy a kód működik?
      - tesztelés…
  - Pl zár fejlesztése majd kipróbálása
    - ez egy tolóajtó : D -> nem jó a zár -> hiányzó információ
    - a zár egy komponens -> önmagában működő a zár (egységteszt - ok), de a rendszer részeként? (integrációs teszt -> failure)



Tesztelési alapelvek

1. **A tesztelés a hibák jelenlétét jelzik**: felfedik azokat, de nem azt fedik fel, hogy nincs
2. **Nem lehetséges kimerítő teszt**: minden bementi kombinációt lehetetlen tesztelni… Magas prioritású részek és triviális esetek tesztelése
3. **Korai teszt**: az életciklus minél korábbi szakaszában elkezdni -> költségek csökkentése
4. **Hibák csoportosulása**: véges idő áll rendelkezése, azokra a modulokra kell koncentrálni ahol a hibák a legvalószínűbbek
5. **Féregírtó paradoxon**: újratesztelés során mindig ugyanazon tesztesetek futtatása, egy idő után nem találnak újabb hibát (alkalmazkodik a féreg a teszthez). -> Tesztszintek bővítése.
6. **A tesztelés függ a körülményektől**: atomerőmű vs. beadandó, 10 nap vs. 1 éjszaka
7. **Hibátlan rendszer téveszméje**: használhatatlan szoftvert nem érdemes tesztelni, hibák javítása <-> elégedetlen megrendelő

Tesztelési technikák

- **Feketedobozos (black-box)**
  - specifikáció alapú, a **specifikáció alapján** készülnek a tesztesetek
  - forráskód ismerete nélkül -> funkcionalitás
  - szükség van a futtatható szoftverre
  - „adott bemenet – adott kimenet” párok tesztelése

- **Fehérdobozos (white-box)**
  - strukturális teszt, a **forráskód alapján** készülnek a tesztesetek
  - ismert a forráskód -> olvasható a konkrét működés
  - lefedettség definíciója
    - a struktúra hány százalékát tudjuk tesztelni a meglévő tesztesetekkel
    - struktúra: általában a kódsor, metódus, elágazás, osztály, modul, funkció stb.
    - struktúra tesztelésére egység (unit) teszt

Tesztelési szintek

- **Fejlesztői teszt (a fejlesztő cég alkalmazottjai végzik)**
  - **Komponensteszt**
    - a rendszer egy komponensét teszteli önmagában
    - általában fehérdobozos teszt (ismert a forráskód)
    - unit-teszt (metódusokat tesztel)
    - modul teszt (nem funkcionális tulajdonságok tesztelése: memóriaszivárgás, sebesség, szűk keresztmetszet)
  - **Integrációs teszt**
    - interfészek tesztelése
    - komponens-komponens között: komponensek közötti kölcsönhatások
    - rendszer-rendszer között: rendszer és más rendszerek közötti kölcsönhatások
  - **Rendszerteszt**
    - a rendszer egészét, az összes komponenst együtt teszteli
    - megfelel-e: követelmény specifikációnak, a funkcionális specifikációnak, a rendszertervnek.

Tesztelési szintek

- **Átviteli teszt (végfelhasználók végzik)**
  - az egész rendszer teszteli, de **felhasználói szempontból**
  - alfa teszt
    - kész termék tesztje a fejlesztő cégnél, de nem a fejlesztő csapat által
    - több millió véletlen egérkattintás
  - béta teszt
    - végfelhasználók egy szűk csoportja végzi
    - Pl. játékoknál még kiadás előtt elküldik néhány fan-nak

Tesztelési szintek

- **Átviteli teszt (végfelhasználók végzik)**
  - felhasználói átvételi teszt
    - majdnem az összes felhasználó megkapja a szoftvert és az éles környezetben használatba veszi
    - installálják és használják, de még nem a termelésben
  - üzemeltetői átvételi teszt
    - rendszergazdák ellenőrzik, hogy a biztonsági funkciók, pl. a biztonsági mentés és a helyreállítás, helyesen működnek-e

Tesztelési tevékenység

- Nem csak tesztek készítéséből és futtatásából áll, magába foglalja:
  - teztterv elkészítése
    - leírja, hogy mit, milyen céllal, hogyan kell tesztelni, mikor sikeres a teszt. Általában a rendszerterv része, azon belül is a minőségbiztosítás (quality assurance, QA) fejezethez tartozik
  - tesztesetek tervezése
    - milyen tesztadattal kell meghajtani a teszt tárgyát, mi az elvárt visszatérési érték vagy viselkedés
  - felkészülés a végrehajtásra
    - teszt környezetre van szükség, a teszt környezet kialakításánál törekedni kell, hogy a lehető legjobban hasonlítson az éles környezetre

Tesztelési tevékenység

- Nem csak tesztek készítéséből és futtatásából áll, magába foglalja:
  - tesztek végrehajtása
    - teszt napló vezetése amiben szerepel, hogy milyen lépéseket hajtottunk végre és milyen eredményeket kaptunk. A tesz napló alapján a tesztnek megismételhetőnek kell lennie
  - kilépési feltételek vizsgálata
    - tesztek után meg kell vizsgálni, hogy sikeresen teljesítettük-e a kilépési feltételt. Ehhez a tesztesetben leírt elvárt eredményt hasonlítjuk össze a teszt naplóban lévő valós eredménnyel
  - eredmények értékelése
    - az eredmények alapján további tesztek készíthetése lehetséges
    - döntés ez alapján: egy komponenst nem érdemes már tovább tesztelni, de egy másikat tüzetesebben kell tesztelni

Tesztelési tevékenység

- Nem csak tesztek készítéséből és futtatásából áll, magába foglalja:
  - jelentéskészítés
    - információ gyűjtése a tesztelésről -> projektmenedzsment, szponzor, ügyfelek, kulcsszereplők
    - kódban hibák találása -> tesztelők támadása -> személyeskedés kerülése
    - hiba oka: rövid idő, nagy nyomás stb.

Teszterv dokumentum

- Fontos dokumentum, amely leírja, hogy mit, milyen céllal, hogyan kell tesztelni

- A teszt célja
  - megtalálni a hibákat,
  - növelni a megbízhatóságot,
  - megelőzni a hibákat

- Leírja a **teszt tárgyát**
- Kigyűjti a **tesztbázisból** a teszt által lefedett követelményeket
- Meghatározza a **kilépési feltételt**
- A **tesztadatokat** általában csak a teszteset határozzák meg, de gyakran a tesztesetek is részei a tesztterv nek

Teszterv dokumentum

- **A teszt tárgya**: A rendszer azon része, amelyet tesztelünk, ez lehet az egész rendszer is.

- **Tesztbázis**: Azon dokumentumok összessége, amelyek a teszt tárgyára vonatkozó követelményeket tartalmazzák.

- **Tesztadat**: Olyan adat, amivel meghívjuk a teszt tárgyát. Általában ismert, hogy milyen értéket kellene erre adnia a teszt tárgyának vagy milyen viselkedést kellene produkálnia. Ez az elvárt visszatérési érték, illetve viselkedés. A valós visszatérési értéket, illetve viselkedést hasonlítjuk össze az elvárttal.

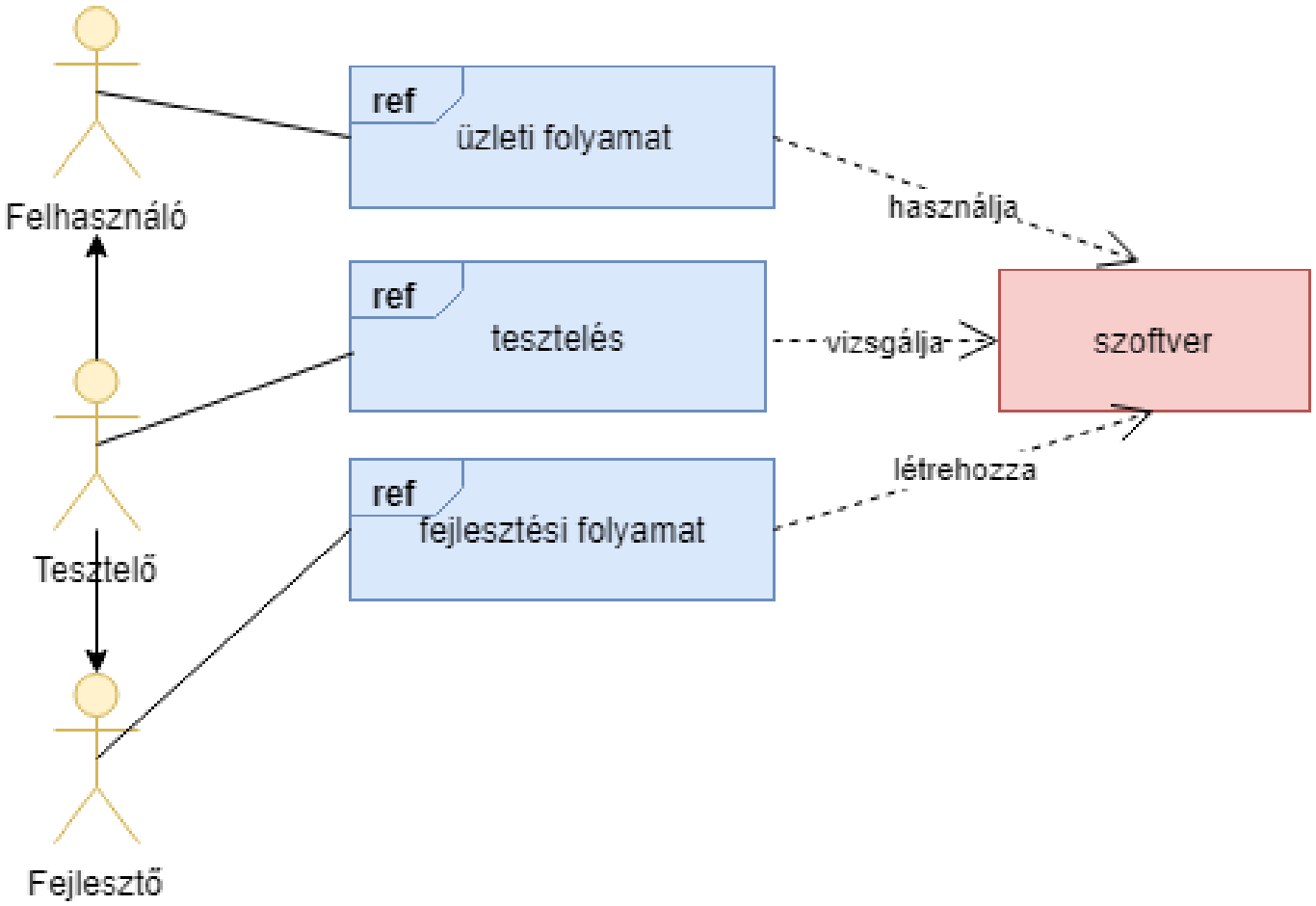
- **Kilépési feltétel**: Minden tesztnél előre meghatározzuk, mikor tekintjük ezt a tesztet lezárhatónak. Ezt nevezzük kilépési feltételnek. A kilépési feltétel általában az, hogy minden tesztest sikeresen lefut, de lehet az is, hogy a kritikus részek tesztlefedettsége 100%

A szoftver kapcsolatai az életciklus során

- **Az életciklus alatt a szoftverrel az egyes szereplők más-más kapcsolatban állnak**

- **Szereplők**
  - **Fejlesztő**
    - fejlesztési folyamat által **létrehozza** a szoftvert
  - **Tesztelő**
    - **vizsgálja** a szoftver megfelelőségét a tesztelés által
  - **Felhasználó**
    - **használja** a rendszert a különböző tevékenységein keresztül

A szoftver kapcsolatai az életciklus során



Vezérlő és rendszerszoftverek teszteléséről

- **Olyan szoftverek amelyek valamilyen hardvereszközt működtetnek közvetlenül**
- Hardverspecifikus működés: jelek előállítás/feldolgozása
  - pl. mikrovezérlők: mosógép vezérlése, hűtő, mikrosütő, TV, autó, repülő, stb.
- Szoftver hibája a hardvereszköz meghibásodásához vezethet
- **Pl. Airbus A320**
  - első olyan utasszállító repülő ami teljesen számítógépes vezérlő rendszerrel működik
  - a repülőgép kormányai nincsenek közvetlen kapcsolatban a kormányfelületekkel
  - joystick és a pedálok segítségével adnak jeleket a vezérlő szoftvernek, amely működtetni a különböző hardver eszközöket
  - rendszer szoftver folyamatosan figyeli a kormányokat és amennyiben úgy dönt, hogy a kormánykitérés "nem megfelelő", akkor felülbírálja a pilóta döntését
  - "megfelelő": hatalmas tapasztalaton alapuló tervezés
  - első prototípus: a vezérlő szoftver felülbírálta a pilótákat, így a leszállás helyett a repülőtér melletti erőbe zuhant a repülőgép. A személyzet hét tagja meghalt.

- **ezen eszközök tesztelésének témaköre túlmutat a tárgy keretein**

Köszönöm a figyelmet!