Szoftvertesztelés

Jeszenszky Péter

2024.04.07.

Felhasznált irodalom

A prezentáció az alábbi források felhasználásával készült:

- Ian Sommerville. *Software Engineering*. 10th edition. Pearson, 2015. https://software-engineering-book.com/
  - Lásd a 8., *Software Testing* című fejezetet (226–254. oldal)
  - A 2006-ban megjelent 8. kiadás érhető el magyar nyelven:
    - Ian Sommerville. Szoftverrendszerek fejlesztése. Második, bővített, átdolgozott kiadás. Panem Könyvkiadó Kft., 2007.
- *ISTQB Certified Tester Foundation Level Syllabus* https://www.istqb.org/certifications/certified-tester-foundation-level
- Pierre Bourque (ed.), Richard E. (Dick) Fairley (ed.). *Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3*. IEEE Computer Society, 2014. https://www.computer.org/education/bodies-of-knowledge/software-engineering
- Vladimir Khorikov. *Unit Testing: Principles, Practices, and Patterns*. Manning Publications, 2020. https://github.com/vkhorikov/UnitTestingPPP

ISTQB (1)

Az *International Software Testing Qualification Board* (ISTQB) egy nemzetközi non-profit szervezet, mely szoftvertesztelési tanúsítványokat kínál.

- Webhely: https://www.istqb.org/
- Tagegyesületek: https://www.istqb.org/certifications/member-board-list/
  - Magyar testület: Magyar Szoftvertesztelői Tanács Egyesület / *Hungarian Testing Board Association* https://www.hstqb.org/

ISTQB (2)

- Az *ISTQB Certified Tester* a világ legnépszerűbb és legelismertebb tanúsítványa a szoftvertesztelés területén.
- Oktatást és vizsgákat akkreditált szolgáltatók nyújtanak.

ISTQB (3)

Tantervek:

- Az oktatás és a vizsgák is tesztelési szakértők által fejlesztett tanterveken alapul.
- A tantervek nyilvánosan elérhetők letöltésre.

ISTQB (4)

Glosszárium:

- Az ISTQB karbantartja a szoftvertesztelésheez kapcsolódó szakkifejezések egy glosszáriumát, amely egy szabványos és következetes terminológiát határoz meg a területhez.
- A glosszárium több nyelven is rendelkezésre áll.
- Weboldal: https://glossary.istqb.org/

Mi a szoftvertesztelés?

- **IEEE**: A szoftvertesztelés annak dinamikus verifikálását jelenti, hogy egy program várt módon viselkedik tesztesetek egy véges halmazán, melyek alkalmas módon kerülnek kiválasztásra egy általában végtelen végrehajtási tartományból.
- **ISTQB**: A szoftvertesztelés egy megoldás a szoftver minőségének megállapításához és a szoftver működés közbeni meghibásodási kockázatának csökkentésére.
- **Sommerville**: A tesztelés célja a a használatba vétel előtt annak megmutatása, hogy egy szoftver azt csinálja, amit kell, valamint a programhibák felfedezése.

A szoftvertesztelés korlátai

- A tesztelés csökkenti a szoftverben maradó felfedezetlen hibák valószínűségét, de a tesztelés még akkor sem a helyesség bizonyítéka, ha egyetlen hiba sem kerül megtalálásra.
- Dijkstra (1972):
  "A tesztelés csak a hibák jelenlétét tudja megmutatni, a hiányukat nem."
- Lásd később a szoftvertesztelés hét alapelvét.

Verifikáció és validáció (V & V)

A szoftvertesztelés egy tágabb folyamat, a szofvter verifikáció és validáció (V & V) része.

- **Verifikáció** (***verification***): annak ellenőrzése, hogy a szoftver megfelel-e a vele szemben támasztott (funkcionális és nem funkcionális) követelményeknek.
  - *Are we building the product right?*
- **Validáció** (***validation***): annak ellenőrzése, hogy a szoftver megfelel-e az ügyefelek elvárásainak.
  - *Are we building the right product?*

V & V módszerek

- **Szoftver ellenőrzés és áttekintés** (***software inspection and review***): statikus módszerek, melyek a szoftver különféle reprezentációit (például a rendszerkövetelményeket, a forráskódot) elemzik a szoftver végrehajtása nélkül.
  - Emberek végzik manuálisan.
- **Szoftvertesztelés** (***software testing***): dinamikus, azaz a szoftver végrehajtásán alapul.
  - Általában manuális és automatikus tesztelés egy keverékét jelenti.

Validációs és hiányosság tesztelés

- **Validációs tesztelés** (***validation testing***): célja bizonyítani a fejlesztő és a vásárló számára, hogy a szoftver megfelel a vele szemben támasztott követelményeknek.
  - A várható felhasználást tükröző teszteseteket használ.
- **Hiányosság tesztelés** (***defect testing***): célja olyan bemenetek találása, melyeknél a szoftver viselkedése helytelen, nemkívánatos, vagy nem felel meg a specifikációjának.
  - Olyan mesterséges teszteseteket használ, melyeket a hiányosságok felfedésére terveztek. Ezek a tesztesetek szándékosan zavarosan lehetnek és nem szükséges, hogy a rendszer normál használatát tükrözzék.

Hibát leíró szakkifejezések (1)

A szoftverfejlesztésben számos szakkifejezést használnak hibák leírására:

- **Tévedés/tévesztés** (**error/mistake**): rossz eredményt adó emberi tevékenység.
- **Hiba** (**defect/fault/bug**): tökéletlenség vagy hiányosság egy munkatermékben, melynél nem teljesülnek a követelmények vagy előírások.
- **Meghibásodás** (**failure**): olyan esemény, melynél egy komponens vagy rendszer nem lát egy megkövetelt funkciót a megszabott határok között.

Hibát leíró szakkifejezések (2)

- Egy személy egy tévedést/tévesztést követ el, mely egy hibát vezethet be a szoftver kódjába vagy valamely más kapcsolódó munkatermékbe.
- Ha végrehajtásra kerül a hiba a kódban, akkor az egy meghibásodást okozhat, de nem szükségszerűen minden esetben.
  - Például bizonyos hibák nagyon sajátos bemenetek vagy előfeltételek mellett váltanak ki meghibásodást, melyek nagyon ritkán vagy sohasem fordulnak elő.
  - Nem minden meghibásodást a kódban lévő hibák okoznak, eredményezhetik őket környezeti feltételek is.

Hibakeresés

- A hibakeresés (*debugging*) a meghibásodások okainak keresése, elemzése és eltávolítása egy komponensben vagy rendszerben.
- A hibakeresés ki tudja javítani azokat a hibákat, melyek az okai a tesztelés révén felfedezett meghibásodásoknak.
    - Azonban nem minden váratlan teszteredmény meghibásodás (hamis pozitív esetek).

Tesztelési alapelvek

A szoftvertesztelés hét alapelve:

1. A tesztelés a hibák jelenlétét mutatja meg, nem a hiányukat
2. Lehetetlen a kimerítő tesztelés
3. A korai tesztelés időt és pénzt takarít meg
4. A hibák csoportosulnak
5. A tesztek elkopnak (óvakodj a kártevőírtó paradoxontól)
6. A tesztelés környezetfüggő
7. A hibamentesség egy tévhit

Teszteset (1)

Definíciók:

- **IEEE**: Teszt bemenetek, végrehajtási feltételek és elvárt eredmények egy halmaza, melyek egy olyan konkrét célból kerültek meghatározásra, mint például a program egy bizonyos végrehajtási útvonalon történő végrehajtása vagy egy bizonyos követelménynek való megfelelés ellenőrzése.
- **ISTQB**: Tesztfeltételek alapján meghatározott előfeltételek, bemenetek, tevékenységek (adott esetben), elvárt eredmények és utófeltételek halmaza.
  - **Tesztfeltétel**: Egy komponens vagy rendszer tesztelhető vontakozás, melyet a tesztelés alapjául választunk.

Teszteset (2)

Magas szintű teszteset: Teszteset, mely absztrakt előfeltételekkel, bemeneti adatokkal, elvárt eredményekkel, utófeltételekkel és (adott esetben) lépésekkel rendelkezik.

Alacsony szintű teszteset: Teszteset, mely konkrét előfeltételekkel, bemeneti adatokkal, elvárt eredményekkel, utófeltételekkel és (adott esetben) a lépések részletes leírásával rendelkezik.

Tesztadatok

- A tesztadatok a tesztvégrehajtáshoz szükséges adatokat jelentik.
- Az ilyen konkrét értékek a használatukra vonatkozó világos útmutatásokkal együt végrehajtható alacsony szintű tesztesetekké teszik a magas szintű teszteseteket.
- Ugyanaz a magas szintű teszteset különböző tesztadatokat használhat különböző végrehajtásoknál.

Tesztfolyamat (1)

- A szoftvertesztelés egy számos különböző tevékenységet magában foglaló folyamat.
- Nem létezik univerzális szoftvertesztelési folyamat.
- Vannak azonban megszokott tesztelési tevékenységek.
- Ezek a tesztelési tevékenységek alkotnak egy tesztfolyamatot.

Tesztfolyamat (2)

Egy tesztfolyamat az alábbi fő tevékenységcsoportokból áll:

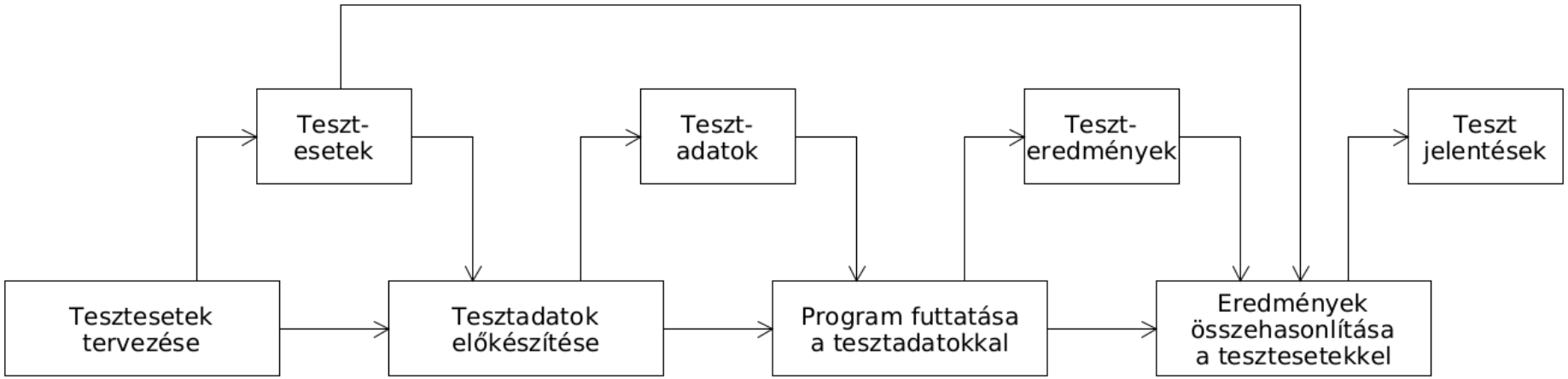
- Teszttervezés (*test planning*)
- Tesztfelügyelet és -irányítás (*test monitoring and control*)
- Tesztelemzés (*test analysis*)
- Teszttervezés (*test design*)
- Teszt megvalósítás (*test implementation*)
- Teszt végrehajtás (*test execution*)
- Tesztlezárás (*test completion*)

Szoftverfejlesztés és szoftvertesztelés

- A szoftvertesztelés mindenre kiterjedő kellene, hogy legyen a teljes fejlesztési és karbantartási életciklus alatt.
- Számos különböző szoftverfejlesztési életciklus modell van, melyek mindegyike különböző megközelítést igényel a teszteléshez.
- Minden fejlesztői tevékenységhez van egy megfelelő tesztelési tevékenység.
- A szoftverfejlesztési életciklus modell választásától függetlenül a tesztelési tevékenységeket az életciklus korai szakaszaiban ajánlott elkezdeni a korai tesztelés elvéhez tartva magunkat.

A szoftvertesztelési folyamat modellje (1)

A hagyományos szoftvertesztelési folyamat absztrakt modellje, ahogy a tervalapú fejlesztésnél használják (Sommerville):



A szoftvertesztelési folyamat modellje (2)

- A tesztadatok néha automatikusan generálhatók.
- Azonban nem lehetséges a tesztesetek automatikus generálása.
  - A várt teszteredmények meghatározásához olyan emberek bevonása szükséges, akik értik, hogy mi lenne a rendszer dolga.
- A tesztelési módszerek (*test techniques*) célja, hogy segítsék a tesztfeltételek meghatározását, a tesztesetek tervezését és a tesztadatok megadását.
- A tesztvégrehajtás automatizálható.
  - A teszteredmények automatikusan összehasonlításra kerülnek a várható eredményekkel, így tehát nincs szükség olyan személyre, aki hibákat és anomáliákat keres a tesztek futásakor.

Fejlesztői tesztelés és felhasználói tesztelés

- **Fejlesztői tesztelés** (***development testing***):
  - A rendszert fejlesztés közben tesztelik hibák és hiányosságok felfedezéséhez.
  - Minden olyan tesztelési tevékenységet tartalmaz, melyet a rendszert fejlesztő csapat végez.
  - A szoftver tesztelésést végeztheti az azt fejlesztő programozó (egységtesztelés, integrációs tesztelés) vagy független tesztelők (integrációs tesztelés, rendszertesztelés).
- **Felhasználói tesztelés** (***user testing***):
  - Felhasználók vagy potenciális felhasználók a saját környezetükben tesztelik a rendszert.
  - Például elfogadási tesztelés.

Tesztelési szintek

Szoftvertesztelés különböző szinteken végezhető a szoftverfejlesztési folyamatokban:

1. **Egységtesztelés/komponens tesztelés (*unit testing*/*component testing*)**
2. **Integrációs tesztelés (*integration testing*)**
3. **Rendszertesztelés (*system testing*)**
4. **Elfogadási tesztelés (*acceptance testing*)**

Tesztelési szintek: egységtesztelés

- A függetlenül tesztelhető komponensekre összpontosít.
- Az egységtesztelést általában az a fejlesztő végzi, aki a kódot írja, de legalább a tesztelt kódhoz való hozzáférés szükséges.
- A fejlesztők gyakran egy komponens kódjának megírása után írnak és hajtanak végre egységteszteket.
  - Azonban az automatikus egységtesztek megírása megelőzheti az alkalmazáskód megírását, lásd például a tesztvezérelt fejlesztést (TDD).

Tesztelési szintek: integrációs tesztelés (1)

- Komponensek vagy rendszerek közötti kommunikációra összpontosít.
- Az integrációs teszteknek magára az integrációra kell koncentrálnia, nem pedig az egyes komponensek/rendszerek működésére.

Tesztelési szintek: integrációs tesztelés (2)

Az integrációs tesztelés két különböző szintje:

- **Komponens integrációs tesztelés**: az integrált komponensek közötti kommunikációra és interfészekre összpontosít.
  - Az egységtesztelés után végzik és általában automatizált.
  - A komponens integrációs tesztelés gyakran a fejlesztők felelősége.
- **Rendszerintegrációs tesztelés**: rendszerek közötti kommunikációra és interfészekre összpontosít.
  - Kiterjedhet külső szervezetekkel és általuk szolgáltatott interfészekkel (például webszolgáltatásokkal) való interakciókra.
  - Történhet a rendszertesztelés után vagy a folyamatban lévő rendszertesztelési tevékenységekkel párhuzamosan.
  - A rendszerintegrációs tesztelés általában a tesztelők felelősége.

Tesztelési szintek: rendszertesztelés

- A rendszer egészének (funkcionális és nem funkcionális) viselkedésére összpontosít.
- Jellemzően független tesztelők végzik jelentős mértékben specifikációkra támaszkodva.

Tesztelési szintek: elfogadási tesztelés (1)

- Annak meghatározására összpontosít, hogy a rendszer kész-e a telepítésre és az ügyfél (végfelhasználó) általi használatra.
- Gyakran az ügyfél vagy a rendszerüzemeltetők felelőssége, de más érintettek is bevonhatók.
- A szoftver kiadása előtt azt néha odaadják potenciális felhasználók egy kis kiválasztott csoportjának kipróbálásra (**alfa tesztelés**) és/vagy reprezentatív felhasználók egy nagyobb halmazának (**béta tesztelés**).

Tesztelési szintek: elfogadási tesztelés (2)

Alfa tesztelés:

- Felhasználók és fejlesztők együtt dolgoznak egy rendszer tesztelésén a fejlesztés közben.
- A fejlesztő szervezet telephelyén történik.

Béta tesztelés:

- Akkor történik, amikor egy szoftverrendszer egy korai, néha befejezetlen kiadását elérhetővé teszik kipróbálásra ügyfelek és felhasználók egy nagyobb csoportjának.
- A felhasználók helyén történik.
- Főleg olyan szoftvertermékekhez alkalmazzák, melyeket sok különböző környezetben használnak.
- A marketing egy formája is.

Teszttípusok

A tesztelés átfogó célja szerint az alábbi teszttípusokat különböztetjük meg:

- **Funkcionális tesztelés** (***functional testing***)
- **Nem funkcionális tesztelés** (***non-functional testing***)
- **Fehér dobozos tesztelés** (***white-box testing***)
- **Változással kapcsolatos tesztelés** (***change-related testing***)
  - **Megerősítő tesztelés** (***confirmation testing***)
  - **Regressziós tesztelés** (***regression testing***)

Bármely teszttípus alkalmazható bármely tesztelési szinten.

Teszttípusok: funkcionális tesztelés

- A rendszer által nyújtott funkciók tesztelése.
- Más szóval annak tesztelése, amit a rendszer csinál.
- Funkcionális teszteket minden tesztelési szinten ajánlott végezni.

Teszttípusok: nem funkcionális tesztelés

- Rendszerek olyan jellemzőinek értékelése, mint például a használhatóság, teljesítmény vagy biztonság.
- Más szóval annak tesztelése, hogy a rendszer mennyire jól teszi a dolgát.

Teszttípusok: fehér dobozos tesztelés

- A rendszer belső felépítésén vagy megvalósításán alapuló teszek.
- A belső szerkezetbe beleérthető kód, architektúra vagy a rendszeren belüli munkafolyamatok.

Teszttípusok: változással kapcsolatos tesztelés

- Teszteket kell végezni, amikor módosítások történnek egy rendszerben egy hiba kijavításához vagy új funkcionalitás hozzáadásához/létező funkcionalitás módosításához.
- A változással kapcsolatos tesztelés két fajtája:
  - **Megerősítő tesztelés**: célja annak megerősítése, hogy az eredeti hiba sikeresen kijavításra került.
  - **Regressziós tesztelés**: Lehetséges, hogy egy változás a kód egy részében, akár egy javítás vagy másfajta módosítás, véletlenül hatással van a kód más részeinek viselkedésére. A regressziós tesztelés célja a változások által okozott akartalan mellékhatások érzékelése.

Feketedobozos tesztelés és fehérdobozos tesztelés

A tesztelési módszereket időnként a feketedobozos (*black-box*) vagy fehérdobozos (*white-box*) kategóriába sorolják:

- **Fekete dobozos tesztelés**: a belső szerkezetének ismerete nélkül vizsgálja egy rendszer funkcionalitását.
  - A tesztelők nem férnek hozzá a forráskódhoz.
  - Az ilyen tesztelés rendszerint specifikációk és követelmények köré épül: azaz hogy mit kellene, hogy csináljon az alkalmazás, nem pedig hogy hogyan.
- **Fehérdobozos tesztelés**: az alkalmazás belső működését ellenőrzi.
  - A forráskód rendelkezésre állását feltételezi.
  - A tesztek a forráskódon, nem pedig követelményeken vagy specifikációkon alapulnak.
  - Üvegdobozos (*glass-box*) teszelésnek is nevezik.

Egységtesztelés (1)

Az egységtesztelés (*unit testing*) a programkomponensek, például a metódusok és az osztályok tesztelésének folyamata.

Egységtesztelés (2)

A jó egységtesztek ismertetőjegyei: FIRST (Robert C. Martin)

- **Gyors** (***Fast***): A tesztek gyorsak kell, hogy legyenek. Gyorsan kell, hogy lefussanak.
- **Független** (***Independent***): A tesztek nem függhetnek egymástól.
- **Megismételhető** (***Repeatable***): A tesztek bármely környezetben megismételhetők kell, hogy legyenek.
- **Önérvényesítő** (***Self-Validating***): A teszeknek logikai kimenete kell, hogy legyen. Vagy átmennek, vagy megbuknak.
- **Jól időzített** (***Timely***): A teszteket kellő időben kell megírni, közvetlenül a tesztelendő kód előtt.

Lásd:

- Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2009.

Egységtesztek szervezése: az AAA minta (1)

A teszteknek az alábbi három része ajánlott, hogy legyen:

- **Elrendez** (***Arrange***): ez a rész felelős a tesztelt rendszer és függőségei egy kívánt állapotba állításáért.
- **Cselekszik** (***Act***): ez a rész szolgál a tesztelt rendszer metódusainak meghívására, az előkészített függőségek átadására és a kimeneti érték elkapására (ha van).
- **Kijelent** (***Assert***): ez a szakasz szolgál a kimenetel ellenőrzésére. A kimenetel ábrázolható a visszatérési értékkel vagy a tesztelt rendszer végső állapotával.

Lásd:

- Bill Wake. *3A – Arrange, Act, Assert*. April 26, 2011. https://xp123.com/3a-arrange-act-assert/

Egységtesztek szervezése: az AAA minta (2)

Alternatív megfogalmazás: Feltéve, hogy-amikor-akkor (*Given-When-Then*)

Lásd:

- Martin Fowler. *GivenWhenThen*. 21 August, 2013. https://martinfowler.com/bliki/GivenWhenThen.html

Egységtesztek szervezése: az AAA minta (3)

Példa: MutablePairTest.java (Apache Commons Lang)

```
@Test
public void testPairOfMapEntry() {
  // Arrange:
  final HashMap<Integer, String> map = new HashMap<>();
  map.put(0, "foo");
  final Entry<Integer, String> entry = map.entrySet().iterator().next();

  // Act:
  final Pair<Integer, String> pair = MutablePair.of(entry);

  // Assert:
  assertEquals(entry.getKey(), pair.getLeft());
  assertEquals(entry.getValue(), pair.getRight());
}
```

Tesztdublőrök (1)

- A tesztelendő kód gyakran függ más objektumoktól, melyek a tesztek végrehajtásához tesztdublőrökkel cserélhetők ki.
- Egy **tesztdublőr** (***test double***) egy objektum, mely úgy néz ki és viselkedik, mint a valós megfelelője, de ténylegesen egy leegyszerűsített változata, mely a bonyolultságot csökkenti és megkönnyíti a tesztelést.
- Maga a név a kaszkadőr (*stunt double*) kifejezésből származik.

Tesztdublőrök (2)

- Egy ilyen tesztdublőr helyettesíthet egy nem implementált objektumot.
- A tesztdublőrök lehetővé teszik továbbá egy kódrész a függőségeitől izolált módon történő tesztelését.
    - Azaz a tesztelés kizárólagosan egy kódrészre fókuszálhat, elválasztva annak viselkedését a külső hatásoktól.

Tesztdublőrök (3)

További hivatkozások:

- Martin Fowler. *TestDouble*. 2006.
  https://martinfowler.com/bliki/TestDouble.html
- Gerard Meszaros. *XUnit Test Patterns*. Addison-Wesley, 2007.
  http://xunitpatterns.com/
  - Lásd: *Chapter 11: Using Test Doubles*
    http://xunitpatterns.com/Test%20Double%20Patterns.html

Automatizált egységteszt keretrendszerek (1)

Egy átfogó lista: *List of unit testing frameworks*
https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

- C++:
  - Catch2 (licenc: *Boost Software License 1.0*)
    https://github.com/catchorg/Catch2
  - GoogleTest (licenc: *BSD 3-Clause License*)
    https://google.github.io/googletest/
    https://github.com/google/googletest/
  - Qt Test (licenc: nem szabad/GNU LGPLv3/GNU GPLv2)
    https://doc.qt.io/qt-6/qttest-index.html

Automatizált egységteszt keretrendszerek (2)

- Java:
  - JUnit (licenc: *MIT License*) https://junit.org/junit5/ https://github.com/junit-team/junit5
  - TestNG (licenc: *Apache License 2.0*) https://testng.org/ https://github.com/testng-team/testng
- .NET:
  - NUnit (licenc: *MIT License*) https://nunit.org/ https://github.com/nunit/nunit
  - xUnit.net (licenc: *Apache License 2.0*) https://xunit.net/ https://github.com/xunit/xunit

Automatizált egységteszt keretrendszerek (3)

- Python:
  - pytest (licenc: *MIT License*) https://docs.pytest.org/ https://github.com/pytest-dev/pytest
  - unittest (licenc: *Python Software Foundation License*) https://docs.python.org/3/library/unittest.html
- R:
  - testthat (licenc: *MIT License*) https://testthat.r-lib.org/ https://github.com/r-lib/testthat/
- PHP:
  - PHPUnit (licenc: *BSD 3-Clause License*) https://phpunit.de/ https://github.com/sebastianbergmann/phpunit

Automatizált egységteszt keretrendszerek (4)

- JavaScript:
  - Cypress (licenc: *MIT License*) https://www.cypress.io/ https://github.com/cypress-io/cypress
  - Jest (licenc: *MIT License*) https://jestjs.io/ https://github.com/facebook/jest
  - Mocha (licenc: *MIT License*) https://mochajs.org/ https://github.com/mochajs/mocha