Tesztautomatizálás

Kundra Zoltán

Készült: 2015.

Terjedelem: 7 ív

A tananyag elkészítését a Munkaerő-piaci igényeknek megfelelő, gyakorlatorientált képzések, szolgáltatások a Debreceni Egyetemen Élelmiszeripar, Gépészet, Informatika, Turisztika és Vendéglátás területen (Munkaalapú tudás a Debreceni Egyetem oktatásában) ***TÁMOP-4.1.1.F-13/1-2013-0004*** *számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.*

TARTALOMJEGYZÉK

Tartalom

1. Bevezetés

Tökéletes, hibáktól mentes, precízen működő szoftver nincs. Azaz minden attól függ, hogy mikor vizsgáljuk az adott szoftvert. Amennyiben egy adott pillanatban, adott teszteseteket végrehajtva nem ütközünk hibákba, úgy elmondhatjuk ezeket. Viszont a folyamatosan változó szoftverkörnyezet és alkalmazás-funkcionalitás változások rákényszerítik a tesztelőket arra, hogy rendszeresen más-más teszteket hajtsanak végre annak érdekében, hogy megbizonyosodjanak arról, az alkalmazás jól működik.

Az üzleti igényeknek időről-időre eleget kell tennünk, változtatni kell már meglévő funkciókat és működéseket a szoftvereinkben. Ez azt hozza magával, hogy a meglévő tesztállományunkat bővíteni kell, valamint változatlanul meg kell bizonyosodnunk, hogy a régi funkcionalitások működnek, az újak pedig az előzetesen definiált követelményeknek megfelelnek. Gondoljunk bele, hogy mennyi tesztet kellene végrehajtanunk, hogy azt tudjuk mondani, tökéletes és hibamentes az alkalmazásunk? Mi az a mennyiség, amire azt tudjuk mondani, hogy ezt végre kell hajtani? Nehéz ezekre a kérdésekre válaszolni, azonban tény, hogy minél több tesztet hajtunk végre, annál biztosabbak lehetünk abban, hogy az elvárt funkcionalitás mentén működik a szoftverünk.

A tesztelőknek nincs egyszerű dolga a folyamatosan változó szoftverkörnyezetben. Minél jobban szélesedik az alkalmazások és funkcionalitások tárháza, annál több tesztet kell végrehajtani. Az automatizált szoftvertesztéléssel az a célunk, hogy olyan teszteket hajtsunk végre, melyek nem kívánnak meg manuális tesztelői beavatkozást, célunk, hogy gyors és pontos eredményeket kapjunk, támogassuk a manuális tesztelői munkát olyan megoldásokkal, melyek nehezen, vagy akár semmilyen módon nem lehetne végrehajtani manuális módon.

Manapság az automatizált szoftvertesztelés sokkal nagyobb teret kezd meghódítani, mint öt, vagy akár hét évvel ezelőtt. Számos nyílt forráskódú eszközt el tudunk érni a világhálón, ugyanakkor a kereskedelmi forgalomban kapható eszközök listája is nagymértékben bővült.

Ennek a tananyagnak a funkcionális tesztautomatizálás megismertetése a célja. Átfogó ismertetést szeretnék adni, hogy mikor és milyen módon célszerű használni. Szeretném megismertetni az olvasóval azt a folyamatot, ami mentén célszerű bevezetni egy automatizálási eszközt, valamint szeretném megmutatni, hogyan lehet ezt a lehető legoptimálisabb módon használni. Természetesen része az anyagnak pár olyan példa is, amit célszerű megfogadni annak érdekében, hogy mindenki a lehető legsikeresebben tudjon elindulni ezen az úton, vagy folytatni azt, amit már elkezdett.

Bízom benne, hogy ezek alapján az érdeklődő sikeresen el tudja kezdeni kalandozását ezen az érdekes, folyamatos kihívásokkal teli ösvényen, amit úgy hívunk: Tesztautomatizálás.

2. Általános áttekintés

Biztos vagyok benne, hogy akik szoftvertesztéléssel foglalkoznak, valamilyen körülmények között szembesültek az alábbi párbeszéddel:

- *Tulajdonképpen mivel is foglalkozol, mi a munkád?*
- *Szoftvertesztelő vagyok.*
- *Értem. És pontosan mit is csinálsz?*

A szoftvertesztelést, mint szakmát és a vele kapcsolatos tevékenységeket igen nehéz elmagyarázni másoknak. Azaz elmondani nem nehéz, hiszen ebben dolgozunk, de hogy ezt meg is értse más is, az bizony nagyon nehéz feladat.
Akik tesztautomatizálással foglalkoznak, azoknak szintén nem egyszerű elmagyarázni, hogy miben is más egy automatizálással foglalkozó szoftvertesztelő feladata.
Próbáljuk meg definiálni, mi is a tesztautomatizálás. Biztos vagyok benne, hogy elsőnek az interneten keresztül próbáljuk megtalálni a választ:

Wikipedia (http://en.wikipedia.org/wiki/Test_automation):
„Szoftvertesztelésen belül, a tesztautomatizálás nem más, mint egy speciális szoftver használata (a tesztelendő szoftvertől független szoftver) tesztek végrehajtásának kontrollálására, és az aktuális eredmények összehasonlítása az előre várt eredményekkel."

http://searchsoftwarequality.techtarget.com/definition/automated-software-testing:
2Automatizált szoftvertesztelés az egy folyamat, amelyben automatizált eszközök futtatnak előre megírt szkripteket a tesztelt alkalmazáson, mielőtt az végső állapotba kerül.2

http://blogs.msdn.com/b/steverowe/archive/2007/12/19/what-is-test-automation.aspx:
„Automatizált szoftvertesztelés egyszerűen egy automatikus módja annak, amit a manuális tesztelők csinálnak, csináltak. Tesztautomatizálás programok sora, ami interfészeket hív meg, gombokat nyomogat, és végül programozottan megvizsgálja, hogy a megfelelő művelet hajtódott-e végre vagy sem."

Ezeken kívül biztos vagyok benne, hogy még számtalan létezik könyvekben, interneten, publikációkban, de vajon melyik lehet a megfelelő definíció?

Automatizált szoftvertesztelés során törekszünk arra, hogy pontosan definiált tesztlépéseket hajtsunk végre előre megírt, számítógéppel végrehajtható programkódokkal, melyekben pontosan azokat a lépéseket hajtjuk végre, mint a manuális tesztelés során, vagy olyan folyamatokat, amivel a tesztelési tevékenységeinket hatékonyabban tudjuk támogatni.

2.1. Mire jó, mire nem?

Ha nagyon röviden akarunk erre a kérdésre válaszolni: automatikusan végrehajtani a manuális teszteket. Ennél azért többről van szó. Tény, hogy **egy automatikus teszt nem csinál annál többet, mint amit „megmondunk" neki**. Pontosan ugyanazokat a lépéseket hajtja végre, mint egy manuális tesztelő. Viszont meg kell jegyeznünk, hogy **a manuális tesztelő munkáját nem tudjuk teljes egészében kiváltani automatizálással**. Nem tudunk olyan mértékben intelligens teszteket írni, mint ahogy egy manuális tesztelő képes dolgozni.

Mivel a teszteket számítógéppel hajtatjuk végre, ezért tökéletes **nagy mennyiségű adathalmaz**ra végrehajtani a teszteket. A gép nem fárad el úgy, mint egy tesztelő, nem megy ki a mosdóba, nem ebédel, nem jár megbeszélésre. És az egyik, talán majdnem a legfontosabb dolgot ne felejtsük el: a **nap 24 órájában használhatjuk**. Automatizált teszteket olyan területeken használhatunk sikeresen, ahol a **regressziós tesztek** pontosan definiálva vannak, és nem történnek a tesztelendő alkalmazásban gyakori változások. **Smoke tesztek**re is kiváló. Gyorsan és hatékonyan tudunk rövid teszteket futtatni annak érdekében, hogy megbizonyosodjunk arról, a tesztelés alatt álló szoftver olyan állapotban van-e, hogy el lehet kezdeni a mélyrehatóbb tesztelést vagy sem. Manuális tesztelő szemszögéből a sok, ugyanolyan lépést tartalmazó tesztesetek egy idő után unalmassá válnak, és igazából nem is egy kihívásokkal teli tesztelési forma ez. Tipikusan ott nagyon jól alkalmazható a tesztautomatizálás, ahol sok **hasonló lépést kell végrehajtanunk**, de változó adattal.

Az elmúlt évek alkalmazásfejlesztési életciklusának időtartama véleményem szerint nagyban változott. A mai tendencia azt mutatja, hogy az üzleti igényekre a lehető leghamarabb és leggyorsabban próbálunk meg megoldásokat kínálni. Ennek az a következménye, hogy különböző alkalmazások tervezésére és implementálására viszonylag gyorsan kell megoldást találni, így a tesztelés időtartama is változhat és változik is. Minél több funkcióval látjuk el az alkalmazásunkat, annál több tesztelést kell végrehajtani. Idővel a folyamatos fejlesztések miatt nem biztos, hogy mindig minden tesztesetet végre tudunk hajtani. Ebben az esetben érdemes tesztautomatizálásban gondolkozni. Képzeljük el, hogy akár egy minimális változtatás az alkalmazásban mit indukálhat. Tesztelési oldalon minden egyes funkcionalitás-változás esetén meg kell győződnünk arról, hogy nem sérült az alkalmazásunk más pontokon. A funkciók száma és a tesztek száma olykor nem tud lépést tartani. Ez bizony kockázatként jelentkezik. Automatizált teszteléssel ki tudjuk küszöbölni, vagy legalábbis hozzá tudunk járulni ahhoz, hogy a szűkös határidők miatt valamilyen teszt kimaradna a végrehajtásból. Megfelelő tesztek automatizált végrehajtásával a tesztelési időt is csökkenteni tudjuk.

2.2. Mi a célja?

A rohamos léptekben történő alkalmazásfejlesztés során szinte biztos, hogy nem tudjuk a tesztjeinket olyan mértékben és ütemben végrehajtani, mint ahogy azt szeretnénk. Automatizált tesztekkel különböző célokat próbálunk meg teljesíteni, amelyek egy adott cégnél, vagy vállalatnál nagyon eltérőek lehetnek. Célszerű szem előtt tartani talán a legfontosabbakat, melyek mentén érdemes haladnunk.

Ideális esetben létezik egy fejlesztő csoportunk, de emellett működik egy tesztelő csoportunk is. Sőt, ne feledkezzünk meg az üzleti elemzőkről sem. Jobb esetben ez a három nagy csoport valamilyen szinten megjelenik egy cégnél, vagy vállalatnál. Előfordulhat például az is, hogy a tesztelést a fejlesztők hajtják végre. Ez magában hordoz bizonyos kockázatot, hiszen a fejlesztők tudják, mit fejlesztettek le, és nem biztos, hogy olyan mértékben és figyelemmel fognak hozzá a teszteléséhez, mint ahogy azt egy dedikált tesztelői csoport kezdené. Legyen adott, hogy létezik egy tesztelői csoport. Képzeljük el, hogy a tesztelők egy bizonyos számú regressziós teszteset-állományt hajtanak végre a vizsgált alkalmazás minden egyes új verziójának megjelenésénél. Tegyük fel, hogy a különböző verziók fejlesztési időszakában megjelennek új funkcionalitások is. Mivel ezek az új funkciók is megkövetelik, hogy a lehető legjobban le legyenek tesztelve, így a tesztelőknek több időre lesz szükségük a munkájuk elvégzéséhez. Verzióról verzióra viszont az **új funkcionalitások előbb-utóbb regressziós tesztekké fognak átalakulni**. Itt van az első olyan dolog, ami egy automatizált tesztelés célja lehet **regressziós tesztesetek implementálása**. Próbáljunk meg az automatizálás segítségével **felszabadítani manuális tesztelői erőforrást** annak érdekében, hogy a tesztelők a legjobban tudjanak koncentrálni az új funkcionalitások tesztelésére, vagy a fejlesztési időszak alatt történő teszttervezésre. Amennyiben sikerült a kiválasztott regressziós teszteket lefejlesztenünk, ne felejtsük el azokat **rendszeresen végrehajtani és használni a tesztelési időszakokban**. A tesztelési időszakok száma nagyban függhet a céges, vállalati kultúrától. A következő cél, amit érdemes szem előtt tartani, hogy **az elérhető tesztjeinket a lehető legoptimálisabb módon próbáljuk meg használni** ezekben az időszakokban. Sokan esnek abba a csapdába, hogy az automatizálás lehetőségeit csak fejlesztési és tesztelési időszakokban használják ki. Ne koncentráljunk csak ezekre a periódusokra. Próbáljuk meg feltérképezni, hogy melyek lehetnek azok a tesztek, amelyeket akár az éles rendszerünkben is tudunk használni. Sokszor nagy segítséget nyújtanak olyan automatikus tesztek, melyek nem járnak tranzakcióval. Ilyenek lehetnek akár olyan tesztek, amelyek tartalmat, objektumokat, folyamatokat ellenőriznek az éles rendszerünkben.

Korábban említettem, hogy jobb esetben létezik egy tesztelői csoport. A csoportban lévő tesztelők száma kötött. Mivel manuális teszteléssel foglalkoznak, így elképzelhető, hogy a teszttervezési időszakban olyan teszteseteket definiáltak, melyek **végrehajtása szinte lehetetlen emberi munkával**. Ez nem azt jelenti, hogy nem is lehet ezeket a teszteket végrehajtani, hiszen automatizált tesztekkel az ilyen feladatok nagyon jól végrehajthatók, és majdnem a lehető legnagyobb nyereséget is tudjuk elérni ez által. Vegyünk egy példát. Adatbázis migráció. Egyik adatbázisból átmozgatjuk az adatokat egy másik adatbázisba. Ha belegondolunk, ilyenkor általában nagy adatmennyiséggel találkozunk. Hogyan győződünk meg arról, hogy a forrás adatbázisból megfelelő módon mozgott át az adat a céladatbázisba? Megszámolni könnyen meglehet, hogy a rekordok megegyeznek-e, vagy sem. Viszont érték alapján ellenőrizni nagy mennyiségben manuálisan szinte képtelenség. Egy jól megírt automatikus teszttel az adatellenőrzés hatékonysága és ideje drasztikusan javulhat.

Mindezeket szem előtt tartva, ha elindulunk az automatizálás útján, hamarosan eljutunk oda, hogy egy **keretrendszer kidolgozására lesz szükségünk**. Egyik fontos cél, hogy azonosítsuk azon kódokat a meglévő tesztjeinkben, melyek többször hajtódnak végre, többször szerepelnek. Ezeket próbáljuk meg modulokba szervezni annak érdekében, hogy **elérjük és megvalósítsuk az újrafelhasználhatóságot**. Az

így keletkezett sok kis elemet egyszerűbben tudjuk használni a későbbi tesztkészítésekkor, nem esünk bele abba a hibába, hogy újra és újra hasonló megoldásokat kell implementálnunk. Ezzel elérjük azt is, hogy a tesztjeink és kódjaink karbantarthatósága egyszerűbb és hatékonyabb legyen.

Az automatizált szoftvertesztelés erőforrás-igénnyel jár. Erőforrás alatt nemcsak emberi óraszámot értünk, hanem a futtatáshoz szükséges gépi szükségletet is. Próbáljuk meg a tesztjeinket úgy futtatni, hogy a **lehető legjobban használjuk ki a gépeink kapacitását**.

Végül talán az egyik legfontosabb szempont, hogy **ne próbáljunk meg mindent automatizálni**. Amennyiben léteznek olyan tesztek, amelyeket lehet automatizálni, még nem biztos, hogy van is értelme hozzákezdeni. Az automatikus tesztelésnek úgy kell megjelennie a folyamatainkban, hogy a lehető legtöbbet tudja nyújtani a tesztelők számára. Vizsgáljuk meg a meglévő, de még nem automatizált teszteseteket, és **jelöljük meg azokat, melyeket lehet automatizálni**. Figyelem! **Ez még nem azt jelenti, hogy kell is!** Ezekből a szelektált esetekből nézzük meg, hogy melyek azok, amelyeket érdemes is implementálni, és azokkal foglalkozzunk első körben. [5]

2.3. Miért nem egyszerű?

Kezdő automatizálással foglalkozó szoftvertesztelők hamar sikerélményként élik meg az első működő teszteket. Ezzel nincs is semmi probléma. Idővel, amikor még több automatizálási igény fog megjelenni, pár nehézséggel fognak szembesülni. Sokan azt gondolják, hogy automatizálni egyszerű. Ez lehet, hogy így is van, hiszen ha valaki jártas egy programozási nyelvben, akkor tud olyan automatikus megoldásokat implementálni, amelyek működnek. Viszont sokan nem látják azt, hogy helyesen és hatékonyan automatizálni nem egyszerű, sőt, nagy precizitást és számos tapasztalatot igényel.

Mindenképpen **szükséges valamilyen programozói jártasság**. Nem feltétlenül szükséges ismerni az összes objektumorientált programozási nyelvet, szkriptnyelvet stb. Fontos, hogy olyan **gondolkodásmód**unk legyen, ami szükséges az alkalmazásfejlesztéshez. Minél több tapasztalatot szerzünk tesztek automatizálása során, annál inkább fogjuk felismerni a tényt, hogy nem áll messze az a munka, amit csinálunk egy programozó feladatától: azaz az **alkalmazásfejlesztéstől**.

Korábban említettem, hogy teszteket automatizálni, valamint azokat végrehajtani **költséggel jár**. Amennyiben kereskedelmi forgalomban lévő automatizált szoftvertesztelésre használt eszközt fogunk használni, akkor a licenszelésnek és a terméknek ára van. Szükségünk van **gépekre, olyan futtató állomásokra**, amelyeken a tesztjeinket fogjuk végrehajtani. Tapasztalt tesztelőkre lesz szükségünk, akik ismerik a használt eszközt, vagy ha nem, akkor képesek lesznek rá, hogy viszonylag hamar megismerjék és használni is tudják azt. A betanulás, a szakmai fejlődés érdekében végzett folyamatos ismeretbegyűjtés mind idővel járnak.

Egy olyan környezetben (vállalat, cég), ahol nem volt még jelen ilyen típusú tesztelés, a körülöttünk dolgozó kollégákat, vezetőséget **meg kell tudnunk győzni az automatizálás előnyeiről és**

hatékonyságáról. Folyamatosan figyelnünk kell, hogy a folyamataink megfelelően működnek vagy sem, valamint a tesztjeinket úgy és olyan formában használjuk-e, ahogy az elvárt.

Mindenképpen tartsuk szem előtt, hogy mikor fog az az idő **megtérülni**, amit a tesztjeink elkészítésére fordítottunk (megbeszélések, kódolás, hibajavítás, tesztelés stb.). Fontos követnünk, hogy hányszor és mennyi ideig futnak a tesztjeink egy tesztelési ciklusban, hiszen ezek alapján tudjuk megmondani, hogy sikeres volt vagy sem az adott teszt lefejlesztése. Tapasztalt automatizált teszteléssel foglalkozó tesztelők a megtérülést már a tervezés során jó eséllyel meg tudják becsülni. Amennyiben nem hozná vissza a tesztek használata azt a befektetést, amit a futtatásokkal visszanyernénk, úgy el kell gondolkoznunk, hogy tényleg bele kell-e fognunk az automatizálásba, vagy sem.

Utolsóként pedig említsük meg, hogy **automatizálással nem lehet kiváltani teljes egészében a manuális tesztelést**. Ez nagyon fontos és úgy gondolom, hogy nem lehet elégszer hangoztatni. És hogy ez miért szerepel itt? Szorosan együtt kell dolgozni a tesztelőkkel annak érdekében, hogy a legjobb megoldások szülessenek meg automatizálás során. A manuális tesztelési folyamatokat lehet, hogy másképpen kell szervezni annak érdekében, hogy a megtérülés a legjobb legyen. Azzal, hogy mindent lefedünk automatizálással, még nem biztos, hogy közelebb kerültünk a minőség javulásához. Még az sem biztos, hogy a tesztlefedettséget növeljük. Nagyon pontosan és precízen kell detektálni, vagy módosítani a megfelelő teszteseteket.

2.4. Mikor javasolt használni?

Itt nem arra próbálok meg választ adni, hogy tesztelési időszakokon kívül mikor használható még ez a módszer. Elsősorban **az a cél, hogy felismerjük, mikor lehet és érdemes**, valamint a körülmények milyenek? Előző példa alapján jól látszik, hogy nagy adathalmaz használatakor elérjük a kívánt célt, hiszen manuálisan szinte lehetetlen több száz rekordot akár ellenőrizni is egy rendszerben, nemhogy létrehozni. Amennyiben a **tesztelési folyamataink megkívánnak több ugyanolyan lépést**, de különböző adathalmazra, úgy az automatikus végrehajtás a lehető legjobb megoldás. A gép nem tud hibázni, ugyanakkor a tesztelő biztos, hogy unalmasnak fogja találni már a huszadik tesztelést is. Gondoljunk bele jobban, ha a funkcionalitás különböző adatok használatával nem változik, akkor ez már majdnem **adatgenerálás**nak minősül, tehát akár tesztadat generálásra is tudjuk használni meglévő tesztjeinket. Míg a manuális másolás és beillesztés folyamatnál akár hibát is véthet egy tesztelő, a gép nem fárad el, pontosan azt hajtja végre, amit kell.

Sikeresen használhatjuk **tesztelési időszakokon kívül is**. Gondoljunk bele, hogy egy fejlesztési folyamat közepén járunk. A meglévő tesztjeink futtatásával megbizonyosodhatunk arról, hogy az újonnan fejlesztett funkcionalitások nem tesznek kárt a meglévő folyamatokban.

Használható a rendszereinkben történő **környezeti beállításokra** is. Amennyiben sok beállítás, előfeltétel szükséges egy új funkcionalitás működésére, akkor automatikus megoldásokkal gyorsabban és pontosabban végrehajthatóak ezek a lépések, mint manuálisan.

Elképzelhető, hogy bizonyos funkciók, vagy folyamatok olyan időpontokban érhetőek el, amikor **manuális erőforrás nem áll rendelkezésre.** Ezekben az esetekben a gép által végrehajtott ellenőrzés, tesztelés akár az esti órákban, vagy hétvégéken is elvégezhetők.

2.5. Mikor nem javasolt használni?

Az előzőekben felsorolt számos érv mellett vannak olyan esetek, amikor nem javasolt az automatizált tesztek készítése. Alapvető ökölszabálynak vehető (funkcionális automatizált tesztelés esetén), hogy akkor kezdünk el automatikus teszteket implementálni, amikor az alkalmazásunk stabil, nincsenek kódmódosítások, funkcionalitásváltozások. Tudjuk, hogy hogyan és miként kell viselkednie az alkalmazásnak. Abban az esetben, ha nem ez áll fent, nagy valószínűséggel nem javasolt elkezdeni dolgozni a teszteken. **Folyamatosan változó szoftverkörnyezetben nehéz automatikus teszteket implementálni és futtatni is.**

Tudjuk, hogy az alkalmazásfejlesztés életciklusában szerepel egy tervezési fázis. Ha a tervezés nem megfelelően van elkészítve (folyamatosan változnak az igények, nincsenek előre pontosítva a szükséges funkcionalitások), azt tapasztaljuk, hogy a fejlesztési ciklusban is folyamatosan változni fognak a működési folyamatok. Tesztelési oldalról lehet, hogy felmerül az igény automatikus tesztek használatára, viszont tartsuk szem előtt, hogy egy olyan környezetben, ahol folyamatosan változnak a funkcionalitások, nem tudunk hatékonyan teszteket készíteni. Miért? Minden egyes funkcionalitás akár minimális változtatása esetén is szükséges az automatikus tesztjeinket finomítani annak érdekében, hogy a lehető legpontosabb eredményeket kapjuk. Ezzel azt érjük el, hogy a **karbantartási fázisunk automatizálási oldalról nagyon sok erőforrást fog igényelni**. Addig, amíg nem egyértelmű, hogy miként is kell működnie az alkalmazásunknak, ne kezdjünk el teszteket implementálni, mert hosszútávon csak nehézségeket fog okozni.

Tegyük fel, hogy a tesztelendő alkalmazás fejlesztése befejeződött és úgy tűnik, hogy a megfelelő funkcionalitással lett ellátva a programunk, az előzetes követelményeknek megfelel a működés. Ekkor a tesztelő csoport - a követelmények alapján - már a fejlesztési fázis során elkezdi feltérképezni, hogy milyen teszteket célszerű végrehajtani. Lehet, hogy a tesztelői csapat nagy szakmai tapasztalattal rendelkezik, viszont **ha nem dokumentálják, hogy pontosan mit és milyen lépésekkel kell leteszteln**i, akkor ez szintén nehézséget tud okozni az automatizálási oldalon. Ha egy automatizálással foglalkozó tesztelő az előzőleg összeállított tesztterv alapján dolgozik, azaz nincs lehetősége konzultációra a tesztelőkkel, akkor számos hibába ütközhet.
Nézzünk egy példát: a követelmények alapján egy bejelentkezési felületnek úgy kell viselkednie, hogy egy e-mail és jelszó párossal való belépést követően egy üdvözlőképernyőnek kell megjelennie. A követelmények között szerepel, hogy nem minden e-mail-re lesz ugyanaz az üdvözlő üzenet a felhasználói felületen. A teszttervben viszont csak annyi szerepel, hogy adjuk meg az e-mail címet és jelszót, majd kattintsunk a belépés gombra. Ez után azt kell látnunk, hogy „üdvözöljük kedves felhasználó”. Ez alapján, aki az automatikus teszteket készíti, azt az ellenőrzési feltételt fogja

implementálni a tesztben, hogy a felületen megjelenik az „üdvözöljük kedves felhasználó” szöveg vagy sem. Viszont a követelmény szerint, ha például yahoo-s email címmel jelentkezünk be, akkor ki is kell írnia az alkalmazásnak a felhasználó keresztnevét. Tehát yahoo-s e-mail címmel így néz ki a felület: „üdvözöljük kedves Tamás felhasználónk”. Látható, hogy ebben az esetben két dolog is különbözik: egyrészt a keresztnév, másrészt a „felhasználó” helyett a „felhasználónk” szöveg jelenik meg. Ekkor az automatikus tesztünk nagy valószínűséggel hibás eredményt fog mutatni. Jegyezzük meg, hogy teljesen jogosan, hiszen a teszterv alapján egységesen az „üdvözöljük kedves felhasználó” szövegnek kell megjelennie. **Azaz a tesztelők is véthetnek hibát a teszttervezés során.**

Beszéljünk arról is, amikor a tesztautomatizáló vét hibát. A tervezési fázis hibátlanul zajlik, a tesztelők megfelelően készítik el a teszterveket, a programozók jól fejlesztik le a funkcionalitásokat, a tesztautomatizáló is tökéletesen készíti el a tesztet, viszont a következő tesztelési ciklusban jön az igény, hogy módosítani kell az automatikus tesztet. Megtörténik a javítás, sikeresen használjuk a tesztet. A következő tesztelési ciklusban is módosítani kell valamit a teszten, hogy rendesen működjön. Vajon mi lehet itt a probléma? Az automatizálással foglalkozó tesztelőnek a feladata lenne az is, hogy **megvizsgálja és kiderítse, hogy lesz-e, vagy sem az elkövetkező időszakban funkcionalitásváltozás** az adott alkalmazáson, amire tesztet készít. Miért lényeges ez? Mint ahogy a példa mutatja, ebben az esetben kisebb, vagy nagyobb módosításokat kell végrehajtani az automatikus teszten. Azaz a karbantartási idő növekedik, ami jelentkezni fog a megtérülés vizsgálata során. Nem a legsikeresebb, ha egy alkalmazáson viszonylag sűrű funkcionalitásváltozás történik egy adott időszakon belül.

Abban az esetben sem célszerű elkezdeni automatizálni, ha a szükséges idő nem áll rendelkezésünkre a teszt elkészítéséhez. **Természetesen az is kockázat, ha a megfelelő erőforrás (emberi, gépi) nem áll rendelkezésünkre.** Fontos szem előtt tartani ezeket is, és ennek fényében eldönteni, hogy lehetséges, vagy sem a teszteken dolgozni.

2.6. Automatizált tesztgenerációk

Elég nehéz megmondani, hogy az automatizált szoftvertesztelés mikor jelent meg a szakmában. Az biztos, hogy 1985-ben jelent meg az első kereskedelmi forgalomban kapható eszköz. [6] Mint ahogy a technika is fejlődött, úgy az automatizált tesztelésben is megfigyelhető a fejlődés. Ahogy az alkalmazásfejlesztési ciklus is változott, úgy jelentek meg a különböző tesztautomatizálási típusok, generációk. Ebben a részben kicsit részletesebben ismertetem a különböző típusokat.

Első generáció – Lineáris automatizálás

Ez a típusú tesztkészítés tekinthető a legegyszerűbbnek. Igazából érdemes is elgondolkozni azon, hogy tesztkészítésről, vagy tényleg egy sima tesztlépés rögzítésről beszélünk. Ez a legelső szint, ami a tipikus **felvétel-visszajátszás elven működik**. Ha jobban megnézzük a tesztvégrehajtást támogató eszközöket, melyek elérhetők a világhálón (legyen az nyílt forráskódú, vagy kereskedelmi forgalomban kapható), bizton állíthatjuk, hogy soknak egyik alapvető funkciója a felvétel és visszajátszás. Ezt tényleg úgy képzelhetjük el, mint egy felvevő „eszközt”. Elindítjuk a „felvételt” és minden olyan lépést, akciót,

műveletet, amit végzünk a tesztelés alatt lévő alkalmazáson, az eszköz felveszi, rögzíti. Tulajdonképpen egy makró rögzítéshez hasonlítható. A felvétel végén el is készül az a teszt, ami rögtön használható is. Ha jobban belegondolunk, ezzel speciális tesztelést nem is hajtottunk végre, arról tudunk megbizonyosodni, hogy a tesztlépések végrehajtódnak-e, vagy sem.
Vegyünk példának egy webes űrlapot. Az űrlap tartalmaz két mezőt, egy szöveges részt, valamint egy gombot, amivel elküldhetjük az adatokat. Az egyik mező egy e-mail címet kér, a másik pedig egy tárgy mező. Amikor a felhasználó rögzíti a lépéseket, akkor nagy valószínűséggel a következő lépések fognak felvételre kerülni:

```
Input "E-Mail" into email textbox
Input "Tárgy" into targy textbox
Input "Üzenet" into uzenet textbox
Click Küldés button
```

Ha megnézzük, egy egyszerű folyamatot látunk, ezek alapján a végrehajtás során pontosan ezek a lépések fognak végrehajtódni. Viszont azon kívül, hogy minden lépés végrehajtódik-e vagy sem, nem teszteltünk semmit. Melyek lehetnek a lehetséges ellenőrző lépések? A legegyszerűbb ellenőrzési pont, ha a Küldés gombra kattintás után a felhasználói felület ad nekünk visszajelzést vagy sem. Ebben az esetben lehetséges egy feltétellel megvizsgálni azt, hogy az elvárt üzenet megjelenik-e, vagy sem. Például így nézhet ki egy ellenőrző rész, ami a fenti kódrészlet után lehetséges:

```
If "Köszönjük az érdeklődést!" exists then
      Test Passed
Else
      Test Failed
End If
```

Vizsgáljuk meg jobban ezt az esetet. Ezzel a módszerrel gyorsan készíthetünk teszteket és nem kíván nagyfokú programozási jártasságot sem. **Egy teszt megfeleltethető a hozzá tartozó manuális tesztesetnek**. Sajnos hosszabbtávon a **karbantarthatóság költséges**. Amennyiben olyan változtatás történik az alkalmazásban, ami több tesztesetet is érint, akkor minden egyes tesztet módosítani, frissíteni kell annak érdekében, hogy a futtatások során ne történjen hiba. Milyen eset fordulhat még elő, amikor tesztet kell frissíteni? Például megváltoztatják az email szöveges mező objektum nevét. Ebben az esetben a teszt nem tudja végrehajtani már az első lépést sem, mert az email szövegdoboz nem szerepel az űrlapon. Általában a felvétel-visszajátszást támogató eszközök a felhasználói felületről rögzítik az objektumokat is. Itt, ha módosul az e-mail szöveges doboz objektum neve, nem tudja megtalálni a programunk.

Tegyük fel, hogy létezik száz olyan tesztesetünk, amiben a küldés gombra kattintunk. Amennyiben a gomb valamilyen tulajdonsága változik, akkor mind a száz esetben frissítenünk kell a különböző teszteket. Nem tudunk rugalmas megoldást kínálni. Akkor tud hasznos lenni ez a fajta automatizálás, ha gyorsan kell megoldást találnunk egy adott tesztelésre, és nem, vagy kevésbé fontos a karbantarthatóság. Abban az esetben is megfelelő megoldás tud lenni a lineáris tesztkészítés, ha nem tervezzük a teszteket hosszútávon használni, hanem ugyanazokat a lépéseket kell végrehajtanunk nagy mennyiségben akár egyszeri alkalommal. Erre is egy egyszerű példa: Tegyük fel, hogy egy bizonyos internetes linket kell a böngészőnkben sokszor (akár ezerszer) meghívni, mert az alkalmazás mögötti funkcionalitás azt írja elő, hogy ilyen esetben egy adatbázisban rögzíteni kell a link meghívásának idejét. Itt ugyan azt az egy linket használjuk, de nagyon sokszor hívjuk meg. Pár soros kódrészlettel meg lehet oldani ezt a feladatot.

Mint látjuk, a lineáris tesztkészítés nem egy bonyolult típus. Előnyként említhető, hogy kevés tréning, betanulási időszak kell azoknak, akik ilyen „eszközt" használnak, vagy ilyen tesztet készítenek. Igazából több hátrányról tudunk beszélni, mint inkább előnyről. A tesztelés alatt lévő alkalmazásnak stabilnak kell lennie. Mondjuk ez igaz a soron következő többi generációra is, de a lineáris típus sokkal érzékenyebb a változtatásokra. A karbantarthatóság sokkal bonyolultabb és időigényesebb. Rövidebb a teszt életciklus. Bármilyen változtatás történik az alkalmazásban, a teszteket frissíteni kell. Ez azzal járhat, hogy több idő szükséges megérteni a jelenlegi tesztünk működését, minthogy elölről felvennénk, vagy elkészítenénk a tesztet és azt használnánk a későbbiekben. Végül, mivel nem tartalmaz logikát a teszt, így nagyobb a valószínűsége a tesztek hibára futásának. [2]

Második generáció – Modularizált tesztek

Az első generáció továbbfejlesztett változata a modularizált tesztek. Itt is jelen van a felvétel és visszajátszás funkció, viszont arra koncentrálunk, hogy ne kelljen többször ugyanazokat a hasonló lépéseket rögzítenünk a különböző tesztesetekre. A felvétellel sikeresen tudunk egy vázat rögzíteni, ami mentén teszteljük az alkalmazást. Amikor készen van, akkor a tesztautomatizáló megpróbál különböző feltételeket, ciklusokat implementálni a tesztbe annak érdekében, hogy sokkal hatékonyabban lehessen használni. Nagyobb szakmai tudást igényel ez a fajta tesztkészítés, hiszen fel kell tudni ismerni, hogy milyen lépéseket, milyen blokkokat lehet és kell különálló egységbe rakni annak érdekében, hogy később fel tudjuk használni a részeket más-más tesztesetek készítésére. Ezeket képzeljük el egyszerű eljárásoknak, vagy függvényeknek. A különböző eljárás- és függvényhívásokkal már rugalmasabb tesztkészítést kapunk. Karbantarthatóság terén is sokkal hatékonyabb, mint a lineáris tesztkészítés.

Nézzük meg az előző webes űrlap példát. Amennyiben van száz tesztesetünk, és mindegyiknek a végén ugyanazt az értéket kell leellenőriznünk, például „Köszönjük az érdeklődést!", akkor ezt az ellenőrzési pontot egy különálló modulként, függvényként definiálva minden tesztünk végén meg tudjuk hívni.

Azaz:

```
Input "E-Mail" into email textbox
```

```
Input “Tárgy” into targy textbox
Input “Üzenet” into uzenet textbox
Click Küldés button
Call Ellenorzes()

Function Ellenorzes()
     If “Köszönjük az érdeklődést!” exists then
     Ellenorzes = Passed
Else
     Ellenorzes = Failed
End If
```

Mint látjuk, az ellenőrző feltétel különálló részként szerepel. Ezt a fajta tesztkészítést még egy dologgal tudjuk kicsivel hatékonyabbá tenni. A függvényben megadott szöveget tegyük paraméterként a függvény hívásába. Mi történik akkor, ha például nem adunk meg tárgyat az űrlapon? Elképzelhető, hogy teljesen más üzenet fog megjelenni. Ebben az esetben, ha paraméterként adjuk meg az ellenőrizendő szöveget, akkor megvan a tesztfuttatónak, tesztelőnek a lehetősége, hogy bizonyos teszteseteknél más értékekkel ellenőrizze az eredményt. Itt érkeztünk el a harmadik generációhoz.

Harmadik generáció – Adatvezérelt automatizálás

A következő szint az adatvezérelt automatizálás. Ez a típus is a lineáris automatizálásból fejlődött ki, megtalálható benne az első és a második generáció felépítése és tulajdonsága. Ha jobban megfigyeljük, látható, hogy a magasabb generációk, automatizálási szintek magukban hordozzák az előzőek tulajdonságait. Abban mutatkozik meg ennek a típusnak az előnye, hogy tesztvégrehajtás során, az adatokat, melyeket használunk, egy külön állományban tároljuk. Legyen ez egy szöveges állomány, dokumentum, táblázat stb. Számos szakmai publikáció, internetes forrás karbantarthatóság szempontjából a legjobbnak említi ezt a típusú automatizálást. Saját tapasztalatom alapján is azt kell mondanom, hogy a lehető legrugalmasabban, legkényelmesebben használható típusról beszélünk. Mivel a tesztadatokat egy külön állományban tároljuk, a fenntarthatóság szempontjából igen rugalmas megoldásnak tekinthető. Képzeljük el, hogy a tesztautomatizálással foglalkozó szakembernek csak a tesztek karbantartására kell ügyelnie. A tesztadatok minőségének biztosítására itt be lehet vonni a tesztelőket, programozókat, üzleti elemzőket. A tesztadat nem része az automatikus tesztünknek, így azt függetlenül tudjuk kezelni. Előnye, hogy a standardizálás felé tudunk haladni, és az automatikus tesztjeinket „mentesítjük” a beégetett adatoktól.

Nézzük meg az előző példát adatvezérelt módszerrel.

```
Open Data Table
Input < UserEmailAddress > into email textbox
Input < UserSubject > into targy textbox
Input < UserGivenMessage > into uzenet textbox
Click Küldés button
Call Ellenorzes(< ExpectedResult >)
Function Ellenorzes(result)
      If result exists then
      Ellenorzes = Passed
Else
      Ellenorzes = Failed
End If
Close Data Table
```

UserEmailAddress	UserSubject	UserGivenMessage	ExpectedResult
teszt.user1@yahoo.com	Érdeklődés termék iránt	Tesz üzenet 1.	Köszönjük!
teszt.user2@yahoo.com	Érdeklődés termék iránt	Tesz üzenet 2.	Köszönjük!

1. táblázat Adatvezérelt automatizálás adattáblája

Mint látjuk, valamilyen módszerrel meg kell nyitnunk azt a szöveges állományt, amiben a tesztadatainkat tároljuk. A tesztesetek végrehajtását és a tesztek számát, iterációját az adatállományban szereplő adatok sora fogja mutatni. Az 1. táblázat alapján két darab teszt iteráció fog bekövetkezni. Az automatikus tesztben látjuk, hogy az adattáblában szereplő fejlécek neve szerepel, ez alapján tudja azonosítani a tesztünk, hogy éppen melyik oszlopból és melyik sorból (iteráció) vegye ki a tesztadatot és használja fel a futás során.

Természetesen implementálhatunk különböző funkcionalitásokat is az adatokhoz. Elképzelhető, hogy a vizsgált (tesztelt) alkalmazásunk különböző e-mail címekre más-más üzenetet ad vissza a felhasználói felületre. A megjelenített üzenet ellenőrzésére implementálhatunk elágaztatásokat a tesztünkben, vagy különböző ellenőrző függvényeket hívhatunk meg a megfelelő szöveg validálására. Ezzel máris megtettük az első lépést ahhoz, hogy a tesztjeinket még használhatóbbá tegyük és az eredmények ellenőrzését tekintve rugalmasabb teszteket készítsünk.

Az előző példa ellenőrzés függvényét egészítsük ki egy feltétellel, ami vizsgálja a bemenő paraméter értékét, és annak megfelelően hajt végre ellenőrzést.

```
Call Ellenorzes(< UserGivenMessage >, < ExpectedResult >)

Function Ellenorzes(message, result)
     If message equal 'Teszt uzenet 1.' then
          If result exists then
               Ellenorzes = Passed
          Else
               Ellenorzes = Failed
          Else
     If message equal 'Teszt uzenet 2.' then
          If result exists then
               Ellenorzes = Passed
          Else
               Ellenorzes = Failed
          Else
     End If
```

Figyeljük meg, hogy az ellenőrző függvényünk paraméterlistája megváltozott. Az üzenet és az eredmény is át lett adva a függvénynek.

Negyedik generáció – Kulcsszó vezérelt automatizálás

Az automatizált szoftvertesztelésnek talán a legkomplexebb tesztimplementálási típusa a kulcsszó vezérelt automatizálás. Szintén tartalmazza az előző típusok tulajdonságait, viszont itt azok a funkciók, melyeket végrehajtunk a tesztelt alkalmazáson, kulcsszavakban vannak tárolva. Igazából kulcsszavaknak feleltetjük meg az egyes kódrészleteket. Szokták ezt a fajta típust táblavezérelt automatizálásnak is nevezni, hiszen a kulcsszavak, az adatok és a várt működés táblában vannak tárolva.

Az ilyen tesztek kidolgozása, implementálása kívánja meg a legtöbb időt és sokkal több odafigyelést igényel a tesztek tervezése és dokumentálása, mint az előzőeké. Viszont ez a típus a leginkább alkalmazás-független. Amennyiben a megfelelő folyamatok mentén működtetjük ezt a típust ésa teszteket megfelelően tervezzük meg, valamint a kulcsszók mögött lévő funkcionalitások fejlesztése és karbantartása kifogástalan, akkor el tudjuk érni, hogy a tesztelőket, vagy akár az üzleti elemzőket is bevonjuk már a tesztek készítése során. Mivel nagy mennyiségű különálló modulokkal rendelkezünk, így a teszteseteket előre el tudjuk készíteni a kulcsszavak felhasználásával. Az így összeállt tesztlista jó forrásként szolgál az automatizálással foglalkozó tesztelőnek, viszont minél több modulunk van, minél több apró funkcionalitást kell lefejlesztenünk, annál több időt kell fordítani a karbantartásra is.

A kulcsszavak (Action) alkalmazás specifikus és alkalmazás független funkcionalitással bíró függvényekkel vagy szkriptekkel vannak kapcsolatban.

Nézzük meg az előző webes űrlap példát átalakítva kulcsszó vezérelt automatizálási típusra.

	Objektum	**Akció**	**Érték**	**Helyreállítás**	**Megjegyzés**
Űrlap	email	Beírás	„teszt.user1@yahoo.com”		
Űrlap	targy	Beírás	„Érdeklődés.”		
Űrlap	uzenet	Beírás	„Árajánlatot szeretnék		
Űrlap	kuldes	Kattintás			
Köszönjük		Ellenőrzés	„Köszönjük.”	Stop_Test	

2. táblázat Kulcsszó-vezérelt automatizálás adattáblája

```
Open Keyword File
Execute Data rows to end of file (EOF)
  If <Akció> == Beírás Then
     Call Beírás(Képernyő, Objektum, Érték)
If <Akció> == Kattintás Then
```

```
    Call Kattintás(Képernyő, Objektum)
If <Akció> == Ellenőrzés Then
     Execute Ellenőrzés(Képernyő, Érték)
Implement Exception Handling Routine Based on Pass/Fail Status
End Loop
```

Ebben az esetben a tesztvégrehajtás a táblázat sorai alapján történik meg. Először az első sorban lévő Action alatti értéket vizsgáljuk és hívjuk meg a megfelelő függvényt a megfelelő paraméterekkel. Kérdésként merülhet fel, hogy miért kell megadnunk a képernyő értéket? Ez egy elég egyszerű példa, de gondoljunk bele, ha egy bonyolultabb üzleti folyamaton kell végigmennünk. Elképzelhető, hogy egy gomb, vagy mező az alkalmazás több pontján is megtalálható, így ezt is ellenőriznünk kell, hogy a teszt-végrehajtás során a megfelelő helyen járunk-e a tesztelt szoftvernek. Érdekes megfigyelni a Helyreállítás oszlopot. Bármilyen automatizálási típust használunk, mindig szem előtt kell tartani, vagy inkább úgy fogalmazok, hogy számítani kell arra, hogy valahol nem tudjuk végrehajtani a megfelelő lépést. Ez számos esetben előfordulhat, például valamiért nem úgy működik az alkalmazás, mint ahogy az elvárt lenne. Ebben az esetben úgy kell „felkészíteni" a tesztünket, hogy ezt képes legyen felismerni, és ne szakadjon meg a teszt végrehajtás, hanem lépjen tovább és kezdje meg az új iterációt a programunk. Igazából ennek a fontosságát akkor érezzük meg, ha nagy adathalmazra használunk tesztet. Amennyiben ezer iterációról beszélünk, nagyon pontosan meg kell találnunk, mi az a helyreállítási forgatókönyv, ami után a tesztünk képes tovább futni. A fenti esetben az van megadva, hogy álljon meg a teszt, ne fusson tovább.

A kulcsszó-vezérelt automatizálás nagy előnye, hogy szinte nehézségek nélkül tudunk újrafelhasználni modulokat. Korán tudunk teszteket készíteni, tervezni, hiszen elérhetőek a kulcsszavak, így bárki képes még a tesztelési időszak előtt ezeket felhasználva teszteket gyártani. Könnyen olvashatóak az elkészített tesztek, amelyek végül is táblázatok. A kódok el vannak rejtve. Mivel nagyfokú a modularizáltság, ezért elkerülhetetlen, hogy valamilyen standard mentén fejlesszünk teszteket. Hátrányként említhetjük meg, hogy magas szakmai ismeretre van szükség az automatizálással foglalkozók oldaláról, a tesztek fenntarthatósága komplexebb és időigényesebb és alapos tervezést igényel mind a tesztelők oldaláról, mind pedig az automatizálással foglalkozó szakemberek részéről.

Ötödik generáció – Hibrid automatizálás

Ez a szint a nevéből adódóan tartalmazza az összes előző típust. Ezeket szoktál „okos" keretrendszereknek hívni. Ennél az esetnél arra is törekednek tesztelők, hogy ne nagyon legyenek teszt

szkriptek, inkább megpróbálják az alkalmazás fázisainak a nézeteit használni és azokon keresztül történik a végrehajtás.

2.7. Eszközválasztás, eszköz tulajdonságok

Biztos vagyok benne, hogy aki elkezd tesztautomatizálással foglalkozni, egyből valamilyen programozási nyelvben próbál meg tesztek készítésére megoldást találni. Az iskolai tanulmányok alatt megszerzett tudás alapján - akár több, akár kevesebb programozási nyelv ismeretében is - el lehet kezdeni próbálkozni és siker esetén előbb-utóbb biztos mindenki fejleszteni szeretné az automatizálásban használható tudását. Felmerül a kérdés, hogy léteznek-e olyan eszközök, melyekkel egyszerűbben lehet teszteket készíteni? Netalán létezik, vagy biztosítanak valamilyen kapcsolódó lehetőséget egy keretrendszerhez, amiben hosszabbtávon jobban és hatékonyabban lehet szkripteket készíteni? Elkezdődik a keresgélés a világhálón. Bevallom, nem töltöttem sok időt a keresgéléssel, próbából beírtam a keresőbe, hogy tesztautomatizáló eszközök. Az első találatok között szerepeltek a legnépszerűbb nyílt forráskódú alkalmazások. Nézzük meg pontosabban, hogy melyek is ezek az eszközök.
Automatizált szoftvertesztelés alatt nagyon különböző tevékenységeket érthetünk. Modulltesztelésre használatos eszközök, funkcionalitás tesztelésére szánt eszközök, különböző teljesítménytesztet támogató eszközök és még sorolhatnánk. Mint azt a tananyag elején is említettem, a funkcionális automatizált tesztelés megismertetése a cél, így én ehhez szeretnék egy kis segítséget adni. Próbáljunk meg inkább angol oldalakon keresgélni, és valami hasonló kereséssel induljunk el: „test automation tool”, „functional test automation tool”.

Sokan esnek abba a csapdába, hogy nem vizsgálják meg, mire is szeretnék használni a kiszemelt eszközt. Azaz azt, hogy milyen tesztelés támogatása a cél, milyen programozási nyelvet és milyen alkalmazásokat képes támogatni. Ezalatt azt értem, hogy .NET, Java és még sorolhatnám, milyen nyelvekben implementált alkalmazások tesztelését meg tudjuk-e oldani vagy sem. Amennyiben egy cégnél, vagy akár nagyobb vállalatnál dolgozunk, akkor a vállalatirányítási rendszer támogatása is fontos tényező tud lenni.

Fontos szempont, hogy nyílt forráskódú, vagy kereskedelmi forgalomban lévő eszközt szeretnénk-e választani. Mindegyiknek megvan a maga előnye és hátránya. Célszerű időt szánni a különböző eszközök dokumentációinak átolvasására, mert sok hasznos infót találunk ezekben is. Mivel egy eszköz használatát általában hosszútávra tervezzük, próbáljuk meg elkerülni, hogy a későbbiekben új, számunkra ismeretlen eszköz használatát kelljen elkezdeni. Az sem feltétlenül megoldás, hogy több automatizált tesztelést támogató eszközt használjunk, mert akkor a tesztek karbantartása lesz nehézkes, ugyanakkor olyan szakemberekre lesz szükségünk, akiknek széles a programozói ismerete.

A mai tendenciák alapján célszerű, ha az alábbi tulajdonságokkal rendelkezik egy tesztautomatizálást támogató eszköz, így döntésünknél a következő szempontokat célszerű figyelembe venni.

Felvétel és visszajátszás funkció (Record & Playback). A tesztlépéseinket rögzítse az eszköz, majd azokat visszajátszva hajtsa is végre azokat.
Objektumtárolás: képes legyen a felvétel során eltárolni azokat az objektumokat, melyeket érintünk a teszt végrehajtása során.
Naplózás: tudjon eredményeket tárolni, a pontos lépéseket és az elvégzett műveleteket elmenteni.
Futási idők mérése: lehetnek akár teljes tesztfutási idők, iterációk ideje, esetleg definiált tranzakciók külön mérése.
Szinkronizálás: legyen összhangban a tesztelés alatt álló alkalmazással, ne hajtsa végre a következő lépést, amíg az alkalmazásunk nincs a megfelelő stádiumban. Ezekről bővebben a negyedik fejezetben lesz szó.

Beszéljünk egy kicsit bővebben az eszköz kiválasztási folyamatról. Tegyük fel, hogy meg vannak a szükséges információink és tudásunk a tesztautomatizálás alapjairól, sőt, akár korábban esetleg megnéztük, milyen eszközök léteznek és használtuk is valamelyiket - akár többet is. Az idő múlásával a vállalatnál fejlesztett, vagy már meglévő alkalmazások fejlődésen mentek keresztül, változott a technológia, és időszerű egy olyan eszköz után néznünk, amivel az automatizálás sikeres és fenntartható tud lenni. Amennyiben erre valamilyen keret (pénz) is rendelkezésre áll, úgy igen megfontolandó, hogy milyen eszközre fordítjuk ezt az összeget. Nagyon sok fázison végig kell menni, hogy kiderüljön, pontosan mire van igény.
Tehát az első lépés, hogy definiáljuk, járjuk körül, pontosan mire is van szükségünk. Írjuk le az általános elvárásainkat, ez lehet akár az is, hogy milyen legyen a felhasználói felülete, vagy akár a menüszerkezete a később használni kívánt eszköznek. Ezután szedjük össze, hogy milyen technikai támogatásra van igényünk. Például lehessen benne programozni VBScript-el, Python-al, Java-val és még sorolhatnám. Ezt követően azt is gyűjtsük össze, hogy milyen funkcionalitást várunk el az eszköztől. Ezek lehetnek az előzőekben tárgyalt felvétel és visszajátszás funkciók, naplózás, szinkronizálás, paraméterezhetőség, helyreállítási forgatókönyv definiálás, külső adatforrás használat, integrálhatóság más eszközökkel. Ezek lehetnek tesztmenedzsment eszközök, követelménykezelő eszközök, incidenskezelő eszközök is.

Vizsgáljuk meg, hogy funkcionalitás terén mit szeretnénk elvárni az eszköztől. Mennyire legyen bonyolult a kezelése, mennyire számítunk egyszerű, felhasználóbarát kinézetre. Például legyen adott olyan opció, hogy egy Excel táblázatból be lehessen importálni az előre megírt tesztlépéseket, és azokat kiegészítve már kapjunk is egy használható automatikus tesztet. Megfontolandó, hogy elérhető legyen mások számára is az eszköz, vagy kifejezetten dedikált szakemberek fognak dolgozni ezzel a programmal? Hogyan lehet teszteket futtatni? Az eszközben tudjuk végrehajtani a teszteket, vagy a futtatás teljesen függetlenül fog történni? A naplózás egy beépített funkció, vagy nekünk kell gondoskodni arról, hogy a megfelelő futási eredményeket rögzítsük? Azt is meg kell vizsgálni, hogy vannak-e az eszközben már előre definiált és implementált funkcionalitások? Értem ezalatt, hogy képes-e elindítani bármilyen alkalmazást az operációs rendszerünkön anélkül, hogy azt nekünk kellene lefejleszteni? Lehetőség van-e például XML állományok generálására a tesztfutás közben, vagy végén? Léteznek előre definiált helyreállítási forgatókönyvek, vagy azokat is nekünk kell lefejleszteni? Alapvető esetek lehetnek: amennyiben nem várt esemény következik be a tesztelt alkalmazáson, akkor az automatikus tesztünk lépjen a következő tesztutasításra, vagy kezdje elölről a tesztet ugyanazzal a tesztadattal, vagy kezdje

elölről, de a következő tesztadatot használja. Hiba esetén esetleg zárja be a tesztelt alkalmazást (akár egy böngésző, vagy Windows-os alkalmazás stb.), nyissa meg újra és folytassa a következő teszt iterációt. Vagy a legegyszerűbb megoldás, ha leállítjuk a tesztfuttatást. Megjegyzem, hogy ez nem a legjobb megoldás, hiszen ebben az esetben nem tudjuk végrehajtani a tesztet az előre definiált adatmennyiségre. Személyes véleményem, hogy ezek a funkciók, mármint a helyreállítási forgatókönyvek, nagyon jó, ha az automatizálásra használt eszközben megvannak, de hosszabb távon érdemes egy általunk definiált, széles körben használható megoldást kitalálni. Természetesen a tesztektől függ, hogy mikor mit várunk el egy olyan helyzetben, amikor nem tud végrehajtódni a tesztünk úgy, ahogyan azt elvárnánk.

Miután feltérképeztük az előző pontokat, érdemes megvizsgálni a továbbiak során, hogy milyen körülmények között szeretnénk használni az eszközt. Állítsunk fel pár esetet, hogy miként is képzeljük el a munkát a kiválasztott eszközzel. Például: dedikált szakemberek fognak foglalkozni vele, és más számára nem lesz elérhető az eszköz. Vagy a tesztelő csoport minden tagja használhatja, rögzítheti a teszteseteit benne, viszont a tesztautomatizálási csoport fogja olyan állapotra hozni a teszteket, ahogy az elvárt és a legjobb működést biztosítja hosszabb távon. Gondoljunk arra is, amikor a végrehajtási szakaszban leszünk. Ebben az esetben is a tesztelők fogják futtatni a teszteket, vagy pedig az automatizálási csapat fogja felügyelni a futtatásokat, és kezelni az éppen felmerülő igényeket? Ezekre érdemes később egy definiált folyamatot kiépíteni, hogy mindenki tudja, mikor mire lehet számítani, és pontosan kitől.

Érdemes beszélni arról is, hogy mekkora forrás van az eszközre. Próbáljunk meg egy minimális, egy közepes és egy maximálisköltséget is lefektetni, mert később a döntésben ez is fontos szerepet fog kapni.

Definiálnunk kell, hogy ki fogja használni az eszközt. Az előző pontokban példákat soroltam fel, viszont célszerű megvizsgálni, hogy kiknek lesz elérhető az eszköz. Az automatizált szoftvertesztelés egy igen költséges befektetés, ami hosszabb távon, ha megfelelő folyamatok mentén működtetjük, megtérülést fog eredményezni. Mérlegelnünk kell, hogy érdemes „megadni” a lehetőséget minden tesztelőnknek, hogy rögzítsen és futtasson teszteket, vagy inkább tényleg azoknak lesz meg a lehetősége használni az eszközt, akik professzionális szinten értenek hozzá. Az automatizálás sikerességének eredményei nem biztos, hogy akkor lesznek a legjobbak, ha mindenki automatizál, ahogy korábban említettem, technikai tudás szükséges, hogy valaki megfelelően tudjon teszteket létrehozni. Teljesen más gondolkozásmód szükséges, és nem biztos, hogy egy manuális tesztelő olyan hatékonysággal fogja tudni megcsinálni, leimplementálni a tesztet, mint ahogy azt egy szakember tenné.

Vizsgáljuk meg azt is, hogy mennyire fontos egy automatizálásra használt eszköznek a beszerzése. Amennyiben egy-két hónapon belül már szükséges lenne egy ilyen eszköz, akkor nagyon gyorsan és céltudatosan kell végigmennünk az eszköz-kiválasztási folyamaton. Természetesen, ha több időnk van erre, akkor alaposabban, jobban körbe tudjuk járni a folyamatot.

Meg kell fogalmaznunk, hogy egy ilyen eszköz, valamint a hozzá tartozó folyamatok bevezetése milyen előnyökkel fog járni. Milyen pozitív hatást eredményez ez a vállalatnak? Melyek azok a jelenlegi pontok,

folyamatok, amelyeken javítani tudunk egy ilyen eszköz beszerzésével? Hol tudunk költséget csökkenteni, és ez nem azt jelenti, hogy manuális tesztelői erőforrások fognak megszűnni. Próbáljunk meg arra rávilágítani, hogy a tesztek futtatásából adódó időnyereség hogyan fog jelentkezni. Nem a tesztelő idejét fogjuk használni, hanem egy automatikus tesztet, ami gyorsabban, nagyobb adatmennyiségre, rövidebb idő alatt fog eredményeket felmutatni. Gondoljunk arra is, hogy ezáltal rendszeresebben, kontrolláltabban lehet teszteket végrehajtani.

Amennyiben ezekkel megvagyunk, szedjük össze, hogy milyen eszközök léteznek. Miket tudunk elérni. Nézzük meg a piacot, hogy a különböző gyártók milyen megoldásokat kínálnak. Gyűjtsük össze az eszközök árait, elérhető funkcionalitásaikat, működési tulajdonságaikat. Vizsgáljuk meg, hogy hosszabb távon mennyire lesz fenntartható a különböző eszközök használata és gyűjtsük össze, hogy milyen alternatíváink vannak.

Ha felmértük a rendelkezésre álló eszközöket, válasszuk ki azokat, melyek számunkra a legjobb megoldásokat nyújtják. Kezdjük el kipróbálni az eszközöket. Amennyiben ingyenes, nyílt forráskódú eszközöket használunk, egyszerűbb a dolgunk. Manapság a kereskedelmi forgalomban kapható eszközök rendelkeznek bizonyos időkorlátos próba verzióval, szerezzük be ezeket a verziókat, és próbáljuk meg használni őket, de szigorúan tartsuk szem előtt azokat a pontokat, melyeket összegyűjtöttünk korábban. Legyünk elfogulatlanok a különböző eszközök használata során, így például ne legyen szempont az, hogy az egyik jobban néz ki a másiknál. Nagyon határozottan vizsgáljuk meg az eszközöket és a funkcionalitásokat, arra összpontosítsunk, amit adni tudnak, és amire képesek.

A próbaverziók használata során folyamatosan dokumentáljuk, milyen eredményeink voltak, mit tudtunk elérni az eszközzel, milyen előnyöket és hátrányokat tapasztaltunk. Készítsünk egy beszámolót ezekről az eredményekről, értékeljünk, sorakoztassuk fel az összes pozitív és negatív tapasztalatot. Ezek mentén pedig készítsünk egy javaslatot. Adjunk egyértelmű és pontos információt, ami alapján a vezetőség képes dönteni az eszköz beszerzéséről. Ennek tartalmaznia kell minden olyan információt, ami érinti a jelenlegi folyamatainkat és természetesen a költségeket is.

Ezek után elkezdődhet a kiválasztási folyamat. Mint látjuk, nagyon sok lépésen kell keresztülmenni, amíg eljutunk a tényleges eszközkiválasztásig. A következő folyamat legelső pontja az eszköz ára lehet. Amennyiben drága, és nem tudunk többet szánni az eszközre, mint ami be volt tervezve, sajnos lehet, hogy el kell vetnünk, akármennyire is jó. Az ár – érték – használhatóság figyelembevételével tudunk megfelelő döntést hozni. Amennyiben több eszköz is van a listánkon, még további vizsgálatokat kell végeznünk. Milyen hatása lesz a jelenlegi folyamatokra? Hol kell módosítani, hogy a lehető legjobban tudjuk használni az eszközt? Ez a módosítás milyen mértékben érinti a tesztelőket, üzleti elemzőket, programozókat? Mennyi idő szükséges a folyamatok változtatásához? Elérhetőek lesznek oktatási anyagok, vagy nekünk kell elkészíteni azokat? Mennyi időbe fog telni az oktatás, és mit várunk el attól? Mennyire lesz fenntartható, karbantartható az eszköz és a folyamat? Milyen minőséget fog eredményezni egy ilyen eszköz? Fognak változni a jelenlegi tesztelési folyamataink minősége ezáltal vagy sem? Hogyan fogjuk mérni azt, hogy ez a minőségjavulás tényleg bekövetkezik?

Ezek mentén el kell jutnunk arra a pontra, hogy megtaláljuk a megfelelő eszközt. Ezzel el is érkeztünk az utolsó fázishoz, amikoris meg kell vásárolnunk az eszközt. Fel kell vennünk a kapcsolatot a gyártóval, tárgyalásokat kell folytatnunk, meg kell vizsgálni, hogy milyen a licenszelés és mit takar az ár. Meg kell néznünk, hogy a gyártó milyen szolgáltatást nyújt, milyen fizetési konstrukció létezik, hiszen vannak olyan gyártók, akik havi szinten kérik az összeget, vannak, akik éves szintre fókuszálnak. A tárgyalások végén meg kell győződnünk arról, hogy az ajánlat a legmegfelelőbb számunkra, ezután lehetséges a tényleges árajánlat kérése, majd a megrendelés véglegesítése. Sikeresen megtaláltuk és beszereztük az eszközt!

3. Automatizált szoftvertesztelés megjelenése

3.1. Milyen dolgok indukálják az automatikus tesztelés megjelenését egy vállalatnál?

Szoftvertesztelőként biztos vagyok benne, hogy a manuális tesztelő munkája során előbb-utóbb valamilyen formában találkozni fog az automatizálással. Elképzelhető, hogy egy új munkakörnyezetben tesztelésre már használnak valamilyen automatikus megoldást, amennyiben viszont nincsenek még jelen a megfelelő eszközök és folyamatok, úgy lehetséges, hogy a manuális tesztelőt kezdi el foglalkoztatni valamilyen tesztelési folyamat automatizálása.

A mai világban hihetetlenül felgyorsult az alkalmazások fejlesztése. A különböző üzleti igények, valamint ezen igények fontossága az üzlet és a felhasználók szemszögéből indukálják a rövid határidőkkel ellátott szoftverkészítési folyamatokat. A tesztelésért felelős csoport megpróbál minél hatékonyabban meggyőződni arról, hogy a legyártott szoftver úgy és olyan módon működik, ahogy az a követelményeknek kell. Minél több alkalmazással rendelkezik a vállalatunk, esetleg létezik egy komplett vállalatirányítási rendszer is, annál több tesztesetet kell végrehajtani minden egyes tesztelési ciklusban. Amennyiben képesek vagyunk detektálni azokat a teszteseteket, melyeket minden egyes tesztelési ciklusban végre kell hajtanunk, és ezeket gép által is lehetséges letesztelni, akkor közelebb kerültünk ahhoz, hogy az értékes és kevésbé unalmas tesztelési feladatokat levegyük a tesztelők válláról.

Egy cégnél, vállalatnál, esetleg kisebb vállalkozásnál, ahol informatikai fejlesztésekkel foglalkoznak, elkerülhetetlen hogy a legyártott alkalmazásokat ne teszteljék valamilyen módon. Jobb esetben erre léteznek tesztelők és nem a fejlesztőknek kell meggyőződniük a programkódok helyes működéséről, vagy esetleg megbizonyosodni az üzleti folyamatok elvárt működéséről.

Előfordulhat, hogy nem tudatos módon kezd el működni egy automatikus tesztelési folyamat. Amennyiben a tesztelő rendelkezik megfelelő informatikai tudással, netalán programozói tapasztalattal, úgy saját magának - a tesztelési folyamatai jobbá tétele céljából - elkezd rövid programkódokat alkalmazni tesztelési feladatai során. Vegyünk példának egy webáruházat. Tegyük fel, hogy olyan módosítás történik a felhasználói felületen, ami azokat a regisztrált felhasználókat fogja érinteni, akik Magyarországról rendelnek. Ebben az esetben minden bizonnyal kell, hogy legyenek olyan tesztesetek, melyek magyar felhasználói profillal fognak végrehajtódni. A tesztelő ezért elképzelhető, hogy a meglévő rendszerben tesztadat-keresésre valamilyen adatbázis lekérdező megoldással keres adatot a teszteléseihez. Itt már a számítógép lehetőségeit kezdi el használni (gyorsaság, pontosság), nem pedig egyesével végignézi a rekordokat az adatbázis táblákban. Előfordulhat az is, hogy új felhasználói profil létrehozására, valamilyen programozási nyelvben egy rövid kódot készít, ami egy szöveges állományból olvassa ki a megfelelő adatokat és illeszti be a webáruház regisztrációs felületén található mezőkbe. Itt a mezők kitöltéséből adódó monoton munkát helyettesíti gépi végrehajtással.

Számos ilyen és ehhez hasonló eset létezhet, amikor nem tudatosan beszélünk a tesztautomatizálás használatáról. Ahol már hosszabb távon jelen van a tesztelés, vagy a feladatokat egy tapasztalt vezető végzi, akinek van automatizálással kapcsolatban tapasztalata, úgy biztos, hogy tudatosan meg fog jelenni az automatizálás iránti igény. A rohamosan fejlődő informatikai rendszerekkel és a rendszerekre épülő

üzleti megoldásokkal nehezen lehet lépést tartani. Elképzelhető, hogy a tesztelő csoport mérete nem tud olyan mértékben növekedni, mint ahogy azt az üzlet megkívánná. A folyamatos fejlesztések mellett meg kell győződnünk arról, hogy a regressziós tesztelési feladatok nem sérülnek, és folyamatosan végrehajtjuk őket. Ezek időről időre problémát okozhatnak, hiszen a tesztelő nem tud olyan mértékben foglalkozni az új funkcionalitások tesztelésével, mint amilyen mértékben kellene, vagy szeretne. Ez egy jó lehetőség az automatizálás használatára, hiszen a jól detektált regressziós tesztek viszonylag könnyen automatizálhatóak (természetesen ez függ attól, hogy milyen komplex a folyamat).

Gondoljunk abba bele, hogy mennyi tesztelést hajtunk végre egy tesztidőszakban. Hányszor van egy évben olyan tesztelési időszak, amikor ugyanazokat a teszteket kell végrehajtani. Mennyi új funkcionalitást kell letesztelnünk időről időre, ami nem volt jelen az előző tesztciklusban. Ideális esetben úgy kellene, hogy működjön a tesztelés, hogy az adott időszakban jelen lévő új funkcionalitás a következő tesztciklusban, mint meglévő (regressziós) funkcionalitás lesz jelen. Tehát időről időre, az új kódrészletek és funkciók átalakulnak regressziós esetekre. Folyamatosan nő a tesztesetek száma, viszont a manuális tesztelői erőforrásainkat nem vagyunk képesek és nem is lehet olyan mértékben növelni, amivel már ezt a mennyiségű tesztesetet le lehetne kezelni. Elérkezik az a pont, amikor felmerül az igény a gép által végrehajtott tesztek iránt. Azonban tartsuk szem előtt, hogy amilyen egyszerűnek is tűnik elkezdeni, olyan nehéz valójában. A következő fejezetek segítenek abban, hogy mire figyeljünk oda a tesztautomatizálás bevezetésekor, használatakor.

3.2. Mielőtt belevágunk, mit tegyünk (mérjük fel, hogy kell nekünk vagy sem)?

Az előző példákon látszik, hogy az automatizálás nagy segítséget tud nyújtani. Viszont ahhoz, hogy ez milyen mértékű legyen, és milyen módon tudjuk kamatoztatni, fel kell mérnünk jelenlegi tesztelési folyamatainkat. Vizsgáljuk meg, hogy melyek azokat a pontok a jelenlegi alkalmazásainkban, informatikai eszközeinkben (weboldal, vállalatirányítási rendszer stb.) melyeket célszerű lenne könnyen és hatékonyan automatizálni. Melyek azok, melyeket sokszor hajtunk végre, melyek kritikusak és viszonylag sok manuális tesztelési munkát igényelnek? Vizsgálati szempont lehet az is, hogy melyek azok a területek, ahol kevés emberi erőforrással rendelkezünk. Sok esetben az automatizálás segítséget nyújt erőforráshiány kiküszöbölésére is, bár itt jegyezzük meg, hogy a manuális tesztelést nem tudja az automatizálás teljes mértékben helyettesíteni.

Automatizált szoftvertesztelés használatával szinte elengedhetetlen, hogy felülvizsgáljuk a meglévő tesztelési folyamatainkat. Akármennyire is jól működnek az aktuális tesztciklusok, automatizálást használva nem biztos, hogy a legnagyobb segítséget és hasznot tudjuk előteremteni, ha a folyamataink változatlanok maradnak. Kisebb-(olykor) nagyobb változtatások nélkülözhetetlenek annak érdekében, hogy hatékonyan tudjuk alkalmazni tesztjeinket. Elképzelhető, hogy egy több hosszú folyamat feldarabolható és manuális valamint automatikus tesztvégrehajtásokat tudunk implementálni. Nem az a cél, hogy mindent teljes egészében leautomatizáljunk! Még ha meg is lehetne csinálni, sokszor értelmetlen, mert valamit a manuális tesztelő pillanatok alatt képes felismerni, sokkal jobban, mint például egy automatikus teszt. Egy egyszerű példa: egy weboldal megjelenítése, elrendezése. Gép által is

valószínűleg meg tudnánk csinálni az ellenőrzést, viszont a teszt elkészítésére sok munkaórát kellene ráfordítani. Egy tapasztalt tesztelő ezt gyorsan képes leellenőrizni.

Fontos megvizsgálni a jelenlegi tesztelőink képességeit is. Automatizált szoftverteszteléssel foglalkozó tesztelők teljesen más képességekkel rendelkeznek, mint a manuális tesztelők. Más gondolkozásmód szükséges, hiszen az egyes tesztek leprogramozása nem úgy történik, mint ahogy egy tesztesetet leírunk. Annak érdekében, hogy sikeresen tudjunk automatizálással foglalkozni, elengedhetetlenek olyan tesztelők, akiknek van kódolási jártasságuk. Képesnek kell lenniük rendszerszinten gondolkozni, előre jelezni, hogy melyek lehetnek a tesztkészítés fázisán belül a nehéz pontok. Tudniuk kell együtt dolgozniuk akár a fejlesztőkkel is. Mivel technikai tudásuk is van, így számos kérdést fel tudnak tenni, ami közelebb viheti őket az automatizált teszt elkészítéséhez.

Amennyiben a jelenlegi csoportunkban nincsenek olyan csapattagok, akiknek van technikai tudásuk, el kell gondolkozni azon, hogy megpróbáljuk bővíteni a csapatunkat technikai szakemberekkel vagy sem. Az is megoldás lehet, ha a meglévő csapattagok között az érdeklődőket elkezdjük fejleszteni ezen a területen. Nehéz dolog, de tudnunk kell megbecsülni, hogy mennyi emberi erőforrásra lesz szükségünk ahhoz, hogy az automatizált szoftvertesztelés sikeresen tudjon működni. Erre egy jó gyakorlati példa, ha egy pilot projektet vezetünk be, aminek keretein belül megpróbáljuk felmérni, hogy milyen mértékben hasznos számunkra az automatizálás, és hogy miként kell a jövőbeni folyamatokat változtatni.

3.3. Pilot bevezetése és kivitelezése

Annak érdekben, hogy a lehető legtisztább képet kapjuk arról, hogy érdemes tesztautomatizálással foglalkoznunk, vagy sem, dedikált időt kell szánnunk a pilot projektre. Ki kell választanunk azokat a tapasztalt tesztelőket, akik ezzel fognak foglalkozni. Amennyiben számunkra teljesen új az automatizált tesztelés fogalma, akár egy évig is eltarthat egy ilyen projekt. Természetesen ez függ attól, hogy mennyi időt tudunk szánni egyéb munkáink mellett a pilot feladataira, és hogy mi is a célja egy ilyen projektnek. Azért, hogy zökkenőmentesen tudjon haladni a projekt, célszerű informálni a környezetünket is arról, hogy bizonyos számú tesztelő egy új, idáig nem használt folyamatot szeretne kipróbálni, kiépíteni. Fontos információ lehet ez a fejlesztő kollégáknak, a csapatunk többi tagjának, társterületek vezetőinek, esetleg a felső vezetésnek is. Tisztázni kell, hogy a tesztautomatizálásra dedikált emberi erőforrás mennyi időben fog foglalkozni a projekttel, hiszen az egyéb manuális tesztelési feladatokat is el kell látni, vagy azokból át kell adni más csapattagoknak. Tartsuk szem előtt, hogy a projekt előrehaladásával számos nehézségekbe fogunk ütközni. Ebből adódóan elképzelhető, hogy egy tesztelési ciklusban a tesztek végrehajtási ideje növekedni fog, hiszen ezekben az időszakokban tudjuk a legjobban ellenőrizni a pilot alatt készített scriptek működését. Ilyenkor tudunk javítani a teszteken, netalán elkészíteni is ilyenkor tudjuk őket (számos esetben a tesztkörnyezet elérhetőségének hiányából adódik ez).

A pilot céljainak kitűzése mellett fontos szempont az is, hogy a tesztjeinket hol és hogyan fogjuk elkészíteni. Mivel automatizált tesztelésről beszélünk, azaz gép által végrehajtott tesztekről, így hardver erőforrásra is kell gondolnunk. Elsőként lehet használni a tesztelő munkaállomását erre. Fontos kérdés,

hogy automatizált tesztelésre alkalmas eszközt tervezünk használni, vagy saját magunk fogunk valamilyen programozási nyelvben teszteket készíteni.
Fontosnak tartom megjegyezni, hogy már a kezdetektől gondolkozzunk szeparált munkaállomások kiépítésén. A sikeres pilot után a megnövekedett tesztek végrehajtása előbb vagy utóbb dedikált hardvert fognak igényelni.

Mi lehet a célja egy pilot projektnek?
Ez nagyban függ az adott üzleti területtől, aminek a folyamatain belül szeretnénk kipróbálni az automatizálást. Függhet attól, hogy milyen alkalmazásokat tesztelnek az adott területen. Nem lehet egy általános érvényű célt definiálni, hiszen mindig az adott időszak, elvárások, munka fogja megszabni, hogy mi mentén felügyeljük a tesztelést.

Általános cél: meggyőződni arról, hogy használható a tesztautomatizálás, és használatával javítunk jelenlegi tesztelési folyamatainkon.

Pár terület, melyek a feltérképezése elég sűrűn szerepel egy pilot alatt:

- Próbáljuk meg feltérképezni a kritikus folyamatokat és az, ahol viszonylag sok tesztesetet hajtunk végre.
- Nézzük meg, hol tart sok ideig egy-egy tesztfolyamat végrehajtása.
- Vizsgáljuk meg, melyek azok a pontok a rendszereinkben, amelyek stabilan működnek és nem fognak változni az elkövetkező egy évben.
- Nézzünk utána, melyek azok a tesztesetek, amelyek ugyanolyan lépéseken mennek keresztül, csak különböző tesztadatokkal.
- Járjunk utána, hol végzünk olyan folyamatot, ami nem szorosan teszteléssel kapcsolatos, inkább adatgenerálás céljából csináljuk.
- Tegyük fel a kérdést, hogy melyek azok a tesztek, amelyekre nagy adatmennyiséget használunk.

Célszerű ilyen és ehhez hasonló célokat kitűzni a pilot idejére, hiszen ezek mentén egyértelműen tudjuk felügyelni, hogy sikeresen tudjuk kivitelezni az automatizálást vagy sem. Amint sikerül elég határozottan körvonalazni, hogy mivel fog foglalkozni a csapat, milyen időintervallumban, milyen munkatempóban hasznos egy megbeszélést tartani az érintett területeken dolgozó tesztelőkkel, vezetőkkel, nekik is egyértelmű lesz, hogy miért is lenne jó automatizált teszteléssel foglalkozni. Javaslom, hogy a felső vezetést is vonjuk be, hiszen a pilot lényege, hogy meggyőződjünk arról, a tesztelési folyamatainkban sikeresen tudjuk alkalmazni a tesztautomatizálást. Meg kell győzni a vezetőket, hogy az automatizálással emberi erőforrásokat tudunk spórolni, hatékonyabban mennek a tesztciklusok, valamint hosszabb távon ez egy befektetés, amibe invesztálni kell és ehhez az ő döntésük is kell. Meg kell tudnunk győzni a környezetünket arról, hogy a kontrollált körülmények között végzett automatizált tesztelés hatékonyabbá tudja tenni a meglévő tesztelési folyamatainkat.

A pilot céljai között még szerepelhetnek az alábbiak is:

- Részletesebben megismerni a használni kívánt eszközt, vagy programozási nyelvet.

- Megvizsgálni, hogy az eszköz, vagy a kiépített keretrendszer miként illeszkedik be meglévő rendszereinkbe.
- Javaslatot tenni a hosszabbtávú standardizálás kivitelezésére.

A következő részben nézzük meg, hogy mire figyeljünk oda a pilot alatt annak érdekében, hogy ne csússzunk el a céljainktól.

3.4. Mit tegyünk a pilot alatt?

Mindenképpen kezdjük el dokumentálni már a projekt elején, hogy mikor min dolgoztunk. Legyen egy külön dokumentáció arról is, hogy milyen célokat tűztünk ki, mikor milyen módon próbálkoztunk és foglalkoztunk az automatizálással, és hogy milyen eredményeket értünk el. Célszerű egy, vagy több tesztelési ciklust is tartalmaznia a pilot projektnek, hiszen ezekben az időszakokban tudjuk meg, hogy milyen sikereink, vagy kudarcaink voltak az elkészült tesztek futtatása során. Rendszeresen tartsunk megbeszéléseket. A projektvezetőnek folyamatosan kontrollálni kell, hogy ki mikor mivel foglalkozott. Abban az esetben, ha valaki problémába ütközik (legyen ez technikai probléma, tesztvégrehajtási nehézségek, tesztrendszer környezetében lévő hibák stb.), vitassuk meg a csoporttal közösen, hogy pontosan miről szól az adott dolog. Nagyon hasznos lehet, ha közösen próbáljuk meg felderíteni, hogy milyen megoldások lehetségesek az adott hibára. Hosszútávon a különböző megoldások nagyban segíthetnek a csapat tagjainak gyorsan és hatékonyan reagálni a felmerülő nehézségekre.

Általában egy pilot alatt kiderül az is, hogy a választott automatizálási eszközünk tényleg a legjobb megoldásokat nyújtja-e számunkra vagy sem. Egy pilot alatt kevésbé probléma, ha sikerül felismernünk, hogy nem a megfelelő eszközt kezdtük használni. Ilyenkor még egyszerű és kevésbé költséges más automatizálást támogató eszközt keresni és kipróbálni.
Elképzelhető, hogy automatizálást támogató eszköz nélkül is sikeres a pilot. Tapasztalhatjuk azt is, hogy nem feltétlen a kereskedelmi forgalomban kapható eszközökkel tudjuk a legjobb eredményeket elérni, hanem nyílt forráskódú lehetőségek és más programozási nyelvek használatával.

Ne felejtsük el, hogy vezessük az automatizálással eltöltött munkaóráink számát. Ide tartoznak például az alábbiak:

- Programozási nyelv megtanulása,
- automatizált eszköz megismerése és betanulása,
- megbeszélésekkel eltöltött idő, melyeket akár fejlesztőkkel vagy tesztelőkkel töltöttünk,
- tesztfolyamatok dokumentálása,
- tesztfuttatási idők,
- tesztek előkészítésének ideje,
- tesztek implementálásának ideje,
- tesztek debuggolásának ideje.

Próbáljuk meg felmérni és kideríteni a megfelelő kommunikációs folyamatot. Megfelelő körülmények között tartsunk rendszeres státusz megbeszéléseket attól függően, hogy milyen hosszú a pilot projektünk, és hogy milyen elvárásaik vannak a kollégáinknak, vezetőinknek az eredmények bemutatásáról. Adjunk visszajelzést a projekt aktuális fázisáról, mit sikerült megoldanunk, milyen sikereink voltak. Mutassuk meg, melyek azok a pontok, ahol nehezen vagy egyáltalán nem lehetett megoldást találni egy automatizálási problémára. Szemléltessük a fenti pontokban gyűjtött eredményeinket táblázatos, vagy grafikonos formában, amelyek lehetnek munkaórák, vagy ha már vannak megtérüléseink, akkor azokat is. Próbáljunk meg a jövőre nézve javaslatot tenni, merre haladjon a projekt, hogy a végén sikerrel zárjuk le azt.
Amennyiben már vannak használható tesztjeink, próbáljuk meg a leghatékonyabban integrálni azokat a meglévő tesztelési folyamatainkba. Ezzel próbáljuk meg elérni és szemléltetni már a pilot alatt, hogy törekszünk a hatékonyságnövelés felé.

Közel a pilot végéhez kezdjünk el dolgozni egy dokumentumon, ami a pontos eredményeinket fogja tartalmazni. A következő részben nézzük meg, hogy miket célszerű belefoglalni egy ilyen dokumentumba, és milyen szempontok szerint értékeljünk.

3.5. Pilot végén hogyan értékeljünk?

A pilot projektünk végén alaposan és körültekintően gyűjtsük össze az eredményeinket egy dokumentum formájában. Készüljünk fel egy projektzáró megbeszélésre, ahová az összes érintett terület tesztelőit, vezetőit célszerű meghívni. Vezetők jelenléte azért fontos, mert hosszabb távon elkerülhetetlen lesz az ő bevonásuk a tesztautomatizálás sikerességéhez.

Elevenítsük fel, hogy mi volt a projekt elején a célunk. Milyen pontokat tűztünk ki, melyek mentén vizsgáltuk és követtük az automatizálás sikerességét. Jegyezzük fel, hogy miket sikerült elérnünk, melyek voltak azok a célkitűzések, melyek csak részben teljesültek, és melyek, amelyek egyáltalán nem.
Ahol kudarcot vallottunk ne érezzük csalódásnak! Ezek a pontok azok, ahol rá lehet mutatni, hogy az adott szervezeten belül mik a hiányosságok, vagy fejlesztési lehetőségek.

Melyek lehetnek ezek? Például:

- Elképzelhető, hogy valamit azért nem tudtunk automatizálni, mert nem volt megfelelő tesztelési folyamat dokumentálva.
- Elképzelhető, hogy a fejlesztéseink nem úgy zajlanak, ahogy egy automatizálási tesztelés megkövetelné.
- Lehetséges, hogy nincsenek meg azok a tapasztalatok, melyek szükségesek egy adott alkalmazás lefedésére.
- És természetesen az is lehet, hogy nem megfelelő automatizálási eszközt, vagy stratégiát kezdtünk használni. Ebben az esetben célszerű lehet egy újabb pilotot csinálni, de ezeknek az információknak a tudatában már másképpen tudunk hozzáállni.

Mutassuk meg, ha lehet számadatokkal:

- Hol, mikor, mennyi erőforrást tudtunk spórolni?
- Mennyi időre volt szükségünk tesztimplementálásra?
- Az elkészült teszt végrehajtás során mennyi ideig futott?
- Vessük össze a futási eredményt a korábbi manuális teszt végrehajtási idővel,
- ezek fényében tegyünk javaslatot, hogy az elkövetkező periódusban (hónapok, fél év, egy év) a tesztfuttatások, tesztelési ciklusok, projektek alatt mennyi nyereséget fog hozni egy adott szkript.
- Nézzük meg, hogy mennyi tesztet és mennyi ideig használtunk akár hétköznapi munkán kívüli időszakban, netalán hétvégeken.

Említsük meg, hogy milyen változást hozott az automatizálás a manuális tesztelők számára! Ez érdekesen hangozhat, de érdemes egy megbeszélés keretén belül összeülnie az automatizáló kollégáknak és a manuális tesztelőknek. Alakuljon ki egy olyan beszélgetés, ahol nyíltan lehet beszélni a pilot alatt tapasztaltakról.

Nekünk, automatizáló szakembereknek fontos szem előtt tartanunk az alábbi pontokat:

- Mérjük fel, hogy miben nyújt segítséget az automatizálás.
- Vizsgáljuk meg, hogy miben nem volt hatékony az automatikus tesztelés.
- Nézzük meg, hol voltak azok a pontok, ahol a meglévő folyamatok javultak és
- melyek azok a pontok, ahol a tesztelési folyamatainkon javítani lehetne.
- Tudnunk kell, hol és milyen fázisban kell megjelennie az automatizálásnak a szoftverkészítési folyamatban.

Számos fontos része van egy pilot lezáró dokumentumnak, megbeszélésnek. Az egyik, talán majdnem a legfontosabb része, hogy kitérjünk benne: szeretnénk-e automatizálásra specializált eszközt használni, vagy sem. Amennyiben úgy döntünk, hogy a tesztautomatizálás bevezetését, használatát és használatának sikerességét egy eszközzel jobban el tudjuk érni, akkor sorakoztassuk fel az érveinket, hogy miért is szeretnénk ilyen megoldást használni.
Elképzelhető, hogy nem azzal az eszközzel szeretnénk a továbbiakban foglalkozni, amit a pilot ideje alatt használtunk, mert kiderült, hogy nem rendelkezik számos olyan funkcionalitással, ami számunkra elengedhetetlen a későbbiek folyamán. Ezeket az érveket egytől egyik tüntessük fel a dokumentumban, hogy egyértelmű legyen, mindenki számára miért nem javasoljuk az adott megoldást.
Mivel egy kereskedelmi forgalomban kapható eszközt meg kell vásárolni, fontos, hogy milyen funkcionalitással rendelkezik. Mivel ez költségként szerepel, így össze kell szedni a vezetőségnek, hogy miért is hasznos nekünk egy olyan eszköz, amiért fizetni kell.

3.6. Eszközkiválasztási szempontok

Szerencsés esetben a környezetünk (főnökeink, cég felső vezetői stb.) úgy dönt, hogy támogatják a tesztautomatizálási eszköz beszerzését. Amennyiben létezik rá megfelelő forrás, vizsgáljuk meg, hogy a

különböző eszközök milyen tulajdonságokkal rendelkeznek, valamint számunkra milyen elvárásoknak kell megfelelnie:

- Határozzuk meg, hogy milyen alkalmazásokat szeretnénk tesztelni,
- ezeknek milyen támogatottsága van a beszerezni kívánt eszköznél és
- képes-e kezelni a tesztelendő szoftverünk felhasználói felületét, értelmezni az objektumokat, vagy sem.
- Nézzük meg, hogy milyen szinten támogat más technológiákat,
- milyen szolgáltatást nyújt a gyártó az automatizálási eszközön kívül, valamint
- Létezik-e felhasználói támogatás, ami abban nyilvánulhat meg, hogy kapcsolatot tudunk-e teremteni egy mérnökkel akár, aki segítséget tud nekünk nyújtani, ha elakadtunk, vagy hibába kerültünk az eszköz használata során.
- Vizsgáljuk meg, hogy milyen csatornák léteznek az eszközzel kapcsolatban. Közösségi oldalak, fórumok, akár letölthető már előre kész sablon tesztek, melyekkel egyszerűbben tudunk előrehaladni.

Mielőtt eszközbeszerzésre szánnánk magunkat és vállalatunkat, ehhez hasonló kérdésekre érdemes magunknak válaszolni: [4]

- Mit szeretnénk megvalósítani az eszközzel? Legyünk nagyon specifikusak, próbáljuk meg lebontani részletesen a céljainkat és győződjünk meg arról, hogy az eszköz, amit vásárolni szeretnénk, megfelel ezeknek a céloknak.
- Ki a gyártó cég? Mennyire ismert a cég neve a piacon, és mennyire stabil. Hosszútávon kell gondolkoznunk, így nem szeretnénk beleesni abba a hibába, hogy egy méregdrága eszköz vásárlása után az eszköztámogatás megszűnik, mert a cég eltűnik a piacról.
- Milyen múltra tekint vissza az eszköz? Mikor jelent meg a piacon? Mi az aktuális verziószáma? Mennyire megbízható, stabil automatizáló eszközről beszélünk? Fejlesztés alatti verzióról beszélünk, vagy már egy stabil verzióról? Tervezi a gyártó cég a jövőben továbbfejleszteni az eszközt, vagy sem?
- Az elkövetkező időszakban várható egy jelentősebb frissítése az eszköznek? Ebben az esetben érdemes megvásárolni, vagy inkább várjuk meg a frissítést és már egy biztosan stabil eszközt vásároljunk?
- Milyen internetes fórumok, tréninget, könyvek léteznek az eszközhöz kapcsolódóan a piacon?
- Előreláthatóan mennyi idő szükséges a tesztautomatizálással foglalkozó csoportnak az eszköz megismerésére, és betanulására?
- A csoportnak megvan a technikai tudása az eszköz használatához?
- Mennyi költséggel jár az eszköz beszerzése és használata? Ne felejtsük el, hogy ez más, mint az eszköz ára. A költségek terén figyelembe kell vennünk az eszköz árát, tréningek árát, esetlegesen olyan számítógépek beszerzésének az árát is, amin használni fogjuk az eszközt. Ezek mind-mind beletartoznak a költségekbe.
- Akármennyire is furcsán hangozhat, de érdemes utánajárni, hogy a gyártó cég fel tudja-e mérni, hogy az automatizáló eszköz és a mi saját alkalmazásaink képesek-e „kommunikálni” egymással.

Elképzelhető, hogy maga a gyártó cég fogja azt mondani, hogy nem javasolják az eszközük megvételét.

- Ismerünk-e valakit, aki használta, vagy használja az általunk megvenni kívánt eszközt? Amennyiben igen, kérdezzük meg tőle, mik a tapasztalatai.
- Nézzünk utána az interneten, hogy a meglévő felhasználóknak vannak-e visszajelzései az eszközről.
- Létezik valamilyen garancia arra, hogy visszakapjuk a pénzünket, amennyiben nem vagyunk elégedettek az eszközzel?
- Elérhető ingyenes kipróbálható próbaverziója az eszköznek mielőtt megvennénk?
- Amennyiben több eszköz is szóba jöhet, próbáljuk ki mindegyiknek az ingyenes próbaverzióját, és hasonlítsuk össze őket.
- Mérjük fel, hogy az eszköz ténylegesen ér-e annyit számunkra, mint amennyiért megvennénk.

Mint láthat, a kérdések szinte kimeríthetetlenek. Elképzelhető, hogy nem minden kérdésre kapunk pozitív választ. Lehetséges, hogy az eszközválasztás valamilyen kompromisszum mentén kell, hogy történjen. Fontos szempont még az is, hogy milyen módon szeretnénk majd használni a tesztautomatizálási eszközünket. Melyik tesztautomatizálási stílust szeretnénk használni? Korábbi fejezetben ismertettem a különböző tesztimplementálási technikákat, így kérdés, például képes az eszköz támogatni akár az adatvezérelt és kulcsszó vezérelt automatizálást is?

4. Automatizált szoftvertesztelő eszköz tulajdonságai

Az előző fejezetekben érintőlegesen már volt szó arról, hogy melyek azok a tulajdonságok, amelyekkel jó, ha rendelkezik egy tesztautomatizálásra használt eszköz. A következő pontokban kicsit részletesebben ejtsünk szót a főbb funkcionalitásokról.

4.1. Felvétel és visszajátszás (Record & Playback)

Az elmúlt évek során hatalmas léptékben fejlődtek a tesztautomatizálást támogató eszközök. A piaci igényeket szinte majdnem minden eszköz képes kielégíteni. A gyártók arra törekednek, hogy fejlesszék ezeket az eszközöket, hogy a felhasználók igényeit minél inkább kielégítsék és a piacon vezető szerepet töltsenek be.

Szinte biztos vagyok benne, hogy majdnem minden eszköz rendelkezik a felvétel és visszajátszás funkcióval. Természetesen ez a megoldás nem azt jelenti, hogy egy egyszerű tesztlépés rögzítéssel már automatikus teszteket kapunk. Véleményem szerint ez egy nagyon jó tulajdonsága az eszközöknek. Ezzel a lehetőséggel viszonylag egy jól használható, működőképes vázat kapunk, amivel sokkal egyszerűbben tudunk dolgozni a tesztkészítés további szakaszában.

Manapság a szakma úgy gondolja, hogy a felvétel és visszajátszás egy elavult megoldás, és elítélendő. Az tény, hogy sokan „belekényelmesedtek" ebbe a funkcióba, és az így generált megoldások nem igazán tekinthetőek automatizálásnak. Viszont az én meglátásom az, hogy ha nem létezne ez a fajta lehetőség az eszközökben, a tesztimplementáció és annak ideje, költsége sokkal több lenne ezen funkció nélkül. Tehát használjuk és éljünk a lehetőséggel, viszont tartsuk szem előtt, hogy az így keletkezett tesztet tényleg nem lehet a szó teljes értelmében automatizált tesztnek nevezni.

Ez a funkció tényleg úgy működik, mint például egy hangrögzítő készülék. Bekapcsoljuk a felvétel gombot, csinálunk valamit, majd visszatekerjük és lejátsszuk. Ugyanazt fogjuk tapasztalni, mint akkor, amikor a rögzítést végeztük, azaz esetleg gyorsabb teszt végrehajtást, de semmi többet. És itt a hátránya a dolognak. Nem kapunk semmi ellenőrzést, feltételeket, függvényeket, eljárásokat, helyreállítási forgatókönyvet, így ez csak egy váz, amit végig lehet hajtani akár ezerszer. Jegyezzük meg, hogy a lépéseken kívül, ha adatot is használtunk, akkor azok sem változnak. Gondoljunk bele, hogy ha ezerszer végrehajtjuk ezt a tesztet ugyanarra az adatra, akkor mivel leszünk előrébb? Nagyon semmivel, hacsak nem az a cél, hogy ugyanazt az adatot használjuk ezerszer. Megjegyzem, elég ritkán találkozunk nagy mennyiségben olyan tesztesetekkel, ahol értelme lenne ilyen tesztvégrehajtásnak.

Mivel számos automatizálást támogató eszköz létezik a piacon - legyenek ingyenes vagy kereskedelmi forgalomban kapható eszközök -, mindegyik más-másmódon és formában képes rögzíti a lépéseket. Nézzünk egy példát.

A teszteset a következő:
Legyen a kezdőképernyő a google levelezőoldala. Írjuk be az e-mail címünket, jelszavunkat, majd kattintsunk a „Maradjon bejelentkezve” checkboxra, és kattintsunk a bejelentkezés gombra. Ezután kattintsunk a kijelentkezés gombra.

A felvétel és visszajátszás funkcióval egy ingyenes eszköz a következő lépéseket rögzítette:

Parancs	Cél	Érték
open	/ServiceLogin?sacu=1&scc=1&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&hl=en&service=mail	
type	id=Email	test.automation@gmail.com
type	id=Passwd	test.automation.password
click	id=PersistentCookie	
clickAndWait	id=signIn	
click	css=span.gb_9.gbii	
clickAndWait	id=gb_71	

3. táblázat Felvétel visszajátszás során rögzített eredmény

A tesztünk futtatás során a fenti táblázat sorain fog végigmenni, és az alkalmazás Cél oszlop alatti objektumain fogja végrehajtani a Parancs oszlopban talált utasításokat, felhasználva az Érték oszlopban lévő adatokat. Felmerülhet a kérdés, hogy hogyan tudja a szkript megtalálni a Cél oszlopban lévő objektumokat. Mint látjuk, egy azonosító alapján (id) próbálja meg megtalálni az Email, Passwd mezőket, melyeken végrehajtja a type, azaz a beírás utasítást. Nézzük meg a google levelezőjének a belépés oldalon található oldal forráskódját. Keressük meg azt a részt, ahogy ezek a mezők vannak.

```
<label  class="hidden-label" for="Email">E-mail</label>
<input  id="Email" name="Email" type="email"
       placeholder="E-mail"
       value=""
       spellcheck="false"
       class="">
<label  class="hidden-label" for="Passwd">Jelszó</label>
<input  id="Passwd" name="Passwd" type="password"
       placeholder="Jelszó"
       class="">
<input id="signIn" name="signIn" class="rc-button rc-button-submit"
type="submit" value="Bejelentkezés">
```

```
<label class="remember">
<input  id="PersistentCookie" name="PersistentCookie"
        type="checkbox" value="yes"
        checked="checked">
<span>
Maradjon bejelentkezve
</span>
```

Jobban megnézve a fenti tesztünk ezt a HTML kódrészletet használja, és ezen információk alapján működik. Minden mezőt az id alapján azonosít.

Nézzük meg ugyanezt a tesztesetet egy kereskedelmi forgalomban kapható eszközben. A felvétel és visszajátszás funkcióval rögzített lépések így néznek ki:

```
Browser("Gmail").Navigate("mail.google.com")
Browser("Gmail").Page("Gmail").WebEdit("Email").Set
" test.automation@gmail.com "
Browser("Gmail").Page("Gmail").WebEdit("Passwd").SetSecure
"54f1db4ff3c45ec47536f9d43497455a11397bb3e75d5b8aa272"
Browser("Gmail").Page("Gmail").WebCheckBox("PersistentCookie").Set "OFF"
Browser("Gmail").Page("Gmail").WebButton("Sign in").Click
Browser("Gmail").Page("Beérkező levelek (3) -").Link("Link").Click
Browser("Gmail").Page("Beérkező levelek (3) -").Link("Kilépés").Click
```

Mint látjuk, a két kód merőben más. A második megoldásban is látjuk azt a fajta megközelítést, mint az elsőben, itt viszont mintegy hierarchiaszerűen van leírva, hogy hol milyen objektumon hajtsunk végre utasítást. Az utóbbinál a böngésző, mint objektumon belül található oldalban keressük a különböző objektumokat: WebEdit, WebCheckBox, WebButton. A sorok végén pedig látjuk azt az utasítást, amit végrehajtunk. Vegyük észre, hogy a második példában a jelszó nem jelenik meg, ezt egy hosszú karakterlánc jelöli. Az automatizálásra használt eszköz már felvétel során átkódolja a jelszót. Miért is jó ez? Gondoljunk abba bele, hogy a tesztünket más is lefuttatja. Lehet olyan adatot és jelszót használunk, amit nem szeretnénk, ha más látna.

Meg kell jegyezni, hogy minden egyes automatikus teszt kezdőpontjának ugyanannak kell lenni. Mivel a teszt különböző iterációkra ugyanezeket a lépéseket fogja végrehajtani, így a tesztelt alkalmazásnak ugyanabban az állapotban kell lennie minden egyes iteráció előtt.

Ezeket a lépéseket körülbelül fél perc alatt meg lehetett oldani mindegyik eszközzel. Melyek a tapasztalatok a két esetben?

- Az eredmény mind a két esetben ugyanaz.
- A kód olvashatósága más.
- Mindkét esetben csak a felvett lépéseket hajtjuk végre.
- Nincs semmilyen feltételvizsgálat.
- Nincs ellenőrzés a tesztekben.

Igazából lehet ezeket tesztnek nevezni?
Amennyiben azt tekintjük sikeresnek, hogy végrehajtódnak a lépések, akkor tekinthető tesztnek mind a két megoldás. Első látásra egyik sem tartalmaz hibakezelést, így mi van akkor, ha rossz jelszót adunk meg? Mivel egyik teszt sem vizsgálja, hogy megjelent vagy sem egy hibaüzenet, nem nevezhető egyik teszt sem intelligensnek. Nos, ez az a pont, amikor a tesztautomatizáló befejezi a felvétel és lejátszás funkció használatát, és manuálisan elkezdi a kódot kiegészíteni és jobbá tenni.

4.2. Objektumtárolás

Az előző példában láttuk, hogy a felvétel és visszajátszás funkciónak köszönhetően lépésről lépésre rögzítettük a tesztünket. Az első esetben, a weboldalon az azonosító („id”) TAG alapján próbáljuk megtalálni azokat az elemeket, objektumokat, melyeken utasításokat hajtunk végre. Ebben az esetben, ha megváltozik a mezők azonosítója, jelen esetben az „id” tag (jelenleg az alapján történik az azonosítás), a tesztünket frissítenünk kell, hogy továbbra is működőképes legyen.

A második esetben, amikor végigmentünk a teszten és felvettük a lépéseket, ezzel párhuzamosan az eszközünk eltárolta azon objektumokat, melyeket érintettünk a tesztünk során. Minden információt mondjuk úgy, hogy lementett egy objektumtároló állományba. A teszt végrehajtása során, ebben az eltárolt halmazban megkeresi az eszközünk az objektumot, és ha azonosítani tudja, akkor végrehajtja a megfelelő utasítást, lépést. Abban az esetben, ha módosítás történik az objektumon, elég, ha ezt az objektum halmazt frissítjük, a tesztünk változatlan marad és továbbra is futtatható formában lesz elérhető.

Nézzük meg, hogy hogyan is néz ki egy ilyen objektumtároló. A belépési oldalon található e-mail mezőről ezeket a tulajdonságokat rögzíti a tesztünk:

Teszt objektum: Email
Az objektum részletes tulajdonságai:

Név	**Érték**
type	text
name	Email
html tag	INPUT
y	
xpath	
x	
width in characters	
width	
visible	True
value	
title	
rows	0
required	

readonly	
placeholder	
pattern	
outertext	
outerhtml	
max length	2147483647
kind	
innertext	
innerhtml	
html id	Email
height	
disabled	
default value	
css	
class	
abs_y	
abs_x	

4. táblázat Tesztelés alatt lévő objektumnál elérhető tulajdonságok és értékek

Mint látjuk, nagyon sok információt eltárolunk az objektumról. Valamelyiknek az értéke üres, így megvan a lehetőségünk, hogy mi magunk adjuk meg azokat. Előfordulhat, hogy a hiányzó értékek teszt-végrehajtás során fognak felvenni valamilyen értékeket. Ezeknek a tulajdonságoknak és a hozzá tartozó értékeknek megfelelően próbálja meg a tesztünk azonosítani az objektumot.

Mint említettem, ha valamilyen tulajdonsága megváltozik egy objektumnak, akár egy új fejlesztésnek köszönhetően, elég, ha itt, ebben a leíró állományban változtatjuk meg az értékeket, azaz frissítjük az objektum tulajdonságait. Jobb esetben lehetőségünk van arra, hogy az objektumhoz tartozó értékeket paraméterezzük, így elérjük azt, hogy nem kötött (beégetett) értékekkel tudjuk végrehajtani a tesztet.

Példa objektumparaméterezésre:
Képzeljük el, hogy ha egy angol személy lép be az oldalra, akkor az „Email cím” felirat „Email address”-re vált, mert az oldalunk képes azonosítani, hogy ki honnan, milyen országból nyitja meg. Ebben az esetben a tesztünk az „Email cím” objektumot fogja keresni, és természetesen nem fogja meg találni, mert „Email address” lesz látható a felületen. Ebben az esetben az objektum „text” tulajdonságát alakítsuk paraméterré, és a tesztünkben vizsgáljuk meg, hogy milyen adatot használunk. Amennyiben angol bejelentkezést szimulálunk, akkor legyen az objektumunk „text” mezője „Email address” érték, amit mi például az adattáblánkból, mint paraméter adunk meg a tesztnek. Ezzel máris rugalmasabb, karbantarthatóbb, modularizáltabb tesztet tudunk készíteni.

Gondoljunk bele, hogy ugyanezen a felületen sok tesztet fogunk végrehajtani. Sok folyamat része ez a belépést végrehajtó teszt. Ezeket az objektumtároló modulokat le tudjuk menteni, és hozzárendelni más

tesztekhez. Nemcsak a tesztlépésekben tudunk definiálni különálló részeket, hanem itt, az objektumleíró modulokban is. Ezzel más tesztek is ugyanazokat az objektumleíró modulokat fogják használni. Amennyiben változik egy objektum - mint korábban említettem -, csak egy helyen kell frissíteni az információt, nem lesz kihatással a többi tesztünkben. A karbantarthatóság sokkal hatékonyabb és rugalmasabb.

A második tesztünkben lehetséges olyan megoldás is, hogy ezeket az objektumleíró modulokat nem használjuk. Ezt a fajta tesztkészítést deskriptív programozásnak hívják. Itt kiküszöböljük az objektumok tárolását. A felvétel és visszajátszás funkció ebben az esetben nem érvényes, hiszen azzal a folyamattal a tesztet rögzítjük, viszont azt a vázat, amit kaptunk, át tudjuk formálni úgy, hogy deskriptív legyen.

A fenti példa deskriptív megoldással:

```
Browser("title:=Gmail ").Navigate("mail.google.com")
Browser("title:=Gmail ").Page("title:=Gmail ").WebEdit("name:=Email").Set
" test.automation@gmail.com "
Browser("title:=Gmail ").Page("Gmail").WebEdit("name:=Passwd").SetSecure
"54f1db4ff3c45ec47536f9d43497455a11397bb3e75d5b8aa272"
Browser("title:=Gmail ").Page("title:=Gmail ")
.WebCheckBox("name:=PersistentCookie").Set "OFF"
Browser("title:=Gmail ").Page("title:=Gmail ").WebButton("name:=Sign in")
.Click
Browser("title:=Gmail ").Page("title:=Beérkező levelek (3) -")
.Link("Link").Click
Browser("title:=Gmail ").Page("title:=Beérkező levelek (3) -")
.Link("text:=Kilépés").Click
```

Mint látjuk, a kódban megadjuk, hogy milyen tulajdonságot keresünk, milyen értékkel, hasonlóan, mint a legelső példában. Itt a böngésző „title” tulajdonságát, továbbá az oldal „title” tulajdonságát, valamint a WebEdit és WebButton objektumok „name” értékét vizsgáljuk.

Miért lehet ez a deskriptív megoldás előnyös?

Gondoljunk bele, hogy éppen az alkalmazás implementálása folyik a programozók oldalán. Még nem rendelkezik stabil felhasználói felülettel az alkalmazásunk, viszont tudjuk, hogy az objektumok milyen névvel és tulajdonságokkal lesznek ellátva. Ebben az esetben a felvétel és visszajátszás nélkül, deskriptív módon már tudunk teszteket készíteni, és akár a fejlesztés fázisában is használható teszteket tudunk készíteni. Mire elkészül a felhasználói felület, már elérhetőek lesznek futtatható tesztek is. Hosszabb távon természetesen a karbantarthatóságot szem előtt tartva érdemes az objektumleíró és tároló modulokat elkészíteni.

4.3. Naplózás

A következő fontos tulajdonsága egy automatizálásra használt eszköznek a naplózás. A teszteket nem feltétlenül napközben szeretnénk és fogjuk futtatni. Nem mindig oldható meg az emberi felügyelet. Nem várhatjuk el a tesztelőtől, hogy mindig, a nap huszonnégy órájában elérhető legyen és teszteket futtasson. Elképzelhető, hogy bizonyos tesztek ütemezve, esti órákban kellenek, hogy elinduljanak. Ekkor nincs rálátásunk a tesztek futására, így olyan naplózási megoldást kell keresnünk, amiből akár a következő napon, vagy egy héttel később is vissza tudjuk keresni, hogy az adott tesztvégrehajtás milyen eredményt produkált. A naplózás másik fontos része lehet, ha hibás tesztfutásokat szeretnénk kielemezni. Nagy segítséget nyújtanak a részletesen dokumentált futások, és sokszor a tesztek karbantartásában is segítenek.

[2a] Miért is fontos még a tesztek futásának naplózása? Ideális esetben egy automatikus tesztfutás emberi beavatkozás nélkül történik. Ezért is fontos a minél részletesebb dokumentálása a tesztvégrehajtásoknak. A dokumentálás, naplózás pedig elvárt funkció lehet egy automatizálásra használt eszköztől. Ez nem azt jelenti, hogy a tesztkészítők nem fejleszthetnek egyedi naplózási funkciókat.
Ezen riportok nélkül nem tudjuk megfelelően kielemezni a tesztfuttatásainkat. A riportok, naplók különbözők lehetnek. Érdemes végiggondolni, hogy milyen módon szeretnénk a naplókat rendszerezni. Egy tesztvégrehajtás során általában a teszteredményekre vagyunk kíváncsiak, viszont elkerülhetetlen, hogy valamilyen hibába ütközzünk végrehajtás során. Például, ha a tesztelt alkalmazás egy idő után nem elérhető, akkor egy automatikus teszttől elvárjuk, hogy hiba esetén valahol rögzítse, miért nem tudtuk folytatni az adott végrehajtást. Tehát lehetséges, hogy lesznek futási naplók és hibanaplók is. Valamikor ez a kettő egy dokumentumban is kezelhető, természetesen megfelelően szemléltetve, hogy az eredmények mit tükröznek.

Egy tesztfutási naplónak általában tartalmaznia kell a sikeres és sikertelen futások listáját. Minden egyes teszteset futási eredményét rögzíteni kell. Ne felejtsük el, hogy egy tesztesetet több különböző tesztadattal is végre tudunk hajtani. Ebben az esetben két tesztfutásról beszélünk, így az eredmények között a tesztesetnek kétszer kell szerepelnie. Ekkor javasolt a naplóba kiíratni a tesztadatot is, hogy éppen milyen értékkel hajtottuk végre. Törekedjünk arra, hogy minél beszédesebb legyen egy ilyen napló, viszont ne legyen túl zsúfolt. Próbáljuk meg megtalálni az egyensúlyt.

Egy futási naplóban célszerű tárolni a futási idők értékét is. Számos megoldás lehet, de az egyik legkézenfekvőbb, ha rögzítjük a teszt végrehajtásának kezdeti idejét és végét. A két eredményből származtatható a futási idő. Ez hasznos lehet a későbbiek folyamán, ezzel tudjuk vizsgálni esetleg azt is, hogy a tesztelt alkalmazásban merült-e fel teljesítményhiba.

Példa:
Tegyük fel, hogy száz tesztesetet kell végrehajtanunk. Lehetséges, hogy a nyolcvanadik tesztfuttatás során a felhasználói felület hirtelen elkezd lassulni, a kért oldalak lassan töltődnek. Ezekkel az időmérésekkel tudjuk ezeket észrevenni, hiszen nem biztos, hogy a tesztfutást látjuk. Egészítsük ki még

egy értékkel a naplót. Tegyünk minden futás mellé egy dátum értéket is. A teljesítménybeli hibákat ezzel konkrét napra és időre meg tudjuk mondani.

Számos automatizáló eszköz rendelkezik olyan opcióval, hogy tranzakciós időt tudunk mérni. Természetesen mi is gyárthatunk ilyen eljárásokat. Ezeknek előnye, hogy bármilyen utasítást a tesztvégrehajtás során közre tudunk zárni, és célzottan tudunk időt mérni egy teszten belül. Ez akkor lehet hatásos, ha tudjuk, hogy az alkalmazásunk valamelyik pontjában véletlenszerűen léphet fel teljesítmény probléma. Ekkor specifikusan azt a pár lépést tudjuk mérni, amit vizsgálnunk kell.

Egy napló tartalmazhatja az aktuális konfigurációt is. Operációs rendszer verzióját, böngészőverziót, a tesztelt alkalmazás verzióját és még sorolhatnánk. Miért is fontosak ezek? Tegyük fel, hogy a tesztünk képes több böngészőtípusban is végrehajtódni. A naplóban így pontosan szerepelni fog, hogy milyen böngészőben és annak melyik verziójában lett végrehajtva a teszt. Lehetséges, hogy egy elvárt funkcionalitás másképpen fog működni egy Explorerben, mint egy Firefoxban.

Végül egy összegzést is tartalmazhat a napló. Mennyi tesztet futtattunk, ebből mennyi sikeres, mennyi hibás. Mennyi volt az összes futási idő.

Nézzünk egy példát a futási naplóra. Az előzőekben két különböző tesztrögzítést mutattam be. A tesztek különböző környezetben futnak, így eltérő naplók készülnek. Az alábbi az ingyenesen elérhető eszköz eredménye.

Test Suite: mail_login
Test cases: 1 total / 0 passed / 1 failed
Commands: 7 total / 5 passed / 1 failed / 1 skipped

Index of test cases
mail_login_sample

Test case: mail_login_sample			
open	/ServiceLogin?sacu=1&scc=1&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F&hl=en&service=mail		
type	id=Email	test.automation@gmail.comm	
type	id=Passwd	test.automation.password	

Test case: mail_login_sample			
click	id=PersistentCookie		
clickA	id=signIn		
click	css=span.gb_9.gbiii		Element css=span.gb_9.gbiii not found
clickA	id=gb_71		

5. táblázat Tesztvégrehajtás utáni napló 1.

A példában látszik, hogy három részre lehet osztani a riportot. A felső rész egy összesítő táblázat, a középső egy tartalomjegyzéknek feleltethető meg, az alsó pedig a tesztesetet és annak lépéseit mutatja. A piros sor a táblázatban azt jelzi, hogy abban a pontban a tesztfuttatás sikertelen volt. A megjegyzés oszlopban pedig látszik, hogy mi volt a hiba forrása: nem lehetett megtalálni a megfelelő objektumot, amin a kattintás műveletet szerettük volna végrehajtani.

A másik, kereskedelmi forgalomban kapható eszköz az alábbi naplózást készíti el. Ez a funkció az eszköz saját beépített naplózási eljárása.

Test: mail_login

Test Iteration 1			
Action: Action1			
Name	Status	Time	Details
Gmail	Done	3/2/2015 - 13:58:39	Browser
Gmail	Done	3/2/2015 - 13:58:39	Page
Email.Set	Done	3/2/2015 - 13:58:39	"test.automation@gmail.com"
Passwd.SetSecure	Done	3/2/2015 - 13:58:39	"54f1db4ff3c45ec47535b8aa272"
PersistentCookie.Set	Done	3/2/2015 - 13:58:40	"OFF"
Sign in.Click	Done	3/2/2015 - 13:58:40	
Beérkező levelek (3) -	Done	3/2/2015 - 13:58:43	Page
Link.Click	Done	3/2/2015 - 13:58:43	
Kilépés.Click	Done	3/2/2015 - 13:58:45	

6. táblázat Tesztvégrehajtási napló 2.

Lehetőség van az automatizáló eszközben beállítani, hogy részletes, vagy rövid tesztriportot szeretnénk. Ennél a naplónál létezik egy kiterjesztett verzió is, ahol minden egyes lépést részletesen lebontva látunk.

A teszt napló fejléce:

Test: mail_login

Product version: 12.02
Results name : TempResults
Time Zone: Central Europe Standard Time
Host Name: HUDEB53
Operating System: Windows 7
Run started: 3/2/2015 - 13:58:38
Run ended: 3/2/2015 - 13:58:46

Látjuk, hogy itt az alapvető információk szerepelnek.

A részletes naplózásnál az e-mail cím beírása így néz ki:

Step Name: **Email.Set** Step

Object	Details	Result	Time
Email.Set	"zollica@gmail.com"		3/2/2015 - 13:58:39

Itt látszik, hogy az objektum is szerepel a listában.

Vannak olyan helyzetek, amikor nekünk kell gondoskodni a napló elkészüléséről. Nem biztos, hogy egy ingyenesen elérhető automatizáló eszköznek pontosan van olyan funkciója, ahogy mi szeretnénk használni. Amennyiben valamilyen programnyelvben készítünk tesztet, akkor mindenképpen számolnunk kell azzal, hogy egy naplózási funkcióval is ellássuk a tesztünket.

A következő példa VBScriptben készült. A teszt célja, hogy egy forráshelyet átmásoljunk egy célhelyre. A másolási funkciókat vizsgáljuk, és egy általunk létrehozott naplóállományba rögzítjük a másolás sikerességét. Tegyük fel, hogy a tesztünket paraméterek meghívásával tudjuk elindítani, ahol a paraméterek a forrás hely, a cél hely és egy naplóállomány megadása.

```
Set fso = CreateObject ("Scripting.FileSystemObject")
Set argsNamed = WScript.Arguments.Named
      If Not ( paramOK("forras") and paramOK ("cel") and paramOK("naplo") )
      Then
        DisplayUsageAndExit
End If
Set logFile = fso.OpenTextFile ( argsNamed.Item("naplo") , 8 , True)
```

```
sForrasMappa = argsNamed.Item("forras")
sCelMappa = argsNamed.Item("cel")
'Checks if the specified folders exist
If Not fso.FolderExists (sForrasMappa) Then
      logFile.WriteLine Date & " " & Time & " : ERROR : A forrás mappa nem
létezik!"
      WScript.Quit
End If
If Not fso.FolderExists (sCelMappa) Then
      logFile.WriteLine Date & " " & Time & " : ERROR : A cél mappa nem
létezik!"
      WScript.Quit
End If
'------------------------------------------------------------------------------
-------------------------------------
logFile.WriteLine Date & " " & Time & " : A teszt elindult ..."
'------------------------------------------------------------------------------
-------------------------------------
```

Ahogy látjuk, a tesztben feltételekkel vizsgáljuk, hogy léteznek vagy sem a megfelelő mappák. Nekünk kell gondoskodni arról, hogy hiba esetén rögzítsük a lépéseket, erre szolgálnak a logFile.WriteLine sorok, ahol a naplóállományba írt felhasználó által definiált szövegek jelennek majd meg.

Végül legyen egy példa Python nyelven. A naplózási funkcióra készíthetünk külön eljárásokat is. Ebben az esetben a tesztünk sokkal olvashatóbb lesz, és bárhol alkalmazhatjuk az elkészített eljárást, nem kell duplikálni a különböző riportokat.

```
# Script Logging method ( Write out information about important steps )
def script_Log(qrt,round,roundtime):
    f1.write("\nQuery running Script V 1.3c - clean 0.1 \n\nRun At time stamp : ");
    f1.write(str(bigtimestamp.strftime("%Y.%m.%d  %H:%M:%S")));
    f1.write("\nQuery Run Time:");
    f1.write(qrt);
    f1.write("\nRound: ");
    f1.write(str(round));
    f1.write("\nRound Time:");
    f1.write(roundtimne);
    f1.write("\n\nFull Run Time : ");
    f1.write(str(datetime.now()-bigtimestamp));
    f1.write(" sec\n\n/================================================/\n");
```

Ez egy előre definiált eljárás. Ezt a következőképpen tudjuk meghívni bárhol a tesztünkben:

```
script_Log(qrt,round,str(datetime.now()-startTimeclient))
```

Mint látjuk, három paramétert is használunk. Ez a teszt SQL lekérdezéseket hajt végre és méri a lekérdezések idejét. A qrt paraméter egy mért érték már, amit felhasználunk a naplózás során. A qrt idő egy tranzakciós idő, amit az sql lekérdezés során mértünk.

[2a] Említsünk meg pár dolgot a hibák naplózásáról is. Amennyiben a tesztünk nem tud valamilyen hibából adódóan tökéletesen végrehajtani, célszerű dokumentálnunk a hiba létrejöttét és körülményét is. Kezeljük külön ezeket a naplókat a tesztfutási (eredmény) naplóktól. Azt, hogy hogyan kezeljük és tároljuk a hibanaplókat és hogy ki milyen megoldást preferál, eltérő lehet még a tesztautomatizálási csoporton belül is. Lehetséges, hogy a projekt követeli meg, hogy egy adott helyre mentsünk minden hibát. Tehát tesztje válogatja, miként lesznek rögzítve az ilyen esetek.

Mit célszerű tartalmaznia egy ilyen hibanaplónak? Mindenképpen hasznos, ha rögzítjük benne a tesztesetet (azonosító, név, stb.), és az automatikus teszt végrehajtásának információit. Értem ez alatt, hogy milyen paraméterekkel lett végrehajtva a tesztünk, és hogy milyen tesztadatot használtunk. Egy hiba nem feltétlenül eredményez incidenst. Elképzelhető, hogy azért lett hibás a tesztvégrehajtásunk, mert olyan tesztadatot használtunk, ami nem elérhető, vagy nem létezik az alkalmazásban. Tipikus példa erre, ha olyan elemet akarunk kiválasztani egy lenyíló menüből, ami nem szerepel benne. Ekkor nem tudja a tesztünk végrehajtani az utasítást, de nem is az alkalmazás hibája ez, hanem a tesztadaté. Elképzelhető, hogy az automatikus tesztünkben van valami hiba. Lehetséges, hogy a tesztünk működése okozta a hibát, akár egy hibás kódrészlettel, vagy egy soha véget nem érő ciklussal, amire nem figyeltünk korábban.
Célszerű valamilyen módon rögzíteni a tesztelt alkalmazás állapotát abban a pontban, amikor a hiba keletkezett. Egy megoldás lehet, ha képernyőképet készítünk, és lementjük a rendszerünkben a megfelelő helyre. Javasolt olyan nevet adni a képernyőképnek, ami egyértelműen azonosítja azt a hibanaplóban is.

Elképzelhető, hogy a hibanapló adott részén például olyan információt tárolunk el, ami a teszt végrehajtó környezetről szól. Például a tesztet végrehajtó gép memóriaállapotát mutatja. Lehetséges, hogy rendszerszintű hiba miatt nem tudott végrehajtódni a teszt. Akár rögzíthetjük a processzorunk kihasználtságát, futó alkalmazások listáját, háttérben futó szolgáltatások listáját és még sorolhatnám. Minél több információt rögzítünk ezekben a naplóállományokban, annál könnyebben lehet kideríteni, hogy egy tesztfuttatás miért volt sikertelen.

4.4. Futási idők mérése

Az előző naplózási résznél már említettem futási időméréseket is. Mielőtt elkezdenénk „haszontalan” információkat tárolni a naplóállományokban, célszerű végiggondolni, hogy pontosan mit is szeretnénk mérni. Futási idő alatt sok mindent érthetünk. Tegyük fel a kérdést: „Mit is szeretnénk mérni”?

Példa:
Legyen egy ötven tesztesetből álló automatikus tesztünk. Itt mérhetjük az ötven teszteset együttes futását, ahol nem mérjük külön az egyes tesztek végrehajtását. Ekkor csak annyit kapunk, hogy az első tesztünk legelső lépésétől mérjük a tesztvégrehajtást, az ötvenedik tesztesetünk utolsó pontjáig. Nem

látjuk pontosan, hogy a különálló teszteknek mennyi ideig tartott a végrehajtása. Ebben az esetben az eredménykiértékelés sem a legpontosabb, hiszen csak annyit tapasztalunk, hogy az ötven teszteset gyorsabban, vagy lassabban hajtódott végre az előző tesztfuttatásokhoz képest.

Előnyösebb, ha az összes tesztvégrehajtási idő mellett monitorozzuk az egyes tesztek végrehajtásának idejét is. Tehát az összes teszteset végrehajtásának ideje, meg kell, hogy egyezzen a különálló tesztek futás idejének összegével. Ekkor a kiértékelés során, még pontosabban tudjuk vizsgálni a végrehajtások idejét.

	Tesztfuttatás ideje / vége	Modulok futásának ideje (ms)
	2015.03.04. 10:40:50	
Bejelentkezés		100000
Rendelésfeladás		150000
Kijelentkezés		50000
	2015.03.04. 10:45:50	
Összesen	300000	300000

7. táblázat Tesztvégrehajtási idők

A táblázatban látjuk, hogy a teszt elindulásának, valamint a tesztfutásának végét is rögzítettük. A különálló modulok külön-külön rendelkeznek a végrehajtásuk idejével, így az összeg megegyezik mindkét esetben.

Elképzelhető, hogy egy adott tesztben csak pár lépést szeretnénk monitorozni. Mikor fordulhat elő ilyen eset? Például akkor, ha tudjuk, hogy a tesztelt alkalmazásunknak abban az állapotában instabil működés is várható. Ekkor az adott részt külön mérési blokkban fogjuk szerepeltetni a kódunkban, hogy legyenek mért értékeink a különböző iterációkban.

Egy egyszerű példa tranzakció idő mérésére. VBScript-ben helyezzük el a következő sorokat a mérni kívánt programrészlet elé és után:

```
'----------------------------------------------------------------------
'Tranzakció mérésének indítása
StartTime = Timer()
'----------------------------------------------------------------------
'
'Programkód
'
'----------------------------------------------------------------------
'Tranzakció mérésének leállítása
EndTime = Timer()
'----------------------------------------------------------------------
'Tranzakció idejének kiíratása
```

```
WScript.Echo("Tranzakció alatt mért érték másodpercben, kettő tized pontossággal: " & FormatNumber(EndTime - StartTime, 2))
'-------------------------------------------------------------------
```

Miért fontos, hogy legyenek futási idők mérve?

Tegyük fel, hogy minden tesztünk rendelkezik a megfelelő tesztvégrehajtási naplóval, hibanaplóval. Előfordulhatnak olyan esetek, amikor nem is számítunk arra, hogy a tesztek végrehajtása során hibába futunk. Tegyük fel, hogy egy esti időpontban sok tesztesetet kell lefuttatnunk. Beütemezzük a tesztjeink futását, és másnap reggel azt tapasztaljuk, hogy még mindig van hátra pár teszt, ami nem hajtódott végre. Miután lefutottak azok a tesztek is, elkezdjük kiértékelni az eredményeket. Azt tapasztaljuk, hogy a tesztjeink tökéletesen hajtódtak végre, a hibanaplóban semmilyen bejegyzés nem szerepel. Vajon mi történhetett, hogy csak reggelre ért véget a tesztek futása, holott terveink szerint két-három órás időintervallumba bele kellett volna férjen az összes teszt végrehajtása? Nézzük meg az egyes tesztek futásának idejét. Azt tapasztaljuk, hogy a kétharmadánál a tesztek futása lelassult. Egy teszt végrehajtása tízszer annyi időt vett igénybe, mint korábban. Ez a jelenség tartott körülbelül öt órán keresztül. Nézzük meg, hogy a fejlesztők jeleztek-e valamilyen problémát az este folyamán. Azt tapasztaljuk, hogy pontosan abban az időszakban egy adatbázis karbantartást kellett végrehajtaniuk és ez kihatással volt a tesztelt alkalmazásunk teljesítményére is. Megvan a probléma forrása! Nem a tesztjeink minősége és állapota okozta a futások idejének nyúlását, hanem környezeti hatások befolyásolták azt. Ebben az esetben, igaz, hogy lefutottak a tesztek, viszont sokkal több időt vettek igénybe. Jogosan vetődik fel a kérdés, hogy újból el kell indítani ezeket a teszteket, vagy a meglévő eredmények elégségesek számunkra? A válasz erre egy újabb kérdés. Mi a célja az automatikus tesztjeinknek? Az alkalmazás funkcionalitásának tesztelése? Az alkalmazás funkcionalitása mellett a válaszidők mérése is? Vagy csak a tranzakciók mérését szeretnénk rögzíteni, hogy kiértékeljük, hogy az alkalmazás egy bizonyos időn belül képes válaszolni? Nagy valószínűséggel, ha az utolsó kettő a cél, akkor érdemes egy újabb tesztfuttatást végrehajtani, és megvizsgálni, hogy az adatbázis karbantartás után fennáll-e még a teljesítménylassulás, vagy sem.

4.5. Szinkronizálás

A következő tulajdonság a szinkronizálás. Szinkronizálás alatt sok mindent érthetünk. Általános értelemben egy tesztautomatizáló eszköznek az a tulajdonsága, amivel képes az automatikus tesztvégrehajtást „moderálni” a tesztelés alatt lévő alkalmazással. Nagyon fontos tulajdonság ez, hiszen ennek hiányában biztos vagyok benne, hogy sok sikertelen tesztfuttatásunk lenne.

Legszemléletesebb példa erre a webes alkalmazások tesztelése. Elkészítünk egy tesztet, ellátjuk minden olyan szükséges tulajdonsággal, amivel rendelkeznie kell egy automatikus tesztnek. Elkezdjük futtatni a tesztünket és azt tapasztaljuk, hogy az első pár lépés (amíg meghívja a tesztelendő webes alkalmazást) tökéletesen működik, utána pedig leáll a tesztünk. Megvizsgáljuk a hibanaplót és azt tapasztaljuk, hogy a végrehajtás során a tesztünk nem találja meg azokat az objektumokat, melyeken műveleteket szeretnénk végrehajtani. Viszont látjuk, hogy megnyílt az alkalmazásunk a böngészőben, és ott is van a szükséges objektum.

Kezdő tesztautomatizálók lassabban fognak rájönni a megoldásra. Itt az történik valójában, hogy a tesztünk - mivel egy program - sokkal gyorsabban végrehajtódik, mint ahogy a tesztelt alkalmazásunk reagál. A tesztünk türelmetlenül végigrohan, azt feltételezve, hogy a weboldalunkon megtalálható minden olyan elem, amin mi műveletet szeretnénk végrehajtani. Igazából ez nem így van. Tudjuk jól, hogy mialatt „interakcióban" vagyunk a tesztelt alkalmazással, az folyamatosan reagál azokra az utasításokra, melyeket mi végrehajtunk. Legyen ez akár egy linkre kattintás, egy lenyíló menüből való érték kiválasztás, rádiógombok kiválasztása stb. Ezeket a lépéseket elég gyorsan végre is hajtja a tesztünk, de gondoljunk abba bele, mi történhet akkor, ha egy lenyíló menüből kiválasztott érték után, az oldalunk frissíti magát, és egy olyan mező jelenik meg, ami még eddig nem. Ezt a mezőt mi használni is szeretnénk a tesztünkben a későbbiek folyamán és bele is fejlesztettük ennek a mezőnek a lekezelését a tesztbe. Amikor manuálisan tesztelünk, látjuk, hogy az oldalunk tölt, várakozik, és egyszer meg is jelenik ez a mező. Ugyanezt a várakozást kell előidéznünk az automatikus tesztünkben annak érdekében, hogy sikeresen tudjon a tesztünk tovább futni. Tipikusan ez az a példa ott, ahol szinkronizálást kell a tesztünkbe beletenni. Számtalan megoldás létezhet erre.

Elképzelhető, hogy létezik beépített szinkronizálási funkció a különböző automatizálásra használt eszközökben, de lehetséges, hogy nem és nekünk kell definiálni ilyen eljárásokat.

Gondoljuk végig, hogy milyen esemény bekövetkezéséhez szeretnénk kötni azt, hogy a tesztünk végrehajtása mehessen tovább. Mik lehetnek ezek:

- Várjuk meg, amíg a weboldal betöltése megtörténik?
- Várjunk egy előre megadott ideig, és csak akkor menjen tovább a teszt?
- Addig várjunk, míg valamilyen objektum meg nem jelenik a felhasználói felületen?

Számos kérdés, és igazából a helyes válasz itt is az, hogy minden attól függ, mit szeretnénk és mi az elvárás a tesztünktől.

Nézzünk meg egy VBScript példát:

Nyissunk meg egy Internet Explorer böngészőt, majd navigáljunk el a http://www.ni.com/ oldalra, és várjunk addig, amíg be nem töltődik az oldal. Tegyük bele az előző fejezetben használt tranzakció mérési kódot is. A lenti példa alapján addig nem hajtódik végre a tranzakció idő mérésének a vége, amíg a feltételünk nem teljesül, ebben az esetben az objIExplorer.ReadyState, ami a böngésző állapotát jelzi. Ez egy szinkronizációs pont.

```
'-----------------------------------------------------
'változók
Dim objIExplorer
Dim StartTime
Dim EndTime
'-----------------------------------------------------
Set objIExplorer = CreateObject("InternetExplorer.Application")
objIExplorer.Visible = True
StartTime = Timer()
```

```
objIExplorer.Navigate "https://www.ni.com/"
Do While objIExplorer.ReadyState <> 4
    WScript.Sleep 2
Loop
EndTime = Timer()
WScript.Echo("Oldal betöltődésének ideje (másodpercben): " & FormatNumber(EndTime -
StartTime, 2))
```

Mint látjuk, az explorer ReadyState tulajdonságát vizsgáljuk, azt, hogy pontosan milyen értékek lehetnek,meg tudjuk nézni a Microsoft MSDN oldalán:
https://msdn.microsoft.com/en-us/library/ie/ms534361%28v=vs.85%29.aspx

Ezzel az apró kódrészlettel máris elértük, hogy a tesztünk várjon addig a pontig, amíg a böngészőnk teljes egészében be nem töltötte az oldalt. Addig nem fog végrehajtódni egyéb utasítás.

Ahogy korábban említettem, jobb esetben automatizálásra használt eszközök rendelkeznek beépített szinkronizációs modullal. A fenti példa (ReadyState) - amikor a böngészőnek a betöltődését vizsgáltuk - így néz ki egy kereskedelmi forgalomban kapható eszközben:

```
Browser("Browser").Sync
```

Itt egy Sync utasítást hajtunk végre a böngészőn. Tegyük fel, hogy létezik több olyan lépés, amit ezután a kódsor után szeretnénk végrehajtani. Felmerül egy kérdés, amit célszerű megfontoltan körbejárni.
Vajon meddig fogja a tesztünk vizsgálni, hogy a böngészőnk betöltődött?
Egy ilyen eszközben, amelyiknek létezik szinkronizációs lehetősége, meg lehet adni, hogy mi legyen az alapértelmezett idő, amíg a tesztünk vár. Nyilván nem szeretnénk azt, hogy egy fél napig álljon a tesztünk abban a fázisban, hiszen lehet, azért nem tudja „megvárni" az oldal betöltését, mert valamilyen hibába ütköztünk és nem képes ezt felismerni a tesztünk. Próbáljuk meg felmérni, hogy mi az a maximális idő, amíg az alkalmazásnak válaszolni kell, és azt állítsuk be alapértelmezett értéknek. Jobb esetben a követelmények között szerepelnie kell egy ilyennek.

Másik szinkronizációs megoldás lehet az is, hogy vizsgáljuk egy bizonyos objektum adott tulajdonságát. Az előző példa alapján, ha egy lenyíló menüből kiválasztunk egy elemet, ami után egy új szöveges mezőnek kell megjelennie, akkor célszerű a szöveges mező valamelyik tulajdonságának értékét ellenőrizni. A kérdés, hogy teljesül-e a tesztelt alkalmazásban az az elvárt eredmény, hogy megjelenik az extra mező.

Példa:
Manuális lépések: Elérhető egy webshop, ahol egy számítógép konfigurációt állítunk össze. Miután kitöltöttünk mindent, legyen adott a weboldalon egy PDF állománygenerálási lehetőség. Miután a PDF generálásra rákattintunk, az oldalon megjelenik egy kis kör és jelzi nekünk, hogy a PDF készítés folyamatban van. Ezután nekünk egy PDF ikonra kell rákattintanunk és lementeni a PDF állományt.

Automatikus megoldás: Miután a PDF generálásra rákattintunk, egy szinkronizációs lépést kell a tesztbe betennünk, hogy ne fusson tovább addig, amíg meg nem jelenik a PDF ikon. Ebben az esetben az egyik megoldás az lehet, hogy a PDF ikon megjelenését vizsgáljuk. Tudjuk, hogy a generálás után az ikonnak meg kell jelenni. Mielőtt rákattintanánk a PDF ikonra, tegyünk be egy objektum ellenőrző lépést, ami így néz ki:

```
Browser("Ajánlatkérés").Page("Ajánlatkérés").Image("pdf_icon").WaitProperty "visible", True
```

Ez a sor annyit csinál, hogy az előzőleg beállított szinkronizációs ideig próbálja vizsgálni a pdf_icon objektum visible tulajdonságát, hogy True értéket vett-e fel vagy sem. Amikor teljesül a feltétel, a tesztünk folytatja a végrehajtást, amennyiben nem, úgy hibát fog jelezni. Ekkor attól függően, hogy mit implementáltunk a tesztünk hibakezelési funkciójában, vagy leáll a teszt, vagy továbbmegy, de a következő iterációval. Természetesen arról ne feledkezzünk meg, hogy a tesztelés alatt lévő alkalmazást a kezdeti, kiindulási állapotba hozzuk!

Kitalálhatunk másfajta szinkronizálást is. Igazából a következő megoldás nem igazi szinkronizálás, hiszen nem a tesztelt alkalmazás állapotát ellenőrizzük. Tegyük fel, hogy az előző példában generált PDF-et a pdf ikonra kattintva lementjük egy adott helyre a merevlemezünkre. Lehet egy olyan feltétel is a tesztben, hogy amíg a lementett pdf állomány nem jelenik meg az előre definiált mappában, addig ne menjen tovább a teszt.

```
FileName = "c:\webshop\pdfallomanyok\arajanlat.pdf"
Set FSO = CreateObject("Scripting.FileSystemObject")
Do
   If FSO.FileExists(FileName) Then
       Exit Do
   End If
   WScript.Sleep 1000
Loop
```

Ez nem feltétlenül a legjobb megoldás. Mi történik akkor, ha a pdf ikon nem is fog megjelenni, mert valamilyen hiba lépett fel az alkalmazásban? A fenti kód a Do Loop részben beragad, hiszen egy végtelen ciklus, nem tartalmaz semmilyen kilépési pontot. Érdemes alaposan végiggondolni, hogy egy tesztben mikor melyik típusú várakoztatási megoldást célszerű alkalmazni, valamint tartsuk szem előtt, hogy a fentiek csak példák, emellett számos olyan megoldás létezhet, melyek kifejezetten egy tesztre tudnak hatékonyak lenni.

5. Automatizált szoftvertesztelés bevezetése

5.1. Automatizálható tesztek detektálása

Tegyük fel, hogy az előző pontokban leírtakat megismertük, és ezek mentén el is szeretnénk kezdeni használni a tesztautomatizálást a megfelelő módon. Biztos vagyok benne, hogy a kezdeti sikerek miatt elsőnek azok a tesztek fognak elénk kerülni, melyek viszonylag gyorsan és egyszerűen implementálhatóak és végrehajthatóak. Sokszor a manuális tesztelők nem tudják felmérni, hogy milyen komplex tud lenni egy manuális teszt automatikus implementálása. Ezért nem biztos, hogy már a legelső pillanatban el kell kezdenünk olyan teszteken dolgozni, melyek elénk kerülnek.

Az alábbi beszélgetés egy üzleti elemző és egy automatizált szoftverteszteléssel foglalkozó tesztelő között zajlott le. A beszélgetést az üzleti elemző kezdi:

- *Szia, Sanyi. Örülök, hogy időt szakítasz rám.*
- *Szia, Orsi. Ugyan már, nagyon szívesen, örülök, ha tudok segíteni.*
- *Hallottam, hogy automatizált szoftverteszteléssel foglalkoztok. Az elmúlt időszakban nagyon sok projektünk volt és szeretnénk megnézni, hogy a megrendelés készítés folyamatát lehet-e automatizálni.*
- *Ez az internetes felületünkről érkező megrendeléseket, vagy a vállalatirányítási rendszerünkben lévő megrendelés kezelő modulokat érinti?*
- *Csak a vállalatirányítási részről lenne szó. Ott kell sok megrendelést lekönyvelnünk.*
- *Értem. Volt már példa arra, hogy a vállalatirányítási rendszerben automatizáltunk. Pontosan mit szoktatok csinálni? El tudod nekem mondani? Én nem nagyon ismerem a megrendelések folyamatát.*
- *Persze. Több oldalról is indulhat maga a folyamat. Ugye lehetséges, hogy meglévő megrendelőnek könyvelünk megrendelést, de elképzelhető, hogy egy teljesen új vevőnek. Ezután megadjuk a szállítási és számlázási adatokat, felvisszük a megrendelt terméket, elképzelhető, hogy kedvezményt adunk az árból, és végül véglegesítjük a megrendelést.*
- *Értem. Általában ez szokott lenni a folyamat?*
- *Igen.*
- *Van valami olyan tényező, ami megváltoztatja a megrendelés-készítés folyamatát? Például, ha más terméket könyvelünk be?*
- *Nem.*
- *Ugye tőlünk rendelhetnek hardvert és szoftvert is a vevők. Ekkor is ugyanaz a folyamat?*
- *Igen. Azaz várj csak. Most, hogy kérdezed, vannak olyan hardver elemek, ahol egy külön ablak is megnyílik a folyamat során és egyéb információt is be kell írnunk.*
- *Kötelező érvénnyel be kell írni valamit azokra az ablakokba, vagy elég, ha nyomunk egy Tovább gombot és kész?*
- *Valamikor elég, ha csak tovább megyünk, de előfordulhat az is, hogy kötelezően meg kell adnunk egyéb paramétert.*

- *Vegyesen szoktak az ilyen esetek előfordulni? Vagy elég, ha csak az egyik fajtát csinálnánk meg?*
- *Vegyesen. A teszteknek körülbelül a fele.*
- *Értem. Azért kérdezem ezeket, mert nekünk ezeket tudni kell. Egy automatikus tesztet fel kell készítenünk olyan dolgokra is, ami nekünk, mint tesztelőknek egyértelmű, de egy automatikus teszt esetén nem. Ugyanis nem tud váratlan helyzetekben gondolkozni.*
- *Ezt nem is gondoltam volna.*
- *Mennyi megrendelést szoktatok egy ilyen tesztelés alatt megcsinálni?*
- *Olyan tíz darabot.*
- *Tíz? Mennyi időbe telik azt manuálisan megcsinálnotok?*
- *Körülbelül egy óra.*
- *Nos, nekem úgy tűnik, hogy az automatikus tesztet elkészíteni jóval több idő kell majd, mint egy óra. A futtatást tekintve picit lesz szerintem gyorsabb, így nem biztos, hogy megéri ezt leautomatizálni.*
- *Értem. Sajnos annyi más teendőnk is van, és próbálnánk a lehető legjobb megoldást megtalálni. Ezek a folyamatok nem változtak már hosszú ideje. Úgy néz ki, hogy nem is lesz projekt ezen a területen.*
- *Aha, ebben az esetben akkor kérdeznék még. A megrendelések és azok menete különböznek más-más országból érkező rendelések esetén?*
- *Nem. Viszont mi csak magyar, olasz és angol adatokkal szoktunk tesztelni.*
- *Csak három országra teszteltek? És csak tíz tesztesetben?*
- *Igen.*
- *Mi garantálja azt, hogy egy német vásárlónál is minden működik úgy, ahogy kell?*
- *Mivel ugyanazok a megrendelések bekönyvelésének lépései, így gondoljuk, hogy nem lesz hiba.*
- *Nos, ha egy automatikus teszt elkészül, az adatok paraméterezetten vannak benne. Amennyiben meglenne ez a teszt, bármennyi adatra végre lehetne hajtani.*
- *Hű de jó hangzik. Akkor azt is meg tudnánk nézni, hogy ötven különböző típusú megrendelőnek működnek a folyamatok?*
- *Persze. Ez lenne a célja. Sok ugyanolyan lépést megcsinálni különböző adatmennyiségre. Ebben az esetben már nem egy óra lenne egy manuális tesztelőnek a tesztelés, hanem jóval több.*
- *Igaz, erre nem is gondoltam.*
- *Annak érdekében, hogy a lefedettséget növeljük, így már el tudom képzelni, hogy érdemes foglalkozni ezzel a kérdéssel.*
- *Nagyszerű. Holnap ráérnél? Megmutatnám, hogyan is csináljuk pontosan a megrendeléseket a rendszerben.*
- *Igen, akartam is kérni. Legalább látnám, hogy milyen felületeket használtok.*
- *Rendben, akkor küldök neked egy meghívót e-mailben a holnapi megbeszélésre.*
- *Ok, köszönöm. Szia.*
- *Köszönöm. Szia.*

A fenti párbeszéd alapján nem minden esetben az a jó megoldás, ha egyből belevetjük magunkat egy teszt elkészítésébe.

De nem minden esetben történnek meg ilyen párbeszédek. Van olyan eset, amikor a tesztautomatizáló feladata kideríteni, hogy mit lehet lefejleszteni. Nagy valószínűséggel már léteznek manuális tesztek nagy számban a vállalatnál. Elsőként meg kellene vizsgálnunk, hogy melyek azok a tesztek, melyeket lehet automatizálni.

Automatizálható tesztek:
Próbáljuk meg összegyűjteni a különböző manuális teszteléssel foglalkozó kollégák teszteseteit és a teszteseteket tartalmazó tesztgyűjteményt. Legyen ez egy táblázatos forma. Tegyünk a táblázatba egy új oszlopot, ebben az oszlopban jelöljünk meg minden olyan tesztesetet, ami véleményünk szerint automatizálható. Mi alapján mondható egy tesztesetről, hogy automatizálható?
Meg kell vizsgálnunk az alábbiakat:

- **Technikai megoldás**
  Fel kell mérnünk, hogy a jelenlegi automatizálási eszközeinkkel meg tudjuk-e csinálni a tesztet vagy sem. Amennyiben nincsen dedikált eszközünk, úgy megvannak-e azok a programozási ismeretek, melyekkel implementálni tudjuk az adott tesztet.

- **Alkalmazás elérhetőség**
  Nézzük meg, hogy az alkalmazás, amin a tesztet végre kell hajtani, elérhető számunkra vagy sem. Elképzelhető, hogy egy speciális csoport tudja csak elérni azt az alkalmazást, így ebben az esetben számunkra lehetetlen a hozzáférés.

- **Területfelelős elérhetősége**
  Tudnunk kell, hogy az adott tesztesethez elérhető, vagy sem a tesztelő, esetleg az üzleti elemző. Elképzelhető, hogy vannak olyan tesztek, melyeket csak végrehajtunk, vagy végrehajtanak más tesztelők, de nincs meg a megfelelő tudásunk és információnk a pontos működéshez. Lehetséges, hogy már nem is dolgozik a vállalatnál az a tesztelő, vagy üzleti elemző, akik ezt a tesztet készítették, és jelen pillanatban sincs olyasvalaki, aki releváns információval tudna szolgálni az alkalmazás adott funkcionalitásáról.

Egy példa: tesztgyűjtemény, amiben jelöltük az automatizálható teszteket:

Tesztazonosító	Teszteset	Automatizálható
1.1	Rendelés bekönyvelése szoftver termékkel.	Igen
1.2	Rendelés bekönyvelése, a megrendelésre kedvezmény és megjegyzés felvitele.	Igen
1.3	Rendelés bekönyvelése, a pénzügyi modulból nyert információk használatával.	

1.4	Bekönyvelt rendelés módosítása egyszerű adatokkal.	Igen
1.5	Bekönyvelt rendelés módosítása, majd ismételt módosítása.	Igen

8. táblázat Tesztgyűjtemény automatizálhatóság jelölésével

Figyeljük meg, hogy az 1.3-as esetet nem jelöltük meg automatizálhatónak. Elképzelhető, hogy a pénzügyi modulhoz nincsen hozzáférésünk és ebben az esetben már nem tudjuk megfelelően végrehajtani a tesztet.

Természetesen egy ilyen lista akár száz darab tesztesetet is tartalmazhat. A következő lépcső legyen az, hogy kiegészítjük a táblázatot még egy oszloppal. Ebben az oszlopban jelöljük meg azokat a teszteseteket, melyeket érdemes leautomatizálni.
Mik a szempontok?

- **Nagy adatmennyiség**
  Egy teszteset automatizálás szempontjából nagyon jó, ha nagy adatmennyiségre hajtjuk végre. Ez nem azt jelenti, hogy egy tesztfutás alatt sok adatot használunk, hanem azt, hogy sokszor futtatva a tesztet, különböző adatokat használunk, és azokra hajtjuk végre a tesztet.

- **Sok ismétlődő lépés**
  Egy tesztesetet többször is végre lehet hajtani, különböző adatra. Viszont ha nem változnak a teszt végrehajtásának lépései, akkor szintén érdemes automatizálást használni, mert nem kell az adatok milyensége alapján a tesztünket „okosítani”.

- **Manuális tesztelők által nehezen, vagy egyáltalán lehetetlen tesztvégrehajtás**
  Vannak olyan esetek, melyeket egy manuális tesztelő nagyon nehezen tud végrehajtani, mert sok időbe telik elejétől a végéig megcsinálni a tesztet. Vannak olyan esetek, amelyeket szinte lehetetlen, hogy végrehajtson a manuális tesztelő. Ilyen lehet például háromszáz darab internetes hivatkozás megnyitása három különböző böngészőben.

- **Alkalmazás stabilitása**
  Itt azt vizsgáljuk, hogy a tesztelt alkalmazásban a közeljövőben lesz-e funkcionális változás, ami esetleg a felhasználói felületet is érinti. Meg kell vizsgálnunk, hogy érdemes-e a változás előtt foglalkozni az automatizálással, vagy majd csak utána, mikor az új funkció belekerült a rendszerbe.

- **Megtérülés**
  Vizsgálnunk kell, hogy az automatikus teszt elkészítéséhez mennyi idő szükséges, és a futások során mikor fog megtérülni a befektetett munka. Amennyiben minimális vagy abszolút lehetetlen a megtérülés, úgy nem javasolt automatizálni.

Nézzük meg a fenti táblázatot, amit kiegészítettünk egy újabb oszloppal:

Tesztazonosító	Teszteset	Automatizálható	Megéri automatizálni?
1.1	Rendelés bekönyvelése szoftver termékkel.	Igen	Igen
1.2	Rendelés bekönyvelése, a megrendelésre kedvezmény és megjegyzés felvitele.	Igen	
1.3	Rendelés bekönyvelése, a pénzügyi modulból nyert információk használatával.		
1.4	Bekönyvelt rendelés módosítása egyszerű adatokkal.	Igen	Igen
1.5	Bekönyvelt rendelés módosítása, majd ismételt módosítása.	Igen	Igen

9. táblázat Tesztgyűjtemény megtérülés jelölésével

A második és harmadik esetben nem szerepel Igen az oszlopban. Az 1.3-as esetben egyértelmű, hiszen nem érjük el a megfelelő modult az alkalmazásban, így nem is tudjuk, hogy mit kellene csinálnunk. Az 1.2-es esetben viszont azért nem tettünk Igen-t, mert a megjegyzést tároló modulon a következő hónapban fejlesztést fognak csinálni. Így most azzal nem érdemes foglalkozni.

5.2. Automatizált folyamat kiépítése

Az automatizált szoftvertesztelés sikeressége nagyban függ attól, hogy azt milyen körülmények között végezzük. Kisebb méretű tesztelői csoportnál teljesen másképpen tud működni, mint egy nagyobb és komplexebb csoport esetén. Minél szélesebb a tesztelendő alkalmazások és feladatok száma, annál bonyolultabb és nehezebb lehet a tesztek feltérképezése és automatizálása.

Időről időre tapasztalni fogjuk, hogy az automatizálandó tesztek utáni igény nagyban meg fog ugrani. Egyre többen fogják látni, hogy milyen előnyökkel jár teszteket leautomatizálni, viszont nekünk fontos szem előtt tartani, hogy nem mindig lehet és célszerű bizonyos teszteket elkészíteni.

Ideális esetben a tesztelésnek párhuzamosan kellene mennie az alkalmazásfejlesztéssel. Miért fontos ez? Nem akkor kellene teszteket detektálni és készíteni, amikor a programozók lefejlesztették az alkalmazást. Mivel a tesztautomatizálónak nem kötelessége érteni minden teszteléshez, így meg kell érteni, hogy az új fejlesztéssel szemben melyek a követelmények. Fel kell térképezni, hogy milyen funkcionalitások vannak lefedve tesztekkel, illetve melyek azok az új tesztek, amelyeket a tesztelők megálmodtak.

Alkalmazásfejlesztés fázisai:

Tervezés	Követelmények	Dizájn	Implementálás	Tesztelés	Karbantartás

Tesztelés fázisai:

Teszttervezés	Tesztesetek	Tesztszkriptek	Tesztfuttatás és karbantartás

Mint látjuk, a tesztelésnek el kellene kezdődnie már az alkalmazás tervezési szakaszában. Sokaknak nehezen elképzelhető, hogy a tesztelés akár már a követelmények specifikálásánál elkezdődik. Gondoljunk bele, hogy mennyi időt lehet megspórolni a tesztek implementálása során, ha minden hasznos információt begyűjtünk már a legelején a projektnek. Sajnos nem minden esetben történnek meg a fenti tevékenységek. Az alkalmazás fejlesztésének ütemtervében is történhetnek változások, melyek kihatással lehetnek a teszt készítésére. Ekkor sajnos kevesebb idő lesz allokálva a tesztelés fázisainak valamelyik részein.

Nézzünk meg egy példát, ami egy tesztautomatizálással foglalkozó szakember és egy üzleti elemző között zajlik le a követelmények specifikálásának szakaszában. A párbeszédet az üzleti elemző kezdi:

- *Szia, János.*
- *Szia, Kornél. Megkaptam a leveledet, amiben írod, hogy szeretnétek automatikus teszteket használni a tesztelés során. Láttam, hogy ötven darab követelményt specifikáltatok. Beszámolnál picit bővebben, hogy miről is szól a projekt?*
- *Természetesen. A cég internetes áruházában tervezünk változásokat implementálni. Mostantól a bejelentkezés után, a vásárló neve alá szeretnénk megjeleníteni, hogy ezüst, arany, vagy platina besorolásba tartozik. Úgy gondolom, hogy ezt ellenőrizni nem lesz nehéz egy automatikus teszttel, hiszen csak ezeket az értékeket kellene vizsgálnotok.*
- *Igen, ez nem hangzik bonyolultnak, viszont lenne pár kérdésem. A követelményspecifikációk között a huszadik pontot megnéznénk? Pontosan erre kérdeznék rá, amit mondtál. Szóval a specifikáció szerint bejelentkezünk és látni fogjuk, hogy mi a felhasználó besorolása.*
- *Igen, így van.*
- *Minden esetben látni fogjuk?*
- *Mire célzol?*
- *Tudomásom szerint, a jelenlegi adatbázis felépítésünkben nem szerepel olyan mező, amiben ilyeneket tárolunk.*
- *Igen, igazad van. Ez is a fejlesztés része lesz. Szükségünk lesz egy új mezőre az adatbázisban.*
- *Ok. Mi lesz a meglévő felhasználókkal? Hiszen ezek az információk nem szerepelnek.*
- *Tényleg. Igazad van. A meglévő felhasználóinkra jelenleg nem fogjuk kiíratni. Ez a funkció majd két hónap után lesz elérhető, és csak az új felhasználóknak tudjuk megjeleníteni.*

- *Ebben az esetben lehetséges, hogy a követelménylistátokat frissítsétek ezzel az infóval? Ez biztos kérdésként is fel fog merülni a későbbiek során. Fontos, hogy pontos legyen a funkció leírása.*
- *Igen, természetesen. Jogos a felvetésed.*
- *És még egy kérdés. Automatizált tesztet szeretnétek használni. Mivel nem csak Magyarországról vannak regisztrált felhasználók, hanem más külföldi országokból is, ezért a kérdésem, hogy a megjeleníteni kívánt ezüst, arany, platina szövegek idegen nyelven is meg lesznek jelenítve?*
- *Azt hiszem, megint igazad van. Hiányos a specifikációnk. Igen, a tervünk az, hogy ha nem Magyarországról lép be valaki, akkor angolul jelenítjük meg ezt az információt.*
- *Mármint, ha magyar felhasználó lép be, de nem Magyarországról, akkor is angol lesz a szöveg?*
- *Nem. Bocs, pontosítok. Nem az adott hely alapján jelenítjük meg, hanem az alapján, hogy a regisztráció során milyen országot adott meg a felhasználó.*
- *Értem, tehát ha Magyarország van a felhasználói profilban, akkor magyar, ellenkező esetben mindig angol.*
- *Igen, így van. Ezt is javítom és belekerül a specifikációba.*
- *Köszönöm szépen. Igen, ezek fontosak, hiszen egy jól megírt specifikáció alapján sokkal hatékonyabban tudunk teszteket tervezni, illetve az automatikus teszt funkcionalitását is megtervezni. Amint javítottad a listát, megtennéd, hogy újra elküldöd nekem?*
- *Persze, és köszönöm, hogy ezekre rávilágítottál.*
- *Igazán nincs mit. Ezzel úgy gondolom, hogy a jövőben kevesebb időt kell fordítani majd a kérdéseink megválaszolására.*
- *Én is így gondolom. Köszi és további szép napot.*
- *Köszönöm. Neked is.*

A párbeszédből látjuk, hogy egy tesztelő miként tud már az alkalmazás fejlesztésének a legelején hatékonyan dolgozni. Fontos, hogy amennyiben elérhetőek a specifikációk, azokat megvizsgáljuk, és adjunk visszajelzést a programozóknak, üzleti elemzőknek.

Annak érdekében, hogy a lehető leghatékonyabban tudjuk kihasználni az automatizálást, foglalkoznunk kell a projektek kezdeti szakaszaival is. Az előző fejezetekben leírt szempontokat fel kell térképeznünk. Ezeket annál nehezebben tudjuk kivitelezni, minél több igény merül fel a tesztautomatizálásra. Valamilyen folyamatot ki kell találnunk és a mentén működtetni az automatizálást, mert hosszabb távon a saját dolgunkat fogjuk nehezíteni. Tartsuk szem előtt, hogy nem feltétlenül projektekre fogunk teszteket készíteni, lehetséges, hogy már meglévő funkcionalitások tesztelésére kell automatikus megoldásokat kitalálnunk, hogy ne kelljen időről időre a meglévő regressziós tesztjeinket végrehajtani.

A jelenlegi agilis módszertanok és alkalmazásfejlesztési stílusok lehetővé tesznek pár hasznos megoldást. Az alábbi naptárban az április 20-i hét jelöli a következő tesztelési ciklust. Tegyük fel, hogy most március 9. hétfő van.

MÁRCIUS						
M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

ÁPRILIS						
M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Az automatizálással foglalkozó tesztelőknek be kell gyűjtenie minden hasznos információt, ami az április 20-i tesztelési időszakra szükséges. Fel kell térképezni, hogy mik azok az új funkciók, melyeket esetleg le kell fejleszteni egy adott projektre, illetve azt is, hogy van-e olyan regressziós teszt, amit lehetne automatizálni és használni a tesztelési ciklusban.

Az automatizálási folyamat részei:

- Automatizálandó tesztek begyűjtése
- Tesztelési ciklus tervezése
- Erőforrás tervezése
- Tesztimplementáció
- Tesztek ellenőrzése
- Tesztek véglegesítése

Ezek az alapfolyamatok. Annak érdekében, hogy mindegyik fázis megfelelően legyen kiépítve, alapos tervezés szükséges. A fenti táblázat alapján az április 20-i hét a tesztelés, tehát addigra elérhetőnek kell lennie az automatikus teszteknek. A fenti fázisok alakulása a következő szerint történjen:

Automatizálandó tesztek begyűjtése

Célja: Összegyűjteni az összes olyan kérést, ami tesztautomatizálással kapcsolatos, továbbá aktualizálni a tesztek listáját.

Fontos, hogy legyen egy listánk a felmerült igényekről, hiszen ezzel tudunk riportálni a menedzsmentnek, illetve szemléltetni, hogy mekkora igény van aktuálisan az automatizálásra. Próbáljuk meg kideríteni az alapvető funkcionalitásokat és terveket, hogy mit tartalmazzon egy automatikus teszt.

Tesztelési ciklus tervezése

Célja: Felmérni, hogy az elkövetkezendő időszakra a tesztautomatizálási folyamatoknak milyen határidőkkel kell bekövetkeznie. A tervezésnek része kell legyen, hogy egyáltalán lehetséges vagy sem kivitelezni egy automatikus tesztet.

Itt meg kell vizsgálnunk azt is, hogy a tesztkörnyezet, vagy bármilyen egyéb környezet (ahol elérhetőek az alkalmazás régi és új funkciói) elérhető lesz-e arra, hogy teszteket kezdjünk implementálni.

Erőforrás tervezése

Célja: Felmérni, hogy az automatizálással foglalkozó tesztelők mennyi időben lesznek elérhetőek az adott időszakban.

Tudnunk kell, hogy ki mikor nem lesz elérhető. Ki lesz szabadságon. Milyen más feladattal kell foglalkoznia a tesztelőnek. Jobb esetben meg kell tudnunk becsülni, hogy hány órában tud automatizálással foglalkozni a tesztelő.

Azt is érdemes felmérni, hogy a teszt elkészüléséhez szükséges információt kitől tudjuk majd begyűjteni. Elérhető lesz az üzleti elemző, vagy a tesztelő? Egyeztetni kell velük is, hogy hány órában szeretnénk kérni a segítségüket.

Tesztimplementáció

Célja: Az előzetesen begyűjtött, priorizált és kiválogatott tesztek implementálása.

Ez az a fázis, amikor ténylegesen automatikus teszteket fejlesztünk.

Tesztek ellenőrzése

Célja: Megbizonyosodni arról, hogy az automatikus tesztünk azt csinálja, amire szükség van.

Ez egy érdekes fázis és célszerű be is tartani. Tulajdonképpen a tesztünket teszteljük. Sokszor tapasztaltam, hogy a tesztek akkorra készültek el, amikor már használni kellett őket - természetesen ezt okozhatta az alkalmazás fejlesztésének csúszása is. Ebben az esetben sajnos a tesztelési időszakban buknak ki olyan hibák, melyek hátráltatják az automatikus teszt használatát. Éppen ezért jobb már a tesztelési ciklus előtt ellenőriznünk a teszteket, hogy a teszt végrehajtásakor ne érjen meglepetés minket.

Tesztek véglegesítése

Célja: Tesztek végleges állapotba hozása, tesztimplementálási standardok mentén történő kódjavítás, tesztadatok megfelelő integrálása és tesztek dokumentálása.

Lehetséges, hogy a legegyszerűbb fázisnak tűnik ez az utolsó, de közel sem az. Jó esetben a tesztünk készítése során már használjuk a standardjainkat, nem használunk beégetett értékeket, megjegyzésekkel látjuk el a teszteket. Viszont gondolni kell arra is, hogy ezeket a teszteket mások számára is elérhetővé kell tennünk. Nem feltétlenül a végrehajtás miatt, hanem azért, hogy használható legyen a tesztben implementált működés másoknak is. Tudja meg más tesztelő is, hogy egy létező automatikus teszt mit tud csinálni és milyen feltételek mentén lehet azt használni.

Ezek alapján a fenti fázisokat próbáljuk meg a legoptimálisabban elhelyezni a naptárban. A következőképpen alakulhat a naptár:

	Automatizálandó tesztek begyűjtése
	Tesztelési ciklus tervezése, erőforrás tervezése
	Tesztimplementáció

	Teszt ellenőrzése és teszt véglegesítése

MÁRCIUS						
M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

ÁPRILIS						
M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Amint látható, a különböző fázisoknál nincsenek éles határok. A tesztelési időszak tervezése és erőforrás tervezése történhet párhuzamosan. A tesztek ellenőrzése és véglegesítése is történhet egyszerre, hiszen a standardok alkalmazása után is kötelességünk leellenőrizni a tesztek működését.

A tesztek begyűjtési és tervezési szakaszában célszerű tartani egy közös megbeszélést az adott terület tesztelőivel, üzleti elemzőivel, esetleg a menedzserekkel is. Számtalanszor előfordulhat, hogy a felmerült igények megvalósítására szükséges idő nem elégséges. Ekkor a tesztek prioritása és az adott alkalmazás fontossága dönthet, mivel kell foglalkoznunk. Nézzünk meg egy ilyen beszélgetést, amikor az automatizálással foglalkozó csoportból egy tesztelő (T), egy üzleti elemző (ÜE) és egy menedzser (M) beszélget.

- *(T) Sziasztok, örülök, hogy eljöttetek erre a megbeszélésre. A megbeszélés célja, hogy felmérjük, milyen automatikus tesztekre lenne szükség, valamint hogyan tudnánk ezeket megvalósítani.*

A listát, ami tartalmazza az összes jelenleg beérkezett igényt, elküldtem nektek a múlt héten. Sikerült megnéznetek?

- *(M) Igen, részemről minden elem benne van.*
- *(ÜE) Igen, én is megnéztem, viszont most lenne még egy kérésem. A pénzügyi modulunkon csinálnánk egy kis javítást. Bizonyos országoknál nem jól jelenik meg a megjegyzés mező, és a programozók most ezen dolgoznak. Szükségünk lenne egy olyan tesztre, ami nagy mennyiségben generálna nekünk számlákat, hogy megnézzük, megfelelő-e a megjegyzés mezőnek a működése.*
- *(T) Máris hozzáadom ezt a listánkhoz. Mikor lenne ennek a fejlesztése, és mikor menne ki élesbe?*
- *(ÜE) A fejlesztés folyamatban van, és májusban már az éles rendszert használnánk.*
- *(T) Az még egy picit távoli időpont. Most van pár fontos teszt, amit április végére meg kellene csinálnunk.*
- *(ÜE) Tudom, igen, viszont még annyit jegyeznék meg, hogy a programozók pont április végén fogják az első részét a kódnak megcsinálni. Már tudnánk tesztelni pár számlát május előtt.*
- *(T) Igen, ebben az esetben ezt célszerű szem előtt tartani.*

- *(M) Annyi lenne a részemről, hogy nem lesz kirakva az éles rendszerre a mi oldalunkról a megrendelés visszaigazolásának funkciója.*
- *(T) Akkor ebben az esetben a prioritását levenném annak a tesztnek, rendben?*
- *(M) Természetesen, azért is szóltam.*
- *(T) Röviden összefoglalnám, hogy milyen tesztek vannak és hogyan is állunk. Szóval automatikus teszteket fogunk implementálni, azaz tervezünk implementálni az elkövetkezendő két hétben. Azaz tizenkét napban, hiszen március 26-án már el is kezdjük a fejlesztéseket. Rajtam kívül még két kolléga fog dolgozni a teszteken. A tizenkét nap alatt Árpinak ötvenhat órája lesz arra, hogy teszteket implementáljon, Istvánnak pedig hatvan órája. István két napot szabadságon lesz, én pedig negyven órát tudok tesztekkel foglalkozni. Egyéb időnkben megbeszélések lesznek és egyéb adminisztratív dolgokkal is foglalkoznunk kell. Tehát összesen százötvenhat óra allokálható tesztkészítésre. A listánkban jelenleg kettőszázötven órányi fejlesztés van megbecsülve. Egy része ezeknek még nem is érhető el a tesztrendszeren, így csökken a cél. Most úgy tűnik, hogy mivel nem lesz készen a megrendelés visszaigazolás funkció, ezért több időt allokálhatunk a számlázás tesztelésére. Egyetértetek ezzel?*
- *(M) Teljes mértékben. Mivel nem leszünk készen, és nem is lesz az éles rendszeren majd csak később, így legyen magasabb prioritású bármelyik teszt, amivel érdemes foglalkozni.*
- *(ÜE) Igen és ennek örülök is. Remélem, hogy a fejlesztés során már találunk olyan hibákat az automatikus teszt használatával, amivel a projekt időben el tud készülni.*
- *(T) Akkor megkérem Árpit, hogy foglalkozzon ezzel a teszttel, valamint Istvánt, hogy folytassa az előző ciklusban elkezdett tesztet, és ha lehet, fejezze is be. Megkérem Istvánt, hogy a megrendelés visszaigazolására készülő funkció miatt keressen meg egy programozót és kezdjen el egyeztetni, hogy mire is lesz szükségetek a későbbiek során.*
- *(M) Nagyszerű, ez tökéletes.*
- *(T) Rendben, örülök, hogy meg tudtuk ezt beszélni. Április végén beszámolunk majd az eredményeinkről.*
- *(ÜE) Köszönöm.*
- *(M) Köszönöm én is.*
- *(T) Sziasztok.*

A fenti párbeszédből is látszik, hogy a tesztelőnek szorosan együtt kell dolgoznia az üzleti elemzőkkel és menedzserekkel is. Az automatizálás sikeressége nagyban függ a közös munkán, valamint a megfelelő döntéseken és egyetértésekben.

[2b] Annak érdekében, hogy a folyamatokat megfelelően ki tudjuk építeni és fenn tudjuk tartani, hosszabb távon szükség lesz egy dedikált automatizálással foglalkozó csapatra. Fontos, hogy a csapatban lévő tagok tudásának összhangban kell lennie azzal a szereppel, amit betölt a csapatban. A fenti párbeszéd alapján a tesztelőnek megfelelő kommunikációs tudása kell, hogy legyen, amivel el tudja érni a célját a különböző területeken dolgozó kollégákkal. Lehetséges, hogy más anyanyelvű kollégával is kell egyeztetni, ekkor már a nyelvismeret is fontos. Elképzelhető, hogy vannak olyan tagok a csoportban, akiknek a programozási tudásuk a kimagasló, ők inkább a tesztkészítésekben tudnak sikereket elérni.

Egy automatizálási csoport struktúrája lehet a következő, ahol megfelelő szerepkörök vannak definiálva:

Automatizálási csoportvezető
Feladatai közé tartozik az automatikus teszterv elkészítése, tesztek és igények felderítése. Stratégiai kommunikáció más vezetőkkel, tesztimplementálási tervek készítése. Fontos, hogy a csapatban lévő tagokat vezesse, teljesítményüket monitorozza, és fejlessze a csapatot. A felmerülő feladatokat kiosztja a csapat többi tagjainak és szigorúan ellenőrzi az ütemtervet.

Tesztfejlesztők
A tesztfejlesztésekért felelős csoporttagok. Ők azok, akiknek az adott alkalmazás funkcionalitásáról megvan a tudásuk. Felelősek a tesztek készítéséért, futtatásáért és az eredmények kiértékeléséért.

Szkriptfejlesztők
A szkriptfejlesztőknek van meg a tudásuk az automatizálásra használt eszköz működtetésére. Ők látják el a keretrendszer karbantartását és a tesztek támogatását.

Bizonyos csoportoknál nem lehet ilyen élesen meghatározni a szerepköröket. A vállalat struktúrája és a tesztelői csoport kiépülése is meg tudja határozni, hogy milyen formában rendszereződnek a tagok. Sokszor a tesztfejlesztők egy külön csoport tagjai. Elképzelhető, hogy a manuális tesztelők látják el azokat a feladatokat, melyeket leírtam.

[2c] Automatizált folyamat kiépítése során szükségünk lesz egy tesztautomatizálási tervre.

Tesztautomatizálási terv
Ennek a dokumentumnak a célja, hogy leírja az automatizáláshoz szükséges lépéseket. Tartalmaznia kell a különböző tesztkomponensek listáját, valamint a szükséges erőforrások listáját is. Ide tartozik az automatizálással foglalkozó tesztelő, az üzleti elemző, manuális tesztelő stb. Amennyiben szükséges valamilyen tréning, akkor azt is fel kell tüntetni a szükséges időkerettel együtt. Rögzíteni kell az ütemtervet, és azokat a pontokat, melyeknek teljesülnie kell, hogy lépésről lépésre tudjunk haladni a tesztkészítésben.
Általában egy ilyen dokumentum a tesztelési ciklus során folyamatosan frissítve van. Nem lehet azt mondani, hogy kötelezően el kell készülnie minden egyes tesztimplementáció előtt, hiszen vannak olyan dolgok, melyek pontosan akkor merülnek fel, melyekor elkezdtünk automatizálni. Ezeket rögzíteni kell ebben a dokumentumban.

Milyen alapvető elemeket célszerű egy ilyen dokumentumnak tartalmaznia?

- **Dokumentumváltoztatások**

Mivel ez egy folyamatosan változó dokumentum, célszerű számon tartani, ki, mikor és mit változtatott a dokumentumon.

Művelet	Ki	Mikor	Megjegyzés
Létrehoz	K. István	2015-03-18	K:/Projektek/TestDokumentaciok/Terv/TestTerv2015.doc
Frissítve	M. Béla	2015-03-19	Tesztkörnyezet hozzáadva.
Frissítve	T. Krisztián	2015-03-20	Fejlesztők neve frissítve.
Módosítva	K. István	2015-03-21	Ütemterv frissítve. Egy héttel tovább tart a fejlesztés.
Jóváhagyva	R. Nándor	2015-03-21	Manuális tesztelési oldalról OK.
Jóváhagyva	B. Katalin	2015-03-23	IT oldalról OK.

- **Alkalmazás**

Röviden írjuk le, hogy mit tudunk a tesztelendő alkalmazásról,

Verzió: 2.4 – Webáruház bejelentkező felület

- **Tesztautomatizálás célja**

Fontos, hogy megfelelően írjuk le, mi és hogyan lesz tesztelve az automatikus teszt által. Írjuk le azt is, hogy mit nem tartalmaz a tesztünk. Fontos, hogy mások is tudják, mi az, amit elvárhatunk egy teszt futtatása során. Sokszor egy félresikerült megbeszélés és tervezés nem realisztikus elvárásokat fog eredményezni a tesztünktől.

Az automatizált teszt használatával célunk az, hogy nagy mennyiségben hajtsunk végre bejelentkezést a webáruházunk bejelentkezési felületén. Bejelentkezés után ellenőrizzük az üdvözlő szöveget. Nem célja a tesztnek, hogy a menetközben felmerülő, nem várt eseményekre megfelelően reagáljon, az a fontos, hogy hiba esetén megpróbálja rögzíteni a hibás lépést. A teszt használatával nem mérjük a különböző tranzakciós időket, teljesítménytesztet a programozók külön hajtanak végre. Nem célja a tesztnek, hogy a bejelentkezési mezőket teljes egészében tesztelje.

- **Tesztelői csapat**

Ebben a részben soroljuk fel, hogy kik szerepelnek a tesztelői csapatban. A csapatban nem feltétlenül csak az automatizálással foglalkozó tesztelők szerepelhetnek. Ide sorolhatóak akár az alkalmazás programozói is, hogy bárki el tudja érni őket akár a projekt alatt.

Név	Szerepkör
Menedzser János	Automatizálási csoportvezető
Automatizáló Krisztina	Automatizált tesztfejlesztő
Tesztelő István	Manuális tesztelő
Tesztelő Barbara	Manuális tesztelő
Fejlesztő Kornél	Alkalmazásfejlesztő

- **Ütemterv**

Fontos ledokumentálni, hogy mikor minek kell teljesülnie annak érdekében, hogy bárki, aki használja a dokumentumot, tiszta képet kapjon, hogy a projekt, valamint a tesztimplementálás milyen fázisban van.

Fázis	Tervezett dátum	Belépési követelmény	Kilépési követelmény	Teljesítve
Teszt követelmények		Tervezési fázis kész	Tesztelői csoport által felülvizsgálva	
Tesztkörnyezet elérhető		Tesztrendszer frissítve	Alkalmazás elérhető a rendszerben	
Automatikus teszttervezés		Manuális tesztek készen vannak	Automatikus teszterv elkészítve az automatizálási csoporttól	
Automatikus tesztimplementálás		Automatikus teszterv készen van	Automatikus teszt elkészült	
Helyreállítási forgatókönyv		Automatikus teszt kész	Helyreállítási forgatókönyv működik	

Emellett a főbb pontok mellett még számtalan részt tartalmazhat egy automatizált teszterv. Minél részletesebb egy ilyen dokumentum, annál pontosabban tudunk teszteket készíteni. Viszont tartsuk

szem előtt, hogy a dokumentum karbantartása is időbe kerül. Törekedjünk arra, hogy csak a lényeges és legfontosabb elemeket tartalmazza a dokumentáció, ne ennek a dokumentumnak a tökéletesítésével menjen el értékes időnk.

Annak érdekében, hogy az automatizálási folyamataink megfelelően működjenek, folyamatosan ellenőrizni is kell azokat. Célszerű minden egyes tesztimplementálás, tesztfuttatási ciklus után csinálni egy visszatekintést a csapattal és kiértékelni, hogy mi ment jól és mi nem.
Egy ilyen megbeszélés során menjünk végig a csoporttal az alábbi kérdéseken:

Mi ment jól?
Próbáljuk meg összegyűjteni azokat a pontokat, momentumokat, melyek jól történtek és mentek az elmúlt időszakban. Szedjük össze, melyek voltak az erősségeink.

Mi nem ment jól?
Próbáljuk meg összegyűjteni azokat a pontokat is, melyek nem mentek jól. Ide beletartozhat bármi, például: nem volt megfelelő a kommunikáció a programozókkal, nem volt az automatikus teszt megfelelően felkészítve, elromlott az asztali gép, amin futtatni szoktuk a teszteket, stb.

Mi az, amit javítsunk a jövőben?
Ide soroljuk fel azokat, amin javítanunk kell. Például legyen egy B terv arra az esetre, ha elromlik egy futtató számítógépünk, esetleg ha megbetegszik valaki a csapatból, ki az, aki át tudja venni a feladatot.

Az automatizált folyamat kiépítésében nagy szerepet játszanak a dokumentumaink is. Időről időre azokat is meg kell vizsgálnunk, hogy milyen információkat tegyünk még bele, vagy miket nem célszerű már tartalmaznia egy ilyen dokumentumnak.

A folyamatok során figyelnünk és monitoroznunk kell a befektetett időt is. Nyitott szemmel kell lennünk minden tevékenységre, hiszen az eltöltött munkaidők száma is mutatja, hogy valami jól megy, vagy javítani kell bizonyos folyamatokon. Ilyen lehet például az is, hogy célszerű több időt szánni a tesztek tervezésére, mert mindig menetközben derül ki, hogy valamit másképpen kell megoldani.

A megfelelő folyamat kiépítésénél gondoljunk arra is, hogy folyamatosan tanítanunk kell a környezetünket. Mire kell odafigyelnie a csoportunknak?

- Az alkalmazások fejlesztése rohamos léptekben halad, így az automatizálási csapatnak folyamatosan figyelni kell, hogy mikor milyen megoldásokkal tudják támogatni a tesztelési folyamatokat.
- Meg kell tudnunk mutatni, hogy melyek azok a tipikus teszt típusok, amelyekre használni lehet az automatizálást és melyek azok, ahol kevésbé?
- Hol és milyen ponton kell, hogy az automatizálás megjelenjen egy alkalmazásfejlesztési ciklusban?
- Mikor mi várható el az automatizálási csoporttól?
- Milyen korlátok vannak a jelenlegi tesztelési folyamatokban?

- Meg kell tudnunk mutatni, hogy az automatizált tesztelés elsősorban a tesztelési folyamatok támogatásáért, jobbá tételéért használjuk, nem pedig azért, hogy bonyolítsuk a folyamatainkat.
- Automatizálással hatékonyságnövelés a célunk, nem pedig a létszámcsökkentés, hiszen automatizálással nem váltható ki a manuális tesztelés teljes egészében.

5.3. Mitől jó egy tesztautomatizálással foglalkozó csoport?

Az előző részben szó volt az automatizálással foglalkozó csoport lehetséges struktúrájáról. Ebben a fejezetben pár olyan dolgot szeretnék megemlíteni, ami jellemző lehet a csoportra és előnyére válhat.
Minél több idő telik el tesztautomatizálással, annál több tapasztalatot gyűjtenek a csoport tagjai. Elkerülhetetlen, hogy különböző területek tesztelői ne lépjenek kapcsolatba az automatizálási csoporttal. Amint elkezdenek dolgozni közösen a tesztelők, üzleti elemzők, úgy lesz egyre több az információ is az automatizálással foglalkozó tesztelőnek. Előbb-utóbb rálátása lesz az adott üzleti területre, tesztelési folyamatokra. Képes lesz olyan kérdéseket feltenni akár az adott terület tesztelőinek, vagy üzleti elemzőinek, amiből sokkal több információt tud begyűjteni egy sikeres automatikus teszt elkészüléséhez. Hosszabb távon nagy előnye lesz az automatizálással foglalkozó csoportnak, hogy olyan tudás lesz a csapattagoknál, ami az adott üzleti területen van. Ezáltal sokkal szélesebb körben tudják felmérni a szükségleteket, jobb rálátást kapnak az integrációs pontokra és folyamatokra. Ezzel segíteni és támogatni is tudják a más területeken lévő tesztelőket, hiszen nem minden esetben lesz meg pár hasznos információ.

Amennyiben új fejlesztések, vagy a jelenlegi folyamatokban funkcionális változások történnek, nem kell a kezdetektől megérteniük az adott folyamatot, hanem kevesebb időbefektetéssel tudnak automatikus teszteket javítani, karbantartani.

Abban az esetben, ha egy adott üzleti területről nem lesz már elérhető az üzleti elemző, vagy új kolléga érkezik a területre, akkor is eredményesen tudunk visszajelzéseket adni, hiszen ismerjük az adott folyamatokat.

A kommunikáció is hatékonyabb. Egy tesztautomatizálási csoport, ami összefogottan működik, sokkal hatékonyabban tud reagálni a felmerülő igényekre, mintha külön-külön foglalkoznának adott tesztelők szkriptkészítéssel. Nagy előnyt jelent, ha a csoport minden tagja egy adott helyen érhető el, és nem külön országokban. Ez miért is fontos? Számos esetben szükséges gyorsan stratégiai döntéseket hozni. Egy adott helyen lévő csoportot gyorsan össze lehet hívni gyors megbeszélésre, és megvitatni olyan kérdéseket, amire gyorsan kell reagálni. Nem jelentenek akadályt a különböző nyelvek.

[7] Cégeknél, vállalatoknál tapasztalható az, hogy centralizáltan jelennek meg ezek az automatizálással foglalkozó csoportok. A csoportfelelős a használt automatizálási eszközökért, esetlegesen a kiépített keretrendszerekért. A csoport tartja karban és figyeli az aktuális licensz kihasználtságot (abban az esetben, ha licenszelt eszközről van szó), a csoport feladata, hogy karbantartsa az eszközt és a teszteket,

ha a vállalat más területein is használják az eszközt. Ezek azok a dolgok, melyek a költségeket a megfelelő módon tudják kontrollálni, valamint a legjobb módja annak, hogy a tesztek készítésekor a megfelelő folyamatok mentén dolgozzanak a tesztelők. Ezek mellett megjelenik az a tudás és felgyülemlett tapasztalat, ami nélkülözhetetlen egy automatizálással foglalkozó csoportnál. A csoport tudja megmutatni a legjobb megoldásokat, tippeket, trükköket és a hátrányokat is. A tapasztalatok mentén tudnak hatékonyan tesztimplementálási ötleteket és terveket készíteni. Nem minden üzleti terület, vagy tesztelendő alkalmazás kívánja meg ugyanazt az automatikus implementációt. A megszerzett tapasztalatok segítenek nekünk abban, hogy a különböző igényekre milyen hatékony megoldásokat tudunk készíteni. Ezek a folyamatok és a megszerzett tudás segíti folyamatosan az automatizálással foglalkozó csoport előre menetelét.

Amikor formálódik, majd kiépül és megfelelő módon működik egy csoport, akkor képesek lesznek befolyásolni a manuális tesztelők, vagy akár az üzleti elemzők munkáját is. A tesztek tervezésében a közös munka egy folyamatos fejlődés lesz. Az automatizálással foglalkozó tesztelő képes lesz felmutatni azokat a megoldásokat, melyek mentén egy manuális teszt leautomatizálása sikeres lehet. Meg kell mutatni, hogy miként és hogyan lehet hatékony egy teszt.

Előbb–utóbb el kell kezdeni kiépíteni a megfelelő tesztimplementálási folyamatokat és standardokat.

Verziókövetés

Nagyon fontos a tesztek karbantarthatósága szempontjából. Sokféle módon lehet a verziókövetést megvalósítani. Egyik lehetséges megoldás, hogy a tesztjeinket tartalmazó mappát minden egyes tesztimplementálási fázis előtt lementjük. Miért fontos ez? Mivel hosszabb távon sok tesztünk lesz, és elképzelhető, hogy a tesztek egymásba kapcsolódó modulokból fognak felépülni, így egy rosszul megtervezett teszt és modulmódosítás más tesztekben működésbeli hibát okoz. Ebben az esetben jó, ha elérhető egy korábbi verzió egy adott tesztről, függvényről, hogy vissza tudjuk állítani a megfelelő állapotot.

Kódolási standardok

Célszerű dokumentálni, hogy milyen standardok mentén készítjük a tesztjeinket. Mivel a tesztek sok részből tevődnek össze, így minden egyes részre célszerű odafigyelnünk.

Ezek lehetnek:

- tesztek elnevezése

  Célszerű egy adott formula alapján képezni az összes tesztünk nevét.

  [A][TERÜLET][AZONOSÍTÓ]_[KÖRNYEZET]_[TESZT_NEVE]

  A következő példa egy rendelésfeladási teszt elnevezése.
  ARF001_WEB_SimaRendelesFelvitele
  Ahol: RF a rendelés feladás rövidítése.

- változók elnevezése

  Fontos, hogy a változókat egységesen kezeljük.

Változó típus	Előjegyzés	Példa
logikai változó (Boolean)	bln	bln_Megtalaltuk
dátum	dtm	dtm_Start
valós (Double)	dbl	dbl_Tolerancia
Hiba (Error)	err	err_MegrendelesSzam
Egész (Integer)	int	int_Darabszam
Objektum (Object)	obj	obj_Test
Tömb (Array)	arr	arr_Tomb
Sztring (String)	str	str_Keresztnev

10. táblázat Kódolási standardok

- kommentelés

  Fontos rögzíteni, hogy a kódunkban milyen módon használunk kommentezést, hogy nézzen ki az egysoros, hogy a többsoros megjegyzés. Rögzítsük, hogy mikor kell kötelezően kommentelni kódrészletet, mikor nem.

- helyreállítási forgatókönyv

  Definiáljunk minden esetben helyreállítási forgatókönyvet. Mi történik abban az esetben, ha nem várt működés lép fel a tesztelés során? Hogyan állítsuk vissza a tesztelés alatt lévő alkalmazást a megfelelő állapotra, hogy a tesztjeink tovább tudjanak futni?

- naplózás

  Miként és hogyan fogjuk naplózni az adott teszt végrehajtását, hol tároljuk a naplóállományokat, meddig tartsuk meg őket, milyen információkat tegyünk bele egy ilyen állományba?

Dokumentálás
Egységes módon kell tárolni a dokumentációkat a tesztekről. Minden egyes fázist, a tervezéstől az implementálásig és tesztfuttatásig, egy közös, bárki által elérhető helyen kell tárolni.

Természetesen léteznek számos más pontok is a standardokban. Ezek lehetnek teljesen csapatfüggőek.

A különböző üzleti területeken dolgozó tesztelők és üzleti elemzők, akár programozók munkáját is hatékonyabban segíti egy centralizált automatizálással foglalkozó csoport. Bármilyen probléma esetén egyértelműen lehet tudni, hogy kit és hol keressenek a kérdésekkel, észrevételekkel a tesztelők. Nem kell időt vesztegetni azzal, hogy kiderítsék, pontosan kihez is fordulhatnak kérdéssel egy adott teszttel kapcsolatban.

5.4. Tesztek kiválogatásának szempontjai

Miután sikeresen azonosítani tudtuk a megfelelő teszteket, bizonyos szempontok alapján érdemes további vizsgálatokat végezni. Az 5.1-es fejezetben bemutatott gyakorlat nem tartalmaz számos egyéb lépést, hisz sokszor a tesztesetek neve nem tartalmaz megfelelő információt és részletességet. Ezeket be kell gyűjtenünk.

Fontos megtudnunk valamilyen módon, hogy az adott teszt milyen prioritású. Valamilyen módszerrel fel kell tudnunk állítani egy rangsort, hogy milyen tesztesetek fontosabbak másoknál.

Pár fontos dolognak teljesülnie kell abban az esetben, ha egy teszteset automatizálható, mielőtt ténylegesen belekezdünk a tesztimplementálásba. Egy lehetséges megoldás, ha egy kérdésekkel teli listát használva megkérdezzük a fejlesztőket, vagy az üzleti elemzőket a hasznos információk begyűjtése céljából.

Kérdés	Igen / Nem	Megjegyzés
Le van-e már fejlesztve a tesztelendő alkalmazás egésze, vagy valamilyen része?		
Rendelkezik-e stabil felhasználói felülettel az alkalmazás?		
Tervbe van véve olyan fejlesztés az alkalmazáshoz kapcsolódóan, ami a felhasználói felületet, vagy az üzleti logikát érinteni fogja?		
Az alkalmazás elérhető lesz még az elkövetkező egy-két évben?		
Elérhető az alkalmazás valamilyen teszt, vagy fejlesztői környezetben?		
Szükséges valamilyen speciális jogosultság az alkalmazás eléréséhez?		

Van üzleti oldalról felelőse az alkalmazásnak?		
Milyen dokumentációk érhetők el az alkalmazásról?		
Elérhetőek teszttervezéssel kapcsolatos dokumentumok?		
Léteznek már megírt manuális tesztesetek?		
Az alkalmazást rendszeresen tesztelik, vagy csak időszakosan, netalán egyszer kellene tesztelni?		
Mennyi időbe telik a manuális tesztvégrehajtás?		
Pontosan elérhetőek azok a lépések, melyeket ellenőrizni kell a tesztfuttatás során?		
Sok adatkombinációra kell végrehajtani a tesztet?		
Egyszerű az eredmények kiértékelése?		
Különböző platformokon is végre kell hajtani a tesztet?		

11. táblázat Teszt kiválogatásának szempontjai

Ennek a táblázatnak a kitöltése után sokkal pontosabb információnk lesz, hogy mennyi időt is kellene rászánnunk a teszt elkészítéséhez.

A tesztek kiválogatásának szempontjai lehetnek továbbá:

Prioritás
Meg kell vizsgálni, hogy az adott teszt, vagy alkalmazás üzleti szempontból mennyire fontos, mekkora prioritása van. Amennyiben sok tesztet kell végrehajtani egy olyan alkalmazáson, vagy az alkalmazásunk valamelyik modulján, ami kritikus, akkor érdemes odafigyelni és automatizálással is megtámogatni a tesztelést. Érdemes a prioritást is három különböző módon rangsorolni: kritikus, közepes, alacsony prioritás.

Komplexitás
Ki kell deríteni, hogy mennyire egyszerűen, vagy bonyolultan tudjuk megvalósítani az automatizált tesztet. Lehetséges, hogy a manuális tesztvégrehajtást kell egy az egyben megoldanunk automatikus módon. A másik lehetőség, hogy olyan bonyolult üzleti és funkcionális logika van a manuális tesztelésben, hogy sokkal több időt igényel az automatikus teszt elkészítése, mint ellenkező esetben. Célszerű a komplexitást is három opcióval ellátni: komplex, kevésbé komplex, nem komplex.

Modularizálhatóság

Érdemes szem előtt tartani, hogy az automatikus teszt elkészülése után, a különböző modulokat mennyire lehet majd a későbbi tesztkészítésekre felhasználni. Abban az esetben, ha sok információnk van, hogy milyen egyéb tesztek használják majd az adott modult, jobban tudunk erre a kérdésre válaszolni. Tegyük fel a kérdést magunknak: mennyire lehet és érdemes különböző egységekre bontani az adott tesztet? Itt is érdemes három opciót használni: modularizálható, kevésbé modularizálható, nem érdemes modularizálni.

A három szempont értékei alapján hozzunk egy összesített eredményt. Ez az eredmény mutatja meg, hogy melyek azok a tesztek, amelyekkel célszerű foglalkoznunk.

A különböző tesztek vagy folyamatok automatizálása eltérő időt vehetnek igénybe. Tartsuk szem előtt, hogy **ne próbáljunk meg mindent egyszerre automatizálni**, csináljuk hatékonyan, lépésről lépésre. Az automatizálás során számos olyan dologba ütközhetünk, melyek ismeretében a fent meghatározott prioritás, komplexitás, modularizálhatóság opciókat felülírhatják. Célszerű időről időre felülvizsgálni az adott tesztesetek értékeit.

Folytassuk a tesztek kiválogatásainak szempontjait.

Költségvetés

Igen, költségvetés. Sokan nem is gondolnák, hogy ez is szempont lehet. Nem mindig evidens, hogy rendelkezésünkre áll egy tesztautomatizáló eszköz. Elképzelhető, hogy olyan alkalmazásokat kell tesztelnünk, amire szükségünk lesz beszerezni valamilyen speciális szoftvert, és bizony annak költsége van. Rendelkezünk a megfelelő forrással? Érdemes költenünk a kiszemelt eszközre csak azért, hogy valamennyi tesztet leautomatizáljunk? Lesz annyi hozadéka a befektetett munkának, hogy megéri beleinvesztálni egy ilyen beruházásba? Ez is okozhatja azt, hogy le kell mondanunk pár teszt vagy tesztfolyamat automatizálásáról.

Ütemterv, időbeosztás

Ki kell derítenünk, hogy az automatikus tesztünket mikorra kell elkészítni, mennyi időnk van rá. A folyamat komplexitása és az automatikus teszttervezés során valamilyen becslést tudnunk kell mondani, hogy mennyi időbe fog kerülni az implementálás és tesztelés. Össze kell ezt vetnünk azzal az ütemtervvel, amikorra szükséges a tesztnek elkészülnie. Amennyiben nem áll rendelkezésünkre a teszt elkészüléséhez megfelelő idő, lehetséges, hogy nem a legjobb megoldás azonnal elkezdeni dolgozni rajta. Viszont, ha kritikus területről és tesztről van szó, akkor mérlegelnünk kell. Lehetséges, hogy több erőforrásra lesz szükség a teszt elkészítésére. Ebben az esetben is változatlanul fel kell tenni a kérdést: megéri ilyen körülmények között lefejleszteni a tesztet? Hogyan néz ki a befektetés megtérülés becslésünk? Nem lenne jobb akkor foglalkozni a teszttel, amikor több időnk lesz rá? Érdemes más potenciális tesztet emiatt nem lefejleszteni?

Szakemberek

Automatizált szoftvertesztelés nem létezik szakemberek nélkül. Meg kell vizsgálni, hogy az automatizálással foglalkozó tesztelőknek megvan-e a megfelelő tudásuk ahhoz, hogy lefejlesszék a teszteket? Ezzel nem azt szeretném mondani, hogy képzetlenek, hanem arra kell gondolni, hogy a mai világban hihetetlen léptékkel történnek a fejlesztések, melyekkel jobb, ha tisztában vannak. Elképzelhető, hogy olyan technológiával készülnek az alkalmazások, melyekhez nincs meg a megfelelő tudásunk. Ekkor meg kell ismernünk az adott technológiát, képezni kell a szakembereinket és azután tudunk hatékonyan megoldásokat szolgáltatni.

Újrahasználható modulok

Célszerű kiemelni ezt a szempontot. Igaz, hogy a fenti példában szerepelt, de vizsgáljuk meg más oldalról is. Tegyük fel, hogy nagy prioritású az alkalmazás, amire automatikus tesztet kell legyártanunk. Lehetséges, hogy nincs értelme a teszt során modulokat létrehozni, mert minimális tesztlépésről beszélünk. Vizsgáljuk meg, hogy az a folyamat, amit leautomatizálunk, rendszerszinten, integrációs teszt szempontból mennyire újrahasznosítható. Lehetséges, hogy olyan központi modulról van szó, amin keresztül más üzleti folyamatok is működnek. Ekkor az integrációs teszt és újrafuttatási szempontok alapján kifejezetten célszerű leautomatizálni. Ezzel más tesztesetek implementálásához is hozzájárultunk.

Manuális tesztelői erőforrás csökkentése

Az automatizálással foglalkozó szakember feladata az is, hogy megvizsgálja, hol lehet a manuális tesztelői erőforrást csökkenteni. Sokan felkapják a fejüket erre, és úgy tekintenek erre a pontra, mint egy veszélyforrásra. Ezzel nem az a cél, hogy a manuális tesztelést és az azzal foglalkozó tesztelők létszámát csökkentsük. Itt arra kell gondolni, hogy miként lehet segíteni a manuális tesztelőket, hogy sokkal fontosabb és kritikusabb tesztelést hajtsanak végre. Nézzük meg újra az 5.1-es fejezetben lévő párbeszédet. Ott arra derült fény, hogy több adatra is hasonlóképpen kell végrehajtani a tesztet. A tesztelést viszonylag kevés tesztadatra hajtották végre a manuális tesztelők. Automatikus tesztet használva viszont a lefedettség növelhető, hiszen minél több adatra meg tudjuk nézni a tesztet, annál biztosabbak lehetünk, hogy az alkalmazásunk jól működik. Ezzel viszont manuális erőforrást csökkentünk, hiszen nem a tesztelőnek kell végrehajtani egyesével a teszteket.

Automatikus teszt végrehajtásának rendszeressége

Meg kell néznünk azt is, hogy milyen rendszeresen lesz használva az adott teszt. Itt is több lehetőség van. Elképzelhető, hogy minden egyes verziónál le kell futtatnunk a tesztet. Ekkor meg kell vizsgálni, hogy milyen gyakran történnek meg verzióváltások. Lehetséges, hogy csak egy egyszeri tesztimplementálásról és futtatásról lenne szó. Ekkor a befektetést és megtérülést kell szem előtt tartani.

<u>Erre egy példa:</u>

Az alkalmazásunkban három mezőt kell kitöltenünk, majd azt elküldeni egy űrlapon. Nem sok lépés, viszont ezt háromezerszer meg kell csinálni, egy adott időintervallumon belül. Manuálisan ez lehetetlen lenne. Egy egyszerű, szimpla megoldással sokkal hatékonyabban tudunk reagálni ilyen tesztelési folyamatokra, mint manuálisan.

Alapvető szempontoknak tekinthetőek a fentiek. Emellett biztos vagyok benne, hogy különböző tesztelési stratégiák, csoportok, munkafolyamatok, fejlesztési stílusok más-más szempontokat tartanak szem előtt. Érdemes ezekre a szempontokra a válaszokat kideríteni az adott terület tesztelőivel, üzleti elemzőivel, esetleg programozóikkal. Mivel az esetek többségében egy adott projektről nem áll rendelkezésünkre a megfelelő információ, így ők azok, akikhez tudunk fordulni és ezek alapján pontosabb képet kapni az automatikus teszt igényekről. Azért is jó, ha valamilyen szinten kapcsolatban vagyunk a területen lévő emberekkel, mert mi is tudunk olyan javaslatokat adni akár tesztelési folyamatok terén, ahol sokkal hatékonyabb automatikus tesztimplementálás és végrehajtás tud elkészülni. Elkerülhetetlen, hogy meghallgassuk és elfogadjuk a javaslataikat. Ugyanakkor a tesztautomatizáló oldalról pedig feladatunk, hogy a folyamatok hatékonyabbá tétele miatt mi is javasoljunk hatékony megoldásokat.

5.5. Kontrollált szkriptkészítés és implementálási folyamat

A megfelelő tesztautomatizálás bevezetéséhez és annak sikerességéhez nagyban hozzájárul, hogy milyen módon milyen folyamatok mentén készítjük a tesztjeinket. Kezdetben lehetséges, hogy nem gondolkozunk folyamatokban. Mivel számunkra is új még az automatizálás területe, így nagyon erősen csak a tesztek létrehozására fókuszálunk, valamint arra, hogy működőképes megoldásokat implementáljunk. Hosszabb távon (amennyiben automatizálással foglalkozó csoport is létezik a vállalatnál) elkerülhetetlen, hogy a tesztek tervezésére és kivitelezésére folyamatokat alkalmazzunk.

A jelenlegi trendek azt mutatják, hogy nagyon elterjedtek az agilis szoftverfejlesztési modellek. Ennek megvannak a maga előnyei és hátrányai is. Számomra az elmúlt évek azt mutatják, hogy tesztautomatizálásra és tesztek implementálására nagyon hatékony megoldást nyújtanak az agilis szoftverkészítési folyamatok.

Elképzelhető, hogy a SCRUM módszertannal már találkozott az olvasó. A SCRUM és felépítése nagyon jó előnyöket kínál a fejlesztői csapatoknak. Abban az esetben, ha a vállalatnál működő alkalmazásfejlesztési folyamatok nem agilis rendszerben történnek, attól még lehet hatékony egy agilis automatikus tesztkészítési folyamat. A kettő nem zárja ki egymást, nagyon jól tudnak együttműködni ilyen formában is.

A SCRUM módszer felépítése és leírása nem célja ennek a tananyagnak, viszont érdemes bemutatni és később kicsit részletesebben tárgyalni azokat az elemeket, melyekkel célszerű kibővíteni az eljárásainkat annak érdekében, hogy az automatizálási procedúráink még hatékonyabbak legyenek.

Alapvető SCRUM elemek és céljaik:

- Terméklista
  Egységes, központi helyen tároljuk azokat az igényeket, melyeket a csapatunk által szeretnének a megrendelőink megvalósítani. Tartalmaznia kell ennek a listának, hogy az adott terméknek (automatikus tesztnek) röviden mi a célja, mit kell magában foglalnia. Itt kell számon tartani,

hogy az adott kérésnek milyen prioritásai vannak. Már itt kell látni, hogy az egyes igények fontossága alapján milyen sorrend mutatkozik. Ezt a listát érdemes naprakészen tartani, hiszen ezzel lehet mutatni, hogy jelenleg mekkora igény van tesztek automatizálására. Jelöljük azt is, hogy ki juttatta el számunkra az adott kérést, hogy kérdés esetén el tudjuk érni a megfelelő kollégát.

- Sprintlista
  A SCRUM része az a folyamat, amit sprintnek hívunk. A sprintben történik a tesztek készítése, maga az implementálás. Mielőtt elkezdődne egy ilyen ciklus, egy elég részletes megbeszélés történik az adott kéréseket beküldők, az automatizálási csapat tagjai, és néha a menedzserek között. Közös döntés alapján lesznek kiválogatva azok a tesztek, melyek automatizálásra kerülnek. Ezeket az igényeket külön gyűjtjük egy sprintlistába és a továbbiakban csak ezzel a listával dolgozunk. Itt megjelenik, hogy a tesztautomatizáló csoporton belül kihez lett hozzárendelve az adott teszt, mennyi időt kell a teszt készítésére szánni, valamint, hogy mennyi ideig is tart az adott sprint.

- Sprint
  Az a része a SCRUM-nak, amikor az automatizálással foglalkozó szakemberek fejlesztik a teszteket. Az előzetes, alapos tervezés mentén tudják, hogy éppen milyen elemekkel kell foglalkozni. Célszerű úgy megállapítani az egyes tesztek és azok moduljainak tervezését, hogy a rajtuk végzendő szükséges munka ne legyen több nyolc óránál. Ez fontos, hiszen napi szinten követjük, hogy ki mivel foglakozott, ki mivel megy tovább, így pontos képet kapunk az aktuális sprint előrehaladásáról.

- Napi SCRUM megbeszélés
  Minden nap egy rövid megbeszélés során mindenki beszámol arról, mivel foglalkozott az elmúlt napban, mivel fog foglalkozni a következő napon, valamint van-e olyan dolog, ami hátráltatja a munkában. Ezen a megbeszélésen ellenőrzik, hogy tartható-e az az ütemterv, amit beterveztek a tesztimplementálási folyamat elején, vagy sem.

- DEMO
  Amikor elérkezett a csapat a sprint végéhez, egy tesztfuttatás történik az automatikus tesztet kérő megrendelőknek. Ekkor ténylegesen futtatni és használni fogják a legyártott automatikus tesztet, hogy meggyőződjenek arról, úgy és olyan formában működik a teszt, mint ahogy az elvárt volt.

A fenti részek csak érintőlegesen lettek bemutatva, viszont látszik, hogy kontrolláltan, folyamatok és fázisok mentén történik a tesztek készítése. A DEMO alkalmával akár menedzsereket is meghívhatunk. Ez fontos, hiszen ekkor tudjuk megmutatni nekik is, hogy miért érdemes és hogyan lehet hatékonyan automatikus teszteket használni és készíteni.

Viszont azt tapasztaltam számos tesztautomatizálással eltöltött év alatt, hogy célszerű más módszereket is használni a SCRUM keretén belül. Ilyen az extrém programozás. Véleményem szerint az extrém programozás tulajdonságainak használatával még hatékonyabban tudjuk bevezetni a folyamatainkba az automatizálást.
A megszokott szoftverfejlesztési folyamat első fázisa azt célozza meg, hogy elkészüljenek a megfelelő követelmények. Azonban tudjuk, hogy számtalan esetben a követelmények hiányosak, nem pontosak, és akkor derülhetnek ki hasznos és lényeges információk, amikor már az implementálási fázisban vagyunk. Minél később derül fény a követelmények szintjén egy változtatásra, annál költségesebb tud lenni a fejlesztés.

[8] [9] Az extrém programozás ezzel szemben más gyakorlatot mutat, és olyan megoldásokat használ, melyekkel a felmerülő költségeket csökkenteni lehet. Az extrém programozás öt alapvető érték köré épül:

1. **Kommunikáció**
   Mindenki tagja a csapatnak. A csapat érdekei közösek és ezt mindenkinek tudnia kell. Folyamatosan kapcsolatban vagyunk egymással, és ez nem csak a tesztautomatizálási csapatot érinti. Ide tartoznak a tesztelők, üzleti elemzők, más programozók stb. Bárki elérhető, és információval tud szolgálni másoknak. A követelmények kidolgozásánál is együttműködnek a résztvevők, hiszen annak érdekében, hogy a leghatékonyabb automatikus teszt legyen implementálva, szükséges az adott terület felelőse és szakértője is. Megpróbáljuk közösen megalkotni a legjobb megoldást az adott igényre. Fontos, hogy a tesztautomatizáló a lehető leghamarabb megkapja a szükséges információkat, valamint a csapat minden tagja egységesen lássa a rendszert.

2. **Egyszerűség**
   A tesztautomatizáló azzal fog foglalkozni, amivel kell és semmi többel. Automatikus teszt készítésekor nem arra törekszünk, hogy a legszebb megoldással álljunk elő, hanem arra, hogy maga a megoldás kerüljön implementálásra. Nem töltünk több időt annál, mint ami szükséges egy adott teszt elkészítéséhez. Ha tudjuk, hogy a jövőben milyen követelmények lesznek, még akkor se feltétlenül kezdjünk dolgozni már azokon a funkciókon, hiszen addig bármikor változhat az adott követelmény, és ezzel a tesztünk is. Mindig csak az adott lehetőségekkel számolunk. Olyan megoldásokat készítünk, melyek hosszútávon, minimális ráfordítással karbantarthatóak. Ne készítsünk és implementáljunk olyan funkciót egy tesztbe, amit később nem fogunk kihasználni, hiszen ebben az esetben később jóformán alig fog valaki emlékezni, hogy mi is volt az és kód karbantarthatóság szempontból nehézséget okozhat.

3. **Visszacsatolás [9]**
   A visszacsatolás több szinten következhet be:

   <u>Rendszerszintű visszajelzés:</u>

A különböző egységtesztek használatával a fejlesztők tudni fogják, hogy milyen állapotban van a tesztelendő alkalmazás. Amennyiben van már elkészült tesztünk, futtatásokkal tudjuk ellenőrizni, hogy működőképes-e az adott teszt, vagy valamilyen változás történt az alkalmazásban. Miért fontos ez? Bárki dolgozhat az adott rendszeren és történhetnek módosítások. A tesztjeinknek összhangban kell lennie ezekkel, és gyakori tesztfuttatásokkal meggyőződhetünk arról, hogy nem történt változás az alkalmazásban.

A tesztelőtől, üzleti elemzőtől érkező visszajelzés:

A tesztelővel, üzleti elemzővel - vagy azzal a kollégával, akitől éppen érkezett az automatikus teszt iránti igény - való szoros együttműködés elősegíti azt, hogy gyorsan és hamar érkezzenek észrevételek. Már az automatikus teszt készítése során próbáljuk meg feltérképezni azokat a pontokat, melyekre nem derült fény a tervezési folyamat során. Ezt csak akkor tehetjük meg a leghatékonyabban, ha rendszeresen megmutatjuk, jelenleg hol is tartunk az implementálásban.

Fejlesztői szintű visszajelzés:

A visszajelzések nem egyoldalúak. Mi, automatizálással foglalkozó szakemberek is szolgáltatunk visszajelzéseket másoknak. Amint módosítás, vagy új funkció iránt van igény, mi - mint tapasztalt tesztkészítők - gyorsan tudunk véleményt adni az észrevételeinkről. Próbálunk hatással lenni a többiekre, hogy mit miért lenne célszerű másként csinálni, mint ahogy azt gondolják.

4. **Bátorság**

A különböző igényekre a lehető legoptimálisabban kell reagálnunk. Feladatunk, hogy egy adott automatikus teszt elkészüléséhez szükséges időt és erőforrást meg tudjuk becsülni. Nem szabad megijedni ezektől a felelősségektől. Azáltal, hogy ezeket meg tudjuk mondani és folyamatosan kezeljük is ezeket, a csapatunk profibbá fog válni. Mások szemében is egy határozott, célorientált csapatnak fogunk látszani. Folyamatosan a megfelelő állapotot kell bemutatnunk az adott tesztkészítésről. Nem próbálunk mást mutatni, mint ami van. Nem törődünk kifogásokkal, nagyon határozottan a sikeres tesztimplementálásra fókuszálunk. Nyitottak vagyunk a változásokra. Legyen bátorságunk megmutatni, hogy szerintünk min kellene javítani, hogy még hatékonyabb legyen egy adott automatikus teszt. Végül az egyik legnehezebb dolog: legyen bátorságunk egy meglévő tesztre azt mondani, hogy nem fogjuk használni tovább, mert nem éri meg.

5. **Tisztelet**

Mindenki megpróbál tisztelettel lenni a másikkal szemben. Ez nemcsak az automatizáló csapat tagjaira érvényes. Tiszteljük a tesztelőket, üzleti elemzőket, programozókat. Ha valamilyen igényre nem derül fény, akkor nem hibáztatunk senkit. Közösen dolgozva tudunk eljutni a célhoz. Mások tapasztalata minket is segít és inspirál.

[8] [9] A gyakorlatban történő extrém programozás során számos fogalommal találkozhatunk, melyeknek célszerű a jelentését bemutatni.

- Inkrementális tervezés
  Komplexebb igények esetén próbáljuk meg a feladatokat felbontani és ezeket folyamatosan, lépésről lépésre implementálni. Ne próbáljuk meg a bonyolult folyamatot nagy falatként implementálni. Vizsgáljuk meg közösen az ügyféllel, hogy pontosan miről is van szó, mikor mire van szükség, és apró lépésekben érjük el a megoldást.

- Minimális befektetés
  Azokat a teszteket készítsük el először, melyeket gyorsan, kevés befektetéssel tudunk megcsinálni, viszont az üzleti igényeket a lehető legnagyobb hatékonysággal szolgálja ki.

- Egyszerű tervezés
  Csakis azt implementáljuk, amire szükség van, nem többet.

- Teszt átírás, átdolgozás (refactoring)
  Törekedjünk arra, hogy a modularizáltság felé haladjunk. Egy adott teszt implementálása során is vizsgáljuk meg, melyek azok a részek, melyeket külön egységekbe, eljárásokban tudunk tárolni, hogy a későbbiek folyamán mások, vagy akár mi is, újrahasznosíthassunk. Elképzelhető, hogy egy adott teszt készítésekor nem látjuk, hogy bizonyos modulok máshol is szerepelnek majd. Ahogy egyre több tesztet készítünk, tudnunk kell, hogy mely teszteket kell újraterveznünk, átdolgoznunk, hogy a kód duplikációját elkerüljük, ezáltal újrafelhasználható részeket kapjunk.

- Páros programozás
  Nagyon érdekes módszer. A lényege, hogy egy gépnél ül két automatizálással foglalkozó szakember és egymás munkáját segítik. Hol az egyiknél van a billentyűzet, hol a másiknál. Aki éppen nem a tesztimplementáláson dolgozik, követi, hogy mit csinál a másik. Ez majdhogynem egy folyamatos kódellenőrzés a másik féltől. Az aktuális megfigyelőtől az elvárás, hogy szem előtt tartsa a kódolási standardokat, kódformázást, megfelelő részegységek kidolgozásának felügyeletét. Az ilyen jellegű működés sokaknak jelenthet nem megfelelő munkafolyamatot. A módszer használata nagyon sok előnnyel jár annak ellenére, hogy a felhasználónak idegennek tűnhet. A fejlesztők arról gondoskodnak, hogy a megfelelő minőségű teszt készül el. Nem kell úgy megélni az ilyen együttdolgozást, hogy az egyik fél oktatja a másikat. Közös a cél, de tény, hogy a tapasztaltabb tesztautomatizáló ezáltal segíti és tanítja is a másikat. Nem mindig fognak ugyanazok a párok együtt dolgozni egy adott teszt implementálásán. A csoporttagok dinamikusan váltakoznak, így folyamatosan más-más csoporttagok dolgoznak együtt. Ezzel javítjuk a csapat szellemét, egységét. Elérhetjük azt, hogy jobb kódminőséget kapunk egy tesztben. Mivel mással kell közvetlenül együttdolgoznunk, így a fegyelemérzet is nagyobb. Tiszteljük a másikat, és építő jellegű kritikákkal szolgálunk egymásnak, amivel a fejlődést célozzuk meg. Mivel különböző tulajdonságú egyének vannak a csoportban, így ez a fajta munka lehet, hogy élvezetesebb kihívásokkal szolgál. Gondoljunk bele, hogy a tesztautomatizáló csapat dinamikusan változik. A páros programozás lehetővé teszi a csapattagok jobb megismerését, könnyebb integrálhatóságát. Kevesebb munkamegszakítást is eredményez ez a fajta módszer. Mivel többen dolgozunk egy teszten, így nem okoz időkiesést, ha valaki éppen reggelizni megy,

mosdóba, vagy egy pohár vízért. Számos cikk megemlíti, hogy fele annyi munkaállomást is kíván egy ilyen munkamódszer, de véleményem szerint nem ez a célja ennek a módszernek. Mivel sok igény és teszt jelentkezhet, így elkerülhetetlen, hogy a mobilitást fenntartsuk és elérhetővé tegyük az automatizálással foglalkozó szakembert más területeken is. Ehhez viszont infrastruktúrára van szükség.

- Közös tulajdonjog
  A tesztautomatizálók nemcsak egy adott területre készítenek teszteket. Elképzelhető, hogy a páros programozás során akár párhuzamosan fejlesztenek, de olyan módon, hogy valamikor, a tesztimplementálás valamelyik fázisában a két különböző tesztnek össze kell kapcsolódnia. Különböző fejlesztők által írt tesztek együtt is működőképesek kell, hogy legyenek. Mivel egy automatizáló szakember nemcsak egy területre fejleszt, és közösen is dolgozik más tesztautomatizálóval, így egy adott teszt közös is lehet. Bárki megváltoztathat a tesztben bármikor bármit. Nem kell attól tartani, hogy ha betegség miatt nem lesz elérhető egy automatizáló szakember. Ekkor a csapatból bárki képes lesz megfelelően reagálni az igényekre.

- Keretrendszerbe való integrálás
  A tesztimplementálás során, amikor feltérképeztük, hogy melyek azok a modulok, amelyeket célszerű külön kezelni, amint csak lehet, emeljük ki azokat és helyezzük el a keretrendszerünkbe a megfelelő helyre. Ezáltal lehetővé tesszük más tesztkészítőknek az adott modul elérését. Ezen felül még azt is jelenti az integrálás, hogy amint egy teszt teljes egészében elkészült, azt tegyük elérhetővé és használjuk, futtassuk. Ezzel meggyőződünk arról, hogy a tesztjeink a későbbiek során nem sérültek, és hogy az alkalmazásunk is a megfelelő állapotban van.

- Fenntartható munkamenet
  Tervezzük és becsüljük meg úgy egy teszt elkészüléséhez szükséges időt és erőforrást, hogy az ne okozzon többletmunkát. Az nem vezet hatékony tesztkészítéshez, ha irreális munkamennyiséget fordítunk egy tesztre és ezáltal akár túlórát is generálunk. Nem attól lesz valami jó, ha több időt fordítunk rá, mint ami szükséges. Valamint ne támasszunk lehetetlen elvárásokat a csapatunk többi tagjával szemben sem.

- Elérhető ügyfél
  Az ilyen típusú tesztkészítés során fontos, hogy kérdés felmerülése esetén elérhető legyen az ügyfél, vagy legalább egy olyan kolléga, aki képes az adott teszthez kapcsolódó információkkal szolgálni a tesztautomatizáló csoportnak. Mivel gyorsan próbálunk megoldást kínálni, elképzelhető, hogy olyan kérdések merülnek fel, melyekre korábban nem tudtunk felkészülni, így szükséges az adott terület szakemberével a gyors konzultáció. Az automatikus teszt készítése előtt tudatnunk kell a környezetünkkel és az ügyféllel, hogy ők bizony a csapat részei. Nemcsak a kezdeti igényfelméréskor és a teszt lefejlesztésekor van rájuk szükségünk. Mivel az ő érdeküket is a teszt elkészülése, így elérhetőnek kell lenniük az automatikus teszt fejlesztésével foglalkozó szakembernek.

A fenti példák mutatják, hogy milyen folyamatok és elvárások szükségesek ahhoz, hogy megfelelően tudjunk implementálni egy automatikus tesztet. Ezeknek a kiépítése és hatékony alkalmazása függ attól, hogy milyen környezetben szeretnénk használni a tesztautomatizálást. Egyes vállalatoknál működő munkafolyamatok eredményezhetnek gyors sikereket, viszont létezhet olyan szigorú alkalmazásfejlesztési és tesztelési folyamat egy cégnél, ahol több időt kíván meg egy hatékony folyamat bevezetése. Mindenképpen tartsuk szem előtt, hogy tesztimplementálás előtt, közben és után folyamatosan figyelnünk kell ezeket a faktorokat.

Említsük meg végül, hogy a tesztimplementálás mellett fontos szem előtt tartanunk a tesztek tesztelését is. Mivel folyamatosan együtt dolgozunk az ügyféllel, így menet közben is kiderülhet, hogy a tesztünk nem feltétlenül azt csinálja, ami az elvárt. Teszteljük rendszeresen a tesztünket, hogy időben kiderüljön, mit és hogyan kell javítani a megfelelő megoldás érdekében.

6. Automatizált szoftvertesztelés a gyakorlatban

6.1. Teszteset detektálás

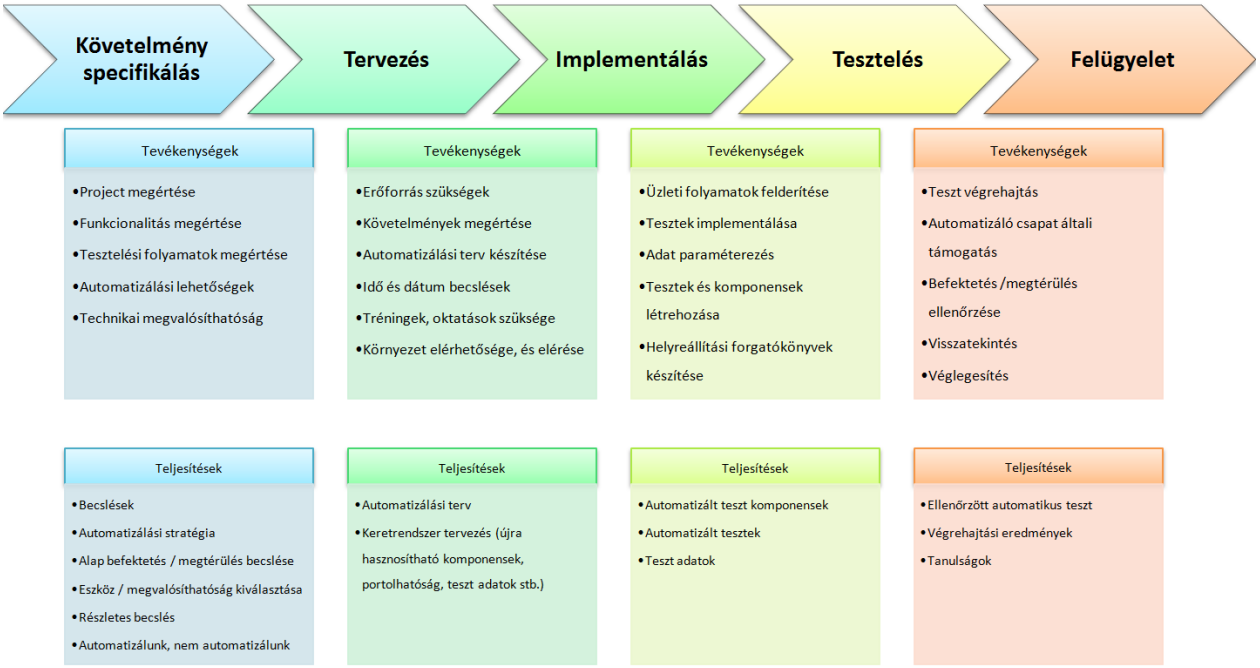
Tegyük fel, hogy elkezdődik egy projekt, melynek keretében bizonyos üzleti igényeknek megfelelően egy alkalmazást fejlesztenek le a programozók. Attól függetlenül, hogy ez egy vízesés, vagy agilis módszertant követő fejlesztés lesz, biztosan megjelennek a projekt során az alábbi fázisok:

- Követelményspecifikálás
- Tervezés
- Implementálás
- Tesztelés
- Felügyelet

Feltételezem, hogy az olvasók többsége ismeri a vízesés modellt. Amennyiben pontosabb információra lenne szükség, azt a *Szoftvertesztelés a gyakorlatban* oktatási anyagban megtalálja a *Szoftverfejlesztési modellek* fejezetben. [1]

Az automatizált szoftvertesztelésre is – mint a manuális tesztelésre is – igaz, hogy minél hamarabb kerül bevonásra a tesztautomatizálási csoport, annál hamarabb és pontosabb, jobban megtervezett teszteket lehet készíteni. Azonban - a manuális teszteléssel szemben, - nagyon eltérő tevékenységek, és teljesítések vannak az alkalmazásfejlesztés különböző pontjain.

Vizsgáljuk meg, hogy a különböző fázisokban melyek azok az alapvető dolgok, amelyeknek teljesülnie kellene egy automatizált tesztimplementálási folyamat során. Az alábbi ábrán az alkalmazásfejlesztési életciklus (felső rész) és az automatizálási folyamat részei (alsó rész) láthatóak. Figyeljük meg, hogy a két különböző folyamat egymáshoz képest átfedésben van. Ez nem véletlen, hiszen az automatizáló csoport akkor tud foglalkozni a megfelelő részekkel, ha azok teljesítve vannak az üzleti elemzők és a programozók által.



1. ábra Automatizálási folyamat kiépítése

A követelmények specifikálásának befejezése és a tervezési fázis elkezdése után zajlódó automatizálással kapcsolatos tevékenységek listája

- Projekt megértése

Tartsuk szem előtt, hogy mielőtt bármit is elkezdenénk automatikus módon implementálni, szükséges begyűjtenünk azokat az információkat, melyekre szükségünk van, hogy megértsük, miről is szól az adott projekt. Azért fontos ez, hiszen sem az üzleti elemzők és fejlesztők, sem a manuális tesztelők nem úgy gondolkoznak és dolgoznak, mint ahogy az elvárható egy tesztautomatizálással foglalkozó csoporttól. Lehetséges, hogy a követelmények specifikálása után a tesztelők más eseteket fognak felépíteni, mint egy automatizálással foglalkozó szakember. Szükség van arra, hogy valaki ismertesse a projekt lényegét és célját az automatizáló szakemberrel annak érdekében, hogy egy általános képet tudjon kapni. El kell tudni helyezni a projektet aszerint, hogy milyen célt szolgál az adott fejlesztés, és hogy kik is pontosan az ügyfelek, akiknek készül.

- Funkcionalitás megértése

A következő lépcső, hogy megértsük, milyen funkcionalitások is kerülnek bele a rendszerbe. Itt különböző lehetőségek vannak, viszont általában két dolog szokott történni. Nagyvonalakban ismerteti velünk a projektcsoportból valaki, hogy milyen működést várnak el az alkalmazástól. Azaz hogy melyek azok a funkciók, amelyek implementálásra kerülnek és milyen módon, milyen elvárásokkal. Mivel a követelmények specifikálásán már túl vagyunk, így nyugodtan próbáljuk meg beszerezni magát a dokumentumot, amiben rögzítették ezeket. Menjünk végig a pontokon, próbáljuk meg közösen

értelmezni azokat – amennyiben van rá lehetőség – a projektben résztvevőkkel. Amennyiben kérdésünk van, tegyük fel, pontosítsunk, hiszen ez egy igen kritikus pontja a későbbi sikeres tesztautomatizálásnak.

Sokszor előfordul, hogy egy követelmény nem tartalmazza a szükséges információkat. Abban az esetben, ha olyan szakember végzi a tesztek automatizálását, akinek több éves tapasztalata van, akkor tehet fel olyan kérdéseket, amivel már az automatikus teszt felépítésére is lehet következtetni.

Az alábbi párbeszéd egy tesztelő és egy automatizálási szakember között zajlik le. T – tesztelő, A – automatizáló:

T: Szia, Kornél, örülök, hogy összejött ez a találkozó. Sikerült végigolvasni a specifikációkat?
A: Szia, Krisztina. Igen. Meg kell vallani, jól összeszedett lista.
T: Köszönjük szépen. Igyekeztünk precízen megfogalmazni az elvárásainkat, hogy a későbbi teszttervezést is megkönnyítsük.
A: Gondolom, nem véletlen, hogy szeretnétek automatikus teszteket is használni. Pontosan mire is lenne szükségetek?
T: Nos, a követelmények között a huszonhármas és a huszonötös az, amire úgy gondoltuk, automatizálható. A lényeg, hogy különböző hivatkozásokat kellene meghívni internetes böngészőben, majd az oldal betöltésének idejét kellene valamilyen módon rögzítenünk. Egy háttér funkciót fejlesztünk a rendszerbe, ami lehetséges, hogy gyorsítaná a meglévő oldalak betöltését.
A: Értem, és hogy gondoltátok, mire is lenne pontosan szükségetek?
T: Jó lenne egy olyan teszt, ami az általunk megadott hivatkozásokat meghívja és az oldal betöltését méri.
A: Ez nem tűnik nehéz feladatnak. Viszont tennék fel pár kérdést, mert volt példa rá, hogy amikor mi már elkezdünk automatikus teszteken dolgozni, akkor derült ki pár olyan dolog, amire nem gondolt az ügyfél.
T: Mire gondolsz?
A: Kérdésem: a hivatkozások meghívása tényleg a böngészőben kell, hogy történjen, vagy elég, ha böngésző nélkül meghívjuk a linkeket és mérjük a válaszidőt?
T: Mindenképpen szükséges, hogy böngészőben nyissuk meg a linkeket.
A: Rendben. A következő kérdésem: A különböző hivatkozások meghívása egymás után történik. Viszont szükséges, hogy legyen böngésző nyitás és zárás a folyamatban?
T: Ez jó kérdés. Igazából lehet, hogy jó megoldás lenne a böngésző bezárása, mert a böngészőben lévő sütiket törölni kell minden hivatkozás meghívás előtt.
A: Igen, ez lett volna a következő kérdésem. Igazából a sütik törlését meg lehetne oldani másképpen is.
T: Milyen előnyökkel jár, ha minden esetben böngészőt nyitunk és zárunk?
A: Nos, picit lehet lassabb lesz a teszt futása, viszont ha bármilyen ponton nem várt esemény következne be, lehet egyszerűbb felkészíteni a tesztet arra, hogy böngészőt zárjon. Ugyanis a teszt legelső pontja az lenne, hogy nyissa meg a böngészőt.
T: Értem. Akkor maradjunk abban, hogy böngészőt nyitunk és zárunk.

A: Szuper. Következő kérdés. Mindegy milyen böngészőt használunk? Elég, ha Internet Explorerben hajtjuk végre a tesztet?
T: Erre nem is gondoltam. Én azt hittem, hogy ha egy teszt elkészül, akkor azt bármilyen böngészőben lehet használni.
A: Nem feltétlenül.
T: Nos, jó lenne, ha Explorerben és Firefoxban is működne a teszt.
A: Rendben, megvizsgáljuk, hogy mit lehet tenni.
T: Bármikor lehet majd futtatni a teszteket?
A: Milyen rendszerességgel lenne szükségetek eredményekre?
T: Terveink szerint, mivel a programozók háromnaponta fognak javításokat és módosításokat lefejleszteni, így úgy gondolom, hogy amint kikerülnek a tesztrendszerre a funkciók, lehetne is alkalmazni a teszteket.
A: Igen, úgy vélem, hogy ez jó terv. Amikor új funkciók kerülnek ki a rendszerbe, célszerű megbizonyosodni, hogy azok nem indukáltak problémát meglévő funkciókba.
T: A huszonötös követelmény pedig az lenne, hogy a rendszernek párhuzamosan el kellene bírnia ezer felhasználói szimulálást.
A: Tehát párhuzamosan szeretnétek meghívni a különböző linkeket ezer szálon?
T: Igen, és úgy mérni a terhelést.
A: Nos, ezt a jelenlegi helyzetben nem biztos, hogy meg tudjuk valósítani. Sajnos nem rendelkezünk a megfelelő hardverekkel, és nem tudunk felhasználókat sem szimulálni ilyen mennyiségben. Még mielőtt megkérdeznéd: ha egy futtató állomáson akarnánk párhuzamosan elindítani a teszteket, nem megfelelő mérési eredményeket okozhat a módszer. Ugyanis a tesztek az adott gép erőforrásait fogják használni, és bizony a memóriatelítettség okozhat lassabb eredményeket.
T: Értem, teljesen világos. Akkor maradjunk annyiban, hogy az közte marad a követelményeknek, és majd megpróbáljuk kideríteni, miként tudjuk ezt kivitelezni.
A: Ezt szerettem volna kérni én is.
T: Rendben, köszönöm, akkor egy hét múlva beszélhetünk, hogy mire jutottatok?
A: Persze, kereslek majd Téged. Köszönöm. Szia.
T: Szia.

A fenti párbeszéd célja, hogy már a követelmények specifikálásakor kiderüljön - vagy legalábbis pontosításra kerüljön -, hogy mit miért és hogyan lehet automatizálni.

- Tesztelési folyamatok megértése

A funkcionalitás megértése mellett nagyon fontos, hogy a tesztelő csoport milyen tesztelési folyamatokat tervez az adott projektre. Ez azért is lényeges, mert látni fogjuk, hogy milyen és mekkora igény jelentkezhet a későbbiek folyamán automatizált tesztekre.
Elképzelhető, hogy a tesztelő csoport csak és kizárólag az új funkcionalitások tesztelésére koncentrál. Ebben az esetben nyugodtan tegyük fel a kérdést a projekt csoportnak, hogy biztosan nem szeretnének regressziós teszteket végrehajtani?

Lehetséges, hogy regressziós és új funkciók tesztelését is tervezi a projekt csapata. Vizsgáljuk meg, hogy a követelmények alapján melyek azok a funkciók, amelyeket gyorsan és hatékonyan tudunk automatikus tesztekkel lefedni és támogatni.
Amennyiben integrációs tesztek is vannak, próbáljuk meg felmérni, hogy az adott folyamat hogyan működik. Melyek azok a pontok, ahol a különböző rendszerek kapcsolódnak? Az integrációs folyamat teljes egészében nyomon követhető a felhasználói felületen, vagy egy bizonyos ponton a háttérben történnek további műveletek, melyeket ellenőrizni kell?

- Automatizálási lehetőségek

A funkciók és a folyamatok megértése és feltérképezése után, próbáljuk meg a lehetőségeket felmérni. Nem biztos, hogy a fenti párbeszéd alapján konkrét automatizálási tervvel jönnek az automatizálási csoporthoz. Lehetséges, hogy egyáltalán nincs tervezve tesztautomatizálás. Pontosan ezért lényeges, hogy a projektek elején legyen bevonva a csoport, hiszen ők tudják, milyen tesztek léteznek már, melyeket futtatni is lehet, illetve melyek a tipikus lehetséges automatizálandó tesztek. Az automatizálással foglalkozó csoport így tud hozzájárulni akár a tesztelési folyamatok kibővítéséhez, vagy azok sikerességéhez.

- Technikai megvalósítás

Amennyiben találtunk olyan teszteket, melyek jó esetben automatizálhatóak, meg kell vizsgálni, hogy miként lehet őket implementálni. Fel kell mérni, hogy a tesztelés alatt lévő alkalmazást lehet-e automatizálni a meglévő eszközünkkel. Már amennyiben létezik ilyen eszköz. Amennyiben nem, akkor a különböző programozási nyelvek, szkript nyelvek oldalán kell megoldást keresnünk. Nem biztos, hogy minden automatikus tesztet abban az eszközben vagy programozási nyelvben kell implementálni, amiben első körben gondoljuk. Lehet, hogy egy része a teszteknek egyszerűbben megvalósítható egy automatizálási eszközzel, mint egy programnyelvvel.
Természetesen azt is meg kell vizsgálni, hogy milyen meglévő tesztállományunk van. Lehetséges, hogy olyan mértékű meglévő funkcionalitás van a birtokunkban, hogy nem lesz kétséges, milyen utat válasszunk a tesztek elkészítésére.

A követelmények specifikálásának befejezése, és a tervezési fázis elkezdése után zajlódó automatizálással kapcsolatos teljesítések listája

- Becslések

Miután megértettük a projekt célját, a tesztelési folyamatokat, és meg is vannak az első körben összegyűjtött automatizálandó tesztek, el kell kezdenünk becsléseket végezni. Ezek lehetnek:

1. Rendelkezésre állás
   Mivel jobb esetben a projekt kezdetétől már bevontak a munkába, így valószínű, hogy tudjuk, milyen ütemezéssel tervezték meg a projektet. Látjuk, hogy mikor vannak betervezve a fejlesztési és tesztelési fázisok, így ezekhez tudjuk igazítani az automatikus tesztjeink fejlesztését és futtatásukat is. A tesztelési időszakra el kell készülnie a tesztjeinknek, tehát úgy érdemes az implementálást tervezni, hogy már addigra elkészüljenek a tesztek.

Nézzük meg, hogy a csoportból terveznek-e szabadságot, vagy szabad napot az elkövetkezendő időszakra. Meg kell nézni, hogy ki, mekkora időkeretben tud tesztimplementálással foglalkozni.

2. Tesztimplementálási idő
   Miután megvizsgáltuk az összegyűjtött manuális tesztekre, hogy alkalmasak-e az automatizálásra, meg kell becsülni, hogy mennyi időt fog igénybe venni az elkészítésük. De ne felejtsük el, hogy azokat az időket is vegyük figyelembe, ami a manuális tesztelőt, vagy esetleg más kollégákat érinti.
   Itt az szokott gondot okozni, hogy sokszor a tesztautomatizáló nem tervezi be a tesztek tesztelését.
3. Alkalmazás elérhetősége
   Járjunk utána, hogy a tesztelt alkalmazást mikor és milyen rendszeren tudjuk elérni. Lehetséges, hogy olyan fejlesztésen kell automatizálnunk, amin a fejlesztők is dolgoznak párhuzamosan, így elképzelhető, hogy bizonyos kódmozgatások miatt a rendszer nem lesz elérhető. Ezt is bele kell kalkulálni az időkbe.

- Automatizálási stratégia

[11] Célszerű készíteni egy automatizálási stratégiát, amit egy dokumentumban rögzítsünk is le. Általában a következő elemeket tartalmazza egy stratégia:

- Miért automatizáljunk?
  Mutassuk meg, hogy miért is fontos az automatikus tesztek használata. Írjuk le, hogy melyek azok a tipikus pozitív esetek, amikor teljes hatékonysággal tudjuk alkalmazni az automatizálást. Például: sok ugyanolyan lépés végrehajtása, nagy adatmennyiséggel történő tesztelés, adategyezés ellenőrzése stb.
  Mutassuk meg, hogy miként járulhat az automatizálás a minőség javuláshoz.
- Miért ne automatizáljunk?
  Határozottan le kell írni azt is, hogy miért ne automatizáljunk. Rögzítsük le nyugodtan, hogy az automatikus tesztek használatának nem minden esetben van értelme. Amennyiben nem garantált a megtérülés és az elvárt hatékonyság, akkor teljesen felesleges teszteket implementálni. Abból, hogy automatizálunk, nem fog egyenesen következni, hogy javítunk is a meglévő tesztelési folyamatokon és azok minőségén.
- Mit lehet automatizálni?
  Soroljuk fel azokat a tipikus eseteket, amikor szinte gondolkozás nélkül meg tudjuk mutatni az automatizálás előnyét. Természetesen ide kell érteni azt is, hogy jelenleg mi mit tudunk megvalósítani. Ne próbáljuk feltüntetni a teljesítmény teszteket, ha nincs meg a megfelelő eszközünk és tudásunk erre a területre. Mindig csak arra fókuszáljunk, amit sikeresen meg tudunk valósítani.
- Melyek a végrehajthatatlan típusok?
  Ide sorakoztassuk fel azokat a példákat, amelyeket teljes mértékben lehetetlen megoldani. Ez egy kicsit nehéz résznek tűnik, viszont ide nyugodtan gyűjtsük össze azokat a példákat is, ahol mondjuk a beszerzésre szánt eszköz ára a korlát. Tüntessük fel, hogy melyek azok a számunkra elérhetetlen eszközök vagy megoldások, amelyekkel

bizonyos típusú teszteket meg tudnánk oldani. Vagy például egy képfelismerő és kép-összehasonlító teszt, ahol egy emberi tesztelés sokkal hatékonyabb lehetne.

- Mit érdemes automatizálni?
  Ez a pont nem keverendő össze a „Mit lehet automatizálni?” résszel. Arra a kérdésre, hogy mit érdemes, teljesen más példákat kell felsorakoztatnunk: tipikusan a regressziós teszteket, egyszerű lépésekből álló monoton teszteket és még sorolhatnánk a végtelenségig. Itt is érdemes azokat felsorolni, ami jelen pillanatban létezik és aktívan végzik is a tesztelők.
- Mit nem érdemes automatizálni?
  Mutassunk rá arra, hogy amennyiben még a tesztelés alatt álló alkalmazás funkcionalitása nem stabil, nem célszerű tesztet készíteni, mert a folyamatos tesztkarbantartás nagyban befolyásolja majd a megtérülés mértékét. Olyan eset is szerepelhet itt, ami arra mutat példát, hogy az alkalmazás elérhetősége a közeljövőben megszűnik. Ne próbáljunk meg tesztet fejleszteni egy olyan szoftverre, ami két-három hónapon belül megszűnik.
- Melyek az automatizálás prioritásai?
  Meg kell vizsgálni, melyek azok a prioritást élvező tesztek, amelyeket mindenképpen jó lenne lefejleszteni. Vegyük figyelembe, hogy az adott projekt milyen fontossággal és milyen ütemezéssel bír. Lehetséges, hogy egy sikeres projektvégrehajtáshoz elkerülhetetlen az automatikus tesztek használata, hiszen a tesztelési időt ezáltal csökkenthetjük, a lefedettséget pedig növelhetjük.
- Mikor automatizáljunk?
  Soroljunk olyan eseteket, amelyeken keresztül meg lehet érteni, mikor célszerű elkezdeni az automatizálást. Minél hamarabb be vagyunk vonva az alkalmazás fejlesztése során – és ide bele kell érteni a tervezést is –, annál hamarabb tudunk hatékony megoldásokat kitalálni. Természetesen itt is fel kell tüntetni, hogy akkor célszerű elkezdeni az effektív automatizálást, ha már valami elérhető az adott alkalmazás funkcionalitásából.
- Hogyan automatizáljunk?
  Ide sorakoztassuk fel azt a folyamatot, amin keresztül a legjobban és leghatékonyabban tudunk automatikus teszteket lefejleszteni. Írjuk le, hogy milyen fázisok szükségesek, mielőtt elkezdünk automatizálni és milyen standardok mentén fejlesszük a tesztünket.

- <u>Alap befektetés / megtérülés mérése</u>

Miután a becsléseket megpróbáltuk a megfelelő módon felmérni, tennünk kell egy gyors befektetés, megtérülés vizsgálatot is. Az alábbi egyszerű formulát alkalmazzuk:

$$Befektetés\ /\ Megtérülés = \frac{haszon - költség}{költség}$$

Fontos megemlíteni, hogy a megtérülés vizsgálatát sokan időben mérik. Véleményem szerint nem feltétlenül csak azt kell vizsgálni. Próbáljuk meg ezeket az értékeket ténylegesen költségként feltüntetni, azaz forintosítsuk, mint számokat. Nagyon egyszerűen úgy lehet megcsinálni, hogy vesszük az eltöltött munkaórák számát, és megszorozzuk a tesztelő, tesztkészítő órabérével. Így megkapjuk, hogy mennyi forintot költöttünk el egy teszt elkészítése során. Amennyiben nagyon pontosak szeretnénk lenni, belevehetjük a hardver igényeket és azok költségeit, az áram használatot és még sorolhatnám, viszont nem kell szerintem ennyire drasztikus számításokat végezni. Határozzuk meg ugyanezt a számot a teszt futásával, azaz hogy mennyi ideig fut a teszt és ezt szorozzuk meg a megfelelő összeggel. Ez mutatja meg, hogy mennyi nyereségünk lett, majd helyettesítsük be a fenti képletbe a számadatokat, így megkapjuk, hogy milyen megtérülést, hasznot tud hozni egy teszt automatizálása.

- Eszköz / megvalósíthatóság kiválasztása

A következő lépcső, hogy megvizsgáljuk, egy adott teszt lefejlesztésére a meglévő eszközünk használható, vagy sem. Lehet, hogy más megoldást kell keresnünk. Tartsuk szem előtt: nem biztos, hogy mindig az adott tesztautomatizáló eszköz tudja nekünk a legjobb megoldásokat szolgáltatni. Lehetséges, hogy olyan megoldást kell implementálnunk, ami nem szerepel az adott eszközben, vagy ha meg is van benne, akkor nem feltétlenül olyan módon szolgáltatja nekünk az eredményt, ahogy azt mi szeretnénk.
Elképzelhető, hogy a csapatunkban vannak olyan programozók, szkript készítők, akik egy adott programozási nyelvben nagyon jártasak, és sokkal gyorsabban, hatékonyabban tudnak egy automatikus tesztet elkészíteni, mint egy automatizáló eszközzel. Lehetséges, hogy egy automatikus teszttől az az elvárás, hogy futtatható legyen bármilyen szoftverkörnyezetben. Ez lehet, hogy nehezebben oldható meg egy adott eszközzel, mint inkább egy VBScript, Python, Java kóddal. Itt is figyelnünk kell arra, hogy mennyi befektetéssel jár egy adott megvalósíthatóság, törekedni kell a gyors és hatékony tesztkészítésre.

- Részletes becslés

Miután megtörtént, hogy milyen megoldással próbáljuk meg az adott tesztet elkészíteni, vissza kell kanyarodnunk, és ismét csinálnunk kell egy – most már sokkal pontosabb - becslést. Látjuk, hogy milyen implementálási megoldás lenne a legjobb a teszt elkészítésére, és ennek fényében az is látszódni fog, hogy mennyi emberi erőforrásra lesz szükség. Itt bele kell venni azokat a dolgokat is, hogy az adott teszt és projekt megértéséhez mennyi idő kell, valamint más területektől - akár az adott információk begyűjtésére - mennyi időre van szükség. Érdemes itt belekalkulálni az alábbiakat:

  - Tréning igények és azok hossza.
  - Amennyiben tesztautomatizáló eszközt kell használnunk, akkor annak az ára.
  - Manuális tesztelői erőforrás, akitől bármikor tudunk információt kérni.
  - Tesztautomatizálók száma.
  - Teszt elkészülésének ideje.
  - Jövőbeni tesztfuttatások száma, és ezek alapján a becsült megtérülés.
  - Mennyiszer fogjuk használni a tesztet, és milyen rendszerességgel?
  - Milyen fejlesztések lesznek az adott területen, és ez indukál-e teszt-karbantartást?

Természetesen van egy határ, ahol érdemes meghúznunk a számokat. Lehetne nagyon részletes becsléseket végezni, de ez lehet, hogy nem minden esetben tud hatékony lenni. Ne hagyjuk, hogy a számadatok alapján ne készüljenek automatikus tesztek.

Szeretnék erre egy nagyon rövid példát mutatni:

> *„Volt egy WEB-et érintő projekt, amiben még nem készültek el a felhasználói felületek, csak a webszolgáltatásokat implementálták a programozók, viszont azok már tesztelhető állapotban voltak. Megkerestek a tesztelők minket, az automatizálási csapatot, hogy szükségük lenne egy tesztre, ami az adott webszolgáltatást teszteli bizonyos bemenetek alapján. Kiderült, hogy nincs egy darab felhasználói felület sem, valamint a külföldi programozó két hét múlva el fog menni a cégtől, és igazából hetente lesznek változtatások az alkalmazásban. Ez a tipikus példája annak, hogy mikor nem kezdünk el automatizálni. Az alkalmazás funkcionalitása még nincs kész, folyamatosan változik, nincs felhasználói felület és még sorolhatnánk. Ebben az időben volt egy gyakornok a csoportban, akit érdekelt az automatizálás, és úgy gondoltam, talán az előrelépésének is jót tenne, ha másokkal együtt dolgozhatna, valamint megtapasztalhatja egy ilyen, és ehhez hasonló projekt nehézségeit. Gondoltam hajrá, csinálja. Több mint nyolcvan órát foglalkozott az automatikus teszt elkészítésével. A teszt futtatása viszont nem volt több mint egy óra. És ezt hetente egyszer kellett futtatni. Mondanom sem kell, hogy teljesen negatív számot mutatott a becslés során a befektetés/megtérülés, viszont egy hét után, mivel nagyobb tesztadat mennyiséggel használták a tesztet, kibukott egy hiba, amiről kiderült, hogy a fejlesztőknek javítani kellett a kódon. Eltelt még három nap, és még két hibát sikerült a teszttel találni. Kritikus hibák derültek ki az automatikus teszt futtatása során.”*

A fenti példa mutatja, hogy bár a becslés alapján egyáltalán nem érte volna meg a tesztet implementálni, viszont hibákat sikerült detektálni, ami abszolút mértékben sikernek könyvelhető el.

- Automatizálunk, nem automatizálunk

Az előző pontok alapján meg kell tudnunk hozni a döntést, hogy egy tesztet megéri-e lefejleszteni, vagy sem. Amennyiben nem tudunk dönteni, akkor feladatunk, hogy jelezzük az adott javaslatot a menedzsmentnek, és kérjük meg őket a döntéshozatalban. Jegyezzük meg, hogy nem minden esetben kell automatizálnunk. Itt az a lényeg, hogy kiderítsük, mit és milyen módon érdemes automatizálni, hogy a megfelelő módon tudjuk támogatni a tesztelési folyamatot.

A tervezés befejezése, és az implementációs fázis elkezdése után zajlódó automatizálással kapcsolatos tevékenységek listája

- Erőforrás-szükségletek

Miután megtörtént a követelmények definiálása, és a tervezés is hellyel-közzel lezajlott, meg kell vizsgálnunk pontosan, hogy az előzetesen felmért erőforrásigények hogyan módosulnak a követelmények függvényben. Mivel a követelmények specifikálása megtörtént, látjuk, hogy melyek azok az elemek, amelyek még pluszként jelentek meg az előzőekhez képest. Itt látnunk kell, hogy pontosan

kitől mennyi időre lesz szükségünk, mennyi automatizáló szakemberre lesz szükségünk és meg kell vizsgálni a szoftver és hardver igényeket is.

- Követelmények megértése

Fontos, hogy amennyiben nem dolgoztunk együtt a tervezési fázisban és a követelmények specifikálásakor a kollégákkal, akkor most tegyük meg, hogy beszerezzük a specifikációkat. Meg kell ismernünk és értenünk kell a követelményeket, hogy a megfelelő kérdéseket fel tudjuk tenni az üzletnek. Biztos vagyok benne, hogy nem minden esetben lesz egy követelmény számunkra teljesen egyértelmű. Nem kell megijedni, nyugodtan keressük meg az adott terület tesztelőjét, vagy üzleti elemzőjét, és kérjünk bővebb információt.

- Automatizálás tervezése

Miután megvan, hogy milyen teszteseteket fogunk automatizálni, érdemes megvizsgálni és egyeztetni is a tesztelőkkel, hogy milyen módon fogjuk implementálni a teszteket. Lehetséges, hogy nem tudunk bizonyos teszteket teljes egészében leautomatizálni. Lehet, hogy valamilyen ponton a tesztelés során manuális ellenőrzés szükséges, és csak ezek után tudunk bizonyos teszteket tovább futtatni. Nem minden esetben az a legjobb tesztelési módszer, amit a tesztelők megálmodnak. Annak érdekében, hogy a lehető leghatékonyabban tudjuk használni az automatizálást, lehetséges, hogy tesztelési folyamatokat kell megváltoztatni.

- Idő- és dátumbecslések

Miután pontosan látjuk, milyen és mekkora mennyiségű munkára lenne szükség, meg kell becsülnünk az időigényt és az ütemezést. Tudnunk kell, hogy mennyi automatizálási szakemberre lehet számítanunk, nekik mennyi idejük van, és hogy az elkövetkezendő időszakban a teszt lefejlesztésére milyen ütemezést kell beterveznünk. Előfordulhat, hogy a fejlesztésre szánt idő sokkal több, mint amennyi erőforrásunk van. Ekkor tudnunk kell javasolni valamilyen alternatív megoldást. Amennyiben az ütemezés nem felel meg, mert a projekt idejéből kicsúsznánk, úgy érdemes megfontolni, hogy elkezdjük-e az adott tesztet fejleszteni vagy sem.

- Tréningek, oktatások szüksége

Olyan esetben, ha számunkra ismeretlen az adott terület és akár a fejlesztés alatt lévő alkalmazás, akkor mindenképpen tervezzük be a szükséges oktatási időket is, hogy megértsük a funkcionalitást. Itt ki kell derítenünk, hogy ki vagy kik azok a személyek, akikhez fordulni tudunk ilyen esetben. Meg kell vizsgálni, hogy elérhető-e írásos dokumentáció az adott alkalmazáshoz.

- Környezet elérhetősége és elérése

Mindenképpen derítsük ki, hogy hol érhető el az alkalmazás. Próbáljunk meg információt szerezni arról, hogy mikor lesznek előreláthatóan leállások, amikor új kódot tesznek ki a programozók, vagy karbantartanak esetleg valamilyen részt. Derítsük ki, hogy szükséges vagy sem bizonyos jogosultságok megkérése a rendszeradminisztrátoroktól, hogy megfelelően érjük el az alkalmazás funkcionalitását. Fontos tudni, hogy szerepel-e vagy sem a kód más környezeteken is.

A tervezés befejezése és az implementációs fázis elkezdése után zajlódó automatizálással kapcsolatos teljesítések listája

- Automatizálási terv

Javasolt készíteni egy automatizálási tervet ennek a szakasznak a végén. Korábban már példaszerűen említettem, hogy mit célszerű tartalmazni egy ilyen tervnek. Nézzük meg részletesen, hogy milyen pontokat foglaljunk bele. [12]

- Automatizálás célja

  Rögzítsük, hogy mi az elvárt az automatizálástól. Mit fogunk tenni és mit nem. Például ilyen lehet, hogy integrált tesztet nem hajtunk végre. Csak és kizárólag regressziós teszteket automatizálunk. Ezek a pontok attól függnek, hogy mit tárgyaltunk le az adott igénylővel.

- Teszt stratégia

  Ez a terv a stratégia része, ide hivatkozhatjuk be a korábban tárgyalt automatizálási stratégiát.

- Erőforrások, felelősségek

  Az előzetes projektfelmérés során látni fogjuk, hogy mennyi erőforrásra lesz szükségünk. Rögzítsük, hogy mennyi automatizálási szakemberre lesz szükségünk, esetleg kitől kell oktatás, vagy egyéb információ, hogy lássa mindenki, kitől mit várunk el. A nevekhez rögzítsük az elvárt felelősségeket is.

- Eszközök

  Listázzuk azokat az eszközöket, melyeket használni fogunk az automatizálás során. Ide sorolhatóak akár a kereskedelmi forgalomban kapható eszközök, vagy akár nyílt forráskódú alkalmazások is.

- Ütemezés

  Tervezzük meg és rögzítsük, hogy milyen menetrend szerint fog történni a tesztimplementálás. Mikor van szükségünk mástól információra, mikor lesz az első tesztvégrehajtás, és mikorra kell teljes egészében elkészülnie a tesztnek. Tartalmazza a terv a tesztelési időszakot is, hogy látszódni tudjon az aktuális tesztelés menete, és hogy mikor lehet számítani az automatikus tesztvégrehajtásra.

- Környezet

  Írjuk le, hogy milyen környezetben fog futni az automatikus tesztünk. Itt kitérhetünk a megfelelő számítógépes környezetre is. Gondoljuk át, milyen konfigurációnak és milyen alkalmazásoknak kell elérhetőnek lennie az adott gépen. Például.: Windows 7 64-bit, Internet Explorer 11, JAVA runtime 1.6.12 és még sorolhatnánk.

Természetesen fel kell tüntetni, hogy mi is az adott tesztelési környezet. Hol milyen szervereken fogjuk a tesztet végrehajtani, fejlesztői, vagy tesztelési környezetben is működnie kell a tesztnek, vagy sem.

- Teljesítések
  Rögzítsük, hogy miket fogunk teljesíteni a tesztimplementálás során. Azért fontos ez, hogy mindenki tisztán lássa, melyek is az elvárások. Kerüljük el, hogy a későbbiek során olyan funkcionalitást is szeretnének kérni a tesztelők, amiről korábban nem volt szó. Amit megbeszéltünk, azt teljesítjük. Ezek lehetnek: automatikus tesztek, modulok, tesztadatok, helyreállítási forgatókönyvek, mérési eredmények, tesztfuttatáshoz szükséges információk.

- Kockázatok
  Soroljunk fel pár kockázatot, melyek előfordulhatnak a folyamat során. Jobb számolni ezekkel még időben, hogy a megfelelő akciókat tudjuk végrehajtani, amennyiben szembesülünk valamelyik kockázattal. Ilyenek lehetnek: nem lesz elérhető a tesztkörnyezet, nem áll a megfelelő eszköz a rendelkezésünkre, valamelyik csapattag betegség esetén nem lesz elérhető.

- Tesztadat
  A tesztadatok fontos részei a tervnek. Egy automatikus tesztnél nagy fontossággal bír a helyes és megfelelő tesztadat. Elképzelhető, hogy a fejleszteni kívánt automatikus teszt, csak és kizárólag egy bizonyos adattal működik. Tegyük egyértelművé, hogy más adattal történő futtatás során helytelen futási eredményeket is kaphatunk. Jegyezzük meg, hogy ez nem az automatikus teszt hibája.

- Eredmények
  Végül az eredményekről is szót kell ejteni. Fogalmazzuk meg, hogy az eredményeket milyen formában fogjuk a tesztelőknek visszaszolgáltatni. Kinek, mikor és milyen formában kell rendelkezésére bocsátani ezeket. Hol fogjuk tárolni az eredményeket kell-e egy közös tárhelyre, megfelelő formában tárolni az eredményeket vagy sem.

- <u>Keretrendszer tervezés</u>

Szem előtt kell tartani, hogy az elkészült automatikus tesztet egy már meglévő keretrendszerbe el kell majd helyezni. Amennyiben még nem létezik keretrendszer, akkor meg kell tervezni, hogyan is nézzen ki. Nem kell egy bonyolult dologra gondolni, viszont az automatizálás céljainak megfelelően kell kiépíteni. Azaz különböző modulokat kell létrehoznunk, újrafelhasználható részeket a tesztünkben, megfelelő tesztadatokat, a helyreállítási forgatókönyveket és természetesen a teszteket is el kell helyeznünk. Íme egy példa, hogy miként lehet akár egy mappaszerkezetből is keretrendszert csinálni:

- Adattáblák, tesztadatok
  - Üzleti_terület_1
  - Üzleti_terület_2

- Függvény könyvtárak
    - Általános
    - Üzleti_terület_1
    - Üzleti_terület_2
- Objektumtároló állományok
    - Általános
    - Üzleti_terület_1
    - Üzleti_terület_2
- Eredmények
    - Üzleti_terület_1
    - Üzleti_terület_2
- Tesztek
    - Üzleti_terület_1
    - Üzleti_terület_2

Mint látjuk, elég jól szervezett ez a lista. Ez még bonyolódhat is, de természetesen egyszerűsödhet attól függően, hogy milyen robosztus környezetben is vagyunk. Az általános mappa célja az, hogy olyan dolgokat tároljunk benne, melyek értelmezhetőek más tesztek szintjén is. Ilyen lehet például a belépési funkció. Amennyiben minden teszt előfeltétele egy bizonyos belépés a tesztelt alkalmazásba, akkor azt emeljük ki, és szerepeljen különálló egységként.

6.2. Tesztimplementálás

Miután sikeresen felderítettük, hogy milyen teszteket célszerű automatizálni, elkezdődhet a tesztimplementálási szakasz.

Az implementálási folyamat során - és a tesztelési fázis elkezdése előtt - zajlódó automatizálással kapcsolatos tevékenységek listája

- Üzleti folyamatok felderítése

Abban az esetben, ha eddig még nem történt volna meg az üzleti folyamatok vizsgálata, akkor most még van lehetőségünk erre. Elképzelhető, hogy korábban nagyon felszínesen lettek érintve ezek a folyamatok, de most, hogy a fejlesztés elindult, sokkal pontosabban tudjuk látni, mire is van szükség, milyen más alkalmazásokkal, folyamatokkal van kapcsolatban a jelenlegi teszt.

- Tesztek implementálása

Végre elérkeztünk ahhoz a folyamathoz, amikor elkezdhetünk a tesztjeinken dolgozni. Mivel a programozók már fejlesztik a tesztelendő alkalmazást, így elérhető számunkra valamilyen környezetben az alkalmazás adott állapota. Ezen már tudunk mi is dolgozni, el tudjuk készíteni az objektumtárolókat, ellenőrzési feltételeket, kezdeti és végpontokat a teszt fázisban.

- Adatparaméterezés

Amikor majdnem kész állapotban vannak a tesztek, törekedjünk arra, hogy a beégetett értékektől szabadítsuk meg a tesztünket. A tesztadatokkal fel kell paramétereznünk a tesztet, hogy a későbbiek során is rugalmasan, bármilyen adattal végre lehessen hajtani a tesztet.

- Tesztek és komponensek létrehozása

Ahogy haladunk előre a fejlesztéssel, úgy fogjuk a tesztekben felismerni azokat a pontokat, melyeket célszerű kiemelni és modulokat készíteni belőlük. Ezeket a modulokat aztán hívjuk meg a tesztjeinkben a megfelelő módon. Itt az újrafelhasználhatóságot helyezzük előtérbe, valamint a későbbi tesztkarbantartási időt próbáljuk meg csökkenteni.

- Helyreállítási forgatókönyvek készítése

Sokan próbálnak meg úgy teszteket létrehozni, hogy megpróbálnak minden apró eshetőséget feltárni arra, hogy hol és milyen esemény hatására akadhat meg egy automatikus teszt futása. Bármennyire is körültekintőek vagyunk, úgysem tudunk felkészülni minden egyes eshetőségre. Igazából ne próbáljuk meg a lehető legmasszívabb teszteket létrehozni és felvértezni a tesztjeinket mindenféle hiba előfordulásokra. Az ilyenfajta fejlesztés sok időt vehet el, és nem is biztos, hogy hatékony lesz. Ezzel szemben inkább azt próbáljuk meg kitalálni, hogy milyen helyreállítási forgatókönyvet definiáljunk. Amikor a tesztfutásunk hibába ütközik és egy adott lépést nem tud végrehajtani, akkor ezzel a helyreállítási folyamattal olyan állapotba hozzuk a tesztelt alkalmazást és ezzel együtt az automatikus tesztünket is, hogy folytatódni tudjon a tesztvégrehajtás. Egy nagyon egyszerű példa, ami nagyon is hatékonyan működik: tegyük fel, hogy egy internetes, weben elérhető felhasználói felülettel rendelkező alkalmazásra készítettünk automatikus tesztet. Azt a forgatókönyvet találtuk ki, hogy amennyiben a teszt nem képes tovább haladni, úgy zárja be egy függvény az adott böngészőt, majd nyissa meg újra, navigáljon el a megfelelő kezdeti oldalra ahonnan a tesztünk indul, majd az automatikus tesztünk futása folytatódjon a következő tesztadat használatával.

Az implementálási folyamat során, és a tesztelési fázis elkezdése előtt zajlódó automatizálással kapcsolatos teljesítések listája

- Automatizált tesztkomponensek

Az automatikus tesztek fejlesztésének végeztével elérhető, a keretrendszerbe illeszkedő komponenseket el kell készítenünk. Ez számos esetben sajnos nem mindig teljesül. Amennyiben a projekt ütemezése csúszásban van, úgy lehet, hogy nem a tervezettnek megfelelő ütemben tudjuk az automatikus tesztjeinket lefejleszteni, azaz igaz a tesztek elkészültek, de a modulok és komponensek nem készültek el. Ezeket pótolni kell.

- Automatizált tesztek

Egyértelmű. Az automatizált szoftvertesztelés célja, hogy működő automatikus tesztek legyenek. Működőképes, végrehajtható, és megfelelően működő teszteket kell implementálnunk.

- Tesztadatok

Menet közben már kellett, hogy használjunk tesztadatokat, amikor fejlesztettük az automatikus tesztet. Ezek vagy az automatizáló szakember által keresett adatok voltak, vagy jobb esetben a tesztelőktől kért adatok. Amikor elkészül egy automatikus teszt, olyan tesztadat mennyiséget kell mellékelnünk, amelyek a tesztelés során értékes és használható eredményeket is fognak szolgáltatni.

6.3. Tesztfuttatás

Elérkeztünk ahhoz a ponthoz, hogy a tesztjeinket futtassuk. Azonban ez nem úgy működik, hogy fogjuk a tesztet és hajrá, elindítjuk. Nagyon sok feltételnek kell / lehet teljesülnie, hogy egyáltalán elindítsuk a teszteket. Vizsgáljuk meg az alábbi kérdésekre a válaszainkat, és nagy valószínűséggel ezek tudatában el tudjuk dönteni, hogy van-e értelme futtatni a tesztjeinket.

- Elérhető a tesztelendő alkalmazás?
    - Ne futtassuk a tesztjeinket, ha nincs meg a környezet. Nem kapunk megfelelő eredményeket úgysem.

- Rendelkezésünkre állnak a megfelelő tesztadatok?
    - Ha nincs adat, amivel teszteljünk, akkor nincs értelme a tesztet sem végrehajtani.

- Megvannak a szükséges jogosultságok?
    - Elképzelhető, hogy a vállalatnál valaki más foglalkozik a jogosultságok kezelésével. Amennyiben ezek engedélyezése nem történt meg, a tesztünk nem tudja megfelelően végrehajtani a lépéseket, hiszen nem tudunk hozzáférni az alkalmazás adott funkcionalitásához.

- Megtörtént a szükséges előfeltétel végrehajtása a manuális tesztelőtől?
    - Lehetséges, hogy létezik olyan előfeltétel, aminek teljesülnie kell, és ezt lehet éppen a manuális tesztelő fogja megcsinálni, majd tőle várjuk az információt, hogy indulhat a teszt, vagy sem. Ez lehet akár a teljes folyamat ellenőrzése is, amit a tesztünk csinál, csak elsőként a tesztelő végrehajt egy kritikus tesztet, hogy meggyőződjünk, lehet futtatni a teszteket.

- Van elérhető erőforrás a teszt végrehajtásához?
    - Meg kell vizsgálni azt is, hogy egyáltalán elérhető-e vagy sem az automatizáló eszközünk, vagy a tesztet végrehajtó. Lehetséges, hogy más fontosabb feladat ellátásával foglalkozik és a teszt végrehajtása később fog teljesülni. Lehet, hogy éppen valaki más használja tesztvégrehajtásra az automatizáló eszközt, így addig várakoznunk kell, amíg a másik tesztfutás befejeződik.

A tesztek végrehajtása során az automatizáló csapat feladata megváltozik. Ebben az időszakban felügyelik a tesztek végrehajtását, ellenőrzik, hogy a tesztek állapota megfelelő-e. Folyamatosan rendelkezésre kell állni más csoportoknak, hiszen a tesztelési időszakban sokan hajtanak végre teszteket. Amennyiben egy adott területnek szüksége van automatikus tesztfutásra, meg kell tudnunk mondani, hogy azt mikor és milyen határidővel tudjuk előreláthatóan teljesíteni. Folyamatosan optimalizálni kell a tesztek végrehajtását, és a lehető leghatékonyabban kihasználni az erőforrásainkat.

Előfordulhat, hogy egy tesztelési folyamat során derül ki, hogy valamelyik tesztünk nem úgy működik, ahogy az elvárt lenne. Azt tapasztalhatjuk, hogy a korábbi állapothoz képest a tesztelt alkalmazás funkcionalitása más. Ekkor a lehető leghatékonyabban reagálnunk kell. Tesztelés során kell végrehajtanunk a tesztek karbantartását, ami bizony idővel jár. Ezek akár hátráltató tényezők is tudnak lenni más tesztek végrehajtásában.

Támogatnunk kell a környezetünket abban, hogy minél szélesebb körben tudják használni az automatizálás előnyeit. Tesztvégrehajtás során különböző módszerekről beszélhetünk:

- Az automatizálással foglalkozó tesztelők indítják a teszteket aszerint, hogy éppen mikor melyik tesztre van szükség, vagy más manuális tesztelők munkáit mikor tudják segíteni.
- Lehetséges, hogy a tesztelési folyamat egy része van csak automatizálva, így folyamatosan kapcsolatban kell lenni más tesztelőkkel, hogy a megfelelő információnak legyünk a birtokában. Ezek fényében tudjuk meg, milyen teszteket kell végrehajtanunk.
- Lehetséges, hogy olyan automatikus tesztek is léteznek, melyeket heti, vagy akár napi szinten ütemezve kell végrehajtani. Elképzelhető, hogy konkrét időpontban kell futtatni a teszteket.
- Amennyiben létezik egy olyan eszköz, amiből elérhetőek a tesztjeink, akár más felhasználók is képesek elindítani a teszteket rajtunk függetlenül.

Tesztvégrehajtás során ellenőriznünk kell a befektetés és megtérülés mértékét. Tesztfuttatás során látjuk tisztán, hogy mennyi haszonnal is jár pontosan az adott automatikus teszt használata.

A tesztvégrehajtások után pedig gondoskodnunk kell a megfelelő eredmények összegyűjtéséről.

6.4. Kiértékelés

Egy tesztelési ciklus végeztével célszerű egy teljes körű kiértékelést végezni. Vizsgáljuk meg, hogy mennyi tesztünk futott és hány alkalommal. Nézzük meg, hogy melyek voltak azok a tesztek, amelyek nem működtek tökéletesen, és módosításokat kellett végrehajtani. Ezeket a teszteket vegyük elő, és priorizálással döntsük el, hogy melyiket fogjuk elsőként kijavítani. Rendeljünk felelősöket is a tesztek kijavítására.

Rendszeresen csináljunk visszatekintést az elmúlt időszakra. Vizsgáljuk meg a tesztkészítések idejét, valamint a tesztvégrehajtási ciklust is. Gyűjtsük össze, hogy melyek voltak azok a dolgok, amelyek jól

mentek és célszerű az elkövetkezendők időszakban is alkalmaznunk. Ez lehet akár a kommunikáció a tesztelőkkel, a gyors tesztvégrehajtások, sikeres teszteredmények és még sorolhatnánk.
Szedjük össze azokat az elemeket is, melyek kevésbé mentek jól. Ebből a listából válasszunk ki egy-két elemet, amelyekre a következő ciklusban odafigyelünk, és javítani szeretnénk. Ez a módszer a SCRUM agilis módszertanból ered és véleményem szerint igen hatékony. Mindig van lehetőség a folyamatokon fejleszteni, és minden egyes ilyen alkalommal a csapatunk tagjai nyíltan és őszintén adnak visszajelzést.

6.5. Metrika

Volt már szó a tananyag során metrikáról. Talán már tudjuk is, hogy miért fontos, hogy metrikák, mérőszámok jelenjenek meg a folyamatainkban. Ebben a részben kicsit részletesebben érintjük a mérésekkel kapcsolatos dolgokat.

[3] A metrikák nem mások, mint egyszerű mérések az automatizálási folyamatainkban. Használatukkal képet kapunk a tesztjeink állapotáról, a tesztkészítési folyamatokról, valamint a tesztek végrehajtása során és után kapott értékekről. Ezekből a számadatokból tudjuk kiértékelni a tesztjeink minőségét, hatékonyságát, felhasználásainak mértékét.

[14]
Érdemes megjegyezni, hogy léteznek már meglévő metrikák az automatizálás terén. Viszont azt, hogy mi is a legjobb nekünk, azt mindig az fogja megszabni, hogy mi is a célunk a metrikákkal. Hiába rögzítünk és követünk számos mérőszámot, ha nem tudjuk belőlük kimutatni azt, ami ténylegesen fontos információ, és ami hatékony visszajelzést ad az automatizálási folyamatainkra.

Egy jó automatizálási metrika a következő karakterisztikákat tartalmazza:

- objektív,
- mérhető,
- van értelme,
- könnyen gyűjthető adatokat tartalmaz,
- általuk meghatározhatóak az automatizálási területek javítása,
- egyszerű.

Említsünk meg pár olyan mérőszámot, ami a manuális tesztelés során is megjelenhet, de hatékonyan használható automatizálás során is.

[3] [14]

- Végrehajtási folyamat mérése (Tesztfolyamat)

Egyszerű számadat. Az összes teszteset, és a végrehajtott tesztesetek számából adódik:

$$TF = \frac{VTE}{ÖTE} = \left(\frac{Végrehajtott\ Teszt\ Esetek\ száma}{Összes\ Teszt\ Eset\ száma} \right)$$

A képlet érdekessége, hogy a végrehajtott tesztesetek száma nem tartalmazza a tesztek eredményét. A végrehajtás során elképzelhető, hogy egy automatikus tesztet többször kell futtatnunk, mert valamilyen hiba merült fel a tesztelt alkalmazásban. Amennyiben azt akarjuk mérni, hogy melyek azok a tesztek, amelyek sikeresen futottak le, akkor a fenti képletbe a VTE helyett, a Sikeresen Végrehajtott Tesztesetek számát kell behelyettesítenünk.

- Követelmény lefedettség

Ez is egy egyszerű számadat. Megvizsgáljuk, hogy mennyi követelményünk van és megnézzük, hogy mennyit fedtünk le tesztesettel. Meg kell jegyezni, hogy itt zömmel a manuális tesztesetek száma és azok lefedettsége fog győzedelmeskedni. Ritkán fordul elő, hogy a követelményeket már a legelső fázisban automatizmussal fedjünk le. Ez a számadat nem azt fogja mutatni, hogy a követelmény teljesült, hanem azt, hogy tesztesettel, vagy tesztesetekkel le van-e fedve. Azt se felejtsük el, hogy egy követelményhez akár több teszteset is tartozhat. Itt nem a tesztesetek számosságát vizsgáljuk, hanem a követelmények lefedettségét.

$$KL = \frac{TKSZ}{KSZ} = \left(\frac{Teljesített\ Követelmények\ Száma}{Követelmények\ Száma} \right)$$

Megjegyzem, hogy ez egy tipikus mérőszám, melynek használatát érdemes megfontolni, ugyanis, ha nem léteznek megfelelően dokumentálva a követelmények, vagy akár nincsenek is, ezt nem tudjuk mérni.

- Automatizálhatóság

Már említettem korábban, hogy miként lehet felmérni, hogy mi automatizálható. Kérjük el a tesztelőktől a kidolgozott teszteket és vizsgáljuk meg, hogy melyek azok, amelyeket automatizálni lehet, és melyek azok, amelyeket érdemes is.

$$A(\%) = \frac{ATESZ}{TESZ} = \left(\frac{Automatizálható\ Teszt\ Esetek\ Száma}{Teszt\ Esetek\ Száma} \right)$$

Fontos és ismét kiemelném: az, hogy valamilyen teszteset automatizálható, nem azt jelenti, hogy érdemes is automatizálni!

- Automatizálási folyamat

Miután megvagyunk azzal, hogy mit is érdemes automatizálni és elkezdünk dolgozni a teszteken, ezzel a számadattal tudjuk figyelni, milyen tempóban haladunk az implementálással.

$$AF(\%) = \frac{KTESZ}{ATESZ} = \left(\frac{Kész\ Teszt\ Esetek\ Száma}{Automatizálható\ Teszt\ Esetek\ Száma} \right)$$

- Klasszikus Befektetés / Megtérülés mérése

Vegyük szemügyre a klasszikus értelemben vett befektetés és megtérülés számítását. Igazából ez a számadat egy nagyon jó becslést is tud szolgáltatni már az automatizálás megkezdése előtt, hogy érdemes vagy sem belevágni egy teszt implementálásába. Legyen példa egy webes környezet. A programozói csapat valamilyen fejlesztés publikálását tervezi minden héten, tehát heti rendszerességgel kellene a tesztjeinket futtatni.

Nézzük meg a következő kérdésekre az értékeket. Szeretném megjegyezni, hogy a táblázatban szereplő értékek nem valósak, csak a szemléltetés kedvéért használom ezeket az értékeket.

Fázisok	Költség	Idő
Tesztelés, tesztvégrehajtás rendszeressége		Hetente
Manuális tesztek elkészítésének ideje	1 tesztelő = 1000 HUF órabér	5 nap
Manuális tesztvégrehajtás	3 tesztelő = 1000 HUF órabér	2 nap
Automatikus tesztek implementálása	1 tesztelő = 1000 HUF órabér	8 nap
Automatikus tesztvégrehajtás	1 tesztelő = 1000 HUF órabér	2 óra
Manuális tesztek karbantartása	1 tesztelő = 1000 HUF órabér	Kéthetente 1 nap
Automatikus tesztek karbantartása	1 tesztelő = 1000 HUF órabér	Kéthetente 1 nap

12. táblázat Klasszikus befektetés / megtérülés mérése

Tudjuk még azt is, hogy az elkövetkezendő egy éven keresztül minden héten meg fog történni ez a fajta tesztelés. Az alábbiak szerint kalkulálható a ROI:

Automatizálás költsége = Automatikus tesztek implementálási ideje + (karbantartás ideje * tesztek végrehajtási száma) + (tesztek végrehajtási ideje * tesztek végrehajtási száma)

Automatizálás költsége = 8 nap * 1000 HUF/óra + (1 nap * 1000 HUF/óra * 26 hét) + (2 óra * 1000 HUF/óra * 52 hét)

Automatizálás költsége = 64000 HUF + 208000 HUF + 104000 HUF

Automatizálás költsége = 376000 HUF

Manuális tesztelés költsége = Manuális tesztek elkészítésének ideje + (tesztek karbantartásának ideje * tesztek végrehajtásának száma) + (tesztek végrehajtásának ideje * tesztek végrehajtásának száma)

Manuális tesztelés költsége = 5 nap * 1000 HUF/óra + (1 nap * 1000 HUF/óra * 26 hét) + (2 nap * 1000 HUF/óra * 52 hét)

Manuális tesztelés költsége = 40000 HUF + 208000 HUF + 832000 HUF

Manuális tesztelés költsége = 1080000 HUF

ROI = Haszon - Költség / Költség

ROI = (manuális tesztelés költsége – automatizálás költsége) / automatizálás költsége

ROI = (1080000 HUF – 376000 HUF) – 376000 HUF / 376000 HUF

ROI = 87 %

Minél magasabb ez a szám, annál biztosabbak lehetünk, hogy az automatizálásunk sikeres lesz.

7. Automatizált szoftvertesztelési folyamat beépülése a tesztelési ciklusba

7.1.Tervezés

Mint általában minden projektet és természetesen a hozzá tartozó tesztelést, az automatizált tesztelést is tervezni kell. Azonban miben térhet el egy automatikus tesztelési ciklus egy manuális tesztelési ciklustól?

Időről időre - amennyiben a folyamataink és az automatizálási igény is megfelelően működik - növekvő tesztállománnyal fogunk rendelkezni. A tesztek között szerepelni fognak olyanok, melyeket minden egyes tesztelési ciklusban végre kell hajtanunk, de olyanok is lesznek, amelyeket csak időszakosan kell végrehajtanunk.

Egy manuális tesztelés egy adott tesztmennyiséggel rendelkezik. Tegyük hozzá, hogy ez a regressziós tesztelés esetén van így. Amennyiben egy tesztelőnek új funkcionalitásokat is kell tesztelnie, akkor azt vizsgálja meg, hogy - a meglévő regressziós tesztelés mellett - mennyi időt kell az új funkcionalitások tesztelésével tölteni. Amennyiben ez az idő nem fér bele a tesztelési ciklusba, úgy jobb esetben a regressziós esetekből fog kihagyni párat, és az új funkcionalitások helyes működésének tesztelésére fordít nagyobb hangsúlyt.

Azonban az automatizálás célja, hogy a meglévő tesztállományt, amikor csak lehet, hajtsuk végre, hiszen ezzel tudunk meggyőződni arról, hogy minden a megfelelően működik még új funkciók implementálása során is. Új funkcionalitások tesztelése nem minden esetben kivitelezhető, lásd a korábbi fejezetek alapján.

Mind a manuális tesztelést, mind az automatikus tesztelést is meg kell terveznünk egy-egy tesztelési ciklus előtt. A következő szempontok okozhatnak nehézséget a manuális teszteléssel szemben (melyekre viszonylag pontos választ kell tudnunk adni):

- Tesztek mennyisége és végrehajtási idejük
  Ahogy azt már korábban említettem, itt a tesztek mennyisége növekszik. Ezeknek a teszteknek a végrehajtása időbe kerül, és minél több tesztünk van, annál több időt kell szánnunk a tesztek végrehajtásához.
  Az nem elvárt, hogy precízen, minden egyes tesztünkről, és az összes tesztadatra történt iteráció futási eredményéről legyen egy kőbe vésett időtartam, amivel számolni tudunk. Inkább próbáljunk meg egy átlagot mondani, hogy a különböző típusú teszteknek a végrehatása mennyi időbe telik. Jobb esetben ismerjük a tesztjeinket, és ezt meg is tudjuk mondani, viszont van, amikor tényleg becsülni tudunk csak. Mivel a tesztjeinkben vannak szinkronizációs pontok, várakoztatások, ezért lehet, hogy a manuális tesztelésnél több időbe telik egy teszt végrehajtása.

Ha tudjuk a tesztjeink és a tesztvégrehajtások számát, akkor az átlag futási idő szorzattal meg tudjuk becsülni, mennyi időt fog igénybe venni a tesztek végrehajtása.

Figyelem! Felteszek egy kérdést. Nem hiányzik valami ebből a becsült értékből? A gondolkodási idő lejártával elárulom, hogy ebben bizony nem szerepel a tesztek előkészítésének, a tesztadatok előkészítésének, valamint, ha szükségesek környezeti beállítások a tesztek futtatásához, azok ideje sem. Ezekkel is számolnunk kell. Tegyük fel, hogy létezik egy automatikus tesztünk, aminek a végrehajtása körülbelül ötven percet vesz igénybe. Attól függően, hogy milyen előkészületeket kell tennünk, nyugodtan becsülhetünk hatvan, hetven percet is a tesztünkre. Mi tartozik még bele az extra húsz percbe? Miután a tesztünk lefutott, az eredményeket ki kell értékelni, esetleg továbbítani a megfelelő tesztelő számára.

Gondot okozhat, ha a tesztelési ciklus előtt nem tudunk olyan tesztekről, melyeket menet közben nem terveztünk be, de időközben a végrehajtásuk szükségessé vált. Ezeknek a teszteknek a végrehajtása plusz időbe telik, valamint az előre felállított tervünk is módosul ezek miatt.

- Emberi erőforrások
  Bizony számolnunk kell azzal is, hogy az automatizálási csapatból ki lesz elérhető. Össze kell hangolni és egyeztetni a tesztfuttatásokat is. Igaz, hogy egy automatikus teszttől elvárt, hogy külső beavatkozás nélkül tudjon végrehajtódni, viszont ha kifejezetten egy szakemberhez tartozik egy teszt, és csakis ő ért az adott üzleti területhez is, akkor számba kell venni azt is, hogy rendelkezésre tudjon állni a szakember, amikor a tesztnek végre kell hajtódnia. Miért is fontos ez? A tesztünk bármikor elakadhat. Abban az esetben, ha ez egy kritikus teszt, és nem tudjuk elérni a megfelelő kollégát, hogy kijavítsa a tesztet, majd újra futtatni tudja, akkor nagyon nagy kockázatnak van kitéve az automatizálás sikere. Törekedjünk arra, hogy ne legyenek olyan területek, melyeknek csak és kizárólag egy tulajdonosa van. Sajnos ez a gyakorlatban nem mindig kivitelezhető.
  Tegyük fel, hogy létezik más valaki is, aki ért az adott területhez. Valaki viszont éppen a tesztelési ciklus alatt lesz szabadságon, így őt helyettesíteni kell. Ekkor több feladat hárul másokra, hiba esetén több időbe telhet a teszteket javítani. Tudnunk kell belekalkulálni ezeket is a tervezés során.

- Végrehajtó állomások száma
  Ezt a számadatot nem lesz nehéz megmondani, amennyiben dedikáltan vannak olyan számítógépek, melyeket automatikus tesztvégrehajtásokra használhatunk. Akkor bonyolódik a helyzet, ha olyan végrehajtó állomásokról is beszélünk, amelyek nem dedikáltan erre a célra vannak kiépítve. Ilyen például, amikor egy tesztelő a saját számítógépén hajt végre tesztet. Hol mutatkozhat probléma?
  Tegyük fel, hogy öt napból áll a tesztelési ciklus. Legyen három végrehajtó állomásunk. Az automatikus tesztállományunk olyan mértékeket ért el, hogy nem tudjuk az öt nap alatt mindegyiket végrehajtani a három gépen. Elkezdődik a tesztelés és azt tapasztaljuk, hogy menet közben a három végrehajtó állomáson kívül más tesztelők hajtanak végre teszteket a saját

gépeiken. Az öt napos tesztelés során már a negyedik napon lefutottak az automatikus tesztek. Ebben az esetben a fennmaradó egy nap pazarlásnak tűnhet, hiszen a tesztállományból kivett tesztek így nyugodtan lefuthattak volna.
A tervezés nem pontos ezekben az esetekben. Emellett még más veszélyeket is hordoz ez a fajta tesztvégrehajtás. Mi garantálja, hogy a tesztelők gépe olyan hardver, szoftver és környezeti beállításokkal vannak ellátva, mint amelyeket a tesztek megkövetelnek? Ki ellenőrzi, hogy a tesztek végrehajtása előtti feltételei teljesülnek? A tesztek futási eredményeinek kiértékeléséért ki a felelős?

- Tesztidőszak hossza
  Vizsgáljuk meg, hogy mennyi időnk lesz az adott tesztelési ciklusban. Elképzelhető, hogy a mindenki számára ismert időintervallumon túl is még elérhető lesz a tesztkörnyezet. Érdemes rákérdezni erről a fejlesztőknél, hiszen ezek az extra idők jól jönnek, amennyiben azt tapasztaljuk, hogy az aktuális időszak szűkös.

- Automatizálási eszközök elérhetősége
  Akár nyílt forráskódú, akár kereskedelmi forgalomban kapható eszközről van szó, meg kell vizsgálnunk, hogy miként lesz elérhető az adott eszköz. Lehetséges, hogy az adott tesztelési ciklusban a csoportunkból valaki éppen automatikus tesztek implementációjával fog foglalkozni. Tehát az adott automatizáló eszköz és a számítógép, amire telepítve van, fejlesztésre és végrehajtásra is kellene. Ekkor egy elég kellemetlen szituáció áll elő, ugyanis a végrehajtás az egy adott időszakra vonatkozik, nem tehetünk mást, futtatnunk kell a teszteket, viszont, ha a fejlesztéstől vesszük el az erőforrást, akkor az automatikus tesztünk elkészülésének ideje fog megnyúlni. Érdemes erre nagy hangsúlyt fektetni a tervezés során!

- Tesztjeink állapota
  A tervezés során meg kell győződnünk arról, hogy a végrehajtani kívánt tesztjeink állapota megfelelő. A tervezés során célszerű megkérdezni a csapatunk tagjait, hogy a kívánt módosítások és javítások megtörténtek-e a teszteken, vagy vannak-e olyanok, melyeket sajnos csak a tesztelési ciklusban lehet kivitelezni.

- Tesztkörnyezetek állapota
  Egyik legfontosabb pontja a tervezésnek. Létezik a tesztkörnyezet, amin végrehajtani szeretnénk a tesztjeinket? Amennyiben nem, úgy nagyon nem tudunk mit kezdeni.
  Próbáljunk meg minél több információt összegyűjteni arról, hogy a tesztelési ciklus előtt éppen mi zajlik a tesztelésre dedikált környezettel. Pár kérdés, amit feltehetünk:
  - Mikor lesz a tesztkörnyezet frissítve?
  - Mikor lesznek a fejlesztők által készített kódok átmozgatva?
  - Mikor kerül sor a rendszerszintű beállításokra?
  - Mikor frissül az adatbázis?
  - Mikor lesznek a rendszerszintű felhasználói jogosultságok kiosztva?
  - Mikor lehet elkezdeni a tesztelést?

- Mikor és hogyan kapunk információt az esetleges tesztkörnyezet karbantartásról?

- Tesztvégrehajtások prioritása
  A tervezés mit sem ér prioritások nélkül. Nem mindegy, hogy melyik tesztünk mikor hajtódik végre. Elsőként, ha a tesztelőknek adatot kell szolgáltatnunk egy automatikus teszt futtatásával, akkor ne tartsuk fel őket a további munkában. Amennyiben egymásra épülő automatikus tesztek vannak, úgy fel kell tudnunk állítani a sorrendiséget, hogy miután mi következzen.
  Vizsgáljuk meg azt is, hogy melyek azok a tesztek, amelyek előreláthatólag többszöri végrehajtást fognak megkövetelni. Ez előfordulhat esetleg egy projekt miatt is. Bizonyos fejlesztések miatt elképzelhető, hogy napi szinten kell ugyanazt a tesztet, teszteket végrehajtani. Ekkor mérlegelni kell, hogy az adott projekt milyen fontossággal bír.

Mint látjuk, a tervezés nagyon sokrétű tevékenység. Végeláthatatlan lehet ez a folyamat, és tapasztalatom alapján nagyon nehéz teljesen precízen is kivitelezni. Mindig van valami, ami keresztülhúzhatja a számításainkat.

7.2. Végrehajtás

A tesztvégrehajtás több ponton is megtörténhet az alkalmazásfejlesztés, vagy a tesztelési ciklus során. A létező regressziós tesztjeinket bármikor végre tudjuk hajtani az új funkcionalitások fejlesztési szakaszában. Ezzel elérjük, hogy folyamatosan meg tudunk győződni arról, hogy a tesztjeink megfelelően működnek, valamint nem történt olyan módosulás az alkalmazásban, amit mondjuk egy új fejlesztés indukált.

Tesztvégrehajtás történhet még természetesen a dedikált tesztelési időszakokban. Általában ez az időszak az, melyekor a legjobban számítunk is a tesztekre. A végrehajtás az előzőleg elvégzett tervezés alapján történik.

A tesztvégrehajtás alatt számos dologra kell figyelnünk:

- Tesztjeink futási ideje
  Előzetesen egy átlagolt idő alapján becsültük meg a tesztjeink futását. A végrehajtás során oda kell figyelnünk, hogy ténylegesen mennyi ideig tartottak a tesztek futásai. Azért fontos ez, mert egyrészt meg tudjuk állapítani, hogy a terveknek megfelelően hajtódnak-e végre a tesztek, vagy sem. Ha túl lassan történik a tesztek végrehajtása, akkor felmerülhet bennünk a kérdés, hogy mi okozhatja ezt? Lehetséges, hogy a tesztkörnyezet állapota nem megfelelő. Ebben az esetben ezt jeleznünk kell a fejlesztőknek. Lehetséges, hogy az általuk fejlesztett valamilyen kódrészlet okoz teljesítmény problémákat.
  Fontos átgondolnunk, hogy abban az esetben, ha előre nem látott tesztek futtatására is igény van, akkor ezek milyen mértékben befolyásolják a tervezett végrehajtási időket? Szükséges, vagy

sem jeleznünk a tesztelők felé, hogy a végrehajtásban csúszás lesz, mert valamilyen projekt miatt elkerülhetetlenül más teszteket is végre kellett hajtanunk?

- Rögzített hibajegyek
  Tartsuk számon, hogy milyen hibákat sikerült találnunk az automatikus tesztek futtatásával. Ezekre szükségünk lesz a végső riport készítésénél. Gyűjtsük össze, hogy aktuálisan melyiknek mi a státusza. Nézzük meg, hogy mikor kellett egy tesztet újrafuttatnunk egy hibajegy újra tesztelése miatt.

- Tesztjeink állapota
  Számtalan esetben megtörténhet az, hogy a tesztvégrehajtási szakaszban derül ki, hogy valamelyik tesztünk nem megfelelően működik. Általában ezt az okozhatja, hogy valamilyen lépést nem tud végrehajtani a tesztünk, így sikertelen tesztvégrehajtás fordul elő. Ez sokszor olyan fejlesztések eredménye, melyekről nem tudtunk.
  Mivel a tesztelés folyamán nagyon kell tartanunk az időbeosztást, ezért számtalanszor van az, hogy gyorsan kell a teszteket javítani. Ez nem mindig jár a megfelelő standardok betartásával. Tehát tartsuk számon, hogy melyek voltak azok a tesztek, amelyeken karbantartást kellett végezni, így ezeket a tesztciklus végeztével újra vegyük elő, és megfelelően frissítsük őket.

- Időben történő válasz a tesztelőknek
  Ne felejtsük el, hogy az automatizált tesztelés nemcsak az automatizálási csoportnak szól. Sőt. Az automatizált tesztelés inkább a tesztelőknek ad segítséget. Így fokozottan kell figyelnünk és szem előtt tartanunk, hogy a tesztek végrehajtását igazítsuk a tesztelők és az igénylők kéréséhez. Nagyon pontos tervezést kíván meg ez a folyamat. A tesztelőknek sok feladatuk van egy tesztelési ciklus alatt, ezért fontos, hogy mindig a lehető legoptimálisabban szolgáltassuk vissza egy adott automatikus tesztfutási eredményét.

- Szoros kommunikáció a csapaton belül
  Az automatizálási csapatnak napi szinten kell kapcsolatban lenniük egymással, valamint a tesztelőkkel. Mivel ebben az időszakban van a legnagyobb terhelés a tesztek végrehajtásában, így tudni kell, hogy éppen ki mivel foglalkozik. Ki az, aki éppen teszteket tart karban, ki az, aki végrehajt és ki az, aki kiértékel. Tisztáznunk kell, hogy ki milyen futtató állomásokat használ, és milyen számban. Amennyiben egy nagy prioritású teszt futására van igény, úgy egymás között tudni kell megosztani az elérhető erőforrásokat. Minden az optimális működésről szól.

7.3. Együttműködés a tesztelőkkel

Hiába a lehető legalaposabb tervezés minden tesztelés előtt, ha nem tudunk együttműködni a környezetünkben lévő tesztelőkkel. Általában ők azok, akiktől az automatizmusra igény van, ők tudják nekünk megmondani, hogy az adott területen mik az üzleti folyamatok.

Amikor elkezdünk együtt dolgozni a tesztelőkkel, számos tudást fogunk felszedni az adott terület teszteléséről, folyamatairól. Nézzük meg, melyek azok a pontok, amelyekben mi is részesei lehetünk az automatikus teszten felül a sikeres tesztvégrehajtásban:

- Lehetséges hibák jelentése
  Tesztfuttatás után az automatikus tesztjeink eredményeit zömével a tesztautomatizáló csapat értékeli ki. Amikor hibát mutat egy teszteredmény, elsőnek az automatizálási csapatban kezdődik meg a kiértékelés és az eredménynapló ellenőrzése. Ki kell deríteni, hogy a hiba miből adódhat. Tesztfutási hiba, vagy tényleg egy futtatás alatt észlelt hibáról van szó?
  Mivel az automatizáló csapat nem mindig ért a legjobban az adott területhez, ezért a manuális tesztelőkkel közösen próbálják meg kideríteni, hogy ténylegesen az alkalmazásban van-e a hiba. Ez nem mindig egyértelmű egy tesztriportból. Mivel a tesztek elkészülése során, szorosan együtt dolgozunk a tesztelőkkel, így mi is nagyban tudunk segíteni a hiba forrásának felderítésében.

- Tesztvégrehajtás újraütemezése
  Tesztelési ciklusokban előfordulhat, hogy valamilyen okból kifolyólag az előzetes tervvel szemben módosításokat kell véghezvinnünk. Lehetséges, hogy a tesztelendő alkalmazás állapota nem megfelelő, netalán a programozók még nem készültek el a kódokkal. Ekkor a tesztelő jelzi a csapatnak, hogy még ne futtassák a teszteket, mert felesleges időpazarlás lenne.
  Mivel sok más automatikus tesztet is végre kell hajtani, így le kell egyeztetni, hogy mi az a legközelebbi időpont, amikor újra be tudjuk illeszteni a tesztet a futtatáshoz.

- Újrafuttatások kezelése
  Természetesen előfordulhat, hogy valamilyen okból teszteket kell újrafuttatnunk, és nem azért, mert azok hibásan hajtódtak végre. Lehetséges, hogy a manuális tesztelők találtak egy incidenst, amit jelentettek. Miután a fejlesztők javították az incidenst, a tesztelők megkérnek minket, hogy hajtsunk végre pár automatikus tesztet, hogy megbizonyosodjunk a kód javítása nem indukált más hibákat.

- Egyszeri, vagy ütemezett tesztvégrehajtás
  Abban az esetben, ha azt tapasztaljuk, hogy több tesztelési ciklusban is jelentkezik bizonyos tesztek újrafuttatásának igénye, célszerű egyeztetni a tesztelőkkel, hogy nem lenne-e hatékonyabb, ha a teszteket mondjuk ütemezett végrehajtással terveznénk be. Lehetséges, hogy adott projekt indukálta a többszöri végrehajtást. Viszont, ha ez független a projektektől, akkor célszerű megtervezni a teszteket ütemezett végrehajtását.

- Tesztfuttatás elvetése
  Tesztvégrehajtási ciklusban - mivel nagyon sok feladat hárul minden egyes tesztelőre így találkozhatunk olyan tesztvégrehajtási kéréssel is, ami nem feltétlen a legjobb megoldás. Az előzőekben említett tesztújrafuttatás példáját hoznám fel. Lehetséges, hogy egy automatikus teszt tervezett végrehajtása csúszásban lesz. Majd a tesztet ismét végre kellene hajtani a tesztciklus legintenzívebb pontján. Az is lehet, hogy minden erőforrásunk foglalt és csak órák

múlva tudnánk a kérést teljesíteni. Ekkor fel kell mérni, hogy mennyi időt venne el az adott teszt manuális végrehajtása. Amennyiben azt tapasztaljuk, hogy jobban megéri, ha a tesztelő manuálisan végrehajtja a tesztet, akkor ezt jelezzük.

Ahhoz, hogy egy automatizált szoftvertesztelést a lehető legoptimálisabban tudjunk beépíteni a meglévő folyamatainkba, nagyon pontos tervezés szükséges. Időről-időre a csapatunk tagjai nélkülözhetetlen tapasztalatokat fognak szerezni, melyek hatékony tesztvégrehajtást, együttműködést, profizmust fognak eredményezni.

8. Automatizált szoftvertesztelés üzemeltetése

8.1. Folyamatosan ellenőrizni tesztjeink állapotát

Amint látjuk, az automatizált szoftvertesztelés is valamilyen szinten alkalmazásfejlesztés. Amitől különbözhet ez a tevékenység egy tényleges fejlesztési munkától az az, hogy a lefejlesztett tesztjeinket folyamatosan karban kell tartanunk. A megszokott automatikus fejlesztési folyamatok, mint egy spirál, folyamatosan mennek előre, és folyamatosan érintjük is a megszokott fázisokat (követelmények specifikálása, tervezés, tesztkészítés, tesztelés és a tesztek használata). Ebben a folyamatban viszont nincs benne a már elkészült tesztek állapotainak ellenőrzése.

Azért készítünk automatikus teszteket, hogy azokat végrehajtsuk és a tesztelési folyamatokat javítsuk ez által. Viszont, ha megfeledkezünk arról, hogy a tesztek minőségét ellenőrizzük, akkor nem biztos, hogy a megfelelő hozadéka van egy adott tesztnek.

Elképzelhető, hogy az igényeknek megfelelően bizonyos funkcionalitások változnak a tesztelés alatt lévő alkalmazásban. Nekünk tudnunk kell, hogy melyek azok a pontok, ahol változások történnek, és ezekhez a változásokhoz az automatikus tesztjeinket is megfelelően kell karbantartani.

Próbáljunk meg figyelemmel lenni azokra a jelekre is, melyek a mérési adatainkból erednek, mint például: milyen rendszerességgel futnak a tesztek, mennyi időt fordítunk tesztelési időszakról tesztelési időszakra bizonyos tesztek karbantartására. Ha azt tapasztaljuk, hogy több idő megy el egy teszt javítgatásán, mint amit nyerünk a futásából, érdemes megvizsgálni, hogy a megfelelő tesztelési folyamat lett automatizálva, vagy nem? Tudja a tesztigénylő, hogy pontosan mit is akar? Eredhet ez akár a folyamatos változásból is, ami a tesztelés alatt lévő alkalmazás oldalán jelentkezik?

Sokszor nem jut el időben egy igény az automatizálásra. Ekkor előfordul, hogy gyorsan, és nem túl szép megoldások készülnek. A szép megoldásokon az automatikus teszt kódját és annak milyenségét értem (kommentelés, formázás, megfelelő standardok használata). Ebben az esetben kimaradhatnak kulcsfontosságú elemek, mint például a megfelelő helyreállítási forgatókönyvek. Bármi is legyen ez, törekednünk kell arra, hogy ha nem is a megfelelő időben, de később visszanyúlva az adott teszthez ezeket pótoljuk. Ne legyünk kényelmesek. Törekedjünk a pontos és precíz tesztkészítésre!

[15]
A tesztjeink állapotát folyamatosan verifikálhatjuk és validálhatjuk. Melyek a különbségek és pontosan mit is jelentenek ezek a szavak?

- Verifikáció
  Helyesen (megfelelő minőségben) készítjük az automatikus tesztet? Tehát megvannak az alapdolgok, melyek elvárhatók egy tesztimplementáció során? Törekedtünk arra, hogy hibamentes automatikus tesztet készítsünk?

- Validáció
  A megfelelő automatikus tesztet készítjük? Tehát a funkcionalitás megfelel annak, ami az elvárt volt? Azokat a lépéseket hajtjuk végre, melyeket kell, és nincs semmi olyan lépés a tesztünkben, ami nem az elvárt lenne?

A verifikáció az a folyamat, amikor az elkészült automatikus tesztet ellenőrizzük, hogy megfelel-e az elvárt igényeknek, míg a validáció pedig az a folyamat, amiben ellenőrizzük, hogy a specifikációk, az elvárások megfelelnek az igénylő kéréseinek.

8.2. Tesztadatok pontossága

Egy adott automatikus tesztet általában egy adott adathalmazzal készítünk el. Ez a teszt az alkalmazás adott verziójában tökéletesen működik. Azonban tesztelési ciklusonként változhatnak a tesztadatok. A tesztadatok változását számos dolog indukálhatja, igazából a tesztelés akkor jó, ha folyamatosan próbáljuk váltogatni az adathalmazt.

Csakhogy előfordulhat az az eset, amikor számos tesztvégrehajtási ciklus után, egyszer csak az automatikus tesztünk hibára fut. A tesztkiértékelés folyamatában azt tapasztaljuk, hogy egy bizonyos tesztadat használata során nem működik megfelelően a tesztünk. Ilyenkor elindul egy manuális hibafeltárási folyamat. Azt vesszük észre, hogy nem az automatikus tesztünkkel van a probléma, hanem a tesztadat használata során egy felugró ablak jelent meg a felhasználói felületen, ami nem volt implementálva a tesztünkben. Ekkor fel kell keresni az adott terület tesztelőjét, és megkérdezni, hogy ez az elvárt működés vagy sem? Amennyiben igen, úgy az automatikus tesztünket fel kell készíteni az ilyen esetekre.

A jó minőségű és helyes tesztadatok használatával elkerülhetünk számos, sok időt igénybevevő beszélgetést a tesztelőkkel. Az alábbi beszélgetés egy tesztelő és egy automatikus tesztvégrehajtó között zajlik, miután a teszt helytelenül futott le. A párbeszédet az automatikus teszt végrehajtója kezdi:

- *Szia. Lefuttattam a kért automatikus tesztet, de az ötödik és hatodik esetben második végrehajtás után sem sikerült helyes eredményt elérni.*
- *Szia. Mit értesz helyes eredményen?*
- *Igazából már a legelső lépésnél nem ment tovább a teszt, amikor a vevő adatait írtuk be.*
- *Értem. Mutasd légy szíves, mi is volt az adat, amit használt a teszt?*
- *Mindkét esetben ugyanaz a vevő volt.*
- *Várj, mindjárt megnézem a rendszerben. Már látom is. Igen, itt lesz a probléma. Ezt a vevőt a múlt hónapban inaktiváltuk a rendszerben. Ezért lehet az, hogy nem találta meg a teszt már az elején.*
- *Így már világos. Ezzel az adathalmazzal hajtjuk végre a tesztet már egy éve. Szerintem ideje lenne, ha megnéznétek a tesztadatokat, és ahol ez szükséges, frissítenétek.*
- *Igen, ez teljesen igaz. Azonban ilyen előfordulhat, de igyekszünk erre a jövőben odafigyelni.*

- *Rendben, köszi. Szia.*
- *Szia.*

A párbeszédből jól látszik, hogy olyan eset is adódhat, hogy egy korábbi tesztadat már nem lesz elérhető a rendszerben. Az automatikus teszt végrehajtása során derül ez ki, amit jelezni kell. Gondoljunk bele abba, mennyivel egyszerűbb lenne, ha minden tesztelési folyamat kezdete előtt ellenőrizni lehetne a tesztadatokat is. Igazából az adatok milyensége és a tesztadathalmaz karbantartása a tesztelők dolga, hiszen ők értenek az adott területhez mélyebben, ők tudják mikor milyen tesztadat a megfelelő.

8.3. Optimalizálás

Az optimalizálás egy olyan része az automatizált tesztelésnek, amelynek mindig a szemünk előtt kell lebegnie. Vajon miket és mit érdemes optimalizálni?

- Tesztfutási idők
  Általában minden végrehajtási ciklus során és után ellenőrizzük a tesztjeink futási idejét. Kiértékeljük, hogy melyik teszt mennyi ideig futott, és hol voltak nagyon nagy eltérések. Amennyiben azt tapasztaljuk, hogy bizonyos tesztjeink futási ideje elég sok időt vesz igénybe, célszerű megvizsgálni, hogy mit lehetne tenni annak érdekében, hogy gyorsítsunk a folyamaton. Miért fontos ez? Kezdetben még lehetséges, hogy nem fog gondot okozni, ha egy tesztünk futása elnyúlik. Azonban mikor növekszik a tesztállományunk, és a futtató állomások is korlátozva vannak, akkor a napi huszonnégy órában bele kell tudnunk sűríteni a legoptimálisabb módon a teszteket. Sok kis apró javítás a tesztek futásán nagy segítség tud lenni.

- Tesztelési folyamatok
  Ez összhangban lehet az előző ponttal. A tesztelés folyamatának optimalizálása a tesztjeink felsorakoztatásával és végrehajtásával függ össze. Egy jól megtervezett, állási idők nélkül, vagy legalábbis minimalizált állási időkkel, sokkal hatékonyabb lehet a tesztelési folyamat. Tisztában kell lennünk a tesztek egymással való függőségeivel, és természetesen az adott üzleti igényből fakadó tesztelési igényekkel.
  Természetesen a tesztek eredményének kiértékeléséhez idő szükséges. Ez nem azt jelenti, hogy ekkor nem lehet teszteket végrehajtani. Törekedjünk arra, hogy a tesztek futtatása előtt a lehető legrövidebb idő teljen el, ezzel folyamatosan közelíteni fogunk a gördülékeny végrehajtáshoz.

- Tesztimplementálás (modularizálás)
  Tesztkészítés során, mint azt már tudjuk, fokozottan ügyeljünk és figyeljünk a tesztjeink modularizálására. A legoptimálisabb az, ha tisztában vagyunk azzal, milyen meglévő moduljaink vannak, melyeket felhasználhatunk tesztek készítése során. Törekedni kell, hogy ne alakuljon ki kódismétlés. Nagy mennyiségű modul esetén egyre nehezebb lesz erre odafigyelni. Így inkább arra fordítsunk időt, hogy a meglévő teszt- és modulhalmazból válasszuk ki a megfelelőt, ami szükséges az adott teszt elkészüléséhez. Ez a hosszú távú tesztkarbantartást fogja segíteni.

- Erőforrások

  Erőforrások alatt természetesen itt emberi és gépi erőforrásokat is értünk. Emberi erőforrás optimális kihasználtsága alatt érthetjük azt, hogy ki-mikor-mivel foglalkozik éppen.

  Tegyük fel, hogy egy tesztelési időszak kellős közepén vagyunk. Az automatizálással foglalkozó tesztelő éppen tesztvégrehajtáson dolgozik, valamint a teszteket monitorozza. Azonban jelentkezik két-három olyan teszt, amit sürgősen javítani kell. Ekkor ne azt tegyük, hogy azon nyomban elkezdünk teszteket javítani. Tervezzük meg, hogy ilyen esetekre hogyan reagálunk. Inkább legyen dedikált idő egy tesztelési ciklusban is arra, hogy teszteket javítsunk. Ilyen esetben, amikor hiba jelentkezik, a legrosszabb, ha rögtön próbálunk javítani a teszten. Ez kihatással van az előzetesen felállított ütemtervvel. Inkább várjunk bizonyos ideig, hátha más teszteknél is jelentkezik, hogy módosításokat kell végrehajtanunk. Ne szabdaljuk szét a tesztvégrehajtási fázist karbantartással.

  Ugyanez a helyzet a gépi erőforrásokkal is. Amikor teszteket javítunk, akkor is gépi erőforrásokra van szükség. Inkább hagyjuk, hogy a gépeket a tesztek használják, és majd egy olyan időintervallumban foglalkozzunk a tesztek karbantartásával, amikor nem hátráltatjuk a tesztvégrehajtást.

- Tesztek minősége

  Időről-időre a tesztjeink minőségét is célszerű ellenőrizni. Minőség alatt itt azt értem, hogy a végrehajtott tesztlépések tényleg azt a folyamatot érintik, amit kell. Menjünk vissza az előző, tesztfutási időponthoz. Igen, vizsgáljuk, hogy egy teszt mennyi ideig fut. Ekkor azt feltételezzük, hogy a megfelelő lépéseket hajtja végre.

  Viszont ahogy haladunk az automatizálással, úgy azt is vizsgálnunk kell, hogy ugyanazt a tesztvégrehajtást nem lehet-e valahogy optimalizálni, és kevesebb lépésből is kivitelezni.

  Erre vonatkozik a tesztek minősége. Tegyük fel, hogy létezik egy automatikus tesztünk, ami tökéletesen működik. Az egyik csoporttagunk azt javasolja, hogy ha bizonyos köztes lépéseket kiváltanánk egy hatékonyabb adatbázis lekérdezéssel, akkor optimálisabb tesztvégrehajtást tudnánk elérni.

  Ekkor meg kell vizsgálni, hogy mi az eredeti tesztünk viselkedése. Készítsük el a javaslat alapján a módosítást, és nézzük meg, hogy az eredmény ebben az esetben mi. Amennyiben a tesztelőknek is megfelel a módosítás, valamint sokkal hatékonyabb ez által a tesztünk végrehajtása, célszerű módosítani a tesztünket az optimálisabb tesztvégrehajtás érdekében.

8.4. Folyamat működésének felügyelete

Egy jól megtervezett és bevezetett automatizálási folyamat kezdetben nagyon nagy hatékonysággal tud működni. Ahogy telik az idő és új kollégák, valamint új alkalmazások jelennek meg, az igény folyamatosan növekedni fog az automatizmus iránt. A kiépített folyamatainkat folyamatosan monitorozni kell. Abban az esetben, ha azt tapasztaljuk, hogy a hatékonyság csökken, kezdjük el felmérni, hogy ezeket milyen tényezők okozhatják.

A főbb fázisok, amelyeket célszerű minden esetben megfelelően felügyelni:

- Manuális tesztek feltérképezése és begyűjtése
- Automatizált tervek megfelelő elkészítése
- Automatizált tesztek implementálásának folyamata
- Tesztelési ciklusok és végrehajtás tervezése
- Optimalizálási pontok feltérképezése
- Tesztek karbantartása
- Metrikák megfelelő megléte

8.5. Mérési adatok pontos kiértékelése

Egy automatizálási folyamat és a folyamat megfelelő üzemeltetése nem mehet megfelelő metrikák nélkül. Azzal, hogy adatokat gyűjtünk, nem oldódik meg minden. Az adatokat olyan formába kell hoznunk, melyek azokat az információkat mutatják, amelyek számunkra szükséges.

Ne próbáljunk meg olyan adatokat készíteni, amelyek elmoshatják a felmerülő, vagy meglévő problémákat. Törekedjünk arra, hogy a legpontosabb kimutatásokat tudjuk készíteni. Igaz, hogy az automatizálás hatékonyságát csakis pozitív számadatokkal tudjuk megmutatni, viszont ha a jól felállított mérések alapján azt látjuk, hogy valami nem a megfelelő módon történik, úgy ez egy jel számunkra, hogy valamit máshogy kell csinálnunk, vagy valamin javítanunk kell.

Egyszer az automatizálási munkám során találkoztam olyan folyamattal, amit nyolc részre lehetett szedni az adott tesztelési feladatoknak megfelelően. Automatikus tesztekkel voltak lefedve ezek a folyamatok és jól is működtek. Pár év elteltével azt tapasztaltam, hogy van két-három olyan pont a folyamatban, ami nem sok hozadékot adott a folyamat egészére nézve. Felkerestem az adott terület tesztelőjét, és elmondtam neki, hogy a folyamatban célszerű lenne nem automatikus teszteket használni, mert mindig javítgatni kellett valamit a folyamatosan változó tesztadatoknak köszönhetően. A mérési adatok azt mutatták, hogy több időt töltöttünk el tesztkarbantartással, mint amit a tesztek visszahoztak a futtatásból. Ezeket a köztes lépéseket kiváltottuk manuális tesztelésre, így bár igaz egy picivel több kommunikációt igényelt a folyamat, viszont nem ment el folyton az idő a tesztek javítgatásával.

A fenti példa is mutatja, hogy bizony léteznek olyan esetek, amikor meg kell válnunk az automatikus tesztektől. Ezek nem azért vannak, mert akkor, amikor a tesztet készítettük, rosszul terveztük meg a folyamatot. Ezt lehet, hogy a jelenlegi helyzet hozta így, hiszen minden folyamatosan változik.

Meg kell tudnunk találni a megfelelő egyensúlyt a manuális és az automatikus tesztek terén. Annak nincs értelme, hogy mindent automatizálni akarunk, a kettő működésének a lehető legjobb egyensúlyban kell lennie.

A pontos kiértékeléshez elengedhetetlen, hogy a megfelelő mértékegységeket használjuk. Például, ha rögzítjük, hogy egy tesztelő mennyi időt tölt el tesztimplementálással, akkor nyílván célszerű órát, vagy percet használni. Ez attól függ, mennyire pontosan akarjuk mutatni az adatot. Viszont, ha oda kerül a sor, hogy a tesztelő egy napjából mennyi idő áll még rendelkezésre, nem mindegy, hogy az adott napot mennyi órának tüntetjük fel. Általában egy nap nyolc óra. Azonban léteznek olyan fejlesztői csapatok is, akik csak hat órával, vagy akár nyolc órával többel számolnak. Ezt mindig az adott csoport és a szervezet határozza meg.

Az ilyen számadatok felállítását nagyon körültekintően kell meghatározni, valamint ezeknek egyértelműnek kell lenni mások számára is.

Képzeljük el, hogy egy riportálás során csak ennyit mutatunk. „Szilárd a héten 85%-os leterhelésen van.” Rendben. Ez viszont nyolc órás, hat órás napi munkamennyiséghez képest van így? Tisztán kell látni, hogy milyen adatot, miből és hogyan származtatunk.

9. Automatizált szoftvertesztelési buktatók, amelyekre érdemes figyelni

Ebben a fejezetben szeretném felhívni a figyelmet három olyan dologra, melyeknek a figyelmen kívül hagyása nagyban befolyásolhatja az automatizálási folyamataink sikerességét. Elképzelhető, hogy tökéletesen működnek a folyamataink, a csoportunk a lehető legoptimálisabban dolgozik. Azonban annak érdekében, hogy ez a folyamatos minőség fenntartható legyen, nagyon nagy energiákat kell mozgósítani. Az elmúlt évek tapasztalatai során azonban szembesültem azzal az állapottal is, hogy valami nem úgy működött, mint a korábbi években. Meg kell vizsgálni, hogy melyek azok a tényezők, amelyek a folyamatos sikereknek gátat szabhatnak, mert jobb felkészülni időben a nem várt eseményekre.

A következő három példát, mint jó gyakorlatot szeretném megosztani az olvasóval.

9.1. Környezetváltozás

A környezetváltozáson nem a fizikai változást értem, azaz, hogy például a cégünk elköltözik egy másik irodába vagy városba. Itt inkább arról van szó, hogy a körülöttünk dolgozók, kollégák, más területen működő szakemberek változnak.

Nem feltétlenül kell hatalmas szervezeti méretben gondolkozni. Például legyen egy ötfős automatizálási csapat, tizenöt fős tesztelői csapat, és egy harmincfős üzleti elemző társaság. Tegyük fel, hogy amikor elkezdtünk automatizálással foglalkozni, akkor ezek a kollégák voltak a cégnél, akik most is. Hallották az összes olyan dolgot, ami a célja az automatizálásnak, így szinte a mindennapi rutinba beleépült az a folyamat is, amit az automatizálás kíván meg. Azonban ez az állapot nem mindig marad meg. A kollégák változhatnak, elképzelhető, hogy növekszik valamelyik csoport.

- Új tagok megjelenése a csapatban

Az első olyan dolog, amire érdemes odafigyelni, hogy a környezetünkben lévő csapatok mérete és a csapatban lévő tagok is változhatnak. Új kollégák jelennek meg, és ez az ütem valahol gyorsabban, valahol lassabban jelentkezik. Azt is célszerű szem előtt tartani, ha létezik más lokációja is a vállalatnak. Főleg ha multinacionális környezetről beszélünk. Ekkor még dinamikusabban jelentkezhet ez a bővülés. Miért is fontos ezt szem előtt tartani?
Az új kollégák, attól függően, hogy honnan érkeztek, elképzelhető, hogy rendelkeznek valamilyen mértékű automatizálási tapasztalattal. Természetesen itt a tesztelői, vagy akár az üzleti elemzői csoportról beszélek, nem pedig az automatizálási csoportról. Tehát lehetséges, hogy van némi tapasztalatuk korábbról automatizálással. Ez viszont nem azt jelenti, hogy pontosan olyan mértékű és tulajdonságú, mint ami a jelenlegi szervezetben létezik. Annak érdekében, hogy a továbbiakban is a megfelelő hatékonysággal tudjunk működni, célszerű az új kollégáknak is megismerni az automatizálás előnyeit, és azokat a folyamatokat, melyek mentén működik az automatizálás a cégen belül. Milyen módon lehet megismertetni ezeket a folyamatokat? Az alábbi lista tartalmazza a legalapvetőbbeket:

- Oktatások

- Általános automatizálási érvek
- A „Mire jó és mire nem az automatizálás?” kérdéskörének átgondolása
- Annak megismerésre, hol helyezkedik el az automatizálás a folyamatainkban
- Pozitív és negatív példák felmutatása

Miért is fontos ez? Kerüljük el azt, hogy az idő múlásával nem jön hozzánk a megfelelő információ, és nem keres minket a környezet, hogy segítsünk a tesztelési folyamatokban. Nem lehetünk mindig, mindenhol ott, így el kell érnünk azt, hogy az alapvető tudást átadjuk a kollégáknak, mégpedig azon a szinten, hogy felismerjék, mikor és hol kell alkalmazni az automatizálást.

- Kollégák távoznak a vállalattól

A következő dolog, ami nagyban meghatározza az automatizálást és az azzal foglalkozók munkáját, ha valaki, vagy valakik elhagyják a céget. Ez mondhatni általános dolog, hiszen nem várható el mindenkitől, hogy elkötelezze magát egy adott cégnek. A probléma akkor kezdődik, ha olyan tesztek és folyamatok voltak automatizálva, melyek az eltávozott kollégát is érintették, netalán ő volt a felelős az automatikus tesztért.
Gondoljunk bele, hogy milyen következménnyel jár, ha megválunk egy olyan kollégától, akinél az üzleti tudás és az automatikus teszt megléte is számon volt tartva. Nagyon nagy nehézséget fog ez okozni hosszabb távon, hiszen nem volt megfelelően dokumentálva az a folyamat, hogy mikor és hogyan használta az adott kolléga a tesztet. Aki a helyére jön, vagy netalán más valaki veszi át a feladatait, nem biztos, hogy tudni fogja, mit is csinál pontosan az az automatikus teszt, és hogy mikor kit kell keresni, ha kérdése, problémája van. Ekkor az automatizálással foglalkozó csapat felelőssége, hogy azt a folyamatot és a tesztet megismertesse az új kollégával. Valljuk be: ez idő- és energiaigényes, valamint nem a leghatékonyabb módszer.
Ehelyett célszerű már az automatizálás során figyelni arra, hogy a megfelelő dokumentálás és folyamat át legyen adva a tesztelőknek, vagy üzleti elemzőknek. Legyen a felelősség is az ő oldalukon, hogy erre figyeljenek oda, hiszen az ő érdekük is, hogy minden a leghatékonyabban működjön.

- Üzleti folyamatok implementálása a tesztekbe

Ezzel önmagában nincs is semmi probléma, hiszen egy automatikus teszt nemcsak mezőket és gombokat nyomogat, hanem sokszor bizony üzleti logika is van implementálva benne. Akkor jelentkezik a probléma, ha az előző pontban említett eset is előfordul, azaz ha egy kolléga távozik. Nem biztos, hogy az az üzleti folyamat is rendesen volt dokumentálva, és ebben az esetben azt tapasztaljuk, hogy „csak” egy automatikus tesztünk marad. Nem kötelessége az automatizálási csapatnak tudnia minden üzleti folyamatot, és azt sem, hogy mi miért zajlik. Nem is felelősségük. Viszont az, ha a másik oldalon, akiknek végrehajtjuk a tesztet, nincs meg az információ, hogy mit is csinál pontosan a teszt, az már probléma. Ez sajnos további félreértéseket is generálhat, hiszen az üzleti elemzői csapat, ha megkapja az eredményeket, nem biztos, hogy a megfelelő eredményeket látja. Elindul egy folyamat, és az automatizálási csapatnál kezdenek el érdeklődni, hogy az eredmények miért azt mutatják, amit. Mivel nem kötelessége a csapatnak tudni, hogy az adott adatmennyiséggel mi is az elvárt eredmény, így nem fogja tudni megmondani, hogy mi is okozta a problémát. Az automatizálási csapat felhívja a figyelmét az üzleti elemzői csapatnak, hogy ez bizony üzleti logika, vagy folyamat miatti eltérés lehet, ami nem a teszt

hibája. Ekkor döbbennek rá, hogy ők maguk sincsenek tisztában azzal, hogy mi okozta a hibás eredményeket.
A következő lehetőség pedig az, ha valamiért (pontosan akkor, amikor az üzleti elemzői csapat szeretné, hogy a teszt végrehajtódjon), nem érhető el az automatizálási szakember. Ez lehet szabadság, betegség stb. Mivel nem tudják, hogy mi is volt a folyamat, és mit miért kell csinálni, mert a teszt jóval korábban készült el még egy akkori kollégával, így nincs információ, hogy pontosan mit is kell tenni. Az automatikus teszt itt egy olyan üzleti folyamatot hordoz magában, ami másnak nincs meg, és ez nagy rizikó, mind az üzleti elemzői, mind az automatizáló csapatnál.

9.2. Egyre tapasztaltabb tesztelők

Ez egy kicsit ellentmondásos lehet. Természetesen nagyon jó, ha egyre tapasztaltabb tesztelők erősítik a csapatot, de most elsődlegesen gondoljunk a manuális tesztelői csapatra. Szerencsére az elmúlt évek tapasztalatai alapján, egyre többet lehet hallani szoftvertesztelésről egyetemi előadásokon, kurzusokon. Ezzel nagyban hozzájárulnak a felsőoktatási intézmények, hogy az egyetemről kikerülve már megfelelő alaptudással rendelkezzenek a hallgatók. Mivel az ilyen fajta előadások, oktatások elég széles körben érhetőek el, így nagy valószínűséggel programozást tanulók is részesülnek benne.

Amint egy ilyen végzős diák elkezdi a munkát, mint szoftvertesztelő, az újdonság erejével hat számára minden. Munkakörnyezet, folyamatok, emberek, céges kultúra és még sorolhatnánk. A bizonyítási vágy és a lendület indukálni fogja, hogy a manuális tesztelési munkákba belevigye az egyetemi évek alatt felszedett programozói tapasztalatot. Bármilyen meglepő, elkezdődik egy „automatizálási” folyamat, amiről lehet nem is tud a dedikáltan automatizálással foglalkozó csoport.

Ez miért probléma? Az alábbi lista tartalmazza a főbb okokat, hogy miért nem javasolt bizonyos esetekben automatizálással foglalkoznia a manuális tesztelőnek:

- Nem ismerik az automatizálási standardokat
- Nem tudják, melyek a buktatók
- Nem tudják megfelelően felmérni, hogy érdemes vagy sem automatizálni
- Megfosztják az automatizálással foglalkozókat a munkától
- Nem biztos, hogy a megfelelő hatékonysággal készítik a teszteket
- Nem gondolnak a hosszú távú karbantarthatóságra
- Nem megfelelően mérlegelik a teszt hozadékát
- Nem tudják, hogy létezik-e már az adott tesztelésre automatizmus

Nincsen azzal probléma, hogy másokat is vonz az automatizálás, és annak előnyei. Próbáljuk meg elérni, hogy mi, mint automatizáló szakemberek segítséget nyújtsunk nekik, és a megfelelő minőségben kerüljenek lefejlesztésre az adott tesztek. Ebben az esetben, ha nem az automatizáló csoport tagjai közül készíti a teszteket valaki, akkor érdemes azt is letisztázni, hogy a karbantartásért is az adott tesztelő felel, hiszen ha nem volt bevonva az automatizáló csoport, akkor a tesztelési eseteket, és folyamatot sem tudjuk. A felelősség a tesztelőnél van.

9.3. Metrikák

A metrikák, mérőszámok abszolút jelleggel nem buktatók, és nem is ezért szerepelnek a lista végén. Görgessük tovább az előző pontot egy kicsit, és gondoljunk bele, hogy mit indukálhat még az, hogy nem az automatizáló csapat végzi a tesztek fejlesztését?

Igen, mint ez a pont mutatja, a metrikákat érintheti, mégpedig sérülhetnek a számadatok. Egy automatikus teszt elkészülésével a tarsolyunkban kell lenni bizonyos számadatnak, melyeket az előző pontokban már említettünk, de elevenítsük fel újra őket. Melyek lehetnek azok a számadatok, amelyeket célszerű követni egy tesztimplementálása során?

- Mennyi időre volt szükség a tesztelési folyamat megértésére?
- Mennyi időre volt szükség esetleg tréningre?
- Mennyi időre volt szükségünk más tesztelőtől?
- Mennyi időt vett igénybe az automatikus teszt elkészülése?
- Mennyi idő volt a tesztünk letesztelése?
- Mennyi időt szántunk a megfelelő tesztadatok elkészítésére?
- Mennyi időt töltöttünk el a teszt karbantartásával, míg a megfelelő formába készült el?
- Mennyi ideig tart a teszt végrehajtása?
- Mennyi tesztelési ciklusban lesz előreláthatóan használva az adott teszt?
- Mennyi tesztadatra fog futni?
- Milyen környezetben kell végrehajtani?

A lista nem véges. Számos olyan kérdés merül fel, melyek adatokat szolgáltatnak az automatizáló csapatnak.

Amennyiben nem a csoportból készíti valaki a teszteket, ezekre nem is feltétlenül gondol. Lehet, hogy egyáltalán nem vizsgálnak számadatokat. Azaz abszolút nem lesz metrikánk, hogy a teszt további sorsáról dönthessünk, vagy dönthessenek. Nem fogják látni, hogy megérte, vagy sem implementálni az adott tesztet.

A jobbik eset, ha a fenti kérdésekből párat szem előtt tartanak és van adatunk. Viszont ebben az esetben sem átfogó a kép.

Miért is fontos, hogy tisztában legyenek a manuális tesztelők is a metrikák létezésével? Akár a tesztelő, akár az automatizáló csapat végzi a tesztek készítését, bizonyos időközönként kimutatásokat kell végezni. Ezek a kimutatások tartalmazhatják a tesztjeink számát, összes tesztfutási időket, tesztfutások számát és még sorolhatnám. Az automatizáló csoport felelőssége, hogy minden egyes automatikus tesztről a megfelelő képet lássuk és kapjuk a végrehajtások és karbantartások során.

Hogy miért is fontos, hogy a megfelelő képet kapjuk egy tesztről? Amikor egy automatikus tesztet elkészítünk, annak a létrejöttét egy adott igény, adott tesztelési folyamat, üzleti prioritás indukálhatta. Ahogy telik az idő, elképzelhető, hogy az alkalmazásaink olyan mértékben változnak, hogy teszteket már nem célszerű, vagy nem éri meg karbantartani. Bizony meg kell fogalmazni magunkban, hogy szükségünk van még az adott tesztre, vagy ne is használjuk, hiszen több a ráfordított idő akár a karbantartásra, tesztadat frissességre, mint korábban. Ekkor nem javasolt tovább használni a tesztet, hiszen csak az időt viszi el, minthogy a nyereséget termelné. Itt nem biztos, hogy a manuális tesztelő képes ezt felismerni.

10. Automatizált szoftvertesztelés, mint szolgáltatás

Elérkeztünk a tananyag utolsó fejezetéhez. Automatizálás, mint szolgáltatás. Elsőként érdemes lehet magát a szolgáltatás fogalmát egy kicsit megfogalmazni.

[13]
„A szolgáltatási szektor, harmadlagos szektor, vagy tercier szektor egyike a fejlett gazdaságok három legfontosabb gazdasági ágazatának.
A szolgáltatás olyan, kézzel nem fogható eredményű munkavégzés vagy jogosultság, mely annak fogadója, vagy élvezője számára értékkel bír, még ha nem is feltétlenül fizet érte.
A szolgáltatás nyújtója általában ellenértéket (gyakran pénzt) vár el a szolgáltatásért cserébe, vagy amennyiben a szolgáltatás egy termékhez kapcsolódik (pl. javítás, karbantartás), úgy előfordulhat, hogy az ellenszolgáltatás már a termék árába volt beépítve."

Itt érdemes háromféle szolgáltatást megemlíteni, ami az automatizált szoftvertesztelléssel lehet kapcsolatban.

1. Cégen belüli automatizálási szolgáltatás

Ahogy azt az egész tananyagon végigvezettem, feltételezzük, hogy a tesztautomatizálást a jelenlegi, aktuális munkaadónk telephelyén végezzük. Mi magunk is alkalmazottak vagyunk, és azért felelünk, hogy a cég tesztelési folyamatait támogassuk automatikus úton. Ebben az esetben melyek lehetnek azok a pontok, amelyek az automatizálási csoporttól, mint szolgáltatás megjelenhet?

- Folyamatos rendelkezésre állás

Elképzelhető, hogy olyan környezetben kell automatikus teszteket készítenünk és végrehajtanunk, melyek megkövetelik, hogy folyamatosan elérhető legyen a csapatunkból valaki. Ezzel elértük a készenlétet. Milyen esetben fordulhat ez elő? Példának szerepeljen egy atomerőmű, amiben igen kockázatos folyamatok zajlanak. Jogosan merülhet fel a kérdés, hogy milyen tesztelés folyhat egy atomerőműben? Itt nem feltétlen arra kell gondolni, hogy tesztrendszerek léteznek. Lehetséges, hogy magát az erőmű működésének felügyeletét végzik automatizált tesztek, melyeknek a leállása, vagy nem megfelelő futása kritikus lehet. Ekkor fontos, hogy valaki, egy automatizáló szakember elérhető legyen, és a lehető leghamarabb tudjon reagálni a felmerült hibára.

- Megfelelő folyamat szigorú követése

Említettem már ezt is korábban. Az automatizálásnak megfelelő folyamatokon keresztül kell történnie. Az egy dolog, hogy mi, automatizáló tesztelők bizonyos igényekkel állunk elő a tesztelők felől, de ugyanez érvényes a csapatra is. Kötelességünk, hogy a teszteket a megfelelő minőségben, a megfelelő dokumentációval ellátva adjuk át a tesztelőknek. Mivel nem elvárt, hogy a manuális tesztelő tökéletesen

értsen az automatizálási folyamatokhoz, ezért a dokumentációknak pontosnak és egyértelműnek kell lenniük.

- Gondoskodni kell az optimális tesztvégrehajtásról

Időről időre azt tapasztaljuk majd, hogy a tesztjeink száma és az igény nőni fog. Attól függően, hogy mennyi erőforrásunk van, ezeknek a teszteknek a végrehajtását a lehető legoptimálisabb módon kell végrehajtani. Mit értünk optimális alatt? Ez lehet a tesztelők igénye, a tesztelés alatt álló alkalmazás elérhetősége, az erőforrásaink rendelkezésre állása és nagyon sok egyéb dolog is. Ezeket szem előtt tartva, tudnunk kell egy megfelelő ütemtervet készíteni, hogy kinek és mikor fog végrehajtódni a tesztje. Nem szabad a saját időnkre korlátozódni, mert a tesztelési folyamatokban nagyon bonyolult folyamatok lehetnek. Elképzelhető, hogy egy tesztelőnek fontos, hogy a lehető leghamarabb hajtódjon végre a teszt, mert a saját tesztelési folyamatainak az az előfeltétele. Nem engedhetjük meg, hogy másokat hátráltassunk. Természetesen itt korlátot szabhatnak a rendelkezésünkre álló futtató állomások vagy eszközök is.

- Tesztek folyamatos karbantartása

Érdekes kérdés ez. Minél nagyobb a teszt-tárházunk, annál kevesebb időt tudunk minden egyes teszttel foglalkozni annak érdekében, hogy az aktuális tesztelési ciklusra tökéletesen működjenek. Sokszor nem jut el az információ az alkalmazásban történt változásokról az automatizálási csapathoz, így előfordul, hogy tesztvégrehajtás során jelentkeznek a hibák. A folyamatos karbantartás itt inkább azt mutatná, hogy amennyiben jelentkezik valamilyen rendellenes működés a teszt végrehajtása során, akkor azt a lehető leghamarabb javítsa az automatizálási szakember. Ez nyilván vonzza magával azt, hogy kapcsolatba kell kerülni az adott terület felelősével, és körbejárni, hogy ez pontosan hiba-e az alkalmazásban, vagy a tesztünk már nem az aktuális folyamatot tesztelné. Szem előtt kell tartani, hogy az adott teszt milyen prioritással bír. Lehetséges, hogy a tesztek végrehajtásában más tesztek is nagy prioritással bírnak, így a teszt karbantartása várat magára.

- Kollégák mentorálása

A 9.1-es fejezetben tárgyaltuk a környezetváltozást és azt, hogy milyen hatása lehet ennek az automatizálásra. Tudnunk kell úgy irányítani a környezetünket, hogy segítsük őket minden olyan pontban, ami számukra nem egyértelmű egy automatikus folyamatban. Melyek lehetnek ezek?

  - Mi is pontosan az automatizálás?
  - Mikor célszerű megkeresni az automatizálással foglalkozó csoportot?
  - Miért automatizálunk más tesztet és nem azt, amit szeretnének?
  - Miért kerül időbe az automatizálás?
  - Milyen információra van szükségünk?

A mentorálás, támogatás nem kell mindig, hogy szigorú keretek között menjen végbe. Elég, ha egy párbeszéd során felmutatjuk azokat a kérdéseket és válaszokat, amiből nagyon sokat tanulhatnak. Az alábbi példa egy párbeszéd egy tesztelő és egy automatizálással foglalkozó szakember között. A párbeszédet a tesztelő kezdi:

- *Szia, Tamás. Lenne rám egy fél órád? A tesztelésünk során kiderült, hogy egy új funkciót is implementálnak soron kívül a webes áruházba. Szükségem lenne egy tesztre, ami végigkattintgat minden gombot a felületen, és ellenőrzi a helyes telefonszámokat az oldalon.*
- *Szia, Sanyi. Jelen pillanatban ez sajnos nem fér bele. Éppen egy fontos integrációs tesztet futtatunk, majd utána szükség lesz egy gyors teszt-karbantartásra, mert nekünk sem szóltak, hogy változás lesz a megrendelések terén.*
- *Értem, de ez nagyon gyors lenne, és nem kerülne sok időbe. Öt lépésről van szó csupán.*
- *Milyen projektről is van szó?*
- *A bevásárlókosárnál minden országra megjelenítjük majd a megfelelő kapcsolattartó telefonszámot. Ezt kellene ellenőrizni.*
- *Elérhető már a felület?*
- *Egy része.*
- *Változni fog még?*
- *Igen, úgy néz ki, hogy folyamatosan, mert még nincs letisztázva, hogy milyen telefonszámok legyenek megjelenítve.*
- *Értem, akkor erre térjünk vissza akkor, ha már nem változtatnak a felületen. Amilyen gyorsan kitalálták, hogy ezt a funkciót most akarják implementálni, akkor lehet még más változás is lesz a felhasználói felületen, ami indukálhatja, hogy még több időt kellene ráfordítani a teszt-karbantartásra.*
- *Értem, igaz.*
- *Igazából a mostani teszt, és ami utána következik, az üzleti elemzők szempontjából kritikusabb, ahogy nézem. Még ha meg is lenne a teljes funkció az áruházban, akkor is azt kell hogy mondjam, prioritást élveznek ezek a tesztek. Hiszen még a következő hét is tesztelési időszak, majd akkor többet tudunk mondani. Tudod, a sikeres tesztkészítéshez szükséges a megfelelő infó és adat.*
- *Igaz, köszi, értem már, akkor megvárom azt is, hogy milyen telefonszámok lesznek kiírva.*
- *Mivel ezek folyamatosan változhatnak, ilyen rövid idő alatt nem hiszem, hogy a legjobb ellenőrzést is bele tudnánk tenni a tesztbe. Szerintem a teszt majd kinyeri az adatot a felületről, és majd Ti leellenőrzitek manuálisan, vagy vizuálisan, hogy jók-e az adatok.*
- *Ez szuper, erre nem is gondoltam. Köszi.*
- *Igazán nincs mit. Jövő héten keressük egymást, rendben?*
- *Igen. Szia.*
- *Szia.*

- Automatizálási csapat fejlesztése

Szolgáltatás alatt oda kell figyelnünk az automatizálási csapatra is. Rohamos léptékben fejlődnek és jelennek meg technológiák, így a csoport szakmai fejlődését is figyelembe kell venni. Amennyiben sikeresek akarunk lenni, és a lehető legszélesebb körben akarjuk elérhetővé tenni az automatizálás előnyeit, úgy fejlődnünk és fejlesztenünk kell a csoportot is. A sikeres szolgáltatás része ez is.

2. Cégen belüli magas fokú automatizálási szolgáltatás

Az első pont lehet a kiindulási pont. Ahogy növekszik a cég alkalmazásainak, funkcionalitásainak száma, úgy követi ezeket minden bizonnyal számos automatikus teszt is. Ez indukálni fogja, hogy a tesztek változhatnak, és folyamatosan karban kell tartani őket. Nem biztos, hogy a csoportunk létszámnövekedése fogja eredményezni, hogy mindig a lehető leghamarabb tudunk teszteket javítani, optimalizálni. Érdemes elgondolkozni azon, hogy mit tehetünk, hogy az automatizált tesztelés, mint szolgáltatás még rugalmasabb legyen.

Azt tapasztaltuk, hogy a tesztek végrehajtása is az automatizálási csoportnál van. Gondoljunk abba bele, hogy ezt tegyük elérhetővé másoknak is. Építsük ki azt a folyamatot és módszert, amin belül mások elérhetik a meglévő teszteket, és képesek azokat futtatni is.

Manapság a felhő-megoldások nagy teret kezdenek nyerni, egy webes alkalmazás tesztelését nemcsak egy böngészőben és egy operációs rendszeren lehet végrehajtani. A tesztek és azok végrehajtásából hogyan tudjuk a lehető legnagyobb nyereséget kinyerni? Egyik lehetőség, hogy kiépítünk olyan platformokat, futtató állomásokat, amelyeken különböző konfigurációk léteznek.

Adjuk meg a lehetőséget a tesztelőknek, hogy ők tudják megadni, kiválasztani, hogy éppen milyen konfiguráción szeretnék a tesztet végrehajtani, ezáltal kiküszöböljük azt, hogy direktben szükség legyen az automatizálási csoportra.

Gondoljunk bele abba, hogy ez milyen szükségletekkel jár. Amennyiben csak arra gondolunk, hogy szeretnénk egy adott operációs rendszer különböző verzióján is teszteket futtatni, már kapunk minimum öt-hat lehetőséget. Amennyiben különböző böngésző-verziókat is szeretnénk használni, úgy ez a szám drasztikusan megugorhat, hiszen egy operációs rendszer adott verzióját számos böngészőn lehet használni. És ez még nem teljes. Fel kell mérni, hogy milyen konfigurációknak van értelme, és azokat megpróbálni kiépíteni.

Amennyiben egy nagyobb vállalatnál vagyunk alkalmazottak, jobb esetben létezik egy integrált vállalatirányítási rendszer. Az itt végzett automatizálás kevésbé jár olyan mértékű kihívással, mint akár egy webet érintő folyamat. A vállalatirányítási rendszer ablakai, gombjai, mezői nagy valószínűséggel verzióról verzióra nem változnak drasztikusan, ellentétben egy webes felhasználói felülettel.

Gondolkozzunk a felhasználók szemszögéből. Lehetséges, hogy valami még Windows XP-t használ, IE 9-el. Valahol már Windows 7, és IE 11 fut. Nekünk meg kell győződnünk arról, hogy az alkalmazásaink a lehető legjobb minőségben kerülnek a felhasználóink elé, ezért az automatikus teszteket célszerű különböző konfigurációkra is megnézni.

Magas fokú automatizálás alatt érthetjük a különböző automatizálási technikák vegyítését. Alkalmazhatóak adatvezérelt, kulcsszó vezérelt és hibrid technikák is. Amennyiben minimalizálni szeretnénk a tesztadat összeállítására, tesztek előkészítésére fordított időt, ezeket a folyamatokat is

automatizálhatjuk. Léteznek olyan megoldások, amikor a tesztadatokat dinamikusan állítjuk elő, valamilyen adatbázisból. Ekkor nem célzottan használunk tesztadatokat, hanem valamilyen metódus gyűjti össze nekünk azokat.

Az összegyűjtött tesztadatokra és a meglévő tesztállományra pedig egy automatikus vezérlő rész fog figyelemmel lenni, hogy mi mikor kerüljön végrehajtásra.

Ez a fajta módszer nagyon nagy tapasztalatot és komplex rálátást igényel mind a tesztelés alatt lévő alkalmazásokra, mind a tesztekre is.

3. Egyedi vállalkozás szintjén értelmezett automatizálási szolgáltatás

Manapság ha valaki az interneten automatizálással kapcsolatos cikkeket keres, előbb-utóbb belebotlik olyan vállalkozásokba, cégekbe, akik automatizálási szolgáltatást nyújtanak másoknak.

Tételezzük fel, hogy egy középvállalatnak létezik manuális tesztelői csoportja, de nincs kiépítve az automatizálási rész. Nem lehetséges ennek kiépítése (költségek, szakemberek, határidők), viszont szükséges lenne valamilyen módon automatikus tesztvégrehajtásra is. Ekkor jönnek képbe az ilyen és ehhez hasonló szolgáltatást nyújtó vállalkozások.

A nehézséget itt az adja, hogy az esetek többségében időre van szükség, míg egy ilyen szolgáltatást nyújtó vállalat megismeri az adott céget, és a cégen belüli folyamatokat. Nagyon komoly és mélyreható tárgyalásoknak kell megtörténnie mind a két oldal részéről. Akkor, ha mi kérünk szolgáltatást, érdemes odafigyelni, milyen szerződés mentén és milyen feltételekkel fogja és tudja az adott szolgáltató a kívánt minőséget hozni. Vizsgáljuk meg, hogy pontosan mit is tartalmaz a szolgáltatás, amiért fizetünk.

Sok szolgáltató hirdeti magát nagyon kecsegtető módon. Számos tesztelői tapasztalat, tapasztalt tesztelők, tesztvégrehajtási eszközök, tesztmenedzsment eszközök, teljesítménytesztelésre használatos eszközök és még sorolhatni lehet, amelyeket fel szoktak tűntetni. Mielőtt belevágunk egy ilyen szolgáltatás igénybevételéhez, az alábbi kérdéseket fontoljuk meg és tegyük is fel a szolgáltatónak:

- Mi a célja az adott szolgáltatónak?
- Milyen múltra tekint vissza a cég?
- Mik a terveik az elkövetkező időszakra?
- Hogyan tervezik a versenyben maradást a piacon?
- Mit tesznek azért, hogy bővüljön az igénybevevők köre?
- Mennyi fővel rendelkezik a vállalkozás?
- Milyen szakterületeken van jártasságuk?
- Milyen tapasztalataik vannak az általuk hirdetett tesztelői eszközökben?
- Tudnak-e mutatni tanulmányokat, mikor milyen cégnél milyen megbízásuk volt?
- Milyen határidőkkel dolgoznak?

- A szolgáltatás igénybevételével szakemberek fognak érkezni a cégünkhöz, vagy sem?
- Direktben együtt dolgoznak majd a saját tesztelőinkkel?
- Milyen dokumentumokat kérnek a szolgáltatás végrehajtásához? (követelményspecifikációk, tesztesetek stb.)
- Milyen határidővel dolgoznak?
- Melyek azok a dokumentációk, amelyeket ők szolgáltatnak?
- Melyek azok az alapszolgáltatások, amit nyújtanak?
- Létezik-e többszintű szolgáltatásnyújtás?
- Milyen tevékenységeket foglal magában a szolgáltatás? (teszttervezés, tesztvégrehajtás, eredmény kiértékelése)

Számos kérdést fel lehet tenni ebben az esetben, hiszen nekünk, mint megrendelőnek nagyon nem mindegy, hogy milyen minőségű szolgáltatásban lesz részünk.

Az biztos, hogy a jelenlegi tendenciákat nézve, nagy igény van automatizált tesztelésre. Csak akkor lehet sikeres egy ilyen szolgáltatást nyújtó vállalkozás, ha folyamatosan követni tudja az adott piaci igényeket, és ezzel együtt az új technológiákat és megoldásokat. Olyan szakemberekkel kell rendelkezni, akiknek széles rálátásuk és nagy szakmai tapasztalatuk van ezekre.

TÁBLÁZATOK JEGYZÉKE

ÁBRÁK JEGYZÉKE

FELHASZNÁLT SZAKIRODALOM

Hivatkozás teljes könyvre:
[1] Szoftvertesztelés a gyakorlatban
Boda Béla, Bodrogközi László, Gál Zoltán, Illés Péter
2014
39. oldal

Hivatkozás könyvrészletre:
[2] The Automated Testing Handbook
Linda G. Hayes
[2a] The Automated Testing Handbook
Linda G. Hayes
81. oldal
[2b] The Automated Testing Handbook
Linda G. Hayes
72. oldal
[2c] The Automated Testing Handbook
Linda G. Hayes
74. oldal
[3] The Automated Testing Handbook
Linda G. Hayes
89. oldal

Hivatkozás internet forrásra:
[4] How to choose the best software test automation tool for your team
http://searchsoftwarequality.techtarget.com/answer/How-to-choose-the-best-software-test-automation-tool-for-your-team
2009. március
[5] Managing Successful Test Automation – Part 1
http://www.testthisblog.com/2013/04/managing-successful-test-automation.html
2015. február
[6] http://www.testingreferences.com/testinghistory.php
2015. február
[7] http://h30499.www3.hp.com/t5/The-Future-of-Testing-Blog/Centralizing-the-test-automation-team/ba-p/2409666#.VRU2cuFUgeG
Centralizing the test automation team
2015. 03. 27
[8] http://www.extremeprogramming.org/
Extreme Programming: A gentle introduction
2015. 03. 27
[9]http://moodle.autolab.uni-pannon.hu/Mecha_tananyag/szoftverfejlesztesi_folyamatok_magyar/ch04.html#d0e1145
Agilis szoftverfejlesztés
2015. 03. 27

[10] http://www.qaiconferences.org/tempQAAC/Automation%20Process%20QAAC.ppt
Test Automation Success: Choosing the Right People & Process
Kiran Pyneni, Automation Manager, Aetna Inc.
2015. 03

[11] https://sqta.wordpress.com/test-automation-strategy/
Test Automation Strategy
2015. 04. 10

[12] http://www.softwaretestinghelp.com/automation-test-palnning/
How Does Test Planning Differ for Manual and Automation Projekts?
2015. 04. 10

[13] http://asztivaniskola.lapunk.hu/tarhely/asztivaniskola/dokumentumok/szolgaltatasi_szektor.pdf
A szolgáltatások
2015. 04. 18

[14] http://www.methodsandtools.com/archive/archive.php?id=94
Implementing Automated Software Testing - Continuously Track Progress and Adjust Accordingly
2015- 04. 19

[15] http://en.wikipedia.org/wiki/Software_verification_and_validation
Software verification and validation
2015. 04. 23