Tesztmenedzselés a gyakorlatban

Mucsina Norbert János

Készült: 2015

Terjedelem: 5,7 ív

Kézirat lezárva: 2015. április 30.

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.*

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

TARTALOMJEGYZÉK

1 Bevezetés 5

1.1 Hogyan használd ezt a tananyagot? 6

1.2 Kiknek, kinek ajánlom ezt a tananyagot? 7

2 Mi is a tesztmenedzsment? 8

3 A tesztmenedzsment elemei 12

3.1 Teszt stratégia 12

3.1.1 Analitikus teszt stratégia: 14

3.1.1.1 A követelmény-központú (requirement based) stratégia 14

3.1.1.2 A kockázat-központú (risk-based) stratégia 15

3.1.1.2.1 Kockázatok azonosítása 16

3.1.1.2.2 A kockázatok kiértékelése 16

3.1.1.2.3 A kockázatok csökkentése – speciális tesztesetek elkészítése 18

3.1.1.2.4 A fennmaradó kockázatok kezelése 19

3.1.2 Modell-alapú (Model Based) teszt stratégia 20

3.1.3 Szisztematikus (Methodical) teszt stratégia 22

3.1.4 Folyamat- vagy szabvány-kompatibilis (Process-Oriented Test Strategy) teszt stratégia 24

3.1.5 Dinamikus teszt stratégia 24

3.1.6 Konzultációs vagy irányított teszt stratégia 25

3.1.7 Filozofikus teszt stratégia (Philosophical test strategy) 26

3.1.8 Visszafejlődés-elkerülő (Regression-averse) 27

3.1.9 A stratégiai dokumentum 29

3.2 Teszttervezés és becslés 36

3.2.1 Teszttervezés 40

3.2.2 A becslés 45

3.2.3 WAG és SWAG – a két mágikus módszer 49

3.2.4 Súlyozott átlag (Weighted Average) 50

3.2.5 Delphi – a nem-programnyelv 51
3.2.6 Összehasonlító becslés (Comparative vagy Historical Estimating) 52
3.3 Tesztmonitorozás, metrikák 53
3.3.1 A mérés tárgyának meghatározása 55
3.3.2 Mérőszámok meghatározása 58
3.3.3 A mérés módszerének meghatározása 62
3.3.4 Határértékek meghatározása 64
3.3.5 Mit tegyünk, ha …? 64
3.4 Konfigurációmenedzsment 65
3.4.1 A konfiguráció beazonosítása/meghatározása (Configuration identification) 68
3.4.2 A konfiguráció felügyelete (Configuration control) 70
3.4.3 A konfiguráció állapotának könyvelése/követése (Configuration status accounting) 71
3.4.4 A konfiguráció auditálása/ellenőrzése (Configuration audits) 72
3.4.4.1 Hogyan ellenőrizzük a konfigurációt? 73
3.4.4.2 Mikor ellenőrizzük a konfigurációt? 74
3.4.4.3 Kinek kell az ellenőrzést végrehajtani? 74
3.5 Kockázatok 74
3.5.1 Kockázat-elemzés 77
3.5.1.1 Kockázatok azonosítása, értékelése 77
3.5.2 Kockázati terv (mitigation plan, contingency plan) 81
3.5.2.1 Kockázati eseményt megelőző terv (contingency plan) 82
3.5.2.2 Kockázati hatást csökkentő terv (mitigation plan) 82
3.6 Incidensmenedzsment 83
4 Tesztelő csapat 88
4.1 A Csapat 88
4.2 Tesztelő (Teszt végrehajtó) – a kubikos 89
4.3 Teszttervező – a mérnök 91

4.4 Tesztmenedzser – a főnök 96

5 Utószó 99

6 Felhasznált szakirodalom 101

1 Bevezetés

“ 1998 végén, 1999 elején el is indult az újabb amerikai szondapáros, melyhez egy harmadik is társult. A Mars Climate Orbiter keringő egységet és a Mars Polar Lander leszállóegységet a Deep Space-2 penetrátor-kettős kísérte el. (A két penetrátor neve Scott és Amudsen lett.) A Climate Orbiter főképp a marsi időjárást és légkört tanulmányozta volna, a Polar Lander pedig a déli pólus körzetét, akárcsak a Scott és az Amudsen.

A még 1999-ben megérkező két űrszonda közül a Mars Climate Orbiter a földi szakemberek hibájából (a metrikus és az angolszász mértékegységek összekeverése) a légkörben pályára állás közben elégett, míg a Polar Lander és a Deep Space-2 rendben megkezdte a leszállást. Ezután azonban róluk sem lehetett hallani.”[1]

Valójában egy szoftverhiba okozta a több millió dolláros katasztrófát. A hiba a szoftverben igen apró volt: az egyik eleme a szoftvernek fontban számolta ki a tolóerő nagyságának mértékét, míg egy másik modul ugyanezt az értéket már newton egységben kezelte. Valamiért ezt nem fedezték fel a szoftver tesztelése során…”

(NASA Mars Mission, 1998-99)

A fenti történet valós – a hír igen tömör, lényegre törő. Semmi mást nem mond el, csak hogy a földi szakemberek hibájából, a kétféle mértékegység összekeveréséből történt a hiba. De vajon miért történhetett ez meg? És pont a NASA-nál? Hiszen ott dolgoznak a világ legjobb koponyái, legnagyobb mérnökei, tudósai… Mégis, mi történhetett? A valós okokat – mint megannyi mást a NASA-nál – sűrű homály fedi. Találgatni persze lehet…

Ennek a tananyagnak azonban nem ez a feladata. Inkább arra törekedtem, hogy egy olyan alapot nyújtsak a tesztmenedzsmentről, ami segíthet elkerülni egy ilyen aprónak tűnő, mégis sok millió dolláros hibát. Megpróbálok rávilágítani a „Nagykönyv”[2] által leírt tesztmenedzsment olyan részeire, amit valóban lehet – és kell – alkalmazni a gyakorlatban. És ez számomra a kulcsszó: ***A GYAKORLATBAN***! Mindig, minden modell, stratégia, definíció és folyamat, amit a szakirodalom megnevez egy steril,

1 Űreszközök a Marsnál 1960-2000, Távoli világok kutatói in http://www.urvilag.hu/ureszkozok_a_marsnal_1960_2000/20040101_a_mars_meghoditasa_6resz_ket_siker_negy_kudarc

2 Patrick Hendrickx, Chris Van Bael, Alain Bultink: Advanced Test Management – ps_testware nv, 2010 - ISBN: 978-90-9025727-3

laboratóriumi körülmények között letisztázott, több száz, esetleg ezer kísérlet/projekt átlagából származik. Éppen ezért ezek így, tisztán a való életben nem fordulnak – nem fordulhatnak – elő. Mindig van valami, ami módosítja, ami hatással van az adott modellre/stratégiára, vagy más modelleket/stratégiákat kell bevonnunk ahhoz, hogy a megfelelő hatást elérhessük.

Gyakorlat – ez a szó az, ami a vezérfonala ennek a tananyagnak. Gyakorlatiasságomból fakadóan rögtön lenne is egy kérésem: tegeződjünk. A tisztelet ugyanúgy megadható, viszont egyszerűbb megfogalmazni bizonyos dolgokat. Remélem hölgyeim és uraim ez nem lesz zavaró számotokra[3].

1.1 Hogyan használd ezt a tananyagot?

Olvasd, értelmezd, dolgozd fel, majd alkalmazd! A fontosabb, hangsúlyosabb részeknél először egy történetet olvashatsz – ez a történet valamilyen módon kapcsolódik az adott fejezet témájához. Nem lesz mindegyik informatikai jellegű, főleg nem teszteléssel kapcsolatos. Ezzel is az életszerűséget próbáltam bemutatni. Lehet, hogy elsőre nem is lesz világos a kapcsolat a történet és az adott szakasz témája között, ezért érdemes újra elolvasni, amikor a fejezetet már feldolgoztad. Nem minden történet valós – ez rögtön ki fog derülni, ahol az írói vénám és a képzeletem erősebb volt – de mindegyik releváns és akár még meg is történhet.

A tesztelés alapnyelve – akár tetszik nekünk, akár nem – az angol. Természetesen megpróbáltam inkább a fontos szavak, kifejezések magyar fordítását használni, de sokszor nagyon körülményes az adott szó magyar fordítása, vagy furcsa a hangzása. Segítségképpen a HTB[4] által kiadott „Szoftvertesztelés egységesített kifejezéseinek gyűjteményét”[5] vettem alapul. Mindennek ellenére, amikor egy ilyen nehézkesnek tűnő fordítást használtam zárójelek között az angol, eredeti megfelelőjét is megjelöltem – de csak az első alkalommal.

Nem minden elemet fejtettem ki olyan mértékben, mint az angol nyelvű szakirodalmakban – ennek oka az, hogy igyekeztem a gyakorlatba átültethető, hasznos dolgokat megmutatni, azokat kiemelni. A tananyag terjedelme nem tette lehetővé, hogy olyan témákat is érintsek, amik a gyakorlatban közvetlenül nagyon ritkán, esetleg sosem jelennek meg, vagy létüket csak közvetve igazolják. Ezeknek a feltárását inkább odabízom – a szakirodalmi jegyzékben minden általam felhasznált irodalom megtalálható, valamit az irodalmak szintén megneveznek jó néhány másik szakirodalmat. Ha pedig abból is kifogytál, akkor meg csak egyszerűen „guglizz rá”.

[3] Ha esetleg mégis zavarna valakit, kérem keressen meg – megbeszéljük.
[4] Magyar Szoftvertesztelői Tanács Egyesület (Hungarian Testing Board), 1117 Budapest, Neumann János u. 1/E
[5] HTB, Szoftvertesztelés egységesített kifejezéseinek gyűjteménye, 2013, ver. 3.13

1.2 Kiknek, kinek ajánlom ezt a tananyagot?

Nehezen tudom jól megfogalmazni azok szűk-tág körét, akik szerintem hasznosítani tudják ezt a tananyagot. Leginkább nem tudom megfogalmazni. Reményeim szerint ugyanis mindenki, aki valamilyen szinten a teszteléssel foglalkozik, talál majd benne hasznos információt, tippet vagy éppen trükköt. Vagy csak egy jópofa történetet, ami tanulságos lesz számára. Azt, hogy valójában kik fogják tudni a leginkább használni – én bevallom: nem tudom. Majd az idő eldönti…

Ha Te úgy döntesz, hogy elolvasod, akkor előre is köszönöm a figyelmedet és az idődet. Ha még hasznos is lesz számodra, akkor még jobban örülni fogok, mert volt haszna annak, hogy megírtam ezt a tanyagot.

Köszönöm.

2 Mi is a tesztmenedzsment?

„Még ifjú diákkoromban történt meg velem az, amit úgy is nevezhetnék: az első találkozás a teszteléssel és a tesztmenedzsmenttel. Az eset pedig így esett:

Az Elektronikai Alapismeretek tantárgyunk vizsgájára készültünk fel éppen – azaz csak próbáltunk. Négyen négyféleképpen értelmeztük a feladatokat, hatféleképpen kezdtük el megoldani a tételekben foglaltakat. Nem tudtuk a váltószámokat a különböző mértékegységek között, stb. Szóval a baj nem volt kicsi.

Erre egyikünk – nevezzük Gézának – egy roppant aranyos ötlettel hozakodott elő (előre is elnézést kérek a tanár kollégáktól):

- Mi lenne, ha írnánk egy apró Java-alkalmazást a telefonokra, ami a vizsgán használhatunk? Nagyjából ugyanolyan telefonjaink vannak, ugyanazt a platformot használja. Egyszerű lenne, nem? – a javaslatot kitörő örömmel fogadtuk. Megszavaztuk, hogy legyen. Négyen két csoportra oszlottunk: egyik csoport programozott, a másik pedig összeszedte, hogy milyen képletekre, váltószámokra és egyéb finomságokra lesz szükségünk a vizsgához. Minden ment, mint a karikacsapás! A program alakult, finomodott, szépült, a lista a képletekről és egyéb szükséges elemekről bővült – a kávé meg a cigi fogyott intenzíven. Azt előre láttuk, hogy nem lesz kész egy nap alatt a program, de úgy gondoltuk: inkább 3-4 nap programozás, mint 2 hét tanulás (akkor még nem tudtuk, mekkorát tévedünk).

A 3. napon mindenki megkapta a késznek mondott programot. Kiéhezve vetettük rá magunkat: letöltés a szerverünkről, telepítés, elindítás, próbálgatás. Úgy tűnt, működik. És ennyiben maradtunk. – Majd a vizsgán használjuk! Boldogan hátradőltünk, megbeszéltük, hogy ez meg kell ünnepelni. Kijelöltük az időpontot – aznap este természetesen, hiszen minek kapkodni – meg a megfelelő társasági helyszínt (mind a négyünkhöz legközelebb található Pénzcseppfolyósító Intézetet), majd megérkezve a helyszínre az adott időpontban, egy itallal a kezünkbe boldogan tervezgettük a sikeres vizsga utáni felhőtlen, iskola nélküli pár hónapot. Szó szót követett, felmerült a program fejlesztése is természetesen, majd a nehézségek. Hogyan oldottuk meg a törtek megjelenítését, stb. Majd előkaptuk telefonjainkat és nézegetni kezdtük tudásunk és munkánk gyümölcsét… Arcunk vidám mosolya előbb csak halványodni kezdett, majd sarkai a nyakunk irányába kezdett lassan, de stabilan mozogni… Nem telt el 10 perc és szomorúan vettük tudomásul: tanulnunk kell. A program ugyanis csak egy eszközön működött, a Gézáén, de ott sem tökéletesen.

– Pedig én megnéztem. Én ellenőriztem. – mondogattuk. De mindenki csak odáig jutott, hogy az első feladathoz tartozó képletet nézte meg… Rádöbbentünk, hogy mi volt a hiba: senki nem fogta össze az egészet. Nem volt tervünk arra, hogy ki mit ellenőrizzen, ki mit fejlesszen, mit teszteljen. Nem volt az egésznek egy vezérfonala. Mivel nem sok időnk maradt, inkább tanultunk – és szerintem így jártunk igazán jól!”

(Norbert J. Mucsina 2015.)

A száraz definíció szerint: a tesztmenedzsment (Test Management) a tesztelési és az ahhoz kapcsolódó folyamatok, dokumentációk és tevékenységek folyamatos tervezése, irányítása, felügyelete, metrikák készítése, majd az eredmények összegzése és továbbítása a projekt vezetése felé.

Az előbbi mondat teljes mértékben igaz. Megpróbálom „konyhanyelven" leírni, hogy mit is takar a tesztmenedzsment fogalma. Tulajdonképpen a tesztelési folyamat egy szoftverfejlesztés életében egy projekt a projektben. Ugyanúgy, ahogyan az adott szoftvert – illetve a köré épített projektet – megtervezik és irányítják, ugyanúgy kell magát a tesztelési folyamatot is megtervezni, irányítani, nyomon követni. Vessünk hát egy pillantást az általánosan elfogadott projektvezetési életciklusra (Project Management Life Cycle = PMLC), illetve annak különböző szakaszaira[6]:

A projektmenedzsment életciklusa öt jól elhatárolható fázisból áll:

1. Felterjesztés/kezdeményezés (Initiation)
2. Tervezés (Planing & Design)
3. Végrehajtás (Execution)
4. Átadás (Deploy)
5. Lezárás (Closure)

A második fázist gyakran még kétfelé bontják, mert megkülönböztetik a projekt tervezését (Planning) valamint a szoftver szerkezeti, technikai, adatbázis, felhasználói felület, stb. tervezését (Design). Az előző fázisban készülnek el az olyan – a projekt vezetése szempontjából – nélkülözhetetlen dokumentumok, mint a projektterv, a kommunikációs terv, kockázati analízis, stb. Az utóbbi fázisban viszont a készítendő szoftver, vagy megoldás funkcionális tervek és a technikai tervek készülnek el.

Nos, most próbáljuk meg felsorolni, hogy mi minden kellhet a tesztelési folyamathoz, annak menedzseléséhez. Csak úgy józan paraszti ésszel…

1. Érdemes megtervezni. Mit, mikor, hogyan, kivel, mennyi ideig? (Nem félreérteni…)

2. A tervet előbb-utóbb, némi kiegészítéssel, de végre is kell(ene) hajtani. (Természetesen csak akkor, amikor a projekt azt megengedi, azaz csúszás esetén nem a tesztelés idejét próbálják meg csökkenteni. Ebben az esetben ugyanis módosítani kell a tervet, majd a módosított határidőket tartalmazó tervet kell végrehajtani.)

3. A tesztelés folyamatát, teszteseteket, eredményeket lehetőleg dokumentálni kell(ene), legfőképpen a talált hibákat. Egyfajta jelentést készíteni a teszt eredményességéről és átadni a megbízónak. (Általában a projekt vezetője, vagy a vezetői.)

[6] Systemation: Fast Start in Project Management, 2014

4. A végén még a hibajavítások tesztelése, nyomon követés következik, mindenféle utólagos hibajavítás, de végeredményben le is kell zárni a tesztelést valamikor.

Ha csak egy pillantást is vettünk a két listára, talán feltűnik némi hasonlóság. Tulajdonképpen az öt lépésből, amit a projektvezetésben felsoroltunk csak a legelső hiányzik a tesztmenedzsment életéből.

Visszatérve a kérdéshez: Mi is a tesztmenedzsment?

A „projekt a projektben" meghatározást alapul véve – és természetesen az előzőekben kielemzetteket figyelembe véve – valóban igen bonyolult egy dolog. „Miért?" – merülhet fel a kérdés – „Hiszen a projekteket is elvezetgetik az emberek, sokan mindenféle előképzettség nélkül. Vagy gyakorlat, nota bene: tudás nélkül. És mégis sikeresek azok is!"

Nos, ez mind igaz – azonban a projektek sikerességét én nem igazán merném firtatni. Ugyanis a projektek „sikeressége" csupán nézőpont kérdése. De ezen nézőpontok kielemzése nem ennek a tananyagnak a témája. Mindenesetre – tapasztalataim alapján – a tudás és/vagy tapasztalat nélküli tesztmenedzsment okozhatja a legtöbb, pénzben is mérhető problémát. Nem kizárólagosan, mert akadhatnak őstehetségek is, de igen sokba kerülhet egy olyan tesztelői csapat, akiknek nincsen jól megtervezett tesztelésük, azaz nem rendelkeznek egy gyakorlott (nem azt jelenti, hogy képzett) tesztmenedzserrel. Természetesen amennyiben ezt a projekt bevállalja, azaz szánnak rá extra költségeket, akkor még lehet egy projekt sikeres mindenféle tesztmenedzsment nélkül is. Azonban lehet, hogy az extra költségek tükrében a siker fogalma egy kicsit más jelentést nyer…

A másik nehézsége a jó tesztmenedzsmentnek az, hogy a tesztmenedzsernek nem csak a saját projektjére kell odafigyelnie – lásd: „projekt a projektben" elv – hanem maximálisan idomulnia, alkalmazkodnia kell a „szülő-projekthez" is, azaz ahhoz a folyamathoz, aminek részegységét képezi. (Sokak szerint ez az egyik legnehezebb feladat…) Nem lehet a megbízó projekttől teljesen elrugaszkodottan megtervezni, majd lefolytatni a tesztelést – azt hiszem, hogy ennek nyilvánvaló okait nem kell részleteznem. Viszont azt is el kell mondanom, hogy ennek az alkalmazkodásnak kétirányúnak kell lennie, azaz a szülő projektnek is tudnia kell némely pontokon alkalmazkodni a teszteléshez. Ilyen pont lehet egy projekt életében például a teszt-ablakok mennyiségének, időpontjának illetve időtartamának a meghatározása. Ezeket inkább a tesztmenedzser feladata meghatározni, mint a projekt menedzseré, ezért a projekt menedzsernek érdemes ezt elfogadnia (amennyiben a minőség fontosabb, mint a projekt zárásának dátuma). Akkor lehet igazán jó

tesztmenedzsmentről – és ezzel egyidejűleg projekt menedzsmentről – beszélni, ha minden más egyéb, fontos jellemző mellett ez a flexibilitás is megjelenik mindkét oldalon.

A harmadik nehézsége az, hogy maga a tesztelés csak egy része annak, amit minőségnek nevezünk, s ami – szerintem – a legfontosabb minden projekt életében. S ezt a minőséget a legtöbbször – igaz, tévesen – a tesztelőktől várják el. Mi, tesztelők tudjuk, hogy a minőség ennél jóval tágabb fogalom, valamint hogy egyedül a teszteléssel még nem lehet minőséget elérni. A minőségbe beletartozik a megfelelő tervezés (projekt), a kiváló, vagy legalábbis meglévő dokumentáció, az üzleti igények megfogalmazása, az ütemterv, projekt főbb dátumainak lefektetése és – amennyiben szükségszerű – azok módosítása, a megfelelő kommunikáció, stb. Nem részletezném tovább – viszont ami nagyon fontos: mindezeket egy jó tesztmenedzser ellenőrzi és a megfelelő módon akár meg is követeli a projekt vezetésétől. Ez a rész összefügg az előbb említett alkalmazkodási kényszerrel, legfőképpen az időzítés tekintetében, valamint feltételez egyfajta szakmai bizalmat a tesztmenedzser irányába.

Ha ezt a mondatot olvasod, akkor feltételezem, hogy nem sikerült elriasztani a szakma mélyebb megismerésétől – ellenére mindazon nehézségeknek, amiket az előzőekben leírtam. Gratulálok, mert az első akadályt már sikerült leküzdened. Most arra kérlek, vegyél egy nagy levegőt, menj ki az utcára, rétre, strandra, vagy a tengerpartra – bárhová, csak tedd le ezt a tananyagot egy kis időre. Gondold végig, mit olvastál, mit várasz ettől a tananyagtól. Mit vártál eddig és mennyire érdekel még. Miután mindezt végiggondoltad és úgy határoztál, hogy mégiscsak végigolvasnád ezt a kis szösszenetet, akkor kérlek, lapozz…

3 A tesztmenedzsment elemei

Amint látom, mégiscsak szeretnél valami közelebbit megtudni erről a 'tesztmenedzsment'-ről. Figyelmeztetlek, az elkövetkezendő oldalak szárazanyag-tartalma drasztikusan magas. Fogadd meg tanácsom és mindig legyen Veled egy pár liter folyadék, amikor ezeket az oldalakat olvasod – az talán enyhíti a szöveg szárazságát…

Miközben írtam ezt a tananyagot – ami persze eléggé hosszadalmas folyamat volt – sokszor visszatértem ehhez a részhez. Nem igazán tudtam, hogyan kezdjem el… Tudtam, éreztem, hogy egy kicsit talán megfélemlítheti azt, aki olvassa majd, hogy ennyi mindent kell tudni – érdemes tudni – a tesztmenedzsmentről. Féltem, úgy járok, mint amit Mikszáth Kálmán leírt „A hályog-kovács" című művében – mindenkit elijesztek attól, hogy tesztmenedzsmenttel foglalkozzon… Gondoltam, bevezetésként egy frappáns kis történettel elveszem az élét, de valahogyan nem találtam a megfelelőt. Nem tudtam, hogyan vezessem be azt a sok fogalmat, leírást és technikát, amit ez a fejezet tartalmazni fog. Mindazon elemeket, amik a tesztmenedzsmentet alkotják: stratégia, tervezés, becslés, monitorozási eljárások, technikák, kockázat-elemzés…

Így maradtam a régi, már jól bevált „in medias res" módszernél… Igaz, történetem azért mégis akadt…

3.1 Teszt stratégia

„Egy alkalommal – még kezdő tesztelőként – megkeresett egy külföldi kollégám egy projekt kapcsán. A projekttel kapcsolatos első telefonos beszélgetésünk – persze nem magyarul és nem ilyen folyamatosan – körül-belül így hangzott (a szokásos udvariassági formulákat kihagytam a leírásból):

- *Szóval lesz majd ez a projekt – mesélte, mintegy csevegő hangnemben David – valamikor pár hónap múlva. Úgy értem: akkor lehet tesztelni.*
- *Pontosan mikor is? – kérdeztem vissza, kicsit bizonytalanul.*
- *Na, egészen pontosan nem tudom, de a májusi ciklusban.*

Nos, nekem sem kellett több: tipikus példája lettem Arany János: „A wales-i bárdok" olyannyit emlegetett szakaszának: 'Szó bennszakad, hang fennakad, / Lehellet megszegik. –' Május ugyanis a következő hónap volt, lévén április havának elején jártunk. Villámgyorsasággal cikáztak át agyamon a lehetőségek: elvállaljam? Ne vállaljam? Mit tegyek? A projekt ígéretes, nagy lehetőség volt számomra, viszont olyan feladatokat követelt meg, amit akkor még nem is tudtam, hogy hogyan nevezik. Még nem voltam gyakorlott tesztmérnök – jobbára végrehajtóként tevékenykedtem…

Valószínűleg furcsállta hosszas hallgatásomat David, mert hirtelen elkezdte ecsetelni, hogy mennyire jó ez a projekt, meg milyen fiatalos a csapat és mennyire kellemes a hangulat… Szinte alig hallottam. Inkább elköszöntem tőle udvariasan azzal az ígérettel, hogy másnap majd még megkeresem, de most mennem kell. A telefon még csak nem is került nyugalmi helyzetébe én már egy tapasztalt kollégát nyaggattam arról, hogy mit és hogyan is kezdjek el csinálni. Pár javaslattal és ötlettel felvértezve leültem és levelet írtam David-nek: ezekre lenne szükségem ahhoz, hogy megfelelően részt tudjak venni a projektben:

- *Projektterv*
- *Követelmények*
- *Kockázatok*
- *Minőségi elvárások*
- *Kapcsolódási pontok*

Miután ezeket, ha nem is azonnal, de még viszonylag időben megkaptam elkezdtem a munkát: a követelmények átvizsgálásával kezdtem, majd a kockázatokkal, s végül maradtak a kapcsolódási pontok. Mindhárom esetben egy-egy hosszú listát készítettem észrevételeimről és kérdéseimről, amiket elküldtem David-nek. Mindeközben a listával párhuzamosan különböző teszteseteket készítettem, amiket egy kezdetleges táblázatba foglaltam, s szintén elküldtem David-nek. Hozzácsaptam még azt is, hogy szerintem mikor érjük el a megfelelő minőséget, azaz mik az én minőségi elvárásaim, valamint hogy mi alapján készítettem el a teszteseteket. Sokáig nem kaptam választ, így vártam, finomítgattam a terveket. Majd egyik nap felhívott David telefonon és egy elég hosszas beszélgetés zajlott le közöttünk, aminek keretében mindent átbeszéltünk, mindent letisztáztunk. Minden információt, ami nekem kellett egy dokumentumba összesítettük, megfogalmaztuk a minőségi követelményeket, valamint lefektettük a tervezés folyamatát. Mindezt a projekt csapat elé tette, s elfogadtatta velük.

Végül minden elkészült, s csak egy hónapot csúsztunk a tervezetthez képest…"

(Norbert J. Mucsina, 2015)

Definíció szerint a teszt stratégia egy magas szintű dokumentum, ami részletesen leírja a fejlesztő szervezetre, programra vagy projektre vonatkozó teszt szinteket, valamint a megnevezett tesztelési szintek részleteit. A teszt stratégiának – mint minden másnak is – különböző típusai, megközelítési módjai léteznek. A későbbiekben részletesen ki fogom elemezni a tudományosan elhatárolt stratégia-típusokat, megközelítéseket. Elöljáróban azonban szeretném leszögezni: egyik stratégia sem élhet,

létezhet tisztán a gyakorlatban, mindenképpen valamelyik másik stratégiával együtt tud megfelelően működni, az elvárt minőséget és tesztelési lefedettséget biztosítani.

3.1.1 Analitikus teszt stratégia:

Az analitikus stratégia – vagy inkább megközelítés – valójában két, hasonló gondolkodásmódnak a gyűjtő fogalma. Mind a kettő valamilyen projekt dokumentációtól függ, arra alapszik és arra építi a stratégiát.

3.1.1.1 A követelmény-központú (requirement based) stratégia

A *követelmény-központú* stratégia alapja egy komoly, nagy körültekintéssel és precizitással létrehozott, minden tekintetben részletes követelmény-rendszer. Ez az a dokumentum, ami leírja, hogy a produktumnak/szoftvernek pontosan milyen feltételeknek, elvárásoknak kell megfelelnie. Másképpen: mit kell, hogy tudjon a kész szoftver. A követelmény-rendszert általában a projekt vezetése, vagy az üzleti elemző(k) szokták létrehozni, de minden esetben segítséget nyújtanak a tapasztalt fejlesztők, vagy rendszermérnökök is. Ezen személyek bevonása a követelmény-rendszer létrehozásába nem feltétel, de igen erősen ajánlott.

Követelmény-rendszerről beszélek – de mit is takar a „rendszer” szó? A kérdés jogos, íme a válasz:

- **Üzleti követelmény (business requirement)**, ami leírja, hogy a megrendelő mit vár el a szoftvertől;
- **Technikai követelmény (technical requirement),** ami leírja, hogy milyen technológiával kell/lehet megvalósítani az üzleti követelményekben foglaltakat;
- **Üzleti terv (business design)** dokumentum, ami azzal foglalkozik, hogy hogyan szeretné, hogy megjelenjen, kinézzen a szoftver. Például a felhasználói felület piros alapon sárga gombokkal és kék feliratokkal;
- **Technikai terv (technical design),** ami a megvalósítás technikai hátterét tervezi meg, tehát a kért technológiák segítségével hogyan fogja tudni megvalósítani;

A követelmény-központú stratégia két fő elemből áll:

1. **A követelmény-rendszer vizsgálata**, annak „tesztelése”. Mindenképpen át kell nézni „kívülállóként” is, hiszen akik dolgoznak vele, összeállítják – a dokumentum méretéből és komplexitásából fakadóan – véthetnek hibát, a leggyakrabban logikai jellegű hibát. Ezek kiszűrése is egy tesztelési feladat, amit nem igazán szépen követelmények felülvizsgálatának (Requirements Review) neveznek magyarul, ami magában foglalja a logikai hibák feltárását, annak megállapítását, hogy a dokumentum teljesen lefedi-e az adott projektet, vagy esetleg

van benne még kérdéses terület, nem teljesen körülírt feladat, követelmény. Miért olyan fontos a követelmények ellenőrzése? A pontos válasz: a projekt korai szakaszában feltárt hibák – mint például a követelményekben vétett hibák – javítása a legolcsóbb. (A tipikusan elkövetett hibákat a 4.3 fejezetben fejtem ki bővebben.) Viszont ha azok a hibák nem kerülnek javításra, akkor a legnagyobb kárt is okozhatják. Gondoljunk csak egy kicsit a NASA csúfos Mars-expedíciójára, amit a bevezetőben említettem…

2. **A tesztesetek megírása a követelmények alapján**. Tulajdonképpen ez a teszt stratégia dokumentációból – és csakis abból – dolgozik. Mondhatjuk, hogy a tesztelési ciklus – a tényleges tesztvégrehajtás – kezdetéig lehet, hogy nem is találkoznak a tesztelendő termékkel. A feltétele viszont a pontos és részletes dokumentáció. Egy jól megírt követelmény-rendszerből felépíthető a teszteset-gyűjtemény anélkül, hogy a terméket működés közben látnánk. Előnye, hogy már a dokumentációból – tehát a projekt korai szakaszában – feltárhatunk esetleges hibákat, valamint a teszt-lefedettség – mivel dokumentációból dolgozunk – jól mérhető, nyomon követhető. Hátránya, hogy amennyiben a dokumentáció hiányos, az esetleges változásokat nem megfelelően dokumentálják a projekt résztvevői, úgy a teszt tervezése és maga a végrehajtás is hiányossá válhat.

3.1.1.2 A kockázat-központú (risk-based) stratégia

A másik analitikán alapuló stratégia a *kockázat-központú (risk-based)* stratégia. Ez a fajta stratégia nagyfokú kockázat-érzékenységet követel. Alapja egy igen részletes és mély kockázat-elemzés. Ahhoz, hogy megértsük ennek a stratégiának a lényegét először tisztáznunk kell, mit is jelent a kockázat a szoftverfejlesztésben.

Mi is a kockázat? Mindazon elemek és események bekövetkeztének a valószínűsége, amelyek hátrányosan érinthetik a projekt megvalósítását, valamint amelyek lényegi befolyással lehetnek annak működésére, a végső produktum (szoftver) minőségére. Magyarul: minden, ami nem a terveknek megfelelően történik/történhet kockázatot jelent.

A kockázat-központú tesztelés a tesztesetek meghatározására a termék minőségi kockázatait veszi alapul, annak fényében rendezi sorrendbe a teszteseteket, rendeli hozzá az erőforrásokat. Különböző technikák léteznek a kockázat-központú tesztelésre, de mindegyik célja az, hogy a végtermék egységes minőségi hibáinak kockázatát csökkentsék.

A kockázat-központú tesztelésnek négy fő lépése/fázisa van:

1. Kockázatok azonosítása (kockázat-elemzés)
2. A kockázatok kiértékelése
3. A kockázatok csökkentése – speciális tesztesetek elkészítése
4. A fennmaradó kockázatok kezelése

Természetesen ezek a fázisok néhol egymásba csúsznak, átlapolódnak, de viszonylag határozottan elkülöníthetőek. A későbbiekben egy kicsit bővebben fogok beszélni ezekről a fázisokról.

Ahhoz, hogy a leghatékonyabban lehessen ezt a tesztelési stratégiát használni mindenképpen be kell vonni a projekt minden területéről egy-egy képviselőt (a fejlesztőktől kezdve a végfelhasználókig).

3.1.1.2.1 Kockázatok azonosítása

A projektben érdekeltek (stakeholders) együttesen képesek csak feltárni a lehető legtöbb kockázatot – egyedül mindez nem, vagy csak nagyon kis mértékben, csekély hatékonysággal sikerülhet. A kockázatok feltárásának különböző technikái léteznek[7]:

- Az adott terület specialistáinak, szakértőinek megkérdezése
- Projekttől független személyek megkérdezése (külső szemlélők bevonása)
- Kockázati sablonok használata
- Előző projektek tapasztalatainak felhasználása
- Kockázatfeltáró műhelymunka
- Ötletbörze
- Hasonló projektek tapasztalatainak felhasználása

A projektben érdekeltek minél nagyobb körű bevonásával lehet a legtöbb minőségi kockázatot feltárni. Itt tipikusan és halmozottan érvényesül a már-már közhelynek számító mondás: „Több szem többet lát, több fej okosabb…”

3.1.1.2.2 A kockázatok kiértékelése

A kockázatok azonosítása után el kell kezdeni a kockázatok kiértékelését. Ez tulajdonképpen egyfajta csoportosítást, sorrendbe állítást jelent. A kockázatokat többféleképpen lehet csoportosítani, például típus szerint, esetleg gyakoriság szerint, belőle fakadó hatás szerint, stb. Sokan használják az ISO 9126

[7] A technikák részletes leírása nem része a könyvnek.

szabvány (ami az ISO 25000 szabványt váltotta fel) minőségi karakterisztikáját a csoportosításhoz, de természetesen más kategória-sémák is használatosak/használhatóak.

A kockázatok szintjének meghatározásakor minden kockázati tényező mellé előfordulási valószínűséget is rendelnek. Ez az érték mutatja meg azt, hogy mennyire valószínű, hogy az adott kockázat meg fog jelenni a fejlesztés során. Más szavakkal ez az érték a technikai kockázatot mutatja meg. Ennek az értéknek sok befolyásoló tényezője van, ilyen például:

- A használt technológia összetettsége és a csapatok összeállítása
- Az üzleti elemzők, tervezők és programozók nem megfelelő szakmai tudása
- Konfliktusok a csapaton belül
- Szerződésbeli hiányosságok a beszállítókkal (például nincsenek megfelelően meghatározva a felelősségi körök)
- A fejlesztő csapat földrajzi és időzónai megosztottsága
- A hagyományos és az új megközelítések ellentéte
- A használt eszközök és technológiák
- Nem megfelelő menedzsment vagy technikai vezetés
- Időbeli, erőforrásbeli, költségvetési és gazdálkodási nyomás
- Korábbi minőségbiztosítási tevékenység hiánya, vagy hiányos dokumentáltsága
- Interfész és az integráció kérdése

A másik érték, vagy mutató, amit figyelembe kell venni, az a kockázati esemény hatása, amennyiben az bekövetkezik. Ezt az értéket befolyásoló hatások:

- Az érintett funkció használatának gyakorisága
- Az érintett funkció kritikussága az üzleti cél elérésének tükrében
- A jó hírnév – reputáció – sérülése
- Üzleti veszteség mértéke
- Potenciális pénzügyi, ökológiai, illetve társadalmi veszteségek, vagy kötelezettségek
- Polgári vagy büntetőjogi szankciók, mint következmények
- Engedélyek veszélyeztetése, elvesztésének kockázata
- Áthidaló megoldások hiánya
- A hiba negatív hatása
- Biztonság

Mindkét érték meghatározása lehet mennyiségi (quantitative) vagy minőségi (qualitative). De nem lehet a kettő keveréke, azaz ha az egyiket, például a bekövetkezési valószínűséget minőségileg (nagyon magas, magas, normál, alacsony és nagyon alacsony értékekkel) határozzuk meg, akkor annak hatását is csak ezen a skálán mérhetjük. Ugyanez érvényes fordítva is. (A mennyiségi meghatározás általában 0-10-es skálán mozog.)

Az előfordulási valószínűség valamint az okozott hatás értékek szorzata (mennyiségi), vagy átlaga (minőségi) adja meg a kockázati értéket, ami alapján lehet egyfajta sorrendet meghatározni a kockázatok között.

3.1.1.2.3 A kockázatok csökkentése – speciális tesztesetek elkészítése

Az előző két pontban sikeresen feltártuk a lehetséges kockázatokat, meghatároztuk azok súlyosságát, sorrendbe állítottuk azokat. Most már tudjuk, hogy hány ponton és milyen mértékben sérülhet a minőség az adott szoftver esetén. Már csak az a kérdés, hogy ezeket a minőségi hiányosságokat hogyan tudjuk észlelni? Miként deríthetünk fényt egy esetleges kockázat – ami nem feltétlen észrevehető átlagos szemlélődés közben – megjelenésére, hatásaira.

A legtöbb kockázatot már a megjelenése előtt lehet kezelni, az előfordulását a lehető legkisebbre korlátozni, vagy a hatását mérsékelni. Ilyen kockázat lehet például az egyes szerverek túlterhelésének kockázata. Ezen kockázatot elsősorban azzal lehet kezelni – mondjuk így – hogy a szerver megtervezésekor már egy olyan teljesítményt nyújtó eszközt jelenítünk meg, amely a várható valós terhelés 120-150%-át ki tudja szolgálni. Természetesen miután az adott szerver felépült, mindenképpen tesztelés alá kell vetni, ezt nevezik teljesítmény-tesztelésnek. Az értékeket minden esetben a várható igény 150%-ára javasolt beállítani.

Az előbbi példa természetesen a hardveres eszközre irányult, de ugyanígy kezelhető a szoftveres kockázat is: vegyünk példának egy adatbázis-alapú szolgáltatást, amit más egyéb rendszerek egy speciális csatornán vehetnek igénybe. Ez esetben felmerülhet a kockázata annak, hogy valamely alkalmazás akkora mennyiségű információt próbál lekérni az adatbázisból (esetleg több adatbázisból és többféle táblából) ami annyira megterheli a rendszert, hogy lelassul a szolgáltatás – esetleg más alkalmazások kiszolgálása is lelassulhat – de még le is állíthatja, „lefagyaszthatja” a rendszert. Erre az esetre lehet olyan szélsőérték-tesztelést tervezni, ami a lehető legnagyobb kombinációval rendelkező lekérést elvégzi, esetleg párhuzamosan többet is. Ez legfőképpen akkor merülhet fel, ha az adatbázis szerkezete már eleve meghatározott (örökölt adatbázis-szerkezet egy régebbi rendszer miatt) és a kapcsolódó

rendszereknek más az igényszintje. Új adatbázis-szerkezet esetén már a tervezésnél gondolnak erre (jobb esetben), tehát az ilyen kockázat megjelenésének esélye igen szerény.

3.1.1.2.4 A fennmaradó kockázatok kezelése

Minden esetben elmondható, hogy 100%-os tesztelés, valamint tökéletesen hibamentes szoftver nem létezik. Így az sem lehetséges, hogy minden kockázatot ki tudjunk szűrni, meg tudjunk szüntetni. Természetes, hogy maradnak még kockázati tényezők, le nem tesztelt kockázati lehetőségek, ki nem javított hibák. Mit lehet ezen apró-cseprő – rosszabb esetben nagyobb – jelenségekkel, mondhatni kockázati tényezőkkel kezdeni?

A kérdésre többféle módon lehet válaszolni, attól függően, hogy miből fakadnak ezek a megmaradt (nevezzük így:) hibák. Tapasztalatom szerint ugyanis a „forrás határozza meg a víz ízét is", itt pedig szintén a forrás határozza meg a kockázat további sorsát.

1. *nem volt részletes és mély kockázat-elemzés.* Igen, ez is előfordulhat. Még így is sikerülhet kiküszöbölni jó néhány hibát, kockázati tényezőt, problémát. De a túlnyomó részük benne marad a rendszerbe. Ezekkel – mivel meg sem találták, nem is tudatosult, hogy léteznek – nem nagyon van mit tenni. Nem is tudunk mit tenni. Majd működés közben, éles helyzetben vagy kibuknak a hibák, vagy nem. Előző esetben – ha bejelentik a hibát, azaz a fejlesztő/támogató csapat tudomást szerez a hibáról – lehet azokat azonnal javítani és egy gyors hibajavító csomagot biztosítani az ügyfél/felhasználók számára. Ez egy eléggé költséges megoldás, de van, akinek megéri… Hogy miért, azt én nem tudom. Ha azonban a kockázati tényező nem avanzsálódik hibává, akkor nincs vele gond. ☺ Vak tyúk is talál szemet…

2. *nem volt idő a részletes tesztelésre.* Nos, még ez is előfordulhat – elég gyakran. Ilyenkor egy jó tesztmenedzser a legnagyobb kockázatú elemeket teszteli le részleteiben, igen mélyen, akár többféle technikával is. A kevésbé fontos elemeket viszont csak érintőlegesen teszteli, éppen annyira, hogy pontosítani tudja a kockázat által előidézett hibák mértékét. Ennek alapján ugyanis a projekt vezetője megfelelő döntést tud hozni, hogy a terméket átadja-e a megrendelőnek egy vagy több későbbi hibajavító csomag tervezetével együtt vagy sem.

3. *túl kicsi volt a hatása a kockázatnak.* Az ilyen kockázatokat nem mindig éri meg kijavítani, főleg, ha van hozzá valami más megoldás, amit nem kód-szinten kell/lehet megvalósítani (workaround). Ezek az úgynevezett „tudott hibák" (known issues), amiket esetleg egy későbbi hibajavító csomagban lehet javítani – már amennyiben a megrendelő kéri azokat.

A kockázat-alapú tesztelés önmagában nem használt megközelítés. A gyakorlatban nincs idő annyi típusú kockázat-elemzést és tesztelési technikai elemzést végrehajtani, mint amennyit ez a stratégia megkövetel. Inkább a teljes stratégia egy részét szokta képezni, mintegy kiegészítésként a többi típus/technika mellett.

3.1.2 Modell-alapú (Model Based) teszt stratégia

Ezt a típusú megközelítést 2006-ban még fekete-doboz (black-box) technikaként határozta meg Mark Utting, Alexander Pretschner és Bruno Legeard[8] egy közösen elkészített tanulmányukban, de egy évvel később, 2007-ben Mark Utting már megkérdőjelezte ezen állítását[9] és valós példákat mutatott be mind a fekete-doboz (black-box) mind a fehér-doboz (white-box) technikákból a modell alapú stratégia kapcsán.

Ezen stratégia tulajdonképpen egyfajta modellt vesz alapul, amire kidolgozza az adott tesztelési ciklus teszteseteit. Ez a modell lehet akár egy elvárt folyamat, egy leképezhető matematikai modell, egyfajta felhasználói hozzáállás modellizálása, de bármi más is, amit modellként lehet használni. Ezen modellt alapul véve készítjük el a teszteseteinket.

A konkrét modell létrehozása nagymértékben függ a tesztelendő rendszertől/alkalmazástól. Csak ízelítőnek megemlítenék egy párat: döntési tábla, Markov-láncok, nyelvtani alap, végesállapotú automaták, különböző hatás-diagrammok, stb.

Amennyiben sikerült rátalálni a megfelelő modellre, esetleg többre is, el kell gondolkodni azon, hogy manuális vagy automatizált legyen a következő lépés. Meglepő fordulat, nemdebár? Igen, el kell gondolkodni ezen a lépésen, mert bár a modell-alapú tesztelés alapvetően automatikus tesztelést feltételez, néhány modell-típusnál elképzelhető a manuális teszteset-generálás és végrehajtás is. Gyakorlatilag bármilyen kombinációja lehetséges az automatikus és manuális lépéseknek a teszteset-generálástól annak végrehajtásáig.

A legegyszerűbb, ha a teszteset-generálást és a végrehajtást automatizáljuk a modell alapján. Az automatizáló eszközök széles választéka fellelhető az interneten. Némelyik fizetős, némelyik ingyenes – de majdnem mindegyikről elmondható, hogy valamilyen egyéb eszköz szükséges vagy már a teszteset-generálás előkészítéséhez, vagy az elkészített tesztesetek futtatásához, esetleg mindkettőhöz.

Amit érdemes ismerni a modell-alapú teszteléshez az elsősorban Grady Booch, Ivar Jacobson és James Rumbaugh közös munkájának eredménye, az UML (Unified Modeling Language). Az UML egy gyakorlati,

[8] Mark Utting, Alexander Pretschner and Bruno Legeard: A TAXONOMY OF MODEL-BASED TESTING, in Working Paper Series ISSN 1170-487X
[9] https://www.youtube.com/watch?v=g4Uo2pyrWCg – 2015.02.13.

objektumorientált modellező megoldás, nagyméretű programrendszerek modelljeinek vizuális dokumentálására alkalmas eszköz. Az UML módszer és leíró nyelv segítségével különböző nézőpontú szöveges és grafikus modellek készíthetőek többek közt:

- Rendszerekről, szervezetekről, azok viselkedéséről, külső és belső kölcsönhatásairól stb.
- Szereplőkről, viselkedésükről egy rendszerben, kapcsolatukról más rendszerekkel stb.
- Üzleti tevékenységről, folyamatokról;
- Logikai összetevőkről, azok viselkedéséről, feladataikról, kommunikációjáról egy rendszeren belül, vagy rendszerek között stb.
- Szoftverekről, programokról. (Az UML az objektumorientált programozás szabványos specifikációs nyelve.)
- Adatbázisokról: elméleti, logikai és fizikai modellekről egyaránt.

Ugyanis ennek a segítségével lehet a leginkább hasznosítani a modell alapú tesztelés előnyeit. Vegyük példának a GraphWalker nevű, nyílt forráskódú tesztelői eszközt[10], amelyik pontosan egy ilyen alapú gráfból tud/képes dolgozni. Egészen pontosan egy XML alapú gráf leíró nyelvből, a GraphML[11]-ből tud teszteket generálni. Azaz, szükséges egy kis előmunka, létrehozni a gráfokat. Ilyen típusú gráfok létrehozására is számtalan program kínálkozik, de kitűnő választás lehet az yEd[12].

Amint elkészültünk a gráfunkkal, majd megetettük a GraphWalker-rel már van is egy alap teszteset-gyűjteményünk. Ez még nem a végleges verzió, hiszen itt még „csak" az alap, gráfból generált esetek jelennek meg. Nem kezeli le a negatív eseteket, a nem elvárt működésre nincs reagálás (hibakezelés) beépítve, stb. Még kell egy kis manuális finomítgatás, egy kis finomhangolás ahhoz, hogy minden megfelelően működjön. Kitűnő leírást készített ennek a két eszköznek a használatához Tóth Árpád, okleveles programozó matematikus a Tesztelés a gyakorlatban portálján[13] egy gyakorlati példán keresztül bemutatva működésüket.

A fent megemlített eszközön kívül még használható, ingyenes eszközök a modell-alapú teszteléshez:

- fMBT (free Model Based Testing) - https://01.org/fmbt Python nyelven íródott, elő- és utófeltételekkel meghatározott modellekből generál teszteseteket.

[10] http://graphwalker.org/index - 2015-02-14.
[11] http://graphml.graphdrawing.org/ - 2015-02-14.
[12] http://www.yworks.com/en/products/yfiles/yed/ - 2015-02-14.
[13] Tóth Árpád: A modell alapú tesztelésről - http://www.tesztelesagyakorlatban.hu/keres_cikk.php?mit=3

- JTorX - https://fmt.ewi.utwente.nl/redmine/projects/jtorx/wiki/ - valós idejű, bemeneti és kimeneti konformancia (input/output conformance – IOCO) reláció vizsgálatára alkalmas eszköz. Számos modellező eszközt támogat, nagyrészük LTS (Labelled Translation System = címkézett transzlációs rendszer) alapú. A JTorX egy futó alkalmazáshoz kapcsolódva végzi a tesztelést.
- ModelJUnit - http://www.cs.waikato.ac.nz/~marku/mbt/modeljunit/ - egyszerű végesállapotú automaták, vagy kiterjesztett végesállapotú automaták tesztelésére szolgáló eszköz.
- OSMO - https://code.google.com/p/osmo/ - Java alapú eszköz, ami feltáró tesztelést végez különböző technikák felhasználásával.
- PyModel - http://staff.washington.edu/jon/pymodel/www/ - képes kapcsolat nélküli tesztelésre is, de főleg a működés közbeni (on-the-fly) tesztelésre lett kifejlesztve.

Néhány fizetős eszköz (nem feltétlenül jobbak, leírásukat megtalálod a linkeken):

- BPM-Xchange - http://www.bpm-x.com/solutions/model-based-testing-for-erp.html
- Conformiq Designer - http://www.conformiq.com/products/
- DTM - http://www.dtmtool.com/
- MaTeLo - http://www.all4tec.net/index.php/All4tec/matelo-product.html
- RT-Tester - https://www.verified.de/products/model-based-testing/
- Smartesting CertifyIt - http://www.smartesting.com/en/product/certifyit
- Spec Explorer - http://visualstudiogallery.msdn.microsoft.com/271d0904-f178-4ce9-956b-d9bfa4902745/
- TestCast - http://www.elvior.com/test-generation/overview
- TestOptimal - http://testoptimal.com/
- T-VEC - http://www.t-vec.com/solutions/simulink.php

3.1.3 Szisztematikus (Methodical) teszt stratégia

Leginkább tapasztalati alapú stratégia. Nem hivatalos, inkább valami formális szabályzatnak, egyfajta ellenőrzési listának lehetne megfeleltetni. Pontról pontra haladva minden egyes alkalommal ugyanazon

listát használva tesztelünk. A lista természetesen folyamatosan változik, mindig belefoglalva az aktuális tapasztalatokat. Nagyjából három különböző típusú lista lehetséges:

1. *Tapasztalati alapú:* ez esetben a lista azokat az elemeket tartalmazza, ahol/amikor hibát talált a tesztelő. Ez azt jelenti, hogy az első tesztciklusban lehet, hogy még nincs ilyen lista, vagy esetlegesen egy másik fejlesztésből „örökölt", tapasztalati lista van csak. A kritikus folyamatbeli pontokat, vagy modulokat, esetleg azokat a helyeket tartalmazza, amit elég csak átfutni. A lista mindig bővül, hiszen mindig lehet új pontokat beiktatni, mindig vannak eddig nem tesztelt, frissen lefejlesztett funkciók, újonnan meghatározott folyamatok, frissen elkészült modulok. Természetesen ezek is felkerülnek erre a „halállistára", amin minden tesztciklus alatt a tesztelő végigmegy.

2. *Funkció-alapú:* ez a fajta lista már a valós tesztvégrehajtás előtt szinte teljesen elkészülhet – természetesen csak abban az esetben, ha egy jól előkészített funkcionális terv dokumentum is a kezünkben van. (Ezt ugyebár nem a tesztelő készíti el, hanem az üzleti elemző – jó esetben bevonva egy fejlesztőt is.) Ezen dokumentum alapján már könnyedén – akarom mondani: **viszonylag** könnyedén – meg lehet írni a teszteset-gyűjteményünket. Azonban ez is bővülhet, pontosan amiatt, hogy az alapdokumentum esetleg nem volt eléggé részletes, vagy olyan funkciókra is fény derül, ami eredetileg nem is volt tervbe véve, de az informatika és a programozók szellemének és védőszentjeinek (beleértve a koboldokat és manókat is) köszönhetően mégis megvalósult. (Nem kell semmi komolyra/komolytalanra gondolni, egyszerűen csak némely funkciók összjátékából egy újabb funkció születhet. Gondoljunk csak a népszerű táblázat-kezelő programban található játékokra…)

3. *Minőség-alapú:* az ilyen lista azokat az elemeket tartalmazza, amit az adott, tesztelendő szoftver vagy megoldás megkíván az általános ISO 9126 szabvány által felsoroltakból. Ilyenek lehetnek: funkcionalitás, lokalizáció (fordítás), használhatóság, teljesítmény, megbízhatóság, stabilitás, telepíthetőség, dokumentáltság, stb. Természetesen ezt a listát is mindig az adott megoldásra szabva kell elkészíteni, figyelembe véve a megrendelő kívánságait.

A Szisztematikus teszt stratégia használatakor a teszt céloknak egy általános készletét, listáját használjuk. Ez a stratégia gyors és hatékony az olyan rendszereknél, amik már stabilan működnek, vagy amely rendszerek nagymértékben hasonlítanak egy, már előzőleg jól tesztelt rendszerre. A tesztelt rendszerben végrehajtott jelentős változtatások/változások néha hiábavalóvá, hatástalanná tehetik ezt a stratégiát –

ilyenkor érdemes egy másik stratégiához folyamodni, még ha csak ideiglenesen is – egészen addig a pontig, amíg a lista nem frissült.

3.1.4 Folyamat- vagy szabvány-kompatibilis (Process-Oriented Test Strategy) teszt stratégia

Ez a fajta stratégia a tesztfolyamatok szabályozásával módosítja, mondhatni szigorítja a szisztematikus stratégiát. Ezzel a szabályozással egy hivatalos, vagy nem hivatalos, de mégis használt szabványt követ a tesztelési folyamat. Ilyen szabvány lehet például az IEEE 829, ami a teszt dokumentálását szabályozza, az IEEE 1008, ami az egységtesztre ad szabványt, vagy az ISO 29119 ami a teljes tesztelési folyamatra ad irányelveket. Ezek a szabványok segítséget nyújthatnak ahhoz, hogy a tesztelés átláthat, érthető legyen mind a programozónak, fejlesztőnek, üzleti elemzőnek – azaz a fejlesztő csapat tagjainak -, mind a megrendelőknek, azaz olyanoknak, akik „nem tesztelők”. Arra azért érdemes odafigyelni, hogy ezekkel a szabványokkal ne generáljunk túlzott bürokráciát, esetleg felesleges papírmunkát. Az ugyebár senkinek sem lenne kellemes…

Nem hivatalos, de egyre népszerűbb az agilis teszt stratégia, ami a szoftverfejlesztői, programozói oldaláról bukkant elő és fejlődik egyre rohamosabb mértékben. Ezek a típusú stratégiák inkább a technikai hibákra fókuszálnak, mint például a dokumentációra. Ez korántsem azt jelenti – mint azt sokan nagyon rosszul értelmezik -, hogy nincs dokumentáció, mert nagyon is van, csak nem arra helyezik a hangsúlyt. Nem kifejezetten szabvány-kompatibilis tesztelési stratégia, de kiválóan alkalmazható, mint kiegészítő stratégia. (A következő sorokban még visszaköszön majd ez a típusú tesztelés.)

3.1.5 Dinamikus teszt stratégia

A dinamikus teszt stratégiák, mint például az agilis teszt stratégia, a kezdeti tervezési feladatokat a minimumra szorítja, inkább a teszt végrehajtására fókuszál. Próbál a lehető legnagyobb teret engedni a hibák felfedésére, azaz egyfajta „hibavadászatra” ösztönöz. Ez nem jelenti azt, hogy nincs semmilyen megelőző tervezés, rendszerezés, dokumentálás, hanem azt, hogy nem annyira részletes, szabályozott, inkább irányvonalakat határoz meg. Ez a típusú teszt stratégia inkább hagyatkozik a tesztelői csapat kollektív tapasztalatára, tudására és a „zsigeri ösztöneikre”. A fő tesztelési irányvonalakat a mindennapos beszélgetések, viták a tesztelendő esetekről, a múltbeli, hasonló projektek tapasztalatainak beépítése adja meg.

Ha feltáró/felfedező teszt-megközelítést (exploratory testing) alkalmazol, akkor mialatt tesztelsz, megtanulod, egyre inkább megismered a rendszer felépítését, működését. Ezáltal folyamatosan tudod

fejleszteni, finomítani a teszt stratégiádat a teszteredményeidre alapozva, esetlegesen újra meghatározni a tesztelés fókuszát ezen eredmények tükrében. Amint azt már említettem, ez a stratégia egyfajta „hibavadászat”, azaz lehet használni hiba-profilokat, amiket a megtalált hibák alapján készítesz el, különböző osztályokat hozol létre és nagy szerepe van a hibasejtésnek is. Ez utóbbi természetesen ismételten a tapasztalataidra épít, azokból merítve – valamint a józan paraszti ész elvét használva – jelentős hibamennyiséget lehet felfedezni.

Ennek a tesztstratégiának a hajlékonysága, a rugalmassága a nagy előnye, valamint az igen magas hibatalálati arány. Hátránya, hogy nem igazán lehet konkrét tesztelési lefedettséget meghatározni, inkább csak megközelítő adat áll erről a rendelkezésünkre. Ugyanígy nincs lehetőség a kockázatok elkerülésére, kiküszöbölésére, valamint a hibák a fejlesztés korai szakaszában (pl. követelmények összegyűjtése) való megtalálására. Mindazonáltal ez a megközelítés még mindig jobb, mint ha nem is tesztelnénk egyáltalán, de ha egy kicsit keverjük az analitikus teszt stratégiával, akkor egy elég hatékony elegyet kapunk, amely némileg kiküszöböli mindkét teszt stratégia hiányosságainak egy részét.

3.1.6 Konzultációs vagy irányított teszt stratégia

Már a nevéből is sejthető, hogy ez a fajta stratégia vonja be leginkább a teljes fejlesztői, vagy akár projekt csapatot is a tesztelésbe. Ez azt jelenti, hogy a tesztelendő területek, egységek meghatározásánál ugyanúgy jelen vannak akár a megrendelők is, mint az egyes tesztesetek meghatározásánál.

Előnye ennek a stratégiának, hogy – amennyiben a végfelhasználót is bevonjuk – már a tesztelés legelső fázisától fogva arra a területre/azokra a területekre koncentrálhatunk, ami a végfelhasználó szempontjából kritikus. Úgy üzleti szempontból, mint a mindennapi munkavégzés, azaz használhatóság szempontjából. De ahogyan említettem, a teljes csapatot illik bevonni, azaz magukat a fejlesztőket is – tőlük kaphatjuk a legfontosabb információt a rendszer esetleges gyengeségeiről, kritikusnak vélt pontjairól. Így technikai szempontból is a fontos területekre tudunk koncentrálni.

Természetesen itt is felmerülnek bizonyos kockázatok, hiszen előfordulhatnak nézeteltérések még a végfelhasználók között is a fontosság megítélésében, esetleg nem megfelelő tudással és/vagy tapasztalattal rendelkezik a bevont felhasználó, ami a tesztesetek kiválasztásánál nem megfelelő lefedettséget generál. Ugyanígy előfordulhat prioritásbeli nézeteltérés, különbözőség, ami szintén a teszteset-gyűjtemény minőségét rombolja.

Akárcsak a dinamikus teszt stratégiát, ezt a típusú megközelítést is érdemes egy másikkal keverve alkalmazni, amely lefedi, kiküszöböli a korábban említett hiányosságokat.

3.1.7 Filozofikus teszt stratégia (Philosophical test strategy)

Ez a típusú stratégia valamilyen filozófiát, egyfajta hitvallást vesz alapul. Arra építve hozza létre a tesztelését, annak elveit használja a tesztesetek létrehozásánál, a tesztelés végrehajtásánál, sőt, még a jelentések és hibajegyek elkészítésénél is. Első pillantásra talán egy kicsit ijesztőnek hangzik, de valójában nem az. Szerintem mindenki használja valamilyen mértékben ezt a stratégiát is, habár nem jellemző, hogy a filozofikus stratégia lenne a mérvadó. Jellemzően csak a háttérben, mint egy segéd-stratégia létezik. Ahogyan a filozófiában, itt is több irányzat létezik:

1. Az *aprólékos*: mindenben és mindenkiben van és lesz is hiba. Valakiről vagy valamiről azt állítani, hogy hibátlan, az elfogadhatatlan. Mindenképpen kell találni hibát. Az a vezető, aki ezt a filozófiát vallja tekintélyes erőket (és erőforrásokat) képes megmozgatni azért, hogy minden hibát megtaláljon a rendszerben. Ilyenkor minden erejét latba vetve tesztel: használja a stratégiai megközelítést, az összes funkciót leteszteli, akár többször is, a kockázat-alapú stratégiát is igénybe veszi, valamint minden egyebet, ami szerinte – és a „főnök" szerint – segíthet abban, hogy az összes hibát megtalálja. Nagyon jól jellemzi ezt a filozófiát egy tréfás amerikai mondás egy ismeretlen tollából: „In God we trust – the rest we test!" (Istenben hiszünk – minden mást tesztelünk.)

2. A *„vadászpuska"*: azt feltételezi, hogy minden és mindenki lehet hibás. Azaz: mindenkiben és mindenben lehet hiba. Ugyanakkor azt is tudja és elfogadja, hogy mindent nem tud letesztelni, nem tud minden hibát megtalálni. Mivel nincsen konkrét ötlete sem, hogy merre, hol találhat hibát ezért „oda lő, ahova éppen gondol". Mint az egyszeri vadász: ahol mozog valami, ott kell lennie valaminek, tehát meglövöm. Vagy talál, vagy nem. A tesztelést véletlenszerűen végzi, nincs szisztematikus lefedettség – pont úgy, mint a puskából kilövellő sörétek: van, ahova több, van, ahova kevesebb jut. Mondanom sem kell, hogy ez akár jól is elsülhet, de inkább erőforrás-pazarlásként jelentkezik a szoftverfejlesztés életében…

3. A *külső irányító*: nemcsak azt fogadja el, hogy nem tud mindent letesztelni, de azt is, hogy fogalma sincs, merre keressen hibát. Éppen ezért bízik abban, hogy más, egy külső irányító tudja, hol lehetnek hibák, hol érdemes „keresgélni", azaz mit érdemes tesztelni. Ezért kéri egy külső személy segítségét. Nemcsak a tesztelés irányítását bízza külsős személyre, de gyakran kér tanácsot annak eldöntésére, hogy a teszteredmény az elvártnak megfelel-e, avagy nem. Ilyen külsős személyek lehetnek a végfelhasználók, programozók, üzleti elemzők, technikai vagy ügyfélszolgálati munkatársak, stb.

Amennyiben a mindezek mögött húzódó filozófia, vagy hitvallás helyes, azaz az adott projektre érvényes, akár még hatásos is lehet. Például ha egy űrrakéta-projektet kellene tesztelni, akkor az aprólékos megközelítést használnám, hiszen a legapróbb hiba is sok-sok millióba, valamint emberéletekbe kerülhet. Ha nem a megfelelő stílust alkalmazzuk – márpedig a legtöbb esetben legalább kettő, ha nem mind a három alkalmatlan a projektre nézve – akkor az veszélyes, esetenként katasztrofális helyzetet teremthet.

Szerény véleményem szerint azon esetekben, mikor egy teljesen új szoftvert, vagy szoftveres megoldás tesztéléséről beszélünk az első filozófiát, az aprólékos megközelítést használnám. Az – bár eléggé idő és erőforrás-igényes – megfelelő lefedettséget biztosítana a szoftver végleges kiadásának minőségi követelményeihez. Mivel azonban a legtöbb cég már rendelkezik valamilyen szoftveres megoldással és projektjeik leginkább azok fejlesztésére („foltozására") irányulnak, mindenképpen javasolnék valamilyen visszatérő, regressziós teszt stratégiának a bevonását.

3.1.8 Visszafejlődés-elkerülő (Regression-averse)

Ez a stratégia használja ki a leginkább az automatikus tesztvégrehajtás lehetőségeit. Ez a stratégia a különböző, már átadott funkciók hibáit próbálja feltárni, amik egy újabb funkció vagy hibajavítás hatására jöhetnek elő. Ezeknek a hibáknak három fő típusa van. Az első, amikor a hiba a helyi rendszerben jelentkezik, azaz a visszafejlődés helyi érdekű. A második típus az, amikor a friss fejlesztés (új funkció, vagy hibajavítás) egy már addig is létező hibát fed fel. A harmadik azon lehetőségek tárháza, amikor az új funkció egy másik, a lokálishoz kapcsolódó rendszerben okoz romlást, azaz hibát. Nem csak a már meglévő funkciókat érheti ilyen típusú visszafejlődés, hanem az éppen fejlesztés alatt állóakat is.

Az okos tesztelő stratégiát épít. Nem mindig és nem minden tesztesetet hajt/hajtat végre. Attól függően, hogy mi változott, vagy melyik kódrészlet módosult, ahhoz mérten állítja össze a teszteset-gyűjteményt is, ami a leghatásosabb az adott szituációban. Mint ahogyan azt már megemlítettem, ez a stratégia igen nagymértékben veszi igénybe a tesztautomatizálási technikákat, technológiát. Nézzünk egynéhány példát ezekre a speciális stratégiákra:

1. *Az összes teszteset ismételt végrehajtása* – a totális támadás (brute force) mindig egy opció lehet. Ez a legegyszerűbb stratégia, ami megvéd minket attól, hogy valamit elfuseráljunk, miközben próbálunk igen okosnak látszani… Ez a fajta stratégia azt feltételezi, hogy a tesztelőnek eleve van egy nagyon jól összeállított teszteset-gyűjteménye, ami tökéletes összhangban van az adott projekt minőségi elvárásaival. Egyértelmű, hogy jelen esetben csak az automatizált

tesztelés jöhet szóba, hiszen egy jól felépített, minden szempontból a megfelelő minőségi elvárásokkal összhangban lévő teszteset-gyűjtemény végrehajtása nem kevés időbe telik. Ezért kell automatizálni a tesztesetek végrehajtását. Az automatizált tesztesetek épülhetnek grafikus felhasználói felületre (GUI – Graphical User Interface), alkalmazás programozási felületen keresztül való végrehajtásra (API – Application Programming Interface), vagy esetleg parancssoros felületen (CLI - Command-line Interface) keresztül is futtathatjuk teszteseteinket.

2. *Egy bizonyos részegység végrehajtása* – még egy hatékony tesztautomatizálásnál is előfordulhat, hogy a teljes tesztelést nem lehet minden alkalommal végrehajtani – költségtakarékosság, esetleg menedzsmenti határozat miatt. Gyakran az is előfordul, hogy nem minden pontját lehet automatizálni a tesztelendő folyamatnak, alkalmazásnak. Lehet, hogy csak kis része van automatizálva, esetleg nincs benne automatikus tesztelés. Nos, ilyenkor mi legyen? Mi a következő lépés? Nos, ilyenkor érdemes egy kisebb készletet, teszteset-gyűjteményt összeállítani, ami megfelel az elvárásoknak, azaz a költségkereten és időkereten belül a leghatékonyabban tudja mérni a változás okozta jelenségeket, feltárni az esetleges hibákat. Ennek a szűkített gyűjteménynek a meghatározására is sokféle módszer lehetséges – nézzünk meg egy párat:

    a. Használhatósági kockázat alapján – ez a megközelítés azt veszi alapul, hogy mely elemeket használnak a leggyakrabban az adott rendszerbe, azaz hol merülhetnek fel a leghamarabb hibák, olyanok, amik fájdalmasan érinthetik az üzleti érdekeket. Három szempontot kell figyelembe venni: a követelményeknek való megfelelőséget, a terveknek való megfelelőséget valamint a minőségi kockázatoknak való megfelelőséget.

    b. Változás-kezelés (change management) alapján – amennyiben a tesztelő nincs tisztában technikai szemszögből a rendszer működésével, szüksége lehet egy fejlesztő, vagy programozó segítségére. Jelen esetben ugyanis azt kell megvizsgálni, hogy a kódban megjelent módosítás hol, milyen mértékben lehet hatással a folyamat későbbi szakaszában. Mely modulokat, funkciókat érintheti és milyen mértékben.

    c. Minőségi kockázatok alapján – ez a harmadik feltételez egy minimális minőségi kockázat-elemzést. Az előző kettő esetében a technikai kockázatokat vontuk nagyító alá és próbáltuk azokat tesztelni, viszont ez a típusú megközelítés inkább a minőséget veszi alapul felhasználva a minőségi kockázat-analízist. Meg kell jegyeznem, hogy minden

alkalommal felül kell vizsgálni a már elkészített kockázat-analízist is, azon okból és célból, hogy az előkészített regressziós teszteset-gyűjtemény a lehető legpontosabb mintát nyújtson és a lehető legnagyobb területet fedje le a kockázatokból.

Leginkább az inkrementális fejlesztési modelleknél látható az ilyen típusú stratégiának a haszna, amikor igen gyors ütemben kell újabb frissítéseket, hibajavításokat kódolni az adott terméken. Ezzel a stratégiával a minimumra csökkenthető a már lefejlesztett modulok, funkciók sérülésének esélye. Hátránya lehet, hogy nincs megfelelő eszköz a tesztek automatizálására, esetleg nincs elég tudás a csapaton belül a megfelelő előkészítéshez. Előfordulhat az is, hogy az új funkciók nem megfelelő alapossággal lettek letesztelve, ezzel adva táptalajt a későbbi regressziós tesztek eredményességének…

Nos, most már van némi fogalmad arról, hogy mi mindent is takarhat a teszt stratégia. Olvasgattad, ízlelgetted a különböző stílusokat, lehet, némelyik meg is tetszett neked. Talán már ki is választottad a kedvenceidet. Igen, többet – legalábbis remélem. Tudod, van egy nagy titka a jó tesztmenedzsernek: **sosem használ kizárólag egy stratégiát!**

Ha figyelmesen olvastad el az egyes stratégiák leírását, akkor biztosan fel is tűnt számodra, hogy valamely másik stratégia párhuzamos használatát szinte minden esetben javasoltam – direkt vagy indirekt módon. Ezért kérlek, olvasd el újra ezt a fejezetet. Igen, az egészet.

És most újra… És megint… És ismét…

Elfáradtál? Kívülről fújod a stratégiákat? Reggel a kávé mellett a pároddal a napi feladatok helyett már a filozofikus tesztstratégia egyetemes kérdéseiről folytattok heves eszmecserét? Jó… Akkor lassan kezded érteni... ☺

3.1.9 A stratégiai dokumentum

Ahogyan azt már biztosan sejtetted, a teszt stratégiát jellemezően tesztmenedzser vagy a tesztelő csapat vezetője alakítja ki. Ez attól függ, hogy milyen metodológia szerint dolgoznak[14]. Valamint befolyásolja még számtalan külső és belső tényező. Lássunk ezekre is néhány példát:

- Külső tényezők:
    - *Tesztelői csapat viszonya a projekthez* – manapság egyre több a **bértesztelés**. Elég csúnya és pejoratív hangzású szó, pedig valójában csak arról van szó, hogy egy, a

[14] A szoftverfejlesztési metodológiákról/modellekről bővebben: Boda-Bodrogközi-Gál-Illés: Szoftvertesztelés a gyakorlatban, 2014.

projekttől teljesen független csapat végzi el a teszteléssel járó feladatokat. Itt a teszt-stratégiát a projekt csapat által nyújtott dokumentáció (vagy annak hiánya) befolyásolja a leginkább. Ugyanígy egyre gyakoribb az úgynevezett „crowd testing" (csúnya magyar fordításban: közönség-tesztelés, vagy tömeges tesztelés), ami annyit jelent, hogy a projekt csapat összeállítja a teszteseteket, teszt-szviteket, amit kiad egy közösségnek végrehajtás céljából. (Vannak már olyan portálok, cégek, akik erre szakosodtak, az ilyen típusú tesztelést segítik.)

- *A megrendelő által emelt minőségi követelmények a végrehajtás (projekt) idejének tükrében* – előfordulhat, hogy bizonyos okok miatt a projekt időzítése módosul, csúszik valamennyit. Ezzel szemben a megrendelő ragaszkodik az eredeti befejező dátumhoz – a tesztelés kárára. Ez egy gyorsabb, ám kevésbé nagy lefedettséget produkáló teszt-stratégiát igényel.

- Belső tényezők:

  - A *cég vagy vállalat magasabb szintű stratégiájának* követése. Azaz a teszt stratégiának illeszkednie kell a magasabb szintű minőségbiztosítási stratégiához.

  - A *cégen belüli folyamatok,* azaz amennyiben egy bürokratikus, mindenféle engedélyeztetési folyamatokkal átszőtt vállalati környezetben kell dolgozni, akkor szinte képtelenség egy agilis fejlesztési-tesztelési stratégiát követni.

  - A *tesztelő csapat belső stratégiai céljainak* összeegyeztetése az adott projekt minőségi céljaival. Itt az okozhat problémát, hogy a megrendelő alacsonyabb szintű minőségellenőrzést (ritkább esetben jóval magasabbat) vár el, mint amit a csapat a saját céljainak kitűzött maga elé.

Szépen felsoroltam a különböző stratégiákat, osztogattam jó tanácsokat, hogy hogyan alkalmazd mindezen tudást, felhívtam a figyelmedet egy pár befolyásoló tényezőre is – bizonyára felmerült benned az is, hogy mindezt hogyan kell egyáltalán érvényesíteni? Van-e valami dokumentum, ami mindezt leírja? Esetleg több ilyen dokumentum?

Igen, van – ha megírod. ☺ Nem minden esetben egyértelmű, hogy a teszt stratégiát írásban is rögzíteni kell. Az általánosan elfogadott nézet az, hogy amennyiben projektről beszélünk legalábbis a

tesztterven, vagy a minőségbiztosítási tervezetben (Quality Plan) érdemes egy szakaszt a teszt stratégiának szentelni.

A tényleges tartalmát a teszt stratégia leírásának, dokumentumának a tesztmenedzser határozza meg. Természetesen van egy ajánlás, mondhatni lista, amit érdemes betartani – persze csak óvatosan, az egyszerűség és a használhatóság elvét szem előtt tartva. Csak a szükséges és az adott projektre értelmezhető részeket javasolt használni.

Na, de lássuk azt a listát:

1. *A tesztelés célja és annak hatásköre* – konkrétan meg kell nevezni a modulokat, vagy magát a szoftvert, esetleg szoftver-csoportot vagy folyamatot. Leginkább akkor fontos ez a szakasz, ha kevés erőforrással kell gazdálkodnunk és már a projekt elején lehet látni, hogy nem lesz elég erőforrás a legnagyobb lefedettségre (azaz a ~99%-os lefedettségre).
2. *Lefedésre kerülő üzleti követelmények köre* – az előző pontban kifejtett területekhez, modulokhoz tartozó üzleti követelményeket tároló dokumentum, vagy rövid kivonata. Tulajdonképpen szinte a tesztesetek és követelmények összekötéséről szól ez a rész. No, nem részletekbe menően, csak úgy nagyjából. Csak annyira, hogy tiszta kép alakulhasson ki mindenkinek, hogy mi és milyen mértékben lesz letesztelve.
3. *A tesztelés szabályai és felelősségi köre* – ki, mikor, mit, milyen mértékben fog tesztelni. Ki miért felelős? Ez egyfajta RACI/RASI mátrix (**R**esponsible-**A**ccoutable-**C**onsultant/**S**upport-**I**nformed), ami szintén fontos.
4. *Jelentések tartalma és előállítási módja* – tulajdonképpen egyfajta kommunikációs terv. Ki, mikor milyen formátumban 'jelent', és kinek? Jelentés alatt értendők a hibajegyek, a tesztriportok, egyéb jelentések.
5. *Tesztelésre alkalmas állapot feltételei* – le kell írni azokat a feltételeket, amiknek teljesülniük kell ahhoz, hogy a tesztelési munka elkezdődhessen. Ez a rész a tesztelési ciklus belépési feltételeit tartalmazza tulajdonképpen. Mivel tudjuk, hogy sosem lesz minden tökéletes, ezért érdemes súlyozni az egyes feltételeket, kiemelni azokat, amiknek mindenképpen teljesülniük kell (pl.: van tesztkörnyezet és a tesztelendő kód/program a tesztkörnyezetre van telepítve), megjegyezni, hogy minden más esetben hogyan kell eljárni. Az eljárás alatt értem azt, hogy ha egy adott feltétel, ami nem alapvető fontosságú, nem teljesül, akkor milyen egyéb lépéseket kell tenni ahhoz, hogy a projekt folytatódhasson, azaz a tesztelési ciklus elkezdődhessen. Tipikusan ilyen plusz lépés lehet egy akcióterv elkészítése az adott feltétel határidőre való teljesítésére.

6. *Tesztelés előfeltételeinek megteremtési módja* – na, ezt jól megírtam. Mit is akar ez jelenteni? Nos, többek között azt is, hogy legkésőbb mikorra kell a tesztrendszernek készen állnia. Pontosabban mi az a végső határ, ami után már nincs értelme új kódokat kitenni a tesztrendszerre, mert annak tesztelése már kérdésessé válik az adott tesztablakban. De ugyanúgy jelentheti azt is, hogy a megtervezett teszteléshez szükséges – amennyiben szükséges – adatokat milyen módon, honnan és kinek a segítségével tudja majd megszerezni a tesztmérnök, vagy maga a tesztelő. Szóval csupa olyan dolog, ami az előző pontnál leírtakhoz közelebb visz minket.
7. *Tesztadatok előállításának módja* – sok esetben a teszteléshez különböző, előkészített adatokat kell használni. Ilyen lehet például egy rendelés-számlázás folyamat tesztelésekor a raktári mennyiségek, a raktáron lévő termékek adata. Ezeket egy másik folyamat segítségével kell létrehozni, lehetőleg még az adott tesztesetek futtatása előtt. (Akár a tesztciklus kezdete előtt is lehet ilyeneket generálni, amennyiben a tesztkörnyezetet már nem fogják frissíteni az adatgenerálás és a tényleges tesztelés közötti időszakban. Mert az ugyebár törölné az adatokat… Biztosan nem esne jól senkinek egy ilyen eset…) Mindezen adat-előállítások természetesen lehetnek manuálisak (pl. speciális árajánlatok kérése, vagy esetleg egy folyamat egy bizonyos pontján való megszakítása, stb.), vagy automatikusak (tipikusan a virtuális árukészlet, esetleg nagy mennyiségű felhasználói fiók létrehozása). Ez utóbbi gyakrabban fordul elő.
8. *Teszteléshez használt eszközök* – ebben a szakaszban fel lehet sorolni a tesztszerverektől kezdve egészen a legutolsó ceruzabélig mindent. De tényleg mindent – ami a teszteléshez használt eszközök fogalmába beleérthető. De ha komolyan akarod leírni az eszközöket, akkor ténylegesen csak a releváns eszközöket fogod megnevezni (pl.: PC, MAC, Virtuális géppark, MS Office alkalmazások, stb.) Persze ide értjük a tesztautomatizáláshoz használt eszközöket is (pl.: HP UTP, JUnit, LoadRunner, stb.)
9. *Alkalmazandó teszt típusok és tesztelési technikák* – ezt most nem írnám le újra. Reményeim szerint ezek a fogalmak már nem újak számodra. Persze, egy kis felfrissítés sosem árt, ezért tessék, néhány példa: statikus tesztelési technika, például az informális felülvizsgálat (Informal review), technikai felülvizsgálat (Technical review), avagy dinamikus tesztelési technika az egységteszt, az integrációs teszt, a rendszerteszt, stb.
10. *Tesztek és futáseredmények követésének módja* – ezek azok a mérőszámok, amik megmutatják, hogy merre jár a tesztelés. Milyen eredményes, mennyi hibát talált, stb. Egyszóval: megmutatja, hogy hogyan is áll a tesztelés. Ennek többféle módja lehetséges: lehet egy tesztmenedzsment

eszköz beépített statisztikai funkcióját használni, vagy esetleg egy egyszerű táblázatból lehet létrehozni mindenféle eredménytáblákat. (Ez természetesen a két véglet.)

11. *Kockázat csökkentési módok* – ez a rész feltételez egy kisebb kockázatelemzést, amivel nincsen gond, ha a kockázat-alapú stratégiát választod. ☺ De akkor sincsen baj, ha nem az a kedvenced – ugyanis mindenféleképpen kell készíteni egy kockázatelemzést az egész tesztelésre. Ennek eredménye lesz egy kockázati lista, aminek elemeihez akciótervet kell kidolgozni. Nem kell rögtön egy James Bond-filmben érezned magad, nem annyira bonyolult és veszélyes dolog. Hadd mondjak egy példát: az egyik legnagyobb kockázat a tesztelési ciklusban az, hogy – mivel kora tavaszra esik – a tesztelői csapat influenzával nyomja majd az ágyat, így nem tud tesztelni[15]. Erre többféle módon is lehet készülni, például előzetesen védőoltásra kötelezed a csapat minden tagját. Vagy a projekt ideje alatt hermetikusan elzárt környezetben tartod őket a NASA által kifejlesztett laborban, ahol csak az asztronautáknak kifejlesztett ételeken és vitamin-komplexumokon élhetnek. Igaz, ezek eléggé a földtő elrugaszkodott megoldások, de lehetséges. De komolyra fordítva a szót: minden súlyos – azaz magas kockázati értékkel bíró – kockázatot le kell fedni egy olyan tervvel, amely valamely mértékben csökkentheti az adott kockázat előfordulásának mértékét, vagy lehetőségét, esetleg mérsékli annak hatását a projektre.
12. *Hibajelentés és nyomon követés módja* – a legegyszerűbben megfogalmazva: a hibakezelő rendszert – ha van ilyen – kell megnevezni és annak használatát kell itt leírni. Ha az egy egyszerű táblázat, akkor az oszlopok és cellák leírása kerül ide. Valamint az, hogy a bejegyzett hibáknak mi lesz az utóélete.
13. *Ügyféligényben történt változások követésének módja* – a követelményekben végbemenő változásoknak leképezése a tesztelés tekintetében. Azaz mi történjék akkor, ha az ügyfél kék, négyszögletes gomb helyett hupilila és kör alakú gombot szeretne – és mindezt már akkor közli a csapattal, amikor a tesztelés éppen folyamatban van… (Azt már inkább nem írom le, hogy ilyenkor milyen gondolatok merülnek fel mind a programozó, mind a csapat többi tagjának elméjében…)
14. *Rendszerkonfigurációban történt módosítások kezelésének módja* – talán ez az egyik legritkábban előforduló esemény egy projekt életében. Mindenesetre érdemes erre is felkészülni. Miről is van szó? Például egy webes alkalmazás fejlesztésekor a tartalomkezelő-rendszer beépített keresőmotorját kezdik el használni, majd a projekt során kiderül, hogy az nem minden igényt elégít ki, ami a vevő részéről felmerült. Így egy külső keresőmotor integrálása kerül a középpontba, ami ugye megváltoztatja a tesztelést is. Ennek kezelése, hatásainak leírása (például a tesztelési

[15] Megjegyezném: én nagyon örülnék, ha ez lenne a legnagyobb kockázat egy adott projekten belül. Sajnos ez a legutolsó helyek egyikén szokott helyet foglalni…

ciklusok számának növekedése, a teszt tervezésének idejének megnövekedése, stb.) jelenik itt meg.

15. <u>*Rendszer oktatási terve*</u> – a fejlesztés és tesztelés alatt jó esetben igen nagy mennyiségű dokumentáció keletkezik. Ezekből a dokumentumokból – és még más egyéb dokumentumokból is – állítja össze az üzleti elemző az adott megoldás oktatási anyagát. Optimális esetben ez az anyag a felhasználói átvételi teszt (user acceptance test) idejére 90%-os készenlétben van. Az említett szakaszra már ki kell „képezni" a felhasználókat a rendszer működésére. Ennek a „kiképzésnek" az ütemezését, vagy legalábbis a hozzávetőleges tervét is tartalmazhatja a teszt stratégia.

„Egy alkalommal megkeresett egy cég fejlesztésért felelős vezetője, hogy segítsek neki az éppen fejlesztés alatt álló egyik termékük minőségének ellenőrzésében. Egyszerűbben fogalmazva: teszteljem le a terméket. (A termék jelen esetben egy internetes áruház volt viszonylag kicsiny termékpalettával, de komoly árakkal és komoly technikai és banki háttérrel.) Igen hízelgő volt rám nézve, de egyúttal óvatosságra is késztetett. Az első kérdésem az volt, hogy mikorra kellene a tesztelést befejezni? A válasz sajnos igen lehangoló volt:

- A tervek szerint a következő hónapban indul az egyik akciónk, amiket csak ezen a webáruházas felületen vehetnek igénybe leendő vevőink. Azaz egy hónap múlva indulnunk kell!

- De a termék, azaz maga a virtuális áruház már tesztelhető állapotban van, igaz? – jött a következő kérdésem.

- Nos, sajnos még nincs… - ismételten nem a leginkább kellemes válasz. Szó szót követett és kiderült, hogy az előző programozói gárda abbahagyta a munkát és egy új csapat vette át, akik ráadásul teljesen elölről kellett, hogy kezdjék, mivel nem volt a fejlesztésekről semmi dokumentáció. Ebből fakadt az is, hogy jelenleg sem készül semmiféle dokumentáció, mivel nincs ideje a csapatnak a dokumentálásra.

Kértem két nap gondolkodási időt. Ezt a két napot arra használtam fel, hogy egyrészről átnéztem a piacon már működő, hasonló szektorban tevékenykedő internetes áruházakat, azok hátterét. Próbáltam ismerősöket keresni a fejlesztők között, és ha találtam, felvettem velük a kapcsolatot.

Másrészről végignéztem az ilyen típusú honlapok tesztelési technikáinak legújabb vívmányait. Konzultáltam szakmabeli barátaimmal, ismerőseimmel, akik már foglalkoztak hasonlóval. Ez utóbbiaktól az időbeliségre, a tesztelési idő szükségleteiről próbáltam információt gyűjteni.

A két nap elteltével újra találkoztunk, s további kérdésekkel bombáztam a megrendelőt: hány tesztelési ciklust terveztek/enged a projekt? Hány fővel számolhatok a tesztvégrehajtás során? Van-e tényleges tesztkörnyezet? Mennyire integrált a rendszer? Lesz-e más, például SAP vagy egyéb vállalatirányítási rendszerrel összekötve? Stb. A fél órára tervezett megbeszélésből egy egész estés diskurzus lett, aminek az eredménye egy majdnem teljesen elkészült teszt stratégia volt – hozzá kell, hogy tegyem, hogy sikerült egy plusz hónapot kicsalni a megrendelőből.

A következő napok intenzív tervezéssel, stratégiák építésével telt. Számos konzultáció, egyeztetés más szakmabeliekkel, megannyi szakkönyv és folyóirat böngészése, tervek és stratégiák felemelkedése, majd bukása…

Egy héttel később – ennyi időt kértem a teszt stratégia elkészítésére – újra találkoztunk és a kész stratégiát átadtam a megrendelőnek. Először nem értette, mi is az a dokumentum, de egy pár perces bemutatás, magyarázat után már minden tisztának tűnt számára is. Megnyugtató volt, hogy elégedett volt a munkámmal. Természetesen itt még nem ért véget az én feladatom, de mivel mindent előkészítettünk maga a tesztelés már nem hozott sok meglepetést. Köszönhető mindez annak, hogy a stratégiát a fejlesztő csapattal karöltve készítettem el támaszkodva az ő rendszerbeli tudásukra és a folyamatos, a fejlesztés során előjött gyenge pontokra. Végül sikerült a tervezettnél 2 héttel korábban – tehát a projekt tervei szerint 1 hónap helyett csak 2 hetes csúszással – a terméket átadni és a webáruházat elindítani. Ez alkalommal a dinamikus és a konzultációs stratégiai modellek ötvözetét alkalmaztam – bátran mondhatom, hogy sikerrel. „

(Norbert J. Mucsina, 2014.)

De – mint a legtöbb elméletben megalkotott dolog – a gyakorlatban ez eléggé ritkán alakul ki ilyen tisztán. A legtöbbször egy belső tesztelői csapat dolgozik együtt a programozókkal/fejlesztőkkel, nem

ritkán ez utóbbiak közül válik ki, alakul ki a tesztelői gárda. Éppen ezért a teszt stratégia a legtöbb esetben – igen finoman szólva – nincs megfogalmazva, vagy csak elméleti síkon létezik.

Szerencsésebb a helyzet, ha ennek a belső tesztelői csapatnak a vezetője egy nagy gyakorlattal rendelkező, esetleg képzett, profi tesztmenedzser. Ilyenkor csak a kezdeti szakaszban, az első 4-5 projektnél lesz nehézkes a teszt stratégia megfogalmazása, bár inkább annak betartása, ahhoz való alakulás, idomulás. A teljesen rutinszerű működéshez egy újonnan formálódott belső tesztelői csapatnál – egy profi vezetővel – átlagosan 8-10 projekt teljes végrehajtása szükséges. (Persze csak abban az esetben, ha a csapatösszetétel minimális mértékben változik a projektek alatt.)

Amennyiben a tesztelést egy külső, teljesen független cég képviseli, a helyzet – és a teszt stratégia – még jobb. Független tesztelői csapat esetén szükségszerű és szinte létfontosságú a teszt stratégia írásba foglalása és mindkét/három/több fél általi elfogadása, majd annak folyamatos és pontos betartása. Ennek hiányában nehéz lenne a tesztelői csapatnak bebizonyítani, hogy a megfelelő munkát végezte el a megfelelő módon. Még nehezebb lenne a végrehajtott munka anyagi vonzatát rendezni…

3.2 Teszttervezés és becslés

„Egy kellemes, tavaszi péntek reggelen – miután már túl voltam a rendszeres edzésemen, meg a reggeli kávémon – éppen vidáman ültem le a laptopom elé gondolván: 'Ma csupa jó dolog történhet velem, hiszen az idő gyönyörű, péntek van, dolgom alig.' Kedélyesen megnyitottam a levelezésemet, meg mellette párhuzamosan még vagy nyolc másik alkalmazást – mint minden reggel, majd elindultam a napi második adag kávémért. Szórakozottan tettem csészémet a kávéautomata megfelelő szegletébe, megnyomtam a megfelelő gombot, s vártam a kávéőrlő számomra oly' kedves zúgását… De hiába: az automata meg sem szólalt. Csak villogott… Pirosan…

Nosza, szokásos karbantartó munka: vizet tölteni, kávébabot pótolni, zacctartót üríteni… De a piros fény csak villog… Röpke 25 perc bütykölés, rimánkodás és egyéb, nem éppen kulturált szavak használata után inkább a büfé felé vettem lépteimet, kissé morcos idegállapotban. Utam az asztalom mellett vitt el, ahol rápillantottam a laptopom kijelzőjére – éppen újraindult egy „tervezett", központi „frissítés" után. Csak nyeltem egyet, nem szóltam semmit, bár az idegállapotom kezdett a morcosból a mérgesbe váltani.

A büféhez érve idegállapotom nem javult, hanem lassan-lassan a haldoklás öt fázisából a harmadik (alkudozás) fázisba kezdtem lépni, mivel a büfés hölgy kedvesen tájékoztatott: kifogytak a friss kávéból. 'De legalább kutyulós, porkávétok van?' – fogtam könyörgőre a dolgot, de mind hiába. Azt nem tartottak…

Nagyot nyeltem, kihúztam magam, majd elindultam vissza az asztalomhoz, ahol a monitoromon a BSOD[16] nevű förmedvény várt engem… Egy villanás erejéig… Majd 20-30 másodperc múlva újra… (4. fázis – depresszió) 'Ennyi' – szóltam – 'én megyek haza!' Összecsuktam a laptopot és elkezdtem pakolászni, hogy hazatérjek és megnyugodjak.

Ebben a lelkiállapotban talált rám egyik, kissé zilált külsővel rendelkező, de egyébként is zaklatottnak tűnő kollegám.

- *Szia – köszönt illedelmesen.*
- *Szia – morogtam vissza, mindenféle finomkodást mellőzve – megyek haza, úgyhogy mondd gyorsan.*
- *Lenne rám egy-két perced? – mivel eléggé csúnyán nézhettem rá egy kissé megszeppent. Láttam, hogy fürgén körbepillant az asztalomon, s mivel ismert engem eléggé, hirtelen elmosolyodott, s így szólt: – Esetleg egy kávé mellett megbeszélhetnénk? Ne aggódj, nem a büféből – tette még hozzá, bár feleslegesen, hiszen én is ismertem: csak otthoni kávét iszik.*
- *Rendben – mondtam megenyhülve – menjünk.*

Csapot-papot otthagyva követtem kollégámat – az egyszerűség kedvéért nevezzük csak Marcinak – a kávé forrása felé. Útközben (4-5 perc) mindenféle dolgokról csevegtünk, időjárás, család, szomszéd legújabb tettei, stb. Szinte el is feledkeztem arról, hogy valamit mondani szeretne, amiért képes megosztania velem az otthonról hozott kávéját is. Persze, mikor már megfelelően lenyugodtam, az ízletes kávé párája mögül rám pillantott, s belevágott. Ő sem finomkodott:

[16] Blue Screen of Death, népszerűbb magyar nevén: Kékhalál.

- *Figyelj, a segítséged kell. – kezdte Marci – Nagy bajban vagyok, s remélem, hogy te tudsz segíteni.*

- *Na, mesélj – mondtam. Legalább meghallgatom, ha már ilyen finom kávéval kenyerezett le, gondoltam, majd meglátjuk, tudok-e segíteni.*

- *Tudod – fogott bele Marci – van ez a projekt. Tudod, az, amin már fél éve dolgozok.*

- *Igen, ismerem – mondtam –, bár részleteket nem tudok. Úgy tudom, hogy eléggé rendben van. Talán nemsokára be is fejeződik, nem?*

- *Nos, ez itt a baj… Nincs minden rendben…*

- *Hogyhogy? – kérdeztem – Mi a baj?*

- *A tesztelés… – nyögte csendben.*

- *Mi a baj a teszteléssel? – kérdeztem gyanakodva.*

- *Hát az a baj a teszteléssel – mondta nagyon lassan és egyre halkabban – hogy nincs…*

Megdöbbentem… A szóban forgó projekt nem volt túl nagy, de ahhoz azért eléggé bonyolult volt, hogy megdöbbenjek a tesztelés hiányán. Csendben kavargattam a kávézaccot a csészém alján (mivel a kávét már megittam) s pörgött az agyam, hogy vajon mit kellene tennem? Hogyan tudnék én ezen segíteni?

Óráknak tűnő másodpercek után Marci megszólalt:

- *Vállalod?*

- *Mit? – kérdeztem gyorsan.*

- *Hát – nézett rám könyörgően reménykedve – a tesztelést. Te… Meg a csapatod… Ugye?...*

- *Persze, megnézzük, mit tehetünk – sajnáltam meg Marcit. No, meg azért be kell, hogy valljam, izgatott is a feladat. Éppen nem volt sok dolgom, sem projektem, a*

csapat is felszabadult, így akár segíthetünk is. A szívességbank-betét meg mindig jól jön… – Na, mesélj, hogy állnak a dolgok?

Az ezt követő két órában a projektről beszélgettünk, mindenféle kérdéseket tisztáztunk. Kiderült, hogy tulajdonképpen már szinte kész az összes kód, mind a 4 programozó gőzerővel dolgozik a maradékokon. Volt némi tesztelés is, hiszen az egységtesztet a programozók megcsinálták, meg Marci is nézegetett dolgokat. Viszont szinte alig vannak ezek – sőt! Az egész projekt alig van – dokumentálva. No, ez már felvetett egy kevés problémát. Mivel a laptopom éppen kék alapon fehér színben tündökölt (pedig nem is MTK-drukker) visszatértem a régi, jól bevált papír+ceruza kombinációhoz, s elkezdtük felvázolni, hogy mit is tehetünk. Mi az, ami meg van, és mi kellhet még. Röpke másfél óra alatt kitisztult, hogy kinek mik az elkövetkezendő napokra a feladatai, miket kell átküldeni, átnézni.

Az elkövetkező napok feszített munkatempóban teltek minden oldalról: Marciéknak létre kellett hozniuk a projekttervet, a követelményeket kellett rendszerbe foglalniuk és átküldeniük nekem, valamint a már elvégzett teszteléseket (tesztesetek, eredmények, talált hibák, eddigi munkamódszereik, stb.). Részemről a tesztelési terv, a stratégia és némi teszteset-gyűjtemény létrehozása volt a feladat. Közbe persze a tesztkörnyezet felállítása, elérhetővé tétele volt még feladat a programozóknak és az adminisztrátoroknak. Nem kevés álmatlan éjszaka – és liternyi kávé elfogyasztása – után végül készen álltunk a tesztelésre. Minden rendben volt, azt a néhány zökkenőt leszámítva – ami a kései bevonásból és a dokumentáció hiányosságaiból származott – sikeresen lezártuk a projektet.

Mint mindig, ekkor is tartottunk egy értékelő-megbeszélést pár nappal az élesre állás után. Marci kitett magáért: ízletes kávé gőzölgött az asztalon, némi rágcsálnivaló is helyet kapott, valamint minden résztvevő előtt toll és papír…

Nem részletezem, mi minden hangzott el akkor, csak egy-két főbb dolgot emelnék ki: mindenki számára egyértelmű volt, hogy sosem szabad a tesztelés előkészítését a legutolsó pillanatra hagyni, hanem ha lehet, akkor már a tervezés fázisában érdemes gondoskodni egy tesztmérnök jelenlétéről. Ennek hiánya egy olyan mértékű kockázat – és anyagi ráfizetés – amit nem minden projekt tud/képes elviselni. Viszont egy –

még ha túl későn is – előkészített, részletes teszttervvel és teszt stratégiával még akkor is megvalósítható a megfelelő tesztelés.„

(Mucsina Norbert János, 2015)

3.2.1 Teszttervezés

A fenti történet sok szempontból tanulságos: először is sose hidd, hogy ha szép a reggeled, akkor jó lesz a napod. Murphy él és munkálkodik! Másodszor: egy jó kávé mindenen segít… Harmadszor: mindig legyen nálad papír és ceruza (vagy toll) – sosem tudhatod.

Na, de komolyan: mindig van egy olyan projekt, ahol éppen a te segítséged kell. Még akkor is, ha későn szólnak. És ilyenkor kell igazán észnél lenni, ilyenkor kell higgadt fejjel gondolkodni.

Projekt a projektben – ebből az alaptételből indultam ki korábban. Most is ezt szeretném tovább boncolgatni.

Minden projektnek van egy terve – vagy írásban, vagy csak szóban, de létezik. Az írásos formája hasznosabb, mert mások is hozzáférhetnek, például a tesztelők. ☺ Így a tesztelésre, annak folyamatára is kell egy tervet készíteni. A tartalma nagyon hasonlít a projekttervre:

Teszterv tartalma (ISTQB alapján):

- Projekt, feladat rövid ismertetése
- Felhasznált dokumentumok
- Leszállítandó dokumentumok
- Tesztelés hatóköre (Érintett rendszerek, Tesztelendő funkciók, Nem tesztelendő funkciók)
- Teszt stratégia
- Teszteredmények kiértékelésének folyamata
- Tesztfázisok, időzítések
- Döntéshozatali fórumok
- Elfogadási kritériumok (Elfogadási kritériumok az egyes fázisokban, Tesztelés felfüggesztésének kritériumai, Tesztelés újraindításának kritériumai)
- Erőforrások (Teszt eszközök, Személyi erőforrás, Tesztkörnyezetek)
- Szerepkörök, felelősök és feladatok
- Tesztadatok
- Kockázatok
- Betanítás, oktatás

Projektterv tartalma (PMF alapján)

- A projekt, feladat rövid ismertetése
- Feltételezések, korlátok
- Munkaelem strukturálás (Work Breakdown Structure) – V-modell esetében
- Product Backlog – Agilis metodológia esetén
- Megvalósítási terv
- Változás-követés és ellenőrzés
- Mérföldkövek
- A projekt fázisai, időzítése
- Függőségek (esetleg más projektektől)
- Belépési, kilépési feltételek
- Költségvetés
- Minőségirányítás
- Emberi erőforrások
- Egyéb erőforrások
- Kommunikációs terv
- Kockázatok elemzése/kezelése
- Hibakezelés
- RACI Mátrix

Mindezek alapján újra megerősítettem az alapfeltételezésemet: a tesztelés projekt a projektben. Úgy, ahogyan a projekt tervét a projektmenedzser, így a tesztelés tervét is a menedzser, jelen esetben a tesztmenedzser készíti el. (Vannak kivételes esetek, amikor egy tapasztaltabb tesztelőre bízzák ezt a feladatot.) Természetesen nem kizárólagosan ő, de ő a felelős azért, hogy a teszterv elkészüljön. (Megjegyzem: a legnagyobb részét jellemzően a tesztmenedzser írja meg.) Jogosan merül fel benned a kérdés: miért ne írhatná meg az egészet

egyedül a tesztmenedzser? Ha egyszer övé a felelősség… – A kérdés jogos, a feltételezés naiv. Ha egyedül írná meg, akkor olyan dolgok kerülnének a tervbe, ami esetleg nem, vagy nem úgy megvalósíthatóak. És nem azért, mert a terv készítője – a menedzser – buta, vagy hozzá nem értő, hanem azért, mert vannak olyan pontok a tervben, ahol mindenképpen kell az adott terület szakértőjének véleménye (pl. biztonsági tesztelés, vagy keresőmotor-tesztelés esetén egy erre specializálódott tesztelő véleménye a szükséges időre vonatkozóan létfontosságú), vagy egyes esetekben a teljes fejlesztői vagy tesztelői csapat véleménye szükséges lehet. Ilyen például a kockázat-analízis, amit egyedül képtelenség jól megírni. Mindenképpen kell a csapat véleménye. Hogy miért? Egyszerűen a „Több szem többet lát, több fej okosabb.” tétel miatt. Itt – azaz a kockázat-analíziskor – ez valóban megállja helyét. Mondok egy példát: adva van egy projekt, ahol a szoftveres megoldás több különálló rendszeren alapul. A különböző rendszerek más és más időzónában, tehát földrajzilag is más területen találhatóak meg. A szervereket felügyelő csapatok szintén szétszórtan helyezkednek el a világban. A tesztmenedzser – és a tesztelői csapat – szemében a szerverek rendelkezésre állása nagy kockázatot rejt magában, hiszen azok koordinálása, az egyes szerverek összehangolt indítása és leállítása több időzónán átívelve finoman fogalmazva is nehézkes. Mivel azonban a teszteléshez a szerverek állandó futása fokozottan szükséges, ez egy nagy kockázat. Ellenben a szerverek üzemeltetői és a projektben résztvevő rendszeradminisztrátorok szemében ez nem kockázat, hiszen nekik van egy jól bejáratott rendszerük ennek biztosítására. A két vélemény teljesen ellent mond egymásnak, mégis mind a kettő releváns. Tehát ha mindkét véleményt figyelembe vesszük, akkor az átlagvélemény nagyjából az lesz, hogy ugyan kockázatos, de nem annyira, hogy külön kellene vele foglalkozni. (Megjegyezném, hogy a csapatnak más tagjai is vannak, akiknek szintén más lehet a véleménye.)

Ez csak egy példa volt, de számos ilyen példát lehetne még felsorolni. Nem csak a kockázat-analízis egy ilyen terület, de akár a tesztelés hatóköre is, vagy az erőforrások, szerepkörök leírásához is érdemes konzultálni, legalábbis a projektmenedzserrel.

Amennyiben végre készen áll a dokumentum, erősen javasolt azt a projektcsapat – legalább a mag (core), a vezetőség – tagjaival azt elfogadtatni, s ennek tényét a teszttervre felvezetni. (Konyhanyelven: alá kell íratni a felelős vezetőkkel a projekten belül.)

Nagyon fontos tény: **a teszterv egy élő dokumentum**, tehát a projekt ideje alatt – bizonyos külső vagy belső események hatására – folyamatosan változhat. És változni is fog – már csak abból kiindulva is, amit már korábban is említettem: Murphy él és munkálkodik. Tehát ami elromolhat, az el is romlik. Mindez azonban nem probléma, hiszen erre való a változás-kezelés és nyomon követés nevű csoda: bármilyen módosítás történik a teszttervben, azt érdemes dokumentálni. Mi, miért, ki, hogyan és mikor változtatott meg, valamint azt ki nézte át és hagyta jóvá? Ja, és mikor? Tulajdonképpen nem is maga a teszterv a fontos, hanem az, hogy mindig a megfelelő kérdést tegyük fel – magunknak is és a csapatnak is – és azokra a kérdésekre a lehető legjobb választ meg is találjuk. Ha kell, akkor döntést hozzunk. A teszterv mindezen válaszok és döntések rögzítése. Bár ezeket a kérdéseket könnyű megbeszélni szóban, tapasztalatból tudom, hogy igen nehéz megfogalmazni azokat; érthető formába önteni mindazon kétségeket, amik bennünk felmerülnek, s gyakran csak hosszas körülírás segítségével tudjuk egymásnak elmagyarázni. Na, EZ az igazi feladat. EZ az, amit a tesztmenedzsernek, - vagy annak a „szerencsés” tesztelőnek, akire rábízták – a legnehezebb és a legtöbb álmatlan éjszakát okozza.

Mindenképpen szeretnék még valamit elmesélni: sok kérdés felmerül egy tesztelői csapatban – még több a tesztmenedzserben – amik néha kényelmetlenek lehetnek. Hogy kinek? Például a projektmenedzsernek, vagy az üzleti elemzőnek, esetleg a programozónak, vagy a tervezőnek – és még sorolhatnám, kiknek. Mik lehetnek ezek a kérdések? Például:

- van-e üzleti követelmény leírás?
- van-e technikai követelmény leírás?
- van-e az egységtesztekről valamilyen dokumentum?
- naprakészek-e a dokumentációk?
- ha még nem tudni, hogy lesz-e tesztkörnyezet, akkor hogyan tervezzünk tesztelést?
- üzleti szempontból mi a projekt sikerességének feltétele?

- mennyi idő van a tesztelésre?
- mekkora erőforrással gazdálkodhatunk?

Egy kezdő tesztmenedzser, vagy egy tesztelő, akire rábízták ezt a feladatot, esetleg egy kissé félénkebb, vagy tapintatosabb ember ezeket a kérdéseket inkább nem teszi fel. **EZ HIBA!!!** Súlyos hiba, kérem! Igen, lehet, hogy egy konfliktust – esetleg többet – az adott pillanatban elkerül az illető, de ezzel lehet – sőt, szinte biztos –, hogy egy sor problémát idéz elő a fejére, zúdít a tesztelői csapat nyakába. Nem! Ilyet nem szabad, nem lehet csinálni!

A KÉNYELMETLEN KÉRDÉSEKET IS FEL KELL TUDNI TENNI!

Nemcsak hogy **fel kell tudni tenni**, hanem **tudni kell feltenni**. Ravasz, mi? Játszok itt a szavakkal, mint Rodolfo a cilinderrel meg a galambokkal? Na, elmagyarázom: nem elég az, hogy felteszed a kérdést. Ha nem megfelelő pillanatban, vagy megfelelő helyen, esetleg módon teszed fel a követelményekre vonatkozó kérdésedet a célszemélynek (projektmenedzser), akkor lehet, hogy végleg keresztet vethetsz a későbbi együttműködésre. Lehet ugyanis, hogy számonkérésként fogja értelmezni. Vagy ami még rosszabb: nyílt támadásként fogja fel. Na, ezek után nem fog segíteni – és még nehezebb lesz a dolgod, mint előtte. Íme, egy nálam jól bevált stratégia:

1. *Ismerd meg az „ellenfeled”*[17] – beszélgess vele sokat. Ne mindig a munkáról, vagy az adott feladatról – néha kérdezz egy kicsit a családjáról, a hobbijáról, esetleg múltbeli tapasztalatairól. Bármiről, ami nem munka.
2. *Találd meg a „zöld gombot”* - Próbáld meg megtalálni azt a pontot, amikor, vagy aminek hatására láthatóan kinyílik. Minden embernek van ilyen pontja. Ezt hívom én „zöld gomb”-nak. Nagyon kell figyelni a testbeszédre is, nem csak a szavakra. Igyekezz elkerülni azokat a témákat, amik kényelmetlenek a számára, vagy nem szívesen beszél róluk. Ezeket a témákat, pontokat nevezem „piros gomb”-nak. Ami mindent leállít és az alany átvált védekező és/vagy támadó módba.

[17] valójában nem ellenfél, csak a szemléletesség miatt nevezem így. ☺

3. *Tálald fel a kérdést, mint egy illatos vadsültet* – azaz rendesen „ágyazz meg” a kényes kérdésnek. Nem kell fejjel menni a falnak. Elég, ha rámutatsz: „ez itt a fal”. Érteni fogja. (Ha jól csinálod, még meg is köszöni.) Nem azt mondom, hogy „hülyére kell venni”, hanem azt, hogy nem kell rámutatni direkt módon a hiányosságokra az adott projekten belül. Ne feledd!: a projekt a projektmenedzsernek a „kicsi babája”. Dédelgeti, szeretgeti – és nem szereti, ha azt mondják rá, hogy csúnya…
4. *Használd fel, amit megtudtál* – **nem** azt jelenti, hogy használd **KI**, amit megtudtál. De ha figyelsz, akkor tudod, hogy mikor lehet olyat kérdezni, ami esetleg kényelmetlen anélkül, hogy megbántanád. Inkább megbeszélitek.

Na, de miért is kell ezeket a kérdéseket feltenni? Ahogy azt már említettem, lehet, hogy egy-két kellemetlen percet, vagy konfliktust elkerülsz. „De hát az jó, nem?” – jöhet a kérdés. Igen, egyrészről jó. De gondolj csak egy kicsit bele: elkerülsz egy kis konfliktust a projektmenedzserrel, mert nem akarod megkérdezni tőle, hogy mikor szeretné pontosan megfogalmazni a követelményeket. (Mikor fogja eldönteni, hogy kék vagy hupilila gombot szeretne piros betűkkel és neon körvilágítással?) Ez eddig rendben van. DE! Pontos követelmények nélkül hogyan tudsz egy pontos teszttervet, vagy teszt stratégiát elkészíteni? Esetleg a tesztmérnököd a csapatodban hogyan fogja megírni a teszteset-gyűjteményt? Hogyan fogtok kidolgozni egy megközelítőleg 100%-os lefedettséget? Én megmondom neked: SEHOGY! És ez további konfliktusok forrása lehet a projektmenedzserrel szemben, de még a csapatod tagjaival szemben is… És akkor még nem beszéltünk a stresszről és a lehetséges plusz munkákról és költségekről, amik ebből fakadhatnak… Amit persze valakin számon fog kérni a megrendelő…

Na, **EZÉRT** kell felvállalni és feltenni a rizikós kérdéseket is. Igen, ehhez egy jó nagy adag emberismeret és szinte profi konfliktuskezelési készség kell. Ne feledjük: kérdéseinkkel, kéréseinkkel befolyásoljuk a teljes fejlesztési folyamatot.

3.2.2 A becslés

„Pár évvel ezelőtt voltam olyan szerencsés, hogy sikerült egy kis házat vennem a párommal vidéken. Szép ház, nagy telek – kert kialakítva. Öröm és boldogság.

Miután rendbetettem a portát (értsd: egészségügyi festés, padló felcsiszolása és újralakkozása, ablakkeretek és ablaktokok újrafestése, valamint nagytakarítás) és beköltöztünk – május környéke volt – azon kezdtem el gondolkodni, hogy mit kezdjek a kerttel.

Egyik reggel kimentem a kertbe és megrémültem attól, amit láttam: rengeteg fa, amit nem metszettem meg, cserébe már szép zöld levelekbe burkolódzott, némelyik már el is virágzott, szőlőtövek, meg derékig érő gaz. Percekig csak némán bámultam, majd elővettem a kerti szerszámokat és nekiveselkedtem, hogy legalább a gazt kitakarítsam, levágjam. Ekkor köszönt át a szomszédasszony:

- *Jó munkát!*
- *Köszönöm, megvan – feleltem – amint látod lesz is egy darabig. – majd széles mosollyal arcomon a gazos kertre mutattam. Büszke voltam arra, hogy 'művelem' a kertet… Azaz: elkezdtem legalább gaztalanítani…*
- *Ez mind ki akarod kapálni? - érkezett a kérdés. – Nem lenne jobb egy fűkaszával nekiesni?*
- *Nem, jobb lesz ez így, mert így a gyökerét is kiszedem. Nem olyan nagy ügy. Látod, jön ez könnyen – mondtam, s igazamat bizonyítván megragadtam egy jó maréknyi derékig érő gyomot és egy rántással kitéptem a földből.*

A szomszédasszony tamáskodva nézett rám, de nem szólt semmit. Pár óra múlva éreztem, hogy kicsit túlbecsültem magam: nem lesz ez olyan könnyű… A nap végére alig haladtam, mert – mint kiderült – a kert azon része, ahol elkezdtem a munkát, volt a könnyebbik rész. Úgy 4 négyzetméternyi területen. A többi… Nos, arról jobb nem is beszélni. Abban az évben mégis inkább vettem egy fűnyírót, s a kertet csak azzal 'gondoztam'.

A következő évben már hamarabb, Februárban tervezgetni kezdtem. Márciusban elkezdtem felásni a kertet. Gondoltam: ásóval könnyebb lesz, mint kapával. Majd egy-két hét és végzek. Hiszen nem olyan nagy ez a kert, meg tudok is ásni, meg van is jó ásóm… De terveim hamar feledésbe merültek, miután alig kétsornyi ásás és jó néhány fűcsomó meg gyökér-rendszer kézzel történő kihúzási próbálkozásából

fakadó hasra, illetve fenékre esés után már láttam: nem fogom tudni az egész kertet felásni. 'Nem baj, most elég lesz ez a harmad is.' Az alig háromszáz négyzetméternyi területre tekintve újra bizakodni kezdtem. Elvégre kivettem pár nap szabadságot, reggeltől estig dolgozva sikerülni fog. Persze nem így történt: a természet újra kifogott rajtam. Sem egész nap dolgozni nem tudtam, mert 'nem szokta a cigány a szántást', ahogy édesapám szokta mondani, sem pedig a fűcsomók szövevényes gyökérrendszere nem engedte a könnyed munkát. Abban az évben azért csak felástam – igaz, egész nyáron ásogattam valamennyit, mindig csak egy kicsit – a kert harmadát. Be is vetettem mindenfélével: hagyma, borsó, bab, sárgarépa, fehérrépa, burgonya, cukkini, patisszon, sütőtök... Csupa olyan, ami mindennap kellhet a háztartásban. Legalább annyi hasznom legyen.

Persze közben intenzíven kutattam, hogy mit lehetne tenni. Mi lenne az a hatékony megoldás, amivel termőre tudnám fordítani a kertet? Vadabbnál vadabb ötletek merültek fel, miközben kicsiny veteményemet kapáltam nyáron, majd ősszel, amikor a termést takarítottam be. De akkorra már volt egy ígéretes tervem: a kert gazos-gyomos részét totális gyomirtásnak vetem alá, majd a következő tavaszon egy kapálógép és egy eke segítségével majd feltöröm a parlagon heverő földet. A gyomirtást sikeresen elvégeztem késő ősszel, majd intenzív kapálógép-keresésbe kezdtem. Hirdetések és üzletek kínálatainak böngészésével telt el az október-februári időszak. Márciusra már világos volt, hogy nem találok megfelelő gépet – marad ismét a kézi ásás. Mihelyst bírta az ásó a földet ki is mentem a kertbe, s nekiláttam a munkának. Az előző évek tapasztalatából tudtam, hogy nem lesz könnyű munka, s nem is lesz rövid idő. Az ásót kikalapáltam, bekészítettem a gereblyét, vizet, zenét. Reggel 6 órakor már ástam… 9-kor reggeli a családdal, majd beszélgettünk kicsit, tervezgettük a hét hátralévő részét. Dél környékén ismét ástam – s a szomszédban a szomszédasszony is ugyanennek a testmozgásnak hódolt. Rövid beszélgetések, két ásónyom közötti 'hogyvagyénjólmivanveletek-merrevoltatokahétvégén?'-típusú tiszteletkörök. Nem nagyon cifráztuk, hiszen mindketten haladni akartunk. Pár órával később azonban éppen mindketten pihenőt tartottunk.

- Mennyi részt fogsz idén be termőre? – kérdezte a szomszédasszony. Halvány mosoly játszadozott arcán – gondolom ismét tamáskodva várta fellengzős, meggondolatlan válaszom. Azonban csalódnia kellett, mert azért néha én is tanulok:

- Még nem tudom – mondtam csendesen – amennyit bírok. Ha jól forgatom, akkor annyit biztos, mint tavaly. Ha sikerül, többet.

- De már nem olyan gyomos – vigasztalt – biztosan több is lesz, mint tavaly. Hiszen nézd, most is milyen jól haladtál.

Ránéztem, majd a felásott területre: szó ami szó, tényleg többre mentem most egy nap alatt, mint tavaly egy hét alatt. De tudtam: ezt a részt már tavaly felástam. A következő szakasz nehezebb lesz… Ahhoz jóval több erő, meg idő kell…"

(Norbert J. Mucsina 2015.)

A teszttervnek van egy-két olyan pontja, ahol adatokat kell(ene) megadni. Mégpedig a lehető legpontosabb adatokat. Ilyen például a tesztfázisok, vagy ciklusok, a mérföldkövek, személyi erőforrások, stb. Ezeknek a pontos megadása szintén esszenciális fontosságú a tesztelésre, és így a projektre nézve is. (Nyugodj meg, ugyanezzel a gonddal küzd a projektmenedzser is, amikor a projekttervet készíti elő.) Namármost jogosan kérdezheted: „Hogyan adjam meg ezeket az adatokat? Nem láthatok a jövőbe!!!..."

Igazad van. Hacsak nincs egy varázsgömböd, ami véletlenül éppen Nostradamus lelkét tartja fogva, akkor nincs esélyed pontosan megadni. (Ebben az esetben is csak akkor, ha tudsz ófrancia nyelven és el is tudod magyarázni Nostradamus-nak, hogy mit is szeretnél. Meg, ha hajlandó és tud segíteni. Ő sem tudott mindent…) Ahogy említettem: nincs esélyed pontosan megadni – de megközelítőleg meg tudod becsülni! És ez már eredmény! Örvendünk!

Szerencsére nem te vagy az első ember, akinek ilyen becslést kell végrehajtania. Könyvek és tudósok garmadája foglalkozott, és foglalkozik a mai napig ezzel a kérdéssel: hogyan lehet a lehető legpontosabban megbecsülni bizonyos elemeket? Rengeteg módszert, képletet és technikát kidolgoztak – némelyik nagyon hatásos és pontos, némelyik kevésbé. Ami biztos: nincs két olyan projekt, ahol egy adott technika ugyanazt a pontosságot mutatná. Nézzünk meg egy pár technikát, hogy lásd, miről is beszélek.

3.2.3 WAG és SWAG – a két mágikus módszer

A két legelterjedtebb módszer. Sajnos… Valamiért úgy érzem, hogy magyarázkodnom kell… Oké, kezdem.

Vegyük először is a WAG-módszert. A „WAG" egy betűszó. A teljes jelentése: **W**ild-**A**ss **G**uess. Nyersfordításban: vadszamár találgatás. Magyarosan fogalmazva: hasra ütés szerű találgatás. Igen, van ilyen és tudományosnak is nevezik. Mert néha bejön… (Biztosan annál az illetőnél van Nostradamus…) Sokszor tapasztalt menedzserek is használják (direkt írtam menedzsereket és nem korlátoztam le a tesztmenedzserek körére, mert minden menedzser használja), legfőképpen akkor, ha igen rövid idő alatt – mondjuk percek alatt – kell valami számot mondaniuk. Ilyenkor gyorsan felvillannak előtte a lehetőségek, megpróbál visszaemlékezni előző projektekre, felmérni a jelenlegi nehézségét, kalkulálgat, de végül is mindenféle tudományos alap nélkül csak bemond egy számot, értéket. Majd reménykedik, hogy helyesen tippelt.

A SWAG egy kicsit más – bár számomra talán egy kicsivel rosszabb fényben tűnik fel. Ennek a betűszónak a jelentése: **S**cientific **W**ild-**A**ss **G**uess… Igen, tudományos hasra ütés… Mitől lett hirtelen tudományos egy hasra ütés? Jogos a kérdés – hadd fejtsem tehát ki a kéz szempontjából, amely ráüt a hasra: az előző esetben csak elindul a kéz egy állapotból/helyzetből, majd addig gyorsul, míg érintkezésbe nem lép a has külső burkával, azaz a hasat takaró bőrfelülettel, esetleg ruhaneművel, aminek hatására csattanó hangot hallunk.

A tudományos hasra ütés esetén a kéz helyzeti energiáját (E_h) átalakítjuk mozgási energiává (E_m), melyet a mozgás irányától függően v_0 kezdő sebességgel és $a=x$ gyorsulási értékkel számolva F nagyságú erővel találkozik $A=tenyér$ méretű felületen a has hámszövettel (esetleg textillel) fedett felületével, melynek hatására $F_{ny}=z$ nyomóerő keletkezik. Ezen nyomóerő a két felület közötti $V=y$ mennyiségű levegőt A felületen présel ki, melynek hatására $\lambda=c/f$ hullámhosszú, v rezgésszámú hangot kapunk eredményül.

Ugye, mennyivel tudományosabbnak hangzik? És mégis ugyanazt jelenti… Éppen ezért nem javaslom egyik módszer használatát sem, amennyiben ez megoldható. Tudom, hogy lesznek olyan helyzetek, amikor az ember gyermeke rákényszerül ezek használatára, viszont azt javaslom, hogy ilyen helyzetekben mindig hangsúlyozd ki: ez csak hozzávetőleges becslés. Pontosabb adatokat később tudsz biztosítani.

De térjünk át a komolyabb becslési technikákra.

3.2.4 Súlyozott átlag (Weighted Average)

Hárompontos technikának vagy statisztikai technikának is hívják ezt a módszert. Önmagában nem alkalmazható, mindenképpen szükség van előző projektek tapasztalataira, esetleg egy specialista bevonása az adott területről, aki megfelelő mennyiségű és minőségű tapasztalattal rendelkezik. Mindjárt ki is fejtem, hogy mit értek ez alatt.

Minden feladat, azaz megbecsülendő érték esetében három fő szempontot kell megvizsgálnunk:

1. Legjobb esetben (Optimista)
2. Normál (elfogadható) esetben (Normál)
3. Legrosszabb esetben (Pesszimista)

milyen és mennyi erőforrás (idő, ember, eszköz, pénz) szükséges a végrehajtásához. A képlet a becsléshez pedig így néz ki:

$$E = \frac{\text{Optimista} + (4\text{xNormál}) + \text{Pesszimista}}{6}$$

Ahol

- E = a becsült érték, azaz a kérdésre adandó válasz maga;
- Optimista = a legjobb esetben, amikor minden megy, mint a „karikacsapás”;
- Pesszimista = ami elromolhat, az el is fog romlani;
- Normál = amikor minden működik kisebb-nagyobb nehézségekkel, de még elfogadható.

Amint az látszik, a három alappillér értékét valamilyen becslés alapján kell megadni. Nincs rá pontos adat. Említettem, hogy statisztikai technikának is nevezik – innen jöhetnek elsősorban az alapadatok a három pillérhez. Előző projektek tapasztalatai alapján – már ha volt dokumentálva – már igen pontosan meg lehet adni a három számot.

A másik lehetőség, ha egy tapasztalt szakember véleményét kérjük az adott témakörben. Olyan szakemberét, aki számos hasonló projektben vett már részt, így meg van a „magához való esze”. (Majdnem olyan, mint a statisztika, csak nem papíron rögzített adatokkal dolgozunk, hanem egy szakember memóriájából előhúzott, s természetesen némileg módosított adatokon.) A tapasztalatok alapján ez a képlet adja a legjobb becslést minden projekt esetében. Használatát erősen ajánlom – tapasztalataim alapján szinte tökéletes adatokat ad vissza. Természetesen ez a módszer sem

tévedhetetlen, hiszen a három tényező (optimista – normál – pesszimista) megállapítása szintén csak becslés. És a kedvenc mondásom itt is valós és érvényes: Murphy él és dolgozik…

3.2.5 Delphi – a nem-programnyelv

Ha te egy programozó vagy, kedves olvasó, akkor most lehet, hogy elmosolyodsz: hogyan lehet egy programozási nyelv nem programozási nyelv? Hát így. ☺ Ennek a technikának ugyanis csak a neve hasonlít a Borland cég által 1995-ben kifejlesztett és kiadott fejlesztői környezetére – tartalmában teljesen más.

Ez a módszer a tesztelő csapatban dolgozó szakemberek, specialisták múltbéli tapasztalataira – és nem kevésbé a memóriáira – épít. A felmerülő kérdések megbecsülendő értékét mindig az adott szakembertől kérik meghatározni. Ha szerencsés a tesztmenedzser, akkor van egy ilyen szakember a csapatában. (Ha meg burokban született, akkor több is.) Ebben az esetben nincs más dolga, mint leülni az adott szakemberrel egy finom kávé/tea/bármi társaságában és elbeszélgetni a kérdéskörről. Akár többször is. Ami fontos: ne csak a kérdést tegyük fel. Szóval ne így nézzen ki a feltett kérdés:

- *Szia Pista[18] bátyám. Lenne itt egy projekt és a kérdésem az lenne, hogy hány tesztelő kell a végrehajtásához?*

Erre ugyanis a legtöbb esetben egy WAG-típusú számot fogsz kapni. Jobb esetben egy SWAG-számot. Ugye nem kell részleteznem ennek „hasznosságát”?

Mindenféleképpen meg kell határozni tehát a projekt jellegét, annak célját, a szoftver különböző specifikumait (például: kapcsolódik-e más rendszerekhez, milyen fázisban van a dokumentáció, mekkora lefedettséget kell produkálni, stb.). Szóval a lehető legnagyobb részletességgel érdemes elmondani mindent, amit a projektről tudunk. Ha van, akkor a projekt dokumentációját is érdemes átadni. Ja, és még valami: ne várjunk azonnali választ. Hagyjuk, hagy olvasson, hagy emésszе meg szegény „Pista bátyám” azt a rengeteg információt, amit a nyakába zúdítottunk. Nagy felelősség ám meghatározni, hogy miből mennyire lesz szükség… Nem kell nagyon sürgetni, inkább bízzunk meg benne. Adjunk elég időt a válasz megfogalmazására. Ha pedig megkaptuk a választ, nem érdemes vitatkozni vele. Elvégre ő a szakértő, nem mi. Ezért kértünk segítséget, nem?...

Mintegy utóiratként azért megjegyzem: néha felül kell bírálni a szakemberek által adott értékeket, de azt csak körültekintően szabad megtenni. Tudott dolog, hogy minden projekt erőforrás-hiánnyal küzd. Néha

[18] Elnézést kérek az István nevű olvasóktól – nem célzás akart lenni.

ennek érdekében kell módosítani a szakemberek által megbecsült értéket, de ilyen esetben is javasolt az eredeti értékeket megmutatni a felelős vezetőnek, s megindokolni a módosításokat, azok mértékét, valamint felhívni a figyelmét a módosítások hatásaira – akár pozitív, akár negatív az.

3.2.6 Összehasonlító becslés (Comparative vagy Historical Estimating)

Nevéből fakadóan valamit valamivel össze fog hasonlítani és abból szűr le következtetéseket. (Jajj, de okos vagyok, igaz?) Igen ám, de nem mindegy, hogy mit hasonlít össze mivel. Zárójelben – angolul – megjegyeztem, hogy van ennek a technikának egy másik neve is: történelmi becslés. Azaz valahol a történelmi eseményekre épít. Na, ne gondold, hogy valahol Napóleon, vagy Szent István környékén kell keresni a gyökereket – bár akár még az is előfordulhat, én mindenesetre ezt az aspektusát ennek a technikának nem ismerem – de én arra gondoltam, hogy a már lefutott projektek dokumentációjából merített adatokat használja fel ez a technika.

Letisztázva a tényeket: ez a technika a már végigvitt, és jól dokumentált (na, ez utóbbi az, amiért ritka ennek a technikának a használata) projektek adatait használja fel a következő, még el nem kezdett projekt erőforrás-igényeinek meghatározására. Feltétele: olyan projektadatok összessége, amik nagymértékben hasonlítanak a jelenleg futó (jobb esetben még csak tervezés alatt álló) projekthez. Amennyiben legalább egyet találsz, már sikeresnek mondhatod magad. Annak adatait felhasználhatod, az abban a projektben történtekből tanulhatsz, annak a tapasztalatait beleforgathatod a projektedbe.

Mivel a multinacionális cégek belső tesztelői csapatánál igen ritka – és nem csak a tesztelői csapatnál – a pontos és nem kozmetikázott dokumentáció, így ez a technika csak egy külső, projekttől független tesztelői csapatnál működhet jól, ahol pontos és valós dokumentálás folyik.

Mint minden más modellek, az előzőekben említett becslési modellek sem állják meg a helyüket önállóan. Valamely másik modellel/technikával ötvözni kell – például a Delphi-t, mert nincs minden területre szakember, mondjuk az összehasonlítóval. És így tovább. Nem szégyen keverni, vagy többféle becslési módszert felhasználni annak érdekében, hogy a lehető legpontosabb számokat kapjad kézhez. (Megjegyzem: a WAG és a SWAG módszereket azért ajánlatos kerülni, vagy ha mégis használjuk, akkor legalább legyünk tudatában, hogy az milyen pontosságú lehet. És ezt a projekt csapatnak is mondjuk meg.) Óvakodj azonban attól is, hogy túl sok modellt/technikát alkalmazz egy adott projekthez. Az nagyon elbizonytalaníthat, valamint túl sok idődbe fog kerülni – nem biztos, hogy megéri…

3.3 Tesztmonitorozás, metrikák

„Egy alkalommal az Egyszeri Projektmenedzser megkereste az Egyszeri Tesztelőt, hogy megtudakolja, milyen állapotban van a projekt. Tulajdonképpen: mennyi idő van még hátra a tesztelésből. Az Egyszeri Tesztelő nagyon szívélyesen fogadta asztalánál az Egyszeri Projektmenedzsert, kávéval, teával és süteménnyel kínálta, leültette, majd kedélyes csevegésbe kezdtek. Miután az Egyszeri Projektmenedzsernek igen szoros volt a napirendje, így szólt:

- *Azért kerestelek meg most, hogy megtudjam, hány százaléka van készen a tesztelésnek?*

Az Egyszeri Tesztelő másodpercekig nem szólt semmit. Kívülről úgy tűnt, hogy éppen végigpörgeti maga előtt, hogy tulajdonképpen hogy is állnak, és egy valós, lehető legprecízebb számot próbál kalkulálni. De az Egyszeri Tesztelő fejében a következő gondolatfolyam hömpölygött keresztül:

»De jó, hogy a múlt héten végig túlóráztam – bár még így sem látom a végét. A könnyű részét ugyan már megcsináltam, de ott van még az a fránya bejelentkezési jogosultság-kiosztási LDAP-os macera. No, azt még nem is tudom, hogyan teszteljem. De ezt nem kötöm az orrára – még a végén kirúgnak. Vagy a nyakamba akasztanak valakit. Neeeem, nem ettem én meszet. Na de akkor mit mondjak? Mondjam azt, hogy készen vagyok? Azt nem tehetem, mert nem vagyok kész. Még csak az kellene, hogy rögtön kérje az eredményeket is! Akkor lennék én csak igazán bajban!

Szóval mit is tehetek? Az adatbázis letesztelve, a felület letesztelve, az LDAP maga letesztelve, beviteli mezők letesztelve, fordítások rendben… Van vagy harminc teszteset-gyűjtemény is…Tesztriportok, hibajegyek… Csak ez a bejelentkezés ne lenne… Igaz, hogy a tesztesetek számát tekintve ez nem lesz túl sok, de… Igen, ez nagyon bonyolult lesz. Ha azt mondom neki, hogy még csak a felével végeztünk, akkor biztos, hogy megkérdezi majd, mit is csinálok én itt… Azt nem mondhatom, hogy készen vagyok, mert az meg nem igaz… Ha azt mondom, hogy majdnem készen vagyok, akkor biztosan rákérdez majd, hogy mit is jelent ez? Mit mondjak neki? Hogyan tudnám ezt neki bebizonyítani? Hoppá! – itt egy szellemi homlokon csapás

jelent meg – majd azt mondom, hogy 90%-ban készen vagyok. Ezt meg is tudom mutatni, hiszen a több ezernyi már végrehajtott tesztesetek mellett a bejelentkezés párszázas teszteset-gyűjteménye kb. 10%-ot tesz ki. Azt meg, hogy azok végrehajtása sokkal több idő, no azt nem kell tudnia…«

Igen, ezek a gondolatok pár másodperc alatt végigfutottak az Egyszeri Tesztelő agyán, majd így szólt:

- Kérlek szépen a tesztelés 90%-a készen van. Szeretnéd látni a számokat is?

- Nem szükséges kérlek – udvariaskodott az Egyszeri Projektmenedzser – megbízom benned. Akkor a héten el is készülsz, igaz?

- Remélem – nyelt egy nagyot az Egyszeri Tesztelő – bár azt meg kell mondanom, hogy ez a folyamat eléggé komplikált, szóval, ha itt találok egy hibát, akkor annak javítási és újratesztelési ideje sokkal több, mint azt gondolnád…

- De még mindig áll a 90%-os készültség? – kérdezte gyanakodva az Egyszeri Projekt Menedzser.

- Természetesen – vágta rá dacosan az Egyszeri Tesztelő – nézd csak meg a számadatokat…

- Nem, tényleg nem szükséges… Báár… - húzta el az „á”-t akkurátusan az Egyszeri Projektmenedzser, minek hatására az Egyszeri Tesztelő gyomra vad vitustáncot kezdett járni – Azt hiszem, hasznos lenne, ha mégis átküldenéd. De csak a tesztesetek száma és állapota érdekel.

Az Egyszeri Tesztelő felsóhajtott, hiszen azokat az adatokat minden további nélkül át tudja küldeni. Az Egyszeri Projektmenedzser pedig megnyugodva sétált vissza asztalához, majd részletes jelentést készített felettesének. Miközben a jelentést írta, imígyen elmélkedett: »Ha a tesztelés ilyen jól áll, akkor én nem írhatom, hogy sok van még hátra, pedig kellene még egy kis idő. Kár, hogy nem sikerült minden követelményt a projekt elején megfogalmazni. Most bajban vagyok… Kell még vagy két hét a fejlesztőknek. Hogyan tudnék időt szerezni?...« Szórakozottan megnyitotta a levelet, amit az Egyszeri Tesztelő épp az imént küldött a megfelelő adatokkal –

majd a homlokára csapott: »Hiszen itt a megoldás!« A levélben ugyanis ott volt, hogy még 1324 tesztesetet kell lefuttatni – ami önmagában nagy szám, mármint mennyiségileg, de valóban, a már végrehajtott esetek számának tükrében elenyésző. Mindenesetre ez a szám már elég indok ahhoz, hogy egy hét csúszást kieszközöljön. Így a számadat és az Egyszeri Tesztelő szava – miszerint bonyolult és időigényes lehet, amennyiben hibát talál a rendszerben – birtokában a következő mondatot fogalmazta meg: »A termék végső tesztelési fázisában több, mint 1000 teszteset lefuttatása szükséges ahhoz, hogy a megfelelő minőségellenőrzést biztosítsuk. Ezen tesztesetek bonyolultságából és hatásmechanizmusából kifolyólag a fennmaradó idő nem elég ezek végrehajtására, ezért további 2 hét tesztelési időt javaslok megadni. Ennek hatására igaz, hogy a projekt határidejéből kicsúszunk, viszont a vevő által kért minőségi garanciát tudjuk biztosítani.«

Ezek után már tényleg megnyugodott, mert tudta, a felettese meg fogja adni a plusz két hetet… Hiszen ő sem akar gyenge minőségű terméket átadni. Az Egyszeri Tesztelő meg majd megmagyarázza…"

(Anominusz, 2015.)

Elérkeztünk ahhoz a részhez, amit én úgy hívok: a „tesztelés tükre". Jogos a kérdés: miért? Annál is inkább, mert ismerjük az adatok feldolgozásának és magyarázatának különböző módozatait, meghatározó indítékait. Ugyanabból az adathalmazból lehet pozitív és negatív képet is festeni egy folyamatról – ahogyan azt az Egyszeri Projektmenedzser tette az előbbi történetben. Csak a megfelelő szemszögből kell nézni/nézetni, magyarázni, látni és láttatni. Ezért is van nagy felelősség egy tesztmenedzseren, ami a metrikákat illeti. De nézzük meg, mit is mérhetünk és hogyan?

3.3.1 A mérés tárgyának meghatározása

Nagyon tudományos cím – na, de valójában mit jelent?

A válasz nagyon egyszerű: meg kell mondanunk, hogy mitől is lesz sikeres a projekt, vagy a tesztelés, majd azt is, hogy ezt hogyan tudjuk bebizonyítani. Magyarán: milyen alapon mondhatjuk azt, hogy elértük azt a célt, amit kitűztünk magunk elé.

Ez ugye például egy maratoni futó esetében nagyon egyszerű képlet:

- A kitűzött cél: lefutni a 42,195 kilométeres távot

- Siker feltétele: a teljes táv lefutása (és nem árt utána azért pár napig még életben is maradni, hogy kiélvezhessük a sikert…)

Egy projekt esetében sokféle ilyen célt meg lehet határozni. Általában a tesztmenedzser a projektmenedzserrel együtt határozzák meg ezeket a célokat. Mint mindenhol, itt is az egyszerűségre kell/érdemes törekedni – nem érdemes túl bonyolult célokat megfogalmazni. Nehézkes lenne mérni a következő cél elérését: projektünk akkor lesz sikeres, ha a projekt hatására a város kulturális élete olyan mértékű növekedést produkál, aminek hatására az ország egyik kiemelkedő zenei központjává minősül 3 éven belül, s így egy nemzetközi kulturális fesztivál lehetséges helyszíne lesz, aminek hatására sikeres pályázatot írhatunk egy újabb, a CIOFF[19] által is támogatott fesztivál rendszeres megrendezésére. (Ha valaki ezt tényleg célul tűzné ki, azt hiszem, komolyan fontolgatnám, hogy vendégségbe hívjam egy olyan intézménybe, ahol csupa fehér ruhás bácsi és néni kedvesen segítené fel rá a „Muß-I”-dzsekit[20], miközben előzékenyen nyitják ki számára élete hátralévő részének kicsiny, ámde annál gumírozottabb szobájának ajtaját…) Ezt a célt egyrészt nehéz lenne mérni, másrészt, ha valaki mégiscsak megtalálná a megfelelő módját, valószínűleg a módszer által generált grafikonokat, görbéket és egyéb vizuális „bizonyítékokat” nem lehetne értelmezni. Vagy legalábbis igen nehézkes lenne…

Szóval, amint már említettem: érdemes az egyszerűség elvét szem előtt tartani. Tipikus célok lehetnek egy projekt életében:

A projekt hatására:

- a honlapunkról érkező megrendelések 25%-os mértékben növekednek;
- az új szoftver forgalma már az első hónapban visszahozza a fejlesztési költségeket;
- a termék előállításának ideje 10%-kal csökken;
- az eladott termékek utáni reklamációk száma 15%-os mértékben csökken;
- a gyártás során 10%-kal kevesebb selejt termék keletkezik;
- a város zöld területei 40%-ban örökzöld növényekből tevődik össze;
- stb.

Na, most nézzük meg, hogy tesztelés szemszögéből milyen mérhető célokat lehet kitűzni:

[19] International Council of Organizations of Folklore Festivals and Folk Arts – Népművészeti és Népi Iparművészeti Fesztiválok Szervezeteinek Nemzetközi Bizottsága

[20] „Muß” – német, „I” – angol; kiejtés szerint olvasd össze és megkapod a megfejtést…

- a követelmények legalább 99,9%-os teszteseti lefedettséggel rendelkeznek;
- a tesztesetek minimum 40%-át automatikus teszteléssel biztosítjuk;
- minden új funkció/modul/egység csak akkor tekinthető késznek, ha nem maradt kritikus, súlyos vagy közepesen súlyos hiba, amit a tesztelők felfedtek;
- a megtalált hibák 100%-a javítva és újratesztelve legyen;
- stb.

Ezen kívül mérhetünk még eléggé sok mindent, ami befolyásolja a projekt életét, a tesztelés – és vele a tesztelők – életét is. Ilyen mérési célok lehetnek a következők:

- A projekt/tesztelés időbeni haladása (mennyire tért el a tervezettől, vagy van időben?)
- A tesztelési lefedettség, esetleg annak hiányosságai
- Tervezett/valós költségek mérlege
- A terheltség, erőforrások használata (van-e túlterhelés, és ha van, mekkora?)
- A kockázatok megjelenése, annak hatásai
- stb.

Amint látható, szinte mindent mérhetünk, monitorozhatunk. Mindenre kaphatunk választ a számokból – csak tudni kell, mi is a kérdés? Na, igen. Ez itt a nagy tudomány: kérdezni tudni kell. És nem akkor, amikor már mérni szeretnénk, hanem még bőven előtte: amikor tervezzük azt az eseménysorozatot, amit tesztelésnek, vagy tesztmenedzsmentnek (esetleg projektmenedzsmentnek) nevezünk. Igen, már a tervezéskor fel kell vetnünk a következő kérdéseket:

1. *Mit szeretnénk mérni?* – Azaz pontosan meg kell határozni, hogy mi az az esemény, amit mennyiségileg szeretnénk mérni. (Erről beszéltem eddig. Azaz írtam…)
2. *Mi az az egység, amiben mérhető lesz?* – Darabszám? Százalék? Esetleg óra? Mérhetjük kilóra? Az egység (mértékegység is!) meghatározása alapvető fontosságú.
3. *Hogyan fogjuk mérni?* – Pontosabban: honnan vesszük majd a megfelelő adatokat? Lesz valami rögzített számlálás? Vagy csak az emberi emlékezet fog segíteni? Van esetleg erre egy szoftveres megoldás? Mindenképpen objektív eszközt érdemes választani. (Ha kozmetikázni kell az eredményeket, akkor meg már úgy is mindegy…)
4. *Milyen értéknél jó/normál/rossz az eredmény?* – Ez talán a legfontosabb rész: egyszerűen megfogalmazva: mikor kell elkezdeni aggódni, vagy éppen harsány örömujjongásba kitörni?

5. *Mit tegyünk, ha … ?* (A pontozott rész tetszés szerint kitöltendő. Néhány példa: ’az eredmény rossz’, ’túl sok a hiba’, ’túl kevés az erőforrás’, stb.) – Ez pedig az a rész, ahol a legtöbb egyszeri tesztmenedzser nem szeret elidőzni, mondván: „úgysem lesz semmi hiba”. Aztán amikor ott a probléma, nem tudja, mit is csináljon.

Nagyon jó! Most, hogy már tudjuk, hogy MIT szeretnénk mérni, monitorozni, szinte készen is vagyunk. Ez már fél siker! A következő kérdés: hogyan szeretnénk mindezeket mérni?

3.3.2 Mérőszámok meghatározása

„Akkoriban még egy művész mozinál dolgoztam, mint pályázatíró és logisztikai menedzser. Sok dolgom volt, hiszen a 2000-es évek elején elég sok ilyen típusú mozi nyílott, viszont a látogatók száma alacsony volt. Mindenkinél, nem csak nálunk. Éppen ezért mindenhol támogatókat kerestünk, források után kutattunk. Nyereményjátékokat hirdettünk, amin mozijegyeket lehetett nyerni, iskolákat kerestünk meg és összefogva velük közös programokat szerveztünk, az iskola tananyagához kapcsolódó filmekből tematikus előadás-sorozatot szerveztünk. Közönségtalálkozóknak adtunk helyet, és ami a legtöbb pénzt hozta: pályázatokat írtunk. Pontosabban: a kiírt pályázatokra adtunk be mindenféle dokumentumot (pályázatot, csak ez olyan furán hangzik: pályázatra adtunk be pályázatot…).

Egyszóval: pályáztunk. Pályáztunk az NKA[21]-nál, az Európai Uniónál, a városnál, a megyénél… Mindegyiknél más és más célokkal, mérőszámokkal, teljesítési ígéretekkel. Kénytelenek voltunk, mert mindenhol kevés volt az anyagi forrás – a fenntartási költség meg magas volt. A pályázatokat jómagam írtam, viszont a számadatokat a mozi teljes személyzete szolgáltatta: három mozigépész, egy jegyszedő és jómagam. Minden nap, amikor előadás volt ott álltunk az ajtóban – mint valami strázsák – és számoltuk a belépőket. Mármint az embereket, akik jöttek és megtekintették az előadást. Egyikünk az egész vetítési idő alatt a bejáratnál silbakolt, hogy ha valaki még bejönne az előadás alatt, akkor arról is tudjunk.

Előadások után a jegyek szortírozásával voltunk elfoglalva: mennyi eladott jegy volt, mennyi bérlet, nyereményjegy, stb. A számokból aztán táblázatokat töltöttünk fel, grafikonokat készítettünk. Ez a folyamat minimum az év elején két hónapig, majd az

[21] Nemzeti Kulturális Alap

év vége felé szintén két hónapig zajlott. Ugyanis az akkori logikánk szerint: a pályázathoz az adatokat az év elején gyűjtjük, mert akkor a legkisebb a látogatószám – viszont a teljesítés igazolásához az év végi adatokat használtuk, mert tapasztaltuk, hogy december környékén szívesebben jöttek az emberek hozzánk egy-egy előadást megnézni. Az is igaz, hogy csak egyféle mérést végeztünk – a pályázatokhoz azonban különbözőeket kellett prezentálni! Na, itt kaptam én enyhe szívdobogást, mikor legelőször kellett a teljesítésről igazolást kiállítanom. (Tulajdonképpen magyaráztam a bizonyítványt, hiszen azt a pályázatot nem én írtam meg, de nekem kellett bebizonyítani, hogy amit vállaltunk, azt teljesítettük.) Szépen sorban haladtam:

1. ***NKA-pályázat.*** *Vállalás: művészeti témájú filmek: 90%, nézőszám minimum a nézőtér kapacitásának 5%-a… Nos, a filmekkel nem volt gond, hiszen szinte csak azt adtunk egész évben – kivéve azokat az előadásokat, amiket az óvodásoknak tartottunk. De a 90%-ot még így is elértük. A nézőszám már problémás volt, mert az átlagos nézőszám 8 fő volt. Ha ez lenne az 5%, akkor ugye – kis matematika – 160 fős a nézőtér. A mi nézőterünk azonban 200 fős volt… Mit lehetne itt csinálni? Napok, éjszakák, nyugtalanság… Egy-egy előadás alatt is ezen morfondíroztam. Majd egyik iskolai előadáskor székeket kellett behozni ahhoz, hogy az iskola minden tanulója le tudjon ülni. És ekkor szinte megvilágosodtam: a nézőtér kapacitása ugyanis valóban 200 fős, de ebbe beleszámoljuk az oldalszékeket és a pótszékeket is – amit viszont a pályázat nem tekint a nézőtér részének (ez a pályázati kiírásban fehéren-feketén le volt írva). Nagykő zuhant le a szívemről – mert hazudni nem szeretek – s újra számolgatni kezdtem. Az új számok alapján ezt is túlteljesítettük.*

2. ***EuroCinema (az Európai Uniós pályázat).*** *Vállalás: a pályázat alatt megnöveljük a nézőszámot 50%-kal, napi 2 előadás hétköznap, hétvégén két matiné és 3 előadás… Itt nem nagyon tudtam tényadatokat közölni a kezdetekről, csak a pályázati anyagra támaszkodhattam. A pályázatban szereplő számadat: átlag nézőszám: 5 fő. Itt megint csak jók vagyunk – gondoltam. Gondolhattam, mert 8 fő bizony 50%-os növekedést jelent. Ami a hétköznapi előadásokat illeti: ez rendben volt, hiszen erre építettünk eleve. A*

hétvégi matinék: nos, ez már problémásabb volt, de mivel itt nem a heti lebontást kérték, hanem a havi értékeket, megint igazat tudtam írni a jelentésbe: havi 8 matiné megvolt, hiszen az iskolásoknak is hétvégén tartottuk a gyermek-előadásokat. Amik matinénak számítanak. A hétvégi három előadást is megoldottam, mert a filmklubok is hétvégén működtek. Annak pedig fix nézőszáma volt: 20 fő.

3. ***Városi pályázat.*** *Vállalás: rendszeres ifjúsági előadások. Szerencsére itt sem volt sok gond, mert nem volt meghatározva a rendszeresség fogalma. Nekünk meg ugyebár volt az iskolával közösen szervezett tematikus előadás-sorozat. Ami éves szinten minimum 12 előadást jelentett.*

Mindegyik pályázathoz meg tudtam végül írni a teljesítési jelentést, fel is adtam, majd a visszaigazolásokat is iktattuk. 'Na, ezzel megvolnék.' – gondoltam, s végre nyugodtan tudtam aludni akkor éjjel. Másnap megkaptam az új, a következő évi pályázati kiírásokat… Nem kellett volna… Várhattak volna még egy kicsit…

Az új pályázati kiírások már nem voltak ennyire kellemesek. Nagyobb nézőszám-növekedést kértek, magasabb lett a minimum nézőszám korlátozása, stb. Na, mit tegyek? Ismét a statisztikához fordultam segítségért: kiválasztottam a 3 legrosszabb hónapot az évben, s azoknak az adatait felhasználva töltöttem ki az Uniós pályázati nyomtatványt – ide kellett a 50%-os növekedés, ami az új pályázatnál már a nézőszám duplázását kérték. Határidőnek a legforgalmasabb hónapunkat, a decembert adtam meg. Ugyanígy jártam el a Városhoz benyújtott pályázatnál is.

Az NKA-nál már bajban voltam, hiszen az oldalszékek is bekerültek a nézőtér-kapacitás számítási módjába. Ami annyit jelentett, hogy az oldalszékeinkkel együtt a nézőtér 170 fő fogadására alkalmas… Ennek az 5%-a: 8,5 fő… Mivel azonban fél embert nem tudok mutatni, így 9 fő átlaglétszámnak kell lennie… 'Nem baj, a statisztika itt is segíteni fog…' – bíztam magamban és a statisztikában.

No, a pályázatok elmentek, el is lettek fogadva – pénz jött, előadások mentek… Teltek a hónapok, s elérkezett November – elkezdtük mérni, számlálni a látogatókat. Alakult a statisztika – közeledett a teljesítési igazolások leadásának határideje. Egyedül az NKA-s pályázat okozott némi fejtörést, mert nem akart összejönni az az átlag 9 fő,

csak december hónapban. Hazudni nem fogok – tehát kell valami megoldás. Végül, sok napos gondolkodás után a következő mondattal igazoltam a teljesítést: >A pályázati kiírás szerint a nézőtér befogadási kapacitása – az oldalszékek bevonása miatt – 21%-kal megnövekedett mozinkban. Ennek alapján a tavalyi teljesítésünk 14%-kal magasabb volt, mint amit a pályázat előírt, tehát idén csak 7%-os növekedést kellett biztosítanunk. Ezt a növekedést az egész éves munkánknak és a Város támogatásának köszönhetően december hónapra elértük.< A dokumentum elment, a Város befogadta, majd értesítettek, hogy soron kívüli támogatást ítéltek oda a mozinknak, mert városunkban ilyen nagy mértékű (7%-os) nézőszám-növekedést egy másik kulturális intézmény sem ért el…"

(Norbert J. Mucsina, 2015)

Egyszerűnek hangzik – vajon tényleg az is? A fenti kis történetből kiderülhet, hogy talán nem annyira egyszerű… Néha a darabszám, néha a százalék adja meg a kívánt eredményt. Minden csak a nézőponton múlik. De vigyázat! A mértékegységeket egységesen kell kezelni a projekten belül – emlékezz csak vissza a NASA-esetre.[22]

Ez az első lépés ahhoz, hogy megfelelően tudjuk mérni az előrehaladást, vagy éppen a talált hibák állapotát, esetleg mást. Nos, tehát mi mindent is mérhetünk? (Csak úgy ismétlésként, ha esetleg az előző fejezetet túl régen olvastad már, és esetleg a kutya is kezelésbe vette a papíros formátumot – a gépet meg nem kapcsolod be azért, hogy visszaolvasd…)

- A projekt/tesztelés időbeni haladását (mennyire tért el a tervezettől, vagy van időben?)
- A tesztelési lefedettséget, esetleg annak hiányosságait
- Tervezett/valós költségek mérlegét
- A terheltséget, erőforrások használatát (van-e túlterhelés, és ha van, mekkora?)
- A kockázatok megjelenését, hatásait
- Talált hibák mennyiségét, azok megoldásának folyamatát
- A projekt céljainak elérésének folyamatát
- A tesztelés folyamatát
- Stb.

[22] A történetet megtalálod a Bevezetésben.

Szóval MINDENT mérhetünk. Tényleg: mindent… Viszont nem mindegy, hogy miként mérjük. Illetve: hogyan fogjuk méréseinket közölni – azaz mi lesz a mértékegység. Első pillantásra talán nem olyan bonyolult dolog ez, hiszen a kenyeret, lisztet kilóra mérjük, a tejet literszám. De mi a helyzet, ha pl. babot szeretnénk venni kedvenc zöldségesünknél… Lehet kilogramm-alapon is kérni, meg literszám is. Akárcsak a borsót, lencsét, de néha még a mézzel is ez a helyzet. Talán mondanom sem kell, hogy egy liter lencse nem egyenlő egy kilogramm lencsével. (Ez az analógia úgy tudom, csak a víz esetében érvényes.)

Hasonló a dilemma néha a tesztelés tekintetében is: százalékos arány, vagy darabszám (esetleg mértékegység?)? A leggyakrabban ez a kérdés merül fel – nem véletlenül. A bevezető történet nagyszerű példa a kétféle mértékegység használatára: amikor a látogatók tényleges számát vizsgáltuk, mert tudni szerettük volna, hogy mennyi volt a forgalmunk, akkor a darabszámot használtuk mértékegységnek. Általában ez volt az alap mindenre. Ellenben amikor a pályázat sikerességét fogalmaztuk meg – mivel tisztában voltunk szerény látogatottságunkkal – már a kitűzött célt, azaz a nézőszám növelését százalékban adtuk meg. Tehettük, hiszen ha csak az akkori lakótársamék, meg még 1-2 barát megjelenik, már elértük a célszázalékos növekedést.

Remélem, most már érthető, hogy miért fontos jól meghatározni a mértékegységeket.

3.3.3 A mérés módszerének meghatározása

Miután meghatároztuk, hogy mit és milyen számokkal fogunk mérni, itt az ideje, hogy azt is eldöntsük, hogy hogyan fogjuk ezeket a számokat „előidézni”, azaz hogyan fogjuk mérni?

Ez sem tűnik túl bonyolultnak, bár van benne egy kis kihívás. Maradjunk csak a mozis példánál: ha meg szeretnénk tudni, hogy mekkora a nézőszám, akkor többféle módszer is lehetséges:

1. Az eladott jegyek számát vesszük alapul. A probléma ezzel az, hogy ha elővételben is lehet jegyet venni, akkor lehet, hogy csak megvásárolták a jegyet, de felhasználni már nem tudja. Esetleg, ha van úgynevezett Joker-jegy, ami egy előadás látogatására jogosít fel. Vagy bérlete van… Így már nem a valós számot kapjuk meg.
2. Megszámoljuk előadásonként – előadás kezdete előtt pár perccel –, hogy hány fő ül a nézőtéren. Itt is akad egy kis probléma, mert az előadás alatt is érkezhetnek még nézők…
3. Megszámoljuk az előadás végén a távozó nézőket – ez sem ad teljes képet, mivel előadás közben is távozhatnak.

Vajon akkor melyik lehet a legjobb módszer? Megvallom, én nem találtam ilyet. Tulajdonképpen a titkos 4. módszer az, ami a legpontosabb adatot adhatja jelen esetben: a jegyszedő. Igen, a jegyszedő, hiszen ő az adott előadás minden látogatójával találkozik. (Azokkal is, akik esetleg nem váltottak jegyet.) Itt csak egy bizonytalanság van: az emberi tényező… Vajon mennyire pontosan tud számolni? Nem felejti-e el, hogy meg kell számlálnia a látogatókat?

Egy szó, mint száz: nincs tökéletes módszer. De ha előre meghatározunk egy módszert (ami akár több más módszer kombinációja is lehet), akkor az lesz a mérésünk alapja. Ahhoz már tudunk viszonyítani, abból már tudunk metrikákat, statisztikákat, gráfokat és diagramokat készíteni.

A kapott eredményeket két fő adatcsoportba, mondhatni két fő típusba lehet sorolni:

1. ***Státusz adat***

   Az ilyen típusú adatok egy pillanatnyi helyzetet, aktuális eredményt mutatnak meg. A mozinál maradva: az adott előadáson pontosan mennyi látogató volt. Ez az adat szintén többféleképpen jeleníthető meg: százalékos arányban a nézőtér kapacitásának fényében, valamint „darabszámban”, azaz ténylegesen hány személy látogatta az adott előadást.

2. ***Trend adat***

   Az ilyen típusú adatok nem egy pillanatnyi állapotot, hanem egy folyamatot írnak le. Ezen adatok segítségével lehet mindenféle „jóslásokba” bocsátkozni. Visszatérve a mozis példára: ha egy bizonyos nap, napszak vagy előadás, esetleg film nézettségét szeretnénk mérni, s felbecsülni, hogy van-e még értelme vetíteni az adott filmet, vagy megtartani az adott előadás időpontját, akkor egy trend adat-alapú mérés sokat segíthet. Ilyen esetben egy adott időintervallumon belül folyamatosan kell számlálni a látogatókat az adott kérdésre specializálva – anno mi a csütörtök késő esti, 20:00 órás előadást fontolgattuk eltörölni. Ennek érdekében 2 hónapon keresztül figyeltük az adott előadás „forgalmát”, majd a kapott eredmények alapján döntöttünk az előadás sorsáról.

Amint látható maga a mérés módja sok mindent meghatározhat. Csak egy valamit nem: mikor jó az érték?

3.3.4 Határértékek meghatározása

Az előző fejezet végén feltettem egy nagyon fontos kérdést: mikor jó az érték? Kicsit bővebben: a kapott adatok, értékek mit is jelentenek? Hol van az a pont, ahol elkezdhetünk aggódni?

Ezek eléggé súlyos kérdések. A választ valahol a projekt- vagy teszttervben kell keresni. Ott, ahol megadtuk a projekt célját, vagy a tesztelés célját. Ott és akkor egy határozott számadatot adtunk meg – most viszont meg kell határoznunk azokat a határértékeket, amikor tökéletesen meg vagyunk elégedve az eredményekkel, amikor még elfogadhatóak azok, vagy éppen amikortól már elkezdhetünk aggódni…

Szóval két határpontot kell meghatározni. Amikor **még nem kell** (alsó küszöb), és amikor **már nem kell** (felső küszöb) aggódni.

Ismét szeretném a mozis történetet venni példám alapjául. Ott ugyebár egy bizonyos pályázathoz a látogatottság növelését vállaltuk 50%-os mértékben. Ezt alapul véve a legjobb eredmény, azaz a felső határ bármi lehet, ami 50%-nál nagyobb növekedést produkál. Tehát az első „határkő”, küszöb az 50%-os határ – ekkor ugyanis teljesítettük, amit vállaltunk. Ha 50% alatt van a növekedés, akkor bizony már nem vagyunk túl boldogok. Maga a pályázat esetleg megenged egy 4-5%-os eltérést negatív irányba is. Ebben az esetben a másik „határkő” a 45%. Ekkor még rendben van minden, nem nagyon kell aggódni. Benne vagyunk a tűréshatárban. Ami azonban ez alatt van, az már nem jó. Akkor már aggódnunk kell.

$$< 45\% \mid 45\% - 50\% \mid 50\% <$$

A fenti példa ékesen mutatja, hogy jó, ha már az elején, a projekt/teszt tervezésénél egy tól-ig határt meghatározni a sikeresség mértékegységeként. Jó, ha van egy kis tűréshatár, amin belül még minden rendben van. Sajnos ez nem minden esetben lehetséges, mert vannak olyan célok, amik nem engednek semmilyen mozgásteret. Az ilyen célokat a legnehezebb pontosan betartani, megmutatni.

3.3.5 Mit tegyünk, ha …?

Sajnálom, de nem találtam jobb címet ennek a fejezetnek. Ez az a fejezet, ami leírja, hogyan lehet reagálni a metrikák által megmutatott helyzetre. Nem fogom tudni pontosan megmondani, hogy milyen helyzetben mit kell tenni, mi a legjobb. Csak egy pár utat tudok felvázolni.

Kezdjük azzal az esettel, ha valami nagyon nem jön össze, azaz a lehető legrosszabbul alakulnak a dolgok. Ez mindenre érvényes. Tehát ha a tesztelés üteme nem megfelelően halad, ha túl sok hibát találtunk, ha minden jel arra mutat, hogy valami nagyon rosszul megy. Erre a legtöbbször a trend-adatok mutatnak rá, nem a státusz-adatok. (Ez utóbbiak inkább a jelentésekhez kellenek, pillanatnyi állapotot mutatnak.)

Ebben az esetben meg kell nézni, visszamenőleg mikor indult hanyatlásnak a folyamat. Górcső alá vesszük a trend-adatokat megkeresve azt a pontot, amikor láthatóan elindult a romlási folyamat. Ha ezt sikerül meghatározni, akkor nagy az esélyünk arra, hogy feltárjuk a probléma valódi okát s azt kiküszöbölve javíthatunk az eredményeinken. Innen még megfordítható a projekt, még van remény.

Sajnos nem minden esetben határozható meg a fordulópont. Azt kell mondjam, hogy a legtöbb esetben nincs fordulópont, vagy nem lehet konkrétan meghatározni. Esetleg – ez a lehető legrosszabb eset – már a kezdetekkor rosszul mértük fel erőinket, azaz sem a célok, sem a metrikák meghatározása nem volt megfelelő. Ilyenkor a legnehezebb meghatározni a következő lépést. Erre nincs általános megoldás, nincs generális válasz. Az érintett felelősöknek együtt kell megvizsgálni a metrikákat, az adott helyzetet és együtt kell döntést hozniuk. Ha minden kötél szakad, akkor meg kell szüntetni a projektet. Igen, ez is egy lehetséges, akár pozitív megoldás is lehet. Természetesen egy új koncepcióval, új tervekkel – tanulva az előző hibákból – még mindig újraindítható az adott projekt.

A másik véglet, amikor minden a legnagyobb rendben – pontosabban: túl nagy rendben, túl jó értékekkel – van, a számok sokkal jobbak, mint amit elvártunk. Gondolod, ilyenkor nincs más teendő, mint pihenni és örülni, ülni a babérjainkon és élvezni a munkák egyre érettebb gyümölcsét? El kell, hogy szomorítsalak: ez majdnem olyan rossz, mint amikor semmi nem jött/jön össze. Miért? Mert akkor a tervezésnél, a célok meghatározásánál volt hiba. Ilyenkor érdemes szintén megvizsgálni az adatokat, a trendeket és folyamatokat. Meghatározni, hogy mi lett volna az az érték, ami reális, ami nem túl alacsony, aminek teljesítése nem túl könnyű feladat. Ne feledjük az örök szabályt: túl korán érkezni a vacsorára legalább olyan nagy hiba, mint elkésni. Lefordítva: túl jól teljesíteni legalább akkora hiba, mint alulteljesíteni. Csak ott nem a teljesítménnyel van a probléma, hanem az elvárásokkal.

Akkor a legjobb a helyzet, ha minden érték a tűréshatáron belül mozog. Akkor lehet egy kicsit örülni, iszogatni, pihengetni. De csak mértékkel. Ezt az állapotot ugyanis fenn kell tartani, meg kell őrizni. Ahhoz, hogy minden rendben legyen nagyfokú odafigyelés, felügyelet, pontos munka szükséges. Ehhez nyújt segítséget a folyamatos monitorozás – státusz-adatok gyűjtése –, ellenőrzés. Minden változás figyelése, azonnali, produktív beavatkozás minden esetben, amikor az szükségessé válik, azaz az adatok alapján; ez adja meg azt a biztonságot, stabilitást, ami egy sikeres projekt/tesztelés elengedhetetlen feltétele.

3.4 Konfigurációmenedzsment

„Évekkel ezelőtt szükségünk volt egy autóra, mivel városból vidékre költöztünk. Anyagi keretünk nem engedett meg egy új autót – még a használtak közül is a majdhogynem-veterán kategóriában nézelődtünk. Sikerült is egy egészen jó állapotot

mutató Fordot vásárolnunk. Örültünk, használtuk. Sajnos azonban egy fél év, tíz hónap elteltével észrevettem, hogy a kaszni finoman szólva is rozsdásodni kezdett. Éppen a műszaki is esedékes volt, így egyik reggel elvittem a családi autószerelőnkhöz, s megbíztam, hogy készítse fel a vizsgára: amit kell, javítson ki, a kasznit lakatolja ki és vizsgáztassa is le.

Egy hét elteltével fel is hívott, hogy kész az autó, lehet menni érte. Boldogan indultam el, de azért magamhoz vettem egy nagyobb összeget is készpénzben, hiszen sejtettem, hogy nem volt ingyen a javítás – és nem is aprópénz lesz… Megérkezvén először körbecsodálgattam a kisautót, beindítottam, s örültem, hogy minden rendben van. Természetesen a számla is ott volt – aminek láttán egy kicsit meghűlt bennem a vér: ugyanakkora összeget mutatott, mint amennyiért vettem az autót egy évvel azelőtt. De nem volt mit tenni: fizettem, mint a katonatiszt. A nagy örömbe egy kis üröm is vegyült…

Hazafelé persze végiggondoltam a dolgot, s úgy döntöttem, hogy mégis megérte. Hiszen most már a kaszni rendben van, láttam a munkát folyamatában is, s tényleg meg lett csinálva rendesen.

Eltelt egy újabb év körüli idő, amikor is egyik hideg, januári napon Gyurika – így neveztük el az autót – nem akart elindulni. »Pöff…« - mondta, mikor indítani próbáltam. Aztán újra: »Pöff… Pöff… Krrrrrr…« Majd már ez sem… Telefon elő, szám beír, szerelő hív. Eredmény: kellemes vontatás a műhelybe, teljes diagnosztika. El is készült a hibaleltár: gyújtáselosztó, vízpumpa, vezérmű-szíj, plusz a járulékos cserélendő dolgok, úgymint olajszűrő, szimering gyűrűk, tömítések (mert csurgott az olaj), volt egy kis horpadás is, akkor már csapjuk hozzá azt is, valamint központi zár, vonóhorog, ködlámpa – ez utóbbi három már az én kéréseim voltak, gondolván: ha lúd, legyen kövér. Akkor még nem sejtettem, hogy mennyire lesz kövér…

Egy hónap szerelés után újra le kellett vizsgáztatni a vonóhorog miatt, valamint új forgalmit is ki kellett váltani. A végösszeg hatalmas volt – de csak az autó korához viszonyítva. Azért mégsem bántam meg, mert most már tudom, hogy az autómba mi az, ami új, mi az, ami várhatóan 1-2 éven belül cserés lesz, mi az, amire oda kell figyelni. Tudom – és ezt több ismerősöm is mondta – hogy az erre az autóra költött

összegből már egy sokkal fiatalabb példányt is vásárolhattam volna, de ekkor két dolog ötlik az eszembe:

1. *Egyszerre sosem volt annyi pénzem, csak annak töredéke. Részletre meg nem adnak egy olyan értékű használt autót. Így nem is vehettem volna egy olyan autót.*

2. *Az is egy használt autó lett volna, aminek rejtett hibáit, felszereltségét, mondjuk így: konfigurációját nem ismertem volna. Csak 1-2 évvel később derült volna ki, ha valami baja van. És akkor arra is lehet költeni rendesen…*

Ennek a régi, öreg Fordnak, Gyurinak már minden baját, nyűgjét, sérülését ismerem. Ha valamit cserélni kell, akkor tudom előre, hogy mikor kell majd cserélni, tudom mi a baj, ha rángatni kezd a motor, ha kicsit 'kacsázva' húz a kormány. Ismerem, mint a tenyeremet. Ha Gyurikának valami alkatrészét cserélni kell, akkor én arról tudni fogok… "

(Norbert J. Mucsina, 2015.)

„A konfigurációmenedzsment olyan tevékenység, ami során az informatikai infrastruktúra (konfiguráció) elemeit azonosítjuk, a közöttük lévő kapcsolatot, változásokat rögzítjük. Célja, hogy csak egy központilag jóváhagyott elem kerüljön a rendszerbe, így minden, a konfigurációban végbement változás egy központi helyen, kronologikusan tárolva legyen, így bármikor visszakereshető a teljes konfiguráció egy adott dátumra levetített állapota."[23]

A fenti sorok nagyon precízen meghatározzák, hogy mit jelent a konfigurációmenedzsment. Nagyon precízen – de nem igazán érthetően. Vizsgáljuk meg, hogy valójában mit is jelent számunkra ez a gyakorlatban.

Bármit nevezhetünk konfigurációnak, ami valamiféle rendszert alkot. Tulajdonképpen a konyhánk is egyfajta konfiguráció, hiszen csak a lábasok, kanalak, kések, villák, poharak, tányérok, asztal, tűzhely, stb. egysége. Együtt, egy „konfigurációban" alkotják azt, amit konyhának nevezünk. Abban az egységben bárminemű változás áll be, jó, ha a háziasszony/konyhafőnök tud róla. (Az én konyhámban mindennek megvan a maga helye – ha valami nem oda kerül vissza, ahol a helye van, akkor vagy mérges leszek és

[23] Boda-Bodrogközi-Gál-Illés: Szoftvertesztelés a gyakorlatban, 2014.

kiigazítom a hibát, vagy abszolválom a változást és beépítem. Az lesz az adott tárgy új helye – azaz a módosított konfiguráció állapotát jóváhagyom.)

A konfigurációmenedzsment több elemből áll össze, azok együttes jelenléte biztosítja a megfelelő verzió-követést, a minőségi felügyeletet egy adott rendszeren belül. Ilyen elemek – a teljesség igénye nélkül – lehetnek a következők:

1. Szolgáltatások
2. Felhasználók
3. Szoftveres elemek
4. Hardveres elemek
5. Dokumentációk
6. Elemek közötti kapcsolatok

Ugyanúgy, mint a metrikáknál, a konfigurációmenedzsment esetében is több lépéssel határozzuk meg a menedzsment fő irányát, elveit és működési szabályait. Ezek a lépések a következők:

1. A konfiguráció beazonosítása/meghatározása (Configuration identification)
2. A konfiguráció felügyelete (Configuration control)
3. A konfiguráció állapotának könyvelése/követése (Configuration status accounting)
4. A konfiguráció auditálása/ellenőrzése (Configuration audits)

Logikus és egyszerű, igaz? Pedig nem volt ez ilyen egyszerű – mármint ezen lépések meghatározása. Számos szabványt írtak/írnak a konfigurációmenedzsment témakörében, mint pl.: ISO 10007:2003, vagy az IEEE 829, de kevésbé ismertebb szabványok is léteznek, mint az ANSI/EIA-649-1998-as szabvány, vagy a STANAG 4159, vagy STANAG 4427. Mindezek azonban a fent említett lépésekben megegyeznek. Vizsgáljuk meg közelebbről ezeket a lépéseket.

3.4.1 A konfiguráció beazonosítása/meghatározása (Configuration identification)

Mint minden más eddig, ez is igen egyszerűnek hangzik. Mi lehet azon bonyolult, hogy megmondjuk, mi a konfiguráció? Miért kell ebből egy külön alfejezetet nyitni? A rövid vagy a hosszú válasz érdekel? Ha a rövid válasz elég, akkor idézném Rózsa Sándort, aki bár betyár volt, erre a kérdésre tökéletes rövid válasza van: „Csak!” Ha ennyi tényleg elég, akkor kérlek, csukd be a tananyagot, csomagold be valami szép papírba és ajándékozd el. Ja, és felejtsd el, hogy valaha is tesztmenedzsmenttel szerettél volna foglalkozni…

Ha esetleg érdekel a hosszabb válasz is, akkor olvass tovább nyugodtan. Hidd el: van értelme.

Naszóval: mi is az a konfiguráció? Hogyan lehet meghatározni? Kettős nézőpontot kell használni: mi az, ami egy adott konfigurációhoz tartozik, és mi az, ami nem. Ez utóbbi legalább olyan fontos, mint az előző. Vegyünk példának egy egyszerű személyi számítógépet, ami majdnem minden háztartásban jelen van. Természetesen nevezhetjük konfigurációnak, hiszen hardveres és szoftveres elemekről, azok közötti kapcsolatokról beszélünk. De mi is tartozik bele ebbe a konfigurációba:

1. *Szolgáltatások* – a gyermek tud játszani, mert játék van telepítve, lehet rajta internetezni, böngészni a világhálón, csevegni a csevegő programon keresztül. Stb.
2. *Felhasználók* – használja az egész család és mindenkinek saját profilja van. Valamint van egy vendég profil is.
3. *Szoftveres elemek* – van operációs rendszer, irodai és grafikai alkalmazások, audio- és video-lejátszó programok, játékok, stb.
4. *Hardveres elemek* – gépház, alaplap, processzor, memória, videokártya, merevlemez, optikai meghajtó, monitor, billentyűzet, egér, nyomtató.
5. *Dokumentációk* – minden program saját kézikönyve.
6. *Elemek közötti kapcsolatok* – a nyomtatóval lehet az irodai alkalmazáson keresztül nyomtatni.

Szinte teljes pontossággal meghatároztuk a mi otthoni számítógép-konfigurációnkat. De csak szinte – hiányzik még az, hogy mi NEM tartozik ebbe a konfigurációba. Nem tartozik bele pl. a szomszéd gyermek versenyautó-szimulátorához tartozó kormány és pedálsor, vagy a különböző családon belüli okostelefonok, vagy laptop(ok). Nem tartoznak bele a vírusok sem.

Miért olyan fontos azt is meghatározni, hogy mi NEM tartozik bele a konfigurációba? Azért, mert a konfiguráción kívüli elemekért a konfigurációért felelős személy (jelen esetben jómagam) nem vállal/vállalhat felelősséget. Így ha az adott számítógépnek valamely hardveres elemét lecseréljük, ami miatt a konfiguráción kívüli eszköz már nem használható, így annak működéséért én már nem vállalom a garanciát. Vagy ha egy, a konfiguráción kívüli eszközben módosítás történik – például egy újabb típusú telefon kerül a családba – ami már nem tud az adott konfigurációval együtt működni, az már nem az én felelősségem. Nekem az a fontos, hogy a számítógép azokkal a paraméterekkel, a feladatát maximálisan ellátva úgy működjön, ahogyan annak megalkotásakor működött. Bármilyen módosítás is történik rajta, arról én tudok, azt úgy vezetem, hogy ha a szükséges – még ha ritkán használt – funkciói valamelyike

nem működik a változás miatt, bármikor vissza tudjam állítani abba az állapotába, amikor még működött.

Levetítve egy tesztelési projektre: a konfiguráció, amit a tesztmenedzser felügyel a következő elemekből áll(hat):

- *A tesztelői csapat* – kivéve azon üzleti elemzőket és programozókat, akik szintén végrehajtanak némi tesztelést, DE nem tagjai a csapatnak, így tesztelésükről nem készítenek jelentést.
- *Tesztelendő szoftver* – csak az elérhető verziók és az a része, hogy a tesztelői csapat mindig a legfrissebb, legutolsó verziót használja. Nem tartozik ebbe a konfigurációba a programozáshoz használt programozási nyelvek leírása, a fordítóprogramok, valamint a különböző, kódszintű ellenőrző programok.
- *A dokumentáció* – azaz minden teszteléssel kapcsolatos dokumentum: teszterv, teszteset-gyűjtemények, automatikus szkriptek, annak működési leírása. Nem tartozik bele a kódolási konvenciók, programozási irányelvek, a követelmények dokumentációja – bár ez utóbbi akár bele is tartozhat.
- *A tesztelői környezet* – ez a rész kérdéses: ha a tesztelői csapat biztosítja a környezeteket, azaz neki kell pl. feltelepítenie az adott szoftvert, akkor akár a tesztelői csapat konfigurációmenedzsmentjéhez is tartozhat. Gyakoribb azonban az, hogy a tesztkörnyezetet – főleg nagyvállalati környezetben – a szerverparkért felelős adminisztrátorok, adatbázis-adminisztrátorok és webmesterek együttesen biztosítják. Ilyenkor a konfiguráción kívüliként, de saját konfigurációmenedzsmentet megkövetelve érdemes kezelni.

3.4.2 A konfiguráció felügyelete (Configuration control)

Igen, a felügyelete – az előző szakaszban már egy kicsit ebbe is belemásztam. A felügyelet ugyanis nagyon fontos. Megmaradva a számítógépes példánál: ha véletlenül a szomszéd, vagy a sógoraim egyike kicserélné a videokártyát a gépben úgy, hogy én nem tudok róla, majd mikor azon ritka pillanatok egyike elérkezik, hogy kedvenc játékommal, vagy tervezőprogramommal játsszak egy kicsit (mindkettőnek elég nagy a videokártya-igénye mind memória, mind sebesség tekintetében) és szembesülök a megrendítő ténnyel: nem működik… Ismerjük az elterjedt operációs rendszer beszélő hibaüzeneteit – képtelenség kitalálni, mi a hiba. Mármint első pillantásra. Természetesen egy kis nyomozómunka – minek során kellemes barátságot kötök Sir Bernard Hogan-Howe -val[24] és James B. Comey-val[25] - és pár órányi zajos

[24] A Scotland Yard jelenlegi vezető biztosa (MPS Commissioner) - 2015
[25] Az FBI jelenlegi igazgatója – 2015.

felemlegetése az ismeretlen tettesek családfájának, nyomra vezet és megtalálom, mi is változott meg a legutolsó állapothoz képest, amikor még kicsiny kedvenceim vígan futkorásztak a „vas”-on.

Nos, mindezt természetesen el lehet kerülni egy folyamatos felügyelettel – esetünkben, ha a cserét nálam kezdeményezik. Természetesen nyomós indok-tömeg alátámasztásával és némi könyörgő tekintet, esetleg a nejem hatékony támogatása/ajánlása, hogy a kérvényezett cserét véghezvigyem. Ebben az esetben is megtörténik a csere – ki tud a nejének ellenállni? – csak annyi a különbség, hogy tudok róla és megspórolok magamnak egy pár órás idegbajt. (De legalább már akkor tudom, hogy kedvenc programjaim nem fognak tovább működni, amikor a cserét megejtem. Egy lemondó sóhajjal nyugtázom a tényt és kész.)

Ugyanez a helyzet egy tesztelői projekt kapcsán is: ha a tesztmenedzser nem tud arról, hogy a költségvetés megvonása miatt a tesztelői csapatból két főt elbocsátanak (tudom, nagyon csúnya példa, de így talán megmarad), akkor igen nagy lesz a meglepetés, amikor nem tudják teljesíteni a megfelelő, előre beütemezett mennyiségű és mélységű tesztelést. Természetesen mindennemű változás, ami a konfigurációt érinti – lásd az előző szakaszban meghatározott konfigurációt – valamilyen változást eredményez. A változást pedig mindenképpen le kell kezelni – akár pozitív, akár negatív irányú a változás. Ha láttad már a Pillangó-hatás[26] című filmet, akkor nem kell magyaráznom, hogy a legapróbb módosítás is mekkora hatással lehet később a tesztelésre, vagy magára a teljes projektre nézve. Ha pedig még nem láttad, akkor – nézd meg! ☺

3.4.3 A konfiguráció állapotának könyvelése/követése (Configuration status accounting)

Már megint a dokumentáció! – Igen, megint. És még mindig. És nem győzöm hangsúlyozni, hogy mennyire fontos a dokumentáció. Mindent érdemes dokumentálni – persze megfelelő önkontrollal. Nem kell leírni mindenki minden napját részletesen, csak azt, aminek köze van az adott projekthez. Persze, ha valaki maga a projekt…

Visszatérve a dokumentációra: a konfigurációmenedzsment egyik alappillére, hogy a változásoknak nyoma legyen. Nem csak annyi, hogy

„Jani tegnap kicserélte a vinyót.”

Mert ez aligha elég. Mert ki is az a „Jani”? A projektben 3, János nevű úriember dolgozik. Azon kívül ki engedélyezte? Mikor? Miért kellett cserélni? Melyik merevlemezt cserélte ki? Milyen merevlemezre

[26] A pillangó-hatás (The Butterfly Effect) - színes, magyarul beszélő, amerikai thriller, 113 perc, 2004, rendezte: Eric Bress, J. Mackye Gruber

cserélte ki? Mi történt az adattárolón lévő adatokkal? Hová került az eredeti merevlemez? Ki ellenőrizte le a csere folyamatát? Ki ellenőrizte a csere utáni állapotot (működés, adatok helyessége, sebesség, stb.)?

Megannyi kérdés, pedig csak egy egyszerű merevlemez csere történt… A fenti műveletet egy kicsit bővebben kellene dokumentálni:

> *„2015.03.15-én a 3-as szerveren (Quantum) adatvesztést tapasztaltunk. A hiba oka egy meghibásodott merevlemez (hdd2), amit azonos típussal (HDD FUJITSU SAS 6G 1TB 7.2K HOT PL 3.5" BC) a vezető szerver-adminisztrátorának, Holló Tibornak engedélyével 2015.03.17-én Kiss János, szerver-adminisztrátor cserélt ki. Az adatok mentését a RAID 10 rendszer elvégezte. Az adatok helyességét Kiss János ellenőrizte, a sérült merevlemezt SD20150317-es jelzéssel a szerverraktárba felvette."*

Ez már elég információt tartalmaz a visszakövetéshez. Manapság azonban már konfigurációmenedzsment megoldások sora van a piacon, a legtöbbjük webes alapú, de van szerver-kliens és lokális típusú is. Kinek-kinek ízlése szerint.

A változások nyomon követése az adott konfiguráción belül azért olyan fontos, mert ha nincs róla dokumentum, akkor, aki tud a változásokról, kulcsfontosságú tudás birtokában van. Ha valami történne vele (felmondana, lebetegedne, sajnálatos módon távozna az élők sorából), akkor az utódjának nagyon nehéz – majdnem lehetetlen – feladat lenne a konfiguráció minőségének szinten tartása. Emlékezzünk csak vissza az otthoni videokártyás példámra… És ez egy akkora kockázat egy projekt életében, ami – szerintem legalábbis – nem megengedhető.

3.4.4 A konfiguráció auditálása/ellenőrzése (Configuration audits)

> *„Az 1860-as években élte virágkorát a kubikosság. A kubikosmunka nehéz munka volt, egész embert kívánt. A fizetség ellenben nem volt annyira jó. Sokan belerokkantak a munkába. A fizetésüket a következő módon számították ki:*
>
> *'A kubikgödrökbe oszlopot, földbabát hagynak a kubikosok mérés céljából. A mérnök a babát méri, amilyen magas a baba, olyan mélységű kubikot számít. A kubikosok úgy segítettek magukon, hogy a figurát, a babát nagyobbították magasságban. Lemetszették a tetejét a gyeppel, a gyökerek alatt és alá raktak egy pár sukk ugyanolyan színű földet egy fölöslegesen meghagyott babából, amit zsákba kötve*

szoktak vinni, hogy széjjel ne menjen, aztán ráfurkózzák és visszateszik rá a lemetszett gyepes földet. Oldalát megnyesik az eredeti babáéval egyenlő felületűre: emberfia nem tudja kinézni, hogy toldva van. Emelik a baba magasságát, különösen agyagos talajnál úgy is, hogy lapáttal ütögetik a baba oldalát, miáltal a baba keskenyebb, de magasabb lesz. A kubikgödör szélét is szokták emelni, különösen ahol feltűnően lapos, nemcsak a középen levő két babát. Aki a mérőlécet tartja, azt is megteszi, hogy lábával gödröt váj a földbe nagy hirtelen, amit a távol álló mérnök nem vehet észre.'[27]

Amint látható, nem mentek furfangért a szomszédba. De persze nem mehetett ez sem örökké – egy idő után gát- és töltés építésénél nem a kihordott földet mérték, hanem a felhordott dombot. Ekkor már a dombot támasztották alá cövekekkel, üregeket képezve, hogy minél nagyobbat mutasson. Persze az első nagyobb esőzés megmutatta a valót, de addigra már messze jártak a kubikosok – velük együtt a fizetség is… „

Beszéltünk már a konfiguráció mibenlétéről, meghatározásáról, a dokumentációról – most beszéljünk egy kicsit az auditról is. Azaz arról a folyamatról, amikor a dokumentált állapotot összevetjük a valós állapottal. Igen-igen. Néha ezt is kell… Tudom, sokan feleslegesnek tartják ezt (is), hiszen ott vannak a szabályok, a dokumentáció, a bizonylatok… A bevezető kis történet azonban megmutatja, hogy a papír, vagy másképpen: a látszat nem minden.

Tehát ellenőrizni kell.

Na de hogyan? És mikor? De legfőképpen: ki? Nagyon jó kérdések, vegyük őket sorba:

3.4.4.1 Hogyan ellenőrizzük a konfigurációt?

Ha rendelkezünk megfelelő dokumentációval, akkor egyszerűnek mondható a dolgunk: csak fogjuk a legutolsó állapot-adatot (konfiguráció-leírást) és összevetjük az élő konfigurációval. Maradva az otthoni számítógépes példánál: megvan a számla, amit a megvásárolt alkatrészekhez kaptam. Valamint a kicserélt videokártya számlája is (hiába, a garancia nagy úr). Tudom azt is, hogy utoljára mikor és mit telepítettem fel – erről nincs papírom, de ez most nem is fontos. Szépen leemelem a gép oldalát és belepillantok: alaplap rendben, processzor+hűtő pipa, memória megvan, videokártya – hoppá! Ez nem is az! (Itt jöhet egy kis mérgelődés, telefonálgatás, stb…) „Pedig nekem azt mondták, hogy ezt teszik bele!" – Nos, ez az a mondat, amit sosem szeretne az ember hallani. De néha előfordul: ami papíron megvan, az

[27] Kiss Lajos: Szegény emberek élete, Gondolat Kiadó, 1981

nem feltétlen egyezik az életben fellelhető dolgokkal. Persze ilyenkor lehet keresni a bűnbakot – de ritka az, amikor megvan a felelős. Még mindig él a régi mondás: „Új kárnak, régi várnak nincs gazdája…”

3.4.4.2 Mikor ellenőrizzük a konfigurációt?

Ez megállapodás, szabvány, szolgáltatás és cégkultúra függvénye. Mindenképpen javallott rendszeresen, félévente vagy évente egy-egy ellenőrzést tartani. Lehet gyakrabban is, ha a konfiguráció komplexitása és gyakori módosítása azt megköveteli. A túl gyakori ellenőrzés azonban nem jó – frusztrálttá teheti a környezetben dolgozókat, bizalmatlanságot szíthat. Az audit rendszerességét érdemes már akkor meghatározni, amikor a konfigurációt összeállítjuk, vagy amikor azt nevesítjük.

A rendszeres auditon kívül vannak rendhagyó auditok is. Ezekre különleges alkalmakkor van szükség: nagyobb módosítások után, komplex felújítások után, esetleg többszöri, rendszeres hibák észlelésekor. Ezeknek nincs előre meghatározott időpontjuk – mindig a helyzet követeli meg az ilyen típusú ellenőrzések végrehajtását.

3.4.4.3 Kinek kell az ellenőrzést végrehajtani?

Az ellenőrző személye nagyon fontos kérdés. Nem szabad szubjektívnek leennie, csúnya szóval élve: megvesztegethetetlennek, pártatlannak kell maradnia. Tudom, hogy nagyon nehéz ilyen embert találni – éppen ezért javaslom, hogy az auditot ne egy ember végezze, hanem egy minimum 3 fős csoport. És még egy fontos dolog: ha lehet, egyik csoporttag se legyen az adott konfigurációért felelős csoport tagja. Szóval: ha lehet, egy kívülálló csapat ellenőrizze a valóságot. Miért? A kérdés jogos, a válasz egyszerű: ha neked az az érdeked, hogy mindent rendben találj, akkor az apróbb hibákat hajlamosabb vagy elnézni. Ha te nem vagy érdekelt abban, hogy hibát találj, vagy mindent rendben találj, akkor az objektivitásod valószínűleg megmarad.

Mindezeket a lépéseket egy tesztelésre levetítve: minden tesztelési ciklus előtt érdemes ellenőrizni az adott konfigurációt, lehetőleg a projekttől független csoporttal. Ha nincs lehetőségünk független csapattal ellenőriztetni, akkor érdemes minden résztvevő csoportból (tesztelő, programozó, üzleti elemző, grafikus, stb.) delegálni egy képviselőt az audit-csoportba ügyelve arra, hogy a létszám páratlan maradjon. (A páratlan létszám a döntésképtelenség elkerülése miatt hasznos.)

3.5 Kockázatok

A kockázatokról, annak meghatározásáról már beszéltem a 3.1.1 –es fejezetben. Természetesen az már eléggé régen volt ahhoz, hogy most szükséges legyen egy kis ismétlésre, az alapok felelevenítésére. Kezdjük akkor magával a definícióval: mi is a kockázat?

A kockázat mindazon elemek és események bekövetkeztének a valószínűsége, amelyek hátrányosan érinthetik a projekt megvalósítását, valamint amelyek lényegi befolyással lehetnek annak működésére, a végső produktum (szoftver) minőségére. (Egy másik meghatározás szerint: a kockázat egy olyan, valamilyen valószínűséggel bekövetkező esemény, ami negatív hatással van a projektre vagy a projekt eredményére (legyen az szoftver, esemény vagy bármi más egyedi produktum).)

A **kockázatok hatásának célját** tekintetbe véve három főbb típus létezik:

1. *Projekt kockázat* – Ez a típus magára a projektre vonatkozik. Azokat a kockázatokat sorolhatjuk ebbe a típusba, amik a projekt működését veszélyeztetik, a költségekre, a tervezett időkeret vagy a szükséges erőforrásokra lehet negatív hatással. Például a betervezett erőforrások mégsem lesznek elérhetőek (esetleg betegség miatt), vagy valamelyik szponzor visszalépett, és így anyagi erőforrás-hiány alakulhat ki.
2. *Termék kockázat* – Ezek a kockázatok magát a terméket, a termék tervezett funkcióit, tulajdonságait, minőségét érinthetik. Leginkább ezzel a típussal foglalkozunk, amikor tesztelői, tesztmenedzseri szemmel vizsgáljuk a kockázatokat. (Természetesen nem kizárólagosan az ilyen típusú kockázatokra koncentrálunk, de ez a fő csapás.) Például a tesztelés hiányából fakadóan némelyik keresztkapcsolt funkció nem működik megfelelően, a grafikai megjelenítés bizonyos extrém esetekben kaotikussá, használhatatlanná teszi a terméket.
3. *Üzleti kockázat* – Ezt a kategóriát ritkábban említi a szakirodalom, ennek ellenére fontos lehet, ha külső beszállítóval dolgozunk együtt, aki termékeivel vagy szolgáltatásaival hozzájárul a projekt eredményeihez. Az ide sorolt kockázatok a külső beszállítón keresztül lehetnek hatással a projektre. Valamint ide tartozik még az is, amikor egy adott termék nem megfelelő minősége miatt hitelt veszít a vállalat – ezáltal esetleges megrendelőket, régi üzletfeleket is veszíthet. Azt hiszem, nem kell bemutatnom a Microsoft által 2007. január 30-án hivatalosan kiadott Windows Vista operációs rendszert. Ez a kiadás nemzetközi szinten bukásnak lett nevezve – hazánkban a 10%-os használati rátát is alig érte el…

A **kockázatok forrását** tekintetbe véve négy fő típust lehet megkülönböztetni:

1. *Megbízótól vagy beszállítótól eredő kockázatok* – az ilyen típusú kockázatok kezelése eléggé nehéz. Ilyen kockázat lehet például, ha a beszállító nem a megfelelő minőségű alapanyagot biztosítja, vagy a megrendelő (megbízó) irreális határidőket próbál betartatni, esetleg az általa meghatározott célok/követelmények egymásnak ellent mondanak.

2. *Célokból, célkezelésből eredő kockázatok* – egy kicsit hasonlít az előző típusra, de itt csak a célokról beszélünk. Nem csak az ellentmondásos célok, hanem a kezdeti célok módosításának hiányos kommunikációja, vagy esetleg negatív irányú módosítás mind-mind kockázatot rejt magában.
3. *Erőforrás-kezelésből eredő kockázatok* – nos, ez elég nagy probléma lehet. Valós kockázat, hiszen nem feltétlen kalkulál megfelelően a megrendelő/megbízó az erőforrások tekintetében. Vagy nem a megfelelő tapasztalattal rendelkező erőforrást sikerül az adott pozícióba állítani, így az erőforrás-igény is változik. Például egy Agile/Scrum típusú projekthez egy nagy „Scrum”-tapasztalattal rendelkező tesztmenedzser hatékony munkaerő. De ha nem találunk ilyen munkaerőt, akkor legalábbis kérdésessé válik a projekt minősége, hiszen a tapasztalatlan munkavállalónak még bele kell tanulnia a „Scrum”-ba és felépítenie a bizalmat, felszednie a megfelelő tudást ahhoz, hogy sikeres lehessen a projekt. Ez természetesen több időt vesz igénybe, mint amennyit előzőleg kalkuláltunk…
4. *Objektív és szubjektív kockázatok* – tudom, eléggé furcsa a megnevezése, hiszen mindenféle kockázatot vagy objektívként, vagy szubjektívként lehet értelmezni. (Leginkább szubjektívként…) Amire én itt gondolok, az inkább az, hogy tőlünk független az adott kockázat megjelenése, avagy tőlünk függő. Független például a természeti kockázat (földrengés, aminek hatására megakad a projekt – lévén nincs pl. áram…), esetleg társadalmi kockázat (egy véletlen háború, vagy kormányrendelet, aminek hatása végzetes lehet a projektre), vagy egy merőben új technológia megjelenése, mint a már említett Windows Vista esetében… Viszont a projekt résztvevőitől függő lehet egy tárgyi tévedés, vagy szabotázs. (Tudom, hogy abszurdnak tűnik, de valójában nem az.)

Egy harmadik csoportosítás viszont a **kockázat időbeni megjelenését** veszi alapul:

1. Előkészítés, vagy tervezés fázisában megjelenő kockázat
2. A lebonyolítás és befejezés fázisaiban megjelenő kockázat
3. Az üzemeltetés fázisában megjelenő kockázat

Fontos, hogy egy adott kockázat akár mindegyik fázisnál megjelenhet – a különbség annak hatásideje, azaz a kockázat időbeni eloszlása és annak hatásának csökkentésére fordított erőforrások mértéke (akár anyagi, akár humánerőforrásról beszélünk).

Bármelyik csoportosítási módszert választjuk is, az biztos, hogy nem lesz egyszerű a folyamat. Sem rövid… Ahhoz ugyanis, hogy egy projekten belül, vagy egy tesztelési ciklusra vonatkoztatva megtaláljuk a lehető legtöbb és legfontosabb(nak vélt) kockázati tényezőket, részletes kockázat-elemzést kell végezni.

3.5.1 Kockázat-elemzés

A kockázat-elemzés egy különálló tudomány, amiből – mint azt korábban már írtam – még tesztelési stratégia is kinőtte magát. A kockázat-elemzésnek több szakasza van – szintén már bemutattam a 3.1.1-es fejezetben –, valamint több módszer is a kockázatok azonosítására.

A kockázat-elemzés szakaszai (csak röviden, kifejezetten ismétlés végett):

1. Kockázatok azonosítása
2. A kockázatok kiértékelése
3. A kockázatok csökkentése
4. A fennmaradó kockázatok kezelése

Maga az elemzés csak két szakaszt foglal magában, az azonosítást és a kiértékelést. A fennmaradó két szakasz mát a kockázati tervhez tartozik, így majd a következő fejezetben fogok erről bővebben írni. Most lássuk, hogyan lehet azonosítani a lehetséges kockázatokat – hiszen ez a kezdet. ☺

3.5.1.1 Kockázatok azonosítása, értékelése

Tulajdonképpen ez az első lépés: meg kell találni, mi is okozhat bajt a projekten belül. Mi az, ami miatt kárunk lehet. Amint azt már a 3.1.1-es fejezetben is írtam, a projektben érintettek együttesen képesek csak feltárni a lehetséges kockázatokat. A kockázatok feltárásának technikáit is felsoroltam már, itt most egy kicsit bővebben is kifejtem azokat:

- Az adott terület specialistáinak, szakértőinek megkérdezése – ez viszonylag kényelmes technika, viszont csak akkor használható, ha ténylegesen rendelkezik a projekt minden területéről egy-egy specialistával. A specialisták ugyanis gyorsan és pontosan fel tudják térképezni a területükkel összefüggő kockázatokat, valamint meg tudják határozni azok mértékét is.
- Projekttől független személyek megkérdezése (külső szemlélők bevonása) – ez egy kicsit macerás és egyedüli technikaként használva nagyon kockázatos. Viszont pozitívumaként kell megjegyeznem, hogy mint külső szemlélő, sok olyan gyenge pontra – azaz kockázatra – mutat rá, amit egy belső tag nem venne észre.

- Kockázati sablonok használata – többféle sablont lehet használni. Van olyan, amit szinte szabványként is lehet kezelni (CRAMM-model[28]), de lehet olyan is, amit egy vállalat a belső projektjeihez dolgozott ki. Mindegyik valamely struktúrát ad és lehetséges megoldásokat is kínál az előre meghatározott problémákra. Nem minden esetben hatékony – inkább úgy fogalmaznék, hogy bizonyos, speciális projekteknél lehet hatékony. (A példának felhozott CRAMM-model például információs rendszerek érzékeny adataival kapcsolatos projektekben lehet hatékony segítség.)
- Előző projektek tapasztalatainak felhasználása – A legegyszerűbb, ha a tapasztalatokra támaszkodunk, azaz egy hasonló, már lezajlott projekt dokumentációját vesszük alapul. Igen, tudom, hogy ez nem mindig lehetséges: vagy azért, mert nem volt eddig olyan típusú projekt, vagy volt ugyan, de nincs jól dokumentálva. Vagy nem is volt, de az sem volt dokumentálva… Maradjunk azért annál a példánál, hogy volt ilyen projekt és még le is lett dokumentálva rendesen. Ebben az esetben viszonylag egyszerű a dolgunk: elővesszük a kockázati elemzését a régi projektnek, meg mellé tesszük a változás-menedzsmentet leíró dokumentumokat és voilà! Már kész is vagyunk – a nehezével. Ugyanis itt azért még nem ér véget a dolog. Minden projekt egyedi. Ezért biztosan nem lehet teljes analógiát vonni a régi és az új projekt között, még a kockázatok tekintetében sem. (Ha mégis, akkor az új projekt nem új, hanem felújított. De szerintem akkor sem lesz tökéletes az analógia.) Szóval „személyre kell szabni” az elemzést. Meg kell vizsgálni, hogy az adott kockázati elemek milyen módosítással és milyen hatással lehetnek az új projektre. A módosításokat viszonylag könnyen elvégezhetjük – akár egyedül is.
- Kockázatfeltáró műhelymunka/Ötletbörze – szerintem az egyik leghatékonyabb módszer. A recept: szedd össze a lehető legtöbb projektrésztvevőt, ültesd be őket egy nagy terembe, adj nekik rengeteg „post-it”-et, meg tollat. Adj nekik még egy kis rágcsálnivalót, meg innivalót, majd tedd fel nekik a kérdést: „Szerintetek milyen kockázati tényezők merülhetnek fel a projekt kapcsán?” Hihetetlenül eredményes és hatékony módszer.

A projektben érdekeltek minél nagyobb körű bevonásával lehet a legtöbb minőségi kockázatot feltárni. Itt tipikusan és halmozottan érvényesül a már-már közhelynek számító mondás: „Több szem többet lát, több fej okosabb…”

[28] http://hu.wikipedia.org/wiki/Cramm-modell - 2015.03.30.

A kockázatok azonosítása után el kell kezdeni a kockázatok kiértékelését. Ez tulajdonképpen egyfajta csoportosítást, sorrendbe állítást jelent. A kockázatokat többféleképpen lehet csoportosítani, például típus szerint, esetleg gyakoriság szerint, belőle fakadó hatás szerint, stb. (Ahogyan azt már korábban említettem.)

A kockázatok szintjének meghatározásakor minden kockázati tényező mellé előfordulási valószínűséget is rendelnek. Ez az érték mutatja meg azt, hogy mennyire valószínű, hogy az adott kockázat meg fog jelenni a fejlesztés során. Más szavakkal ez az érték a technikai kockázatot mutatja meg. Ennek az értéknek sok befolyásoló tényezője van, ilyen például:

- A használt technológia összetettsége és a csapatok összeállítása
- Az üzleti elemzők, tervezők és programozók nem megfelelő szakmai tudása
- Konfliktusok a csapaton belül
- Szerződésbeli hiányosságok a beszállítókkal (például nincsenek megfelelően meghatározva a felelősségi körök)
- A fejlesztő csapat földrajzi és időzónai megosztottsága
- A hagyományos és az új megközelítések ellentéte
- A használt eszközök és technológiák
- Nem megfelelő menedzsment vagy technikai vezetés
- Időbeli, erőforrásbeli, költségvetési és gazdálkodási nyomás
- Korábbi minőségbiztosítási tevékenység hiánya, vagy hiányos dokumentáltsága
- Interfész és az integráció kérdése

A másik érték, vagy mutató, amit figyelembe kell venni, az a kockázati esemény hatása, amennyiben az bekövetkezik. Ezt az értéket befolyásoló hatások:

- Az érintett funkció használatának gyakorisága
- Az érintett funkció kritikussága az üzleti cél elérésének tükrében
- A jó hírnév – reputáció – sérülése
- Üzleti veszteség mértéke
- Potenciális pénzügyi, ökológiai, illetve társadalmi veszteségek, vagy kötelezettségek
- Polgári vagy büntetőjogi szankciók, mint következmények
- Engedélyek veszélyeztetése, elvesztésének kockázata

- Áthidaló megoldások hiánya
- A hiba negatív hatása
- Biztonság

Mindkét érték meghatározása lehet mennyiségi (quantitative) vagy minőségi (qualitative). De nem lehet a kettő keveréke, azaz ha az egyiket, például a bekövetkezési valószínűséget minőségileg (nagyon magas, magas, normál, alacsony és nagyon alacsony értékekkel) határozzuk meg, akkor annak hatását is csak ezen a skálán mérhetjük. Ugyanez érvényes fordítva is. (A mennyiségi meghatározás általában 0-10-es skálán mozog.)

Az előfordulási valószínűség valamint az okozott hatás értékek szorzata (mennyiségi), vagy átlaga (minőségi) adja meg a kockázati értéket, ami alapján lehet egyfajta sorrendet meghatározni a kockázatok között. Viszont a hatás mértékét, vagy az előfordulás valószínűségét már nagy felelőtlenség egyedül meghatározni. Ezt inkább egy csoportra kell bízni – a legjobb az, ha az érintettek (a projekt résztvevői) maguk is pontozzák az egyes kockázati elemeket előfordulási és okozott hatásbeli szempontból. Majd a kapott értékeket összegezzük és átlagoljuk (külön az előfordulási valószínűséget és az okozott hatás mértékét), majd a kapott eredményeket összeszorozzuk. Így már egy viszonylag releváns súlyozást kapunk.

Még egyszer tehát a képlet mennyiségi mértéknél:

$$előfordulási\ valószínűség * okozott\ hatás = kockázati\ érték$$

Valamint minőségi mértéknél:

$$\frac{előfordulási\ valószínűség + okozott\ hatás}{2} = kockázati\ érték$$

Miért fontos, hogy ne egyedül egy ember véleménye alapján állítsuk sorba a lehetséges kockázatokat? A válasz egyszerű: mindenki számára más és más a kockázat, mást és mást jelent annak hatása. Például egy tesztelőnek elemi fontosságú, hogy az adott tesztelési ablak ideje alatt a tesztkörnyezet folyamatosan elérhető legyen. Ellenben egy programozónak mindez nem fontos, mert számára a fejlesztői környezet a fontosabb. Így ő valószínűleg kisebb értéket fog adni a tesztkörnyezetre vonatkozó kockázat által okozott hatásra, mint egy tesztelő.

3.5.2 Kockázati terv (mitigation plan, contingency plan)

Nos, most már megvannak a kockázataink, le is pontoztuk mindet, felállítottunk egy sorrendet – és most akkor mi legyen? Most akkor mit csináljunk?

A kérdés jogos, de újabb kérdések is felmerülnek:

- Miért kellett kiszámolni a kockázati értéket?
- Van-e egy határérték, ami alatt nem foglalkozunk a kockázattal? (Kockázati értékre vonatkoztatva, természetesen.)
- Minden kockázatot meg kell vizsgálnunk?
- Mit csináljunk a súlyozott kockázatokkal?

Szép kérdések, igaz? Vegyük sorra mindet.

Miért kell kiszámolni a kockázati értéket? Azért szükséges ez, hogy fel tudjunk állítani egy sorrendet.

Van-e egy határérték, ami alatt nem foglalkozunk a kockázattal? Igen, van ilyen. Pontosan ezért kell a sorrendet felállítani. Viszont az, hogy a határ hol helyezkedik el, csak is a csapaton és a kockázatok mennyiségén múlik. Ezt lehet a kockázati érték alapján, vagy egy bizonyos kockázati mennyiség alapján eldönteni. Általában egy nagyméretű projekt esetén (több éves lefutású, vagy esetleg nagyon komplex projekt) a top 25 kockázat az, amivel foglalkozni kell. Közepes projekt esetén (5-10 hónap lefutású, mérsékelten komplex projekt) 10-15, míg kis projektek esetén (1-5 hónap lefutású projekt minimális komplexitással) az 5 legsúlyosabb kockázat az, amivel foglalkozni kell. Azt hiszem, ezzel a „*Minden kockázatot meg kell vizsgálnunk?*” kérdést is megválaszoltam.

Mit csináljunk a súlyozott kockázatokkal? Ha jól belegondolok, ezt a kérdést is megválaszoltam már: sorrendbe állítani a kockázati pontérték – azaz a súlyozás – alapján, majd azokat a kockázatokat kiválasztani, amivel foglalkozni kívánunk. A kérdés inkább úgy hangzik jól: **mit tegyünk a kiválasztott kockázatokkal?** (Azaz: most akkor mit csináljunk?)

Mivel már tudjuk, hogy mik azok a legsúlyosabb események, amik a legnagyobb kockázatot jelentik a projektre/tesztelésre nézve, megpróbálhatunk ellene védekezni is. Ezt kétféleképpen tehetjük meg: vagy a bekövetkezésének valószínűségét próbáljuk meg csökkenteni, vagy az okozott negatív hatását próbáljuk mérsékelni. Akár mind a kétféle módon is megpróbálhatunk védekezni. Mindezeket érdemes írásban is lefektetni, azaz tervet készíteni az egyes kockázatok kivédésére, hatásainak mérséklésére. Ezt nevezzük kockázati tervnek.

3.5.2.1 Kockázati eseményt megelőző terv (contingency plan)

Ahogyan azt már említettem, kétféleképpen lehet egy adott kockázatra felkészülni. Az egyik lehetőség az, hogy megpróbáljuk csökkenteni a lehetőségét, hogy egy adott kockázati tényező bekövetkezhessen. Hogyan is lehet ezt megvalósítani? Nézzünk egy egyszerű példát:

Legyen a „projekt" egy családi nyaralás. Mindig érdemes feltérképezni, hogy mi baj történhet a nyaralás alatt. És érdemes felkészülni is a „bajra". Mivel a nyaralás helyszínére autóval tervezünk menni, az egyik legnagyobb kockázat maga az öregedő autó lesz. Vajon bírni fogja-e az utat? Nem fog véletlenül lerobbanni? Természetesen az ember mindent megtesz azért, hogy az autó rendben legyen: az utazás előtti héten elviszi a család autószerelőjéhez és teljes átvizsgálást kér, kisebb javításokat végez, ellenőrzi van-e tartalék olaj, fékolaj, üzemanyag-kanna, ami fel is van tölte, égőkészlet, stb. Szóval „megelőző csapást mér" a rettegett eseményre…

A fenti példában a bajok feltérképezése jelenti a kockázat elemzést, az autó átvizsgálása pedig a megelőzést, azaz az előfordulási valószínűség csökkentését. Egy tesztelésre levetítve a dolgot az egyik leggyakoribb kockázat a tesztkörnyezet rendelkezésre állása – illetve annak hiánya. Azért, hogy ennek valószínűségét a minimálisra csökkentsük a legegyszerűbb egy részletes, mindenre kiterjedő tervet készíteni a különböző tesztelési ciklusokra, különös tekintettel azon időpontokra, amikor a tesztkörnyezetre szükségünk van. Ezt a részletes tervet mindenkivel el kell fogadtatni, vagy legalábbis megismertetni.

3.5.2.2 Kockázati hatást csökkentő terv (mitigation plan)

Amennyiben nem tudjuk csökkenteni az előfordulási lehetőséget, akkor marad a kockázati tényező hatásának mérséklése. Ezt is előre meg lehet tervezni, hiába akkor kell alkalmazni, amikor a „baj" már megtörtént.

Az előbbi, családi nyaralás példájánál maradva: egy másik kockázat – állandó jellegű kockázat – az időjárás. Mi legyen akkor, ha az időjárás esős lesz, meg hideg? Ezt ugye nem tudjuk befolyásolni – legalábbis egyelőre. De azt igen, hogy mit tegyünk, ha már megtörtént. Például érdemes szétnézni, hogy abban a városban, vagy közeli környékén milyen más, nem szabadtéri rendezvények, szórakozási lehetőségek vannak. Például pillangóház, vagy színház, múzeum, játszóházak, kézműves foglalkoztatók, stb. A családot megkérdezve így már könnyen kellemessé tehetünk egy-egy esős napot, vagy hideg estét.

Ami a tesztelést illeti: kockázatként könyveljük el azt, hogy a tesztelőinket valami előre nem látható esemény meggátolja a tesztelés végrehajtásában (lehet ez betegség, baleset, bármi). Ezen események

ellen nem sokat tehetünk, hogy elkerüljük, viszont az okozott hatást csökkenthetjük. Készíthetünk egy olyan részletes teszttervet és teszteset-gyűjteményt, kiegészítve egy pontos és minden lépést magába foglaló teszt leírással, aminek segítségével egy olyan kolléga is végre tudja hajtani a tesztelést, aki előzőleg nem volt bevonva a projektbe. Ezzel az erőforrás-hiány által okozott negatív hatást egy kissé mérsékeltük, hiszen maga a teszt végrehajtása megoldott – még ha nem is olyan precíz és pontos, mint az által lenne végrehajtva, aki teljes mértékben be volt vonva a projektbe.

Természetesen minden, az alaptervtől eltérő, kiegészítő tervnek van költsége, időbeni hatása is. Mind a megelőző, mind pedig a negatív hatást csökkentő tervek esetén érdemes időbeni és költségbeli kalkulációt is végezni – főképpen azért, hogy lássuk, a tervezett módosítást megéri-e alkalmazni, azaz nem kerül-e túl sokba?

Amit még szeretnék megjegyezni: nem kell kizárólagosan az egyik típusú tervet kidolgozni egy-egy kockázathoz. Lehet olyan is, hogy mindkét (megelőző és negatív hatást csökkentő) tervet elkészítjük, így biztosítva azt, hogy a kockázat előfordulása és hatása lehetőleg minimális legyen.

3.6 Incidensmenedzsment

A tesztmenedzsment egyik eleme az incidensmenedzsment, azaz a tesztelők által talált hibák rendszerezése, azok karbantartása, jelentése az üzleti elemzők és programozók felé. Ehhez tartozik még a jelentett hibák nyomon követése, esetleges újra tesztelése, illetve mindezeknek a menedzselése.

Mint azt már tudjuk, a tesztelés célja az, hogy a szoftverben található hibákat feltárjuk, valamint, hogy a szoftver minőségét mérjük. Ez utóbbi leginkább az agilis (Agile) típusú szoftverfejlesztési módszertan esetén elvárás.

Az előző definíció egyik eleme a „szoftverben található hibák”. Magyarul csak ennyi: hiba. De angol nyelven több szó is létezik arra az eseményre, ami egy, a szoftver általánostól eltérő, nem várt működését fogalmazza meg: *incident*, *issue*, *bug, fault/failure, mistake, anomaly*, *defect*, *error*, *problem*. Mindegyiket lehet úgy fordítani, hogy ’hiba’ – de a valós jelentésük egy kicsit árnyaltabb. Vizsgáljuk meg egy kicsit közelebbről ezeket a szavakat:

- Ha a programozó vét egy hibát programozás közben, azt *mistake*, vagy *error* szóval illetik. Ennek oka lehet a program belső struktúrájának nem megfelelő mélységű ismerete, vagy a memória-vezérlés túlterhelése, esetleg egy adat kiszámításának nem megfelelő módja;

- Ha a programozó egy hibát helyez el (*bug*, *defect*) a kódban, ami egy *'mistake'* program-szintű megjelenése;
- A tesztelő a program egy hibás részét futtatja (hiba = *bug, issue*);
- Egy jól megtervezett tesztelés során előre meghatározható az program működése: ezt hívjuk elvárt működésnek. Ha ettől eltérő működést tapasztal a tesztelő, azt hívjuk anomáliának (*anomaly*), vagy egyszerűbben: hibának;
- Amikor a tesztelő egy anomáliát vizsgál, meghatározva annak forrását, azaz megtalálva az alap hibát (*fault/failure*);
- Egy anomália vizsgálatának az eredménye a hibajelentés (*bug report, incident report, defect report, issue, problem report*).

Mindezen angol szavakra a magyar nyelvben egy, idegen eredetű gyűjtőszót használunk: incidens. Az incidens definíciója az ISO 20000 szabvány szerint egy olyan esemény, amely nem része a normál üzemeltetésnek, vagy működésnek, és a szolgáltatás vagy üzemelés minőségének csökkenését, vagy az adott szolgáltatás megszűnését okozhatja.

A sorok között azért érződik, hogy ez a megfogalmazás inkább az olyan rendszerekre igaz, amik már „éles" rendszerek, tehát át lettek adva a végfelhasználónak. Az ezeken a rendszereken észlelt incidenseket különböző súlyossági szintekre bontva különítik el, más és más csapatok foglalkoznak a különböző súlyosságú eseményekkel. Ennek a rendszernek az ismertetése nem tárgya ennek a tananyagnak.

Tesztmenedzsment szempontjából azonban fontos azt tudni, hogy a tesztelés során talált hibákat is rangsorolni kell. Erre azért van szükség, mert a fejlesztők – akármilyen jól dolgoznak is – nem tudnak minden megtalált hibával foglalkozni, csak azoknak egy részével. Természetesen azokkal a hibákkal fognak foglalkozni, amiknek a javítása a legtöbb hasznot fogja hozni – akár minőségi, akár anyagi szempontból. A talált hibákat súlyosságuk és sürgősségük szempontjából lehet a legkönnyebben rangsorolni.

- **Súlyosság (*Severity*):** a hiba hatásának a mértékét jelzi a rendszerre. Lehet üzleti szempontú (pl.: a megrendelések könyvelése nem működik, ezáltal üzleti veszteséget generál a hiba), vagy rendszer szempontú (Pl. az adott hiba hatására a rendszerben végtelen visszacsatolás, hurok jön létre, ami a rendszer teljes összeomlásához vezethet). Fokozatai:

- **Kritikus (Critical) –** a hiba hatására a teljes rendszer összeomlik, nagymértékű adatvesztés, vagy pénzbeli veszteség lép fel. A hibára nincs ideiglenes megoldás.
- **Súlyos (Major) –** a hiba hatására rendszeregységek állhatnak le, egy vagy több modul működésében zavarok következhetnek be, kisebb mértékű adatvesztés alakulhat ki. A hiba nem végleges, alternatív módja létezik a hiba ideiglenes kiküszöbölésére.
- **Mérsékelt (Moderate)** – a hiba nem okoz rendszerleállást, vagy adatvesztést, de a hatására a rendszer hiányos, helytelen vagy egymásnak ellentmondó adatokat szolgáltathat.
- **Kevésbé súlyos (Minor)** – a rendszer működését és használhatóságát nem befolyásolja, az okozott kár könnyen kiigazítható.
- **Kozmetikai (Cosmetic)** – a hiba nem okoz sem rendszerbeli, sem működésbeli fennakadást.

❖ **Sürgősség (Priority):** ez az érték a hiba megoldásának időbeliségét határozza meg, más szavakkal: milyen gyorsan kell az adott hibát kijavítani. Ennek a fokozatai:

- **Magas (High) –** a hiba javítását amilyen gyorsan csak lehet meg kell cselekedni, mivel a hiba komolyan befolyásolja a rendszer működését, vagy nagy hatással van a végfelhasználókra, esetleg arculatromboló hatása van. (Ilyen lehet például a cég honlapján egy tárgyi elírás, ami ugyan alacsony súlyossággal bír, hiszen a rendszert működésében nem befolyásolja, viszont mivel a honlapon jelenik meg a hiba, ezáltal mindenki számára látható, így a sürgőssége magas.)
- **Közepes (Medium)** – a hiba megoldását a normális fejlesztési ciklusban el lehet végezni.
- **Alacsony (Low)** – a hiba, bár irritáló, de a javítása várhat addig, amíg a súlyosabb hibákat ki nem javítják.

Visszatérve a tesztmenedzsmentre: a jelentett hibákat most már tudjuk rangsorolni. Sorrendbe tudjuk állítani – ha tudunk róla. Igen, ha tudunk a hibáról. Egy ici-pici apróságot azonban még elfelejtettem megemlíteni: a hibákat valahol regisztrálni kell, valahogyan nyilvántartásba kell venni, hogy nyomon követhető legyen. És hogyan lehet ezt megvalósítani? Nos, a legegyszerűbb megoldásoktól a legbonyolultabb szoftveres megoldásokig igen széles a választék.

A legegyszerűbb egy táblázatot használni. Megfelel erre akár egy MS Office, egy OpenOffice, vagy bármilyen táblázatkezelő szoftver. A legfőbb dolgok, amiket regisztrálni kell egy hibával kapcsolatban:

- **Egyedi azonosító** – ami alapján később hivatkozhatunk a hibára;

- **Észlelő személy** – aki a hibát észrevette, majd a hibajegyet rögzítette. Valószínűleg ugyanez a személy fogja ellenőrizni a javítást is.
- **Összegzés** – egy rövid leírása a hibajelenségnek;
- **Leírás** – részletes leírása a hibának, magába foglalva az újra előidézés lépéseit (részletesen);
- **Tesztkörnyezet** – pontos megnevezése, ha esetleg több teszt szerveren átnyúlik a tesztelendő folyamat, akkor mindegyiket érdemes megjegyezni. Valamint mindenféleképpen le kell írni a teszteléshez használt lokális gép paramétereit is – ha van ilyen;
- **Felelős személy** – az a személy, aki ki lett jelölve a hiba javítására;
- **Státusz** – a hibajegy státusza, állapota, azaz éppen milyen fázisban van (nyitott, fejlesztés alatt, tesztelés alatt, javítva);
- **Súlyosság**
- **Sürgősség**

Természetesen ez csak egy nagyon kicsi projektnél funkcionálhat – mármint az Excel-alapú incidensmenedzsment. Nagyobb, bonyolultabb projekteknél mindenképpen hasznos egy incidensmenedzsment megoldásra – lehetőleg szoftveres megoldásra – van szükség. A piac igen széles választékkal büszkélkedik ilyen szoftveres megoldások terén könnyű fajsúlyú, egyszerű megoldásoktól kezdve az egészen bonyolultakig. Íme, néhány példa[29]:

- Apache Bloodhound (Apache Software Foundation)
- AssemblaTickets (Assembla)
- Axosoft (Axosoft LLC)
- BMC Remedy Action Request System (BMC Software)
- Bontq (Bontq LLC)
- Brimir (Ivaldi)
- Bugzilla (Mozilla Foundation)
- Debbugs (Debian)
- FogBugz (Fog Creek Software)
- Fossil (D. Richard Hipp)
- GLPI (INDEPNET)
- GNATS (Free Software Foundation)
- Google Code Hosting (Google Code)

[29] Az eszközökről a http://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems oldalon bővebb információ is található (2015.03.31.)

- HP Quality Center (Hewlett-Packard)
- IBM Rational ClearQuest (Rational Software)
- IBM Rational Team Concert (Rational Software)
- JIRA (Atlassian)
- Kayako SupportSuite (Kayako)
- Launchpad (Canonical Ltd.)
- Liberum Help Desk (Doug Luxem)
- LibreSource (Artenum)
- MantisBT (nyílt forráskódú szoftver)
- Microsoft Dynamics CRM (Microsoft)
- org-mode (Carsten Dominik)
- OTRS (otrs.org)
- Pivotal Tracker (Pivotal Labs)
- Plain Ticket (Turbine interactive)
- Planbox (Planbox)
- Redmine (Jean-Philippe Lang)
- Request Tracker (Best Practical Solutions, LLC)
- Roundup (Ka-Ping Yee, Richard Jones)
- StarTeam (eredetileg Starbase Corporation, jelenleg Borland)
- Supportworks (Hornbill Systems)
- SysAid (SysAid Technologies)
- Team Foundation Server (Microsoft)
- Teamwork (Open Lab)
- TechExcel's DevTrack (TechExcel)
- TestTrack (Seapine Software)
- The Bug Genie (nyílt forráskódú szoftver)
- Trac (Edgewall Software)
- TrackerSuite.Net (Automation Centre)
- Web Help Desk (MacsDesign Studio, LLC)
- Wrike (Wrike, Inc.)
- YouTrack (JetBrains s.r.o.)

- Zoho BugTracker (Zoho Corp.)

Ezen szoftveres megoldások bármelyike jó választás lehet, figyelembe véve az adott projekt, vállalat érdekeit, valamint a hosszú távú terveket. Nincs egyértelműen legjobb megoldás, hiszen mindenkinek más és más a fontosabb.

Bármelyik megoldást is választjuk, arra mindenképpen érdemes odafigyelni, hogy hosszú távon, azaz a későbbi projekteknél is hasznosítható, használható legyen ez a megoldás. Arra is érdemes gondolni, hogy a tesztmenedzsmentet később szintén valami szoftveres megoldással váltsuk ki, amit esetleg összeköthetünk az incidens menedzsment szoftverünkkel. Így közvetlenül a hibajegyekhez kapcsolhatjuk majd az egyes teszteseteket, valamint a követelmények tesztesetekkel való lefedettségét is pontosan mérhetjük, hiszen a követelményeket szintén hozzákapcsolhatjuk a teszteseteinkhez.

4 Tesztelő csapat

Nagyon sok dologról beszéltem idáig: tesztelési technikákról, stratégiákról, mindenféle eszközökről, hibakezelésről, ilyen-olyan metrikákról. Volt benne könnyebb hangvételű, érthetőbb rész, meg volt bizonyára nehezen emészthető szakasz is. Szóval sok minden. Csak arról, azokról nem beszéltem még, akik mindezt használják, akik ezen dolgok előnyeit élvezik, hátrányait, kínjait elszenvedik. Arról a csapatról, azokról az emberekről, akik a tesztelést megtervezik, irányítják, végrehajtják, monitorozzák. A Tesztelőkről. Igen, nagy kezdőbetűvel, vagy akár csupa nagybetűkkel: NEMECSEK ERNŐ. Vagyis: A TESZTELŐKRŐL!

Engedd meg nekem, kedves Olvasó, hogy ezt a hibát most kiküszöböljem.

4.1 A Csapat

Elsősorban szeretném, ha mint egységes egészről ejthetnék egy pár szót. A csapat ugyanis több, mint az egyes emberek, vagy szerepek összessége. Mint azt az alaklélektan alapjaiból is tudjuk: az egész mindig több, mint a részek összessége és prioritást élvez a részekkel szemben.[30] Így a tesztelői csapat is több, mint a tesztmenedzser, a tesztmérnök és a tesztvégrehajtó hármasa. Nem elég, ha mindannyian professzionálisan végzik a munkájukat, a legpontosabban betartják az utasításokat és maximálisan a „nagykönyv” szerint járnak el. Van valami, amit nem lehet megvenni, nem lehet kierőszakolni: a csapatszellem. Ha ez nincs meg, akkor hiába specialista minden tag, nem fognak tudni egy csapatként dolgozni.

[30] Alaklélektan, vagy Gestalt-pszihológia. Forrás: http://hu.wikipedia.org/wiki/Gestaltpszichol%C3%B3gia – 2015.04.07

Egy jó csapatban mindenki tudja, hogy mi az, amit hozzá tud tenni a siker eléréséhez. Nincs olyan, hogy valaki jobb, vagy rosszabb. Nincs alá-fölérendeltségi viszony – maximum papíron, vagy bizonyos extrém esetekben. De akkor sem olyan komolyan és határozottan érződik, mint az élet más területén. (Szándékosan nem akarok megemlíteni sem céget, sem területet – nem szeretnék semmit és senkit bírálni.) A tesztelői csapatban mindenki egyenlő – azaz egyenlőnek kellene lennie.

Vessünk egy pillantást most a csapat „egyes elemeire", azaz a tagokra.

4.2 Tesztelő (Tesztvégrehajtó) – a kubikos

Nem véletlen a fejezet címében megjelenő „kubikos" szó. Pár fejezettel ezelőtt már tettem említést a hódmezővásárhelyi kubikosokról – igaz, nem a legszebb oldalukat mutattam be. De az is igaz, hogy a kubikos munka volt az egyik legnehezebb, viszont ennek ellenére bizalmi munka a XIX. századi Magyarországon. Miért mondom én ezt? Azért, mert a kubikos volt az, aki a nehéz, szikes, vagy agyagos talajt feltörte, a feltört földet egyik helyről a másikra szállította azért, hogy töltést építsenek az árvizek ellen, vagy a vasúti sínpár számára. Sőt, még az országház alapját is ezek a kubikosok ásták ki. Gépek nélkül… Ha őket nem fizették meg rendesen és csalásra kényszerültek – lásd aládúcolt töltések – akkor az később nagy kárt okozott. Ezért volt bizalmi állás: megbíztak bennük, hogy a töltés, amit ők készítettek olyan tömör lesz és stabil, ami elbírja a mozdony és a szerelvények súlyát, vagy éppen az áradó Tisza, vagy Duna vad hullámait.

A tesztelő, tesztvégrehajtó hasonlóan bizalmi, de nehéz munka. Látszólag semmi más dolga nincs, mint hogy az előkészített teszteket végrehajtsa, majd az eredményt jelentésében leírja, a talált hibákról pedig hibajegyet készítsen.

Azért ennyire nem egyszerű a dolga.

Valóban, a tesztvégrehajtó munkája akkor kezdődik, amikor a tesztelési ciklus is elkezdődik. Optimális esetben a tesztelési ciklus kezdetére megkapja a végrehajtandó teszteset-gyűjteményt a hozzá tartozó tesztleírásokkal. Tisztában van a teszttervvel, az elvárásokkal. Minden tesztesetnél szerepel az adott esetre jellemző elvárt működés.

Amint elkezdődik a tesztelés – ismételten optimális esetről beszélek – a tesztvégrehajtó szépen sorban elkezdi végrehajtani az egyes teszteseteket. Ha hibát talál – tulajdonképpen először még csak anomáliának nevezzük – akkor megpróbálja azt újra előidézni. Ha többször is sikerül előidézni ugyanazt a hibát, akkor a tesztelő leírja a hibához vezető lépések pontos sorrendjét. Van, amikor több oldalról, több úton is el lehet jutni az adott hibához, elő lehet idézni ugyanazt a hibás eseményt. Ilyen esetekben a tesztelőnek ezeket az utakat is jelölnie kell. Ha esetleg valamit felfedezett, úgy értem van javaslata, hogy

a hibát, a konkrét hibát hol kell keresni, vagy netalán van ötlete a javításra, akkor azt is leírja az adott hibajegyben. (A hibajegy részletes leírását megtalálod a 3.6-os fejezetben.)

Miután a teljes teszteset-gyűjteményt végrehajtotta, minden hibát lejelentett, készít egy tesztjelentést (test report). Ez a jelentés tartalmazza a következőket:

- A tervezett és a valós tesztelési időt;
- A tervezett és a végrehajtott tesztesetek számát;
- Az újra futtatott tesztesetek számát;
- A talált hibák számát, típusát;
- A tesztelési ciklus alatt esetlegesen javított hibák számát;
- Minden, a tesztkörnyezettel kapcsolatban észlelt hibát;
- A tesztvégrehajtó véleménye a tesztelt termék minőségéről.

Ha úgy hiszed, vagy gondolod, hogy ezzel be is fejeződött a tesztelő feladata, akkor nagyon tévedsz. Van még valami, ami nagyon fontos: a tesztvégrehajtó a tesztelés során igen mélyen megismeri a tesztelt alkalmazást vagy rendszert. Jobban, mint a programozó, vagy az üzleti elemző, aki „megálmodta” az alkalmazást. Jobban, mint a két említett projekt résztvevője együttvéve – felhasználói szemszögből. Éppen ezért tudása nagyon sokat jelent – és nagyon hasznos azok számára, akik a felhasználói kézikönyveket írják a tesztelt alkalmazáshoz/rendszerhez. Szóval miután a tesztekkel végzett, a tesztvégrehajtó nekilát és elkezdi írni azokat a dokumentumokat, amik egy része, vagy teljes egésze a majdani felhasználói kézikönyv része lesz.

Amint azt már mondtam, mindez csak akkor van így, ha az ideális esetet tekintjük. Vannak olyan események, amik hatására az „egyszerű” tesztvégrehajtónak sokkal több dolga, feladata keletkezik, mint ami eddig leírtam.

Ilyen eset lehet az, ha nem minden követelmény, vagy technikai leírás jutott el a tesztmérnök kezébe. Ebből az következik, hogy bizonyos funkciók, vagy esetek lefedésére nem készült teszteset. Jobb esetben ezt a tesztvégrehajtó észreveszi, s még jobb esetben a teszt végrehajtója ismeretei alapján újabb teszteseteket készít azért, hogy ezeket a hiányosságokat lefedje. (Halkan megjegyezném: ez már tesztmérnöki feladat.)

Egy másik ilyen eset lehet az, amikor egy anomália kivizsgálása során egy olyan elemi, vagy alapvető hibát talál, ami meggátolja a teljes teszt további végrehajtását. Ilyenkor ismét a tesztvégrehajtó

tapasztalatán és szakmai tudásán múlik, hogy csak az adott hibás funkció által blokkolt eseteket nem hajtja végre, vagy megvizsgálja a teszteseteket, majd amennyiben talál még olyan esetet, ami a hibától függetlenül végrehajtható, esetleg egy kis különmunkával a hibás rész kikerülhető, akkor azon teszteseteket végrehajtja, hogy időt takarítson meg. Ez utóbbira – különmunkával végrehajtható teszteset – példa lehet, ha egy folyamat elején a letároláskor a dátum formátum nem magyar típusú (éééé-hh-nn), hanem európai (nn-hh-éééé), holott a felhasználói felület a magyar formátumot igényel, akkor az adatbázis módosításával ideiglenesen kiküszöbölhető a probléma. Így a folyamat további része tesztelhető.

Tudom, ez két eléggé tipikus példa, de ezeken kívül még sok olyan tényező zavarhatja meg az általam leírt optimális folyamatot, aminek hatására a teszt végrehajtónak néha tesztmérnöki „kalapot", néha pedig tesztmenedzseri „kalapot" is kell viselnie. (A „kalap" kifejezés alatt szerepköröket értek.)

A fejezet elején a kubikosmunkával hasonlítottam össze a tesztvégrehajtó feladatát. Külön kiemeltem, hogy bizalmi feladat – és ezt továbbra is fenntartom. De érzem, hogy egy kevés magyarázatot igényel, annak ellenére, hogy most már talán van némi sejtésed arról, hogy miért is mondtam ezt.

Szóval azt állítottam, hogy a tesztvégrehajtás bizalmi feladat. Annak tartom, hiszen a végrehajtáskor lehet a legnagyobb hibát véteni. Minden addigi munka kárba veszhet, ha a végrehajtás hiányosságot szenved. Ezért nevezem én bizalmi feladatnak: mindenki megbízik a teszt végrehajtójának ítéletében, annak precizitásában. E nélkül hogyan is fogadhatnák el a tesztjelentést, annak teljes tartalmát? A teszt végrehajtójának véleménye, ítélete mérvadó – arra alapozzák az adott alkalmazás, vagy modul további sorsát. Ha nem bízunk meg maradéktalanul a tesztek végrehajtójában, akkor a tesztelés ténye nem is ér semmit. Igen, lehetséges, hogy vannak tesztelőnek nevezett személyek, akik csak úgy tesznek, hogy minden tesztesetet végrehajtottak, majd átnyújtanak egy szép, cizellált jelentést olyan tartalommal, amit a megrendelő (tesztmenedzser, tesztmérnök, vagy projektmenedzser – akárki) látni szeretne… Őszintén remélem, hogy ilyenek csak elméletben, az én fantáziámban léteznek. Ugyanis ha ez nem így van, akkor vannak emberek, akik miatt alkalmazások, szoftverek kerülnek végfelhasználók kezei közé úgy, hogy nincsenek letesztelve, azaz nincsenek feltárva hibáik… Remélem így már érthető, hogy miért neveztem bizalmi feladatnak a teszt végrehajtását.

4.3 Teszttervező – a mérnök

A következő állomás: a teszt tervezése. (Tulajdonképpen előző állomás, ha a teljes tesztelési feladatkört tekintjük. ☺) A teszttervezőket másképpen tesztmérnököknek (Test Engineer) is nevezik – mindezt azért,

mert valóban mérnöki feladatokat látnak el. Na, nem a klasszikus értelemben, logarléccel és papírral-ceruzával – nem, ez azért túlzás. De feladatuk nagyon hasonlít a mérnökökéhez. Lássuk, mik is azok (ismételten az optimális esetet veszem alapul):

Egy tesztmérnök alapvető feladata elkészíteni a teszteset-gyűjteményt a hozzá tartozó teszt leírással, valamint segíteni a teszterv létrehozásában, vagy – bizonyos esetekben – saját teszterv elkészítésében. Ez is olyan egyszerűnek hangzik, igaz? Nos, ez sem annyira egyszerű, mint ahogyan hangzik. Először is, mi mindenre van szüksége feladata elvégzéséhez egy tesztmérnöknek? Lássuk:

- *Projekt dokumentáció*, ami magában foglalja a projekt mérföldköveit, a projekt leírását, tervét, stb.
- *Felhasználói esetek (user stories)* – olyan esetek, amik a termék átlagos használatát mutatják be;
- *Követelmények, üzleti és technikai* egyaránt. Ez képezi a teszteset-gyűjtemény alapját;
- *Technikai tervek*, azaz az a dokumentum, ami leírja, hogy hogyan fogja a programozó megvalósítani a követelményekben foglaltakat;
- *Felhasználói felület tervei* (ha van ilyen), ami jó alapot biztosít a felhasználó átvételi tesztesetek létrehozásának.

Ezen információk alapján kezd el dolgozni a tesztmérnök. A legelső feladata teljes mértékben átlátni és megérteni a projekt tervét. A mérföldköveket, az időbeli határokat, kockázatokat, környezetet, stb. Ennek alapján tud segíteni a teszterv létrehozásában is. Igaz, hogy a teszterv elkészítése elsősorban *nem* a tesztmérnök feladata, de igen gyakran segít annak létrehozásában. Sok esetben a tesztelendő szoftver megköveteli, hogy ne csak egy, hanem több teszterv is készüljön. Ilyen esetekben a tesztmérnökök is létrehoznak egy-egy, a fő tesztterhez alkalmazkodó, de a kisebb modulokra, egységekre érvényes tesztterv. Így a tesztmérnök feladatai közé kell, hogy soroljuk a teszterv elkészítését is. A teszterv tartalmának részletes leírását a következő fejezetben fogom megejteni. Most koncentráljunk inkább a tesztmérnök legfőbb feladatára: **a teszteset-gyűjtemény elkészítésére**.

A tesztesetek létrehozása többlépcsős folyamat. (A különböző tesztterv ezési technikákat, azok leírását megtalálod a „Szoftvertesztelés a gyakorlatban” című tananyagban[31].) Az első feladata a tesztmérnöknek a **követelmények átvizsgálása (requirement review)**. Ez a tesztelés első lépése. Ahogy már említettem, a tesztmérnök a projekt dokumentumokból, valamint a tervekből és felhasználói esetekből dolgozik.

[31] Boda-Bodrogközi-Gál-Illés: Szoftvertesztelés a gyakorlatban, 2014.

Mielőtt azonban a követelményeket felhasználná, megvizsgálja, hogy megfelelő-e, azaz nem tartalmaz-e hibát.

Milyen típusú hibákat tartalmazhat egy követelményrendszer, amit egy vagy több tapasztalt, hozzáértő ember állított össze a legjobb tudása szerint? A paletta elég széles – de előre kikötöm: **NEM** azért keletkeznek ezek a hibák, mert a követelmények kidolgozásáért felelős személyek nem értenek hozzá, hanem azért, mert egy ilyen dokumentum akár több száz oldalas is lehet. Mire azokat az oldalakat megírja/megírják a felelős személy/személyek már elfáradnak, s ezért átsiklanak bizonyos hibákon. Szóval nézzük meg ezen hibák típusait:

- **Ellentmondás** – a követelmények elkészítése, leírása hosszadalmas folyamat. Ez a folyamat napokig, hetekig, akár hónapokig eltarthat. Ezen folyamat alatt a követelmények változhatnak, módosulhatnak. Olykor olyan, új követelmények ébrednek, amik ellent mondanak az eddigi követelmények egészének, vagy némelyikének. Hadd mondjak egy példát: az első, alapvető követelmény az, hogy az adott szoftver csak ablakban fusson, teljes képernyős módra ne legyen alkalmas. A későbbiekben azonban a megrendelő úgy gondolja, hogy a teljes képernyős módnak is van létjogosultsága, így ezt is beleíratja a követelményekbe. A dokumentum eddigre már több oldalas, így az első követelmény is benne marad. Ez azonban ellentmondás, mivel az első követelmény *kizárja* a teljes képernyős működési módot.
- **Nem eléggé részletes specifikáció** – azt hiszem, ezt nem nagyon kell fejtegetnem. Inkább rögtön egy példával illusztrálom: a programnak tudnia kell adatbázissal kommunikálni. Nos, ez eléggé tág fogalom, mármint az „adatbázis” fogalma. Nem mindegy ugyanis, hogy milyen típusú adatbázisról beszélünk. Ha ezt a követelményt egy programozónak le kellene fejlesztenie, biztos vagyok benne, hogy további kérdéseket vetne fel. De azok a válaszok már nem biztos, hogy szerepelni fognak a követelmények között. Ez viszont már probléma a tesztelés szempontjából.
- **Többszörös megfogalmazás** – másképpen fogalmazva: duplikált tartalom. Ez igen egyszerű dolog: miközben a követelményeket gyűjtik össze, néha többször is felmerül ugyanaz az igény. Van, hogy észreveszik az ismétlődést, de néha nem. Ilyenkor ugyanazt az igényt többször is megfogalmazzák, csak más módon. Ezeket a duplikátumokat is ki kell szűrni – nem, mintha valami nagy problémát okozna, hanem inkább azért, hogy a dokumentum átláthatóbb legyen.
- **Nem megvalósítható specifikáció** – ez nem túl gyakori hiba, de néha előfordul. Van, hogy a projekt egy adott szoftverre épül, ami már használatban van. A projekt egy fejlesztési projekt, aminek keretében az adott szoftvert a mai napi követelményeknek megfelelően, azok igényei

szerint kibővítenék. A baj csak az, hogy a régi szoftver nem képes bizonyos funkciók ellátására, még egyedi kódok beépítésével sem. Ez az, amit úgy neveznek: nem megvalósítható. Ha ilyet talál egy tesztmérnök, akkor azt kiemelten kezeli, hangsúlyozottan kiemeli jelentésében.

- **Visszafejlesztési javaslat** – ez is elég ritka, de néha előfordul. Nem az üzleti elemzők hibája, ha egy ilyen típusú követelmény kerül bele a fejlesztési javaslatokba. Erre elég nehéz igazán jó, egyértelmű példát mondani. Talán egy tartalomkezelő rendszer esetén, ahol különböző munkafolyamatok felügyelik az egyes tartalmak megjelenését, amely munkafolyamatokba több ponton is ellenőrzési és jóváhagyási procedúra. A fejlesztés az lenne, hogy ezeket a bonyolult folyamatokat leegyszerűsítsék, néhány ellenőrzési pontot kivegyenek, jóváhagyási procedúrát megszüntessenek. Ezzel csak az a probléma – jelen esetben – hogy azokat az ellenőrzési pontokat azért tették bele, hogy biztosítsák a megfelelő minőségű tartalmakat. Ha ezeket megszűntetik, akkor az visszafejlődés a minőség tekintetében.

Nem szeretném azt mondani, hogy csak ezek a típusok fellelhetőek – inkább azt mondanám, hogy ezek a leggyakoribbak.

Amint a tesztmérnök a követelmények vizsgálatával végzett, már van egy alapkoncepciója a tesztelés „hogyan?”-járól – azaz hogy az adott projekthez melyik tesztelési techniká(ka)t fogja használni. A követelmények átvizsgálásakor már készít jegyzeteket, talán kész teszteseteket is megír. Ez lesz az alapja a felhasználói átvételi tesztnek (User Acceptance Test).

A következő feladatat a tesztmérnöknek a **tervek (design) átvizsgálása**. A tervek adják az alapját az integrációs és a rendszer szintű teszteléshez készített teszteset-gyűjtemények létrehozásához. Mind a technikai, az architekturális, mind pedig a felhasználói felület terveit át kell néznie, hogy abban nincsenek-e véletlenül hasonló, rejtett hibák, mint a követelményeknél. Ezen kívül megvizsgálja azt is, hogy a tervezett rendszert, felépítést és felhasználói felületet hogyan lehetne tesztelni, milyen tesztesetekkel lehet lefedni. Ezekkel az információkkal már egy eléggé komplex tesztesett-gyűjteményt tud létrehozni a tesztmérnök, amit a tesztvégrehajtó majd a megfelelő szakaszokban, tesztelési fázisban használni tud.

A **végleges teszteset-gyűjteményt** a projekthez azonban akkor tudja elkészíteni, ha megkapta a programozók által végrehajtott egységteszteket, illetve azok eredményeit. Ez ugyan már csak apróbb simításokat jelent, hiszen a munka oroszlánrészét már elvégezte, de mégis fontos. Lehet ugyanis, hogy a programozók által végrehajtott tesztesetek olyan területekre mutatnak rá, amit mindenképpen sokkal

részletesebben kell tesztelni – akár a stabilitás hiánya, akár a hibák gyakorisága, vagy esetleg más, egyszerűbb ok miatt – mint ahogyan azt az előzetes dokumentumok alapján a tesztmérnök eltervezte. De az is előfordulhat, hogy egy modul, amit nagyon kockázatosnak vélt a tesztmérnök az előzetes dokumentációk alapján kiderül, hogy sokkal sikeresebben működik, sokkal stabilabb, mint azt remélni lehetett. Ami annyit jelent, hogy kevesebb teszteset is elegendő a teszteléséhez.

A következő feladata már csak az első tesztelési kör után következik: **összeszedi, összegyűjti a tesztjelentéseket, összegzi azok eredményét**. A végrehajtott tesztesetekből kiemeli azon eseteket, amelyek a regressziós tesztekhez hasznosak lehetnek. Egészen pontosan: **létrehozza, vagy aktualizálja, frissíti a regressziós teszteset-gyűjteményt**. Ez szintén elég fontos lépés a projekt egészére nézve – legfőképpen akkor van ennek jelentősége, amennyiben a projekt egy több szakaszból, több fejlesztési ciklusból álló, hosszadalmasabb folyamat. Ilyen esetekben ugyanis mindig javallott a már letesztelt modulok újratesztelése annak érdekében, hogy az új kódok, kódrészek illetve modulok nehogy valami olyan hibát okozzanak, aminek hatására a már letesztelt részekben hiba keletkezzen.

Van még egy fontos feladata a tesztmérnöknek a tesztjelentések összegzése és kiértékelése során: a megtalált és jelentett hibák nyomon követése, valamint azoknak a lehető legnagyobb mértékű újratesztelésének megtervezése. Ehhez szüksége van természetesen a tesztvégrehajtó jelentésére, a hibajegyekre, valamint a fejlesztők technikai megoldásainak, vagy hibafeltárásának a leírására. Ez utóbbiak – technikai megoldás, hiba tényleges forrásának (root cause) feltárása – azért fontos, mert ezek alapján ki tudja a tesztmérnök szűrni, hogy az adott hiba még hol lehet esetleg hatással a rendszer vagy folyamat későbbi szakaszában.

A tesztmérnök feladata még a fentieken kívül a **tesztadatok gyűjtése vagy azok generálása**. Mit is jelent ez? Ez nagyon egyszerű: vannak olyan tesztesetek, amihez szükség lehet valamely, a rendszer által már korábban letárolt adatok lehívására, vagy valószerű adatok tárolásával imitáljuk a rendszer végleges használatát. Ilyen lehet az a teszteset, amikor egy adott termékhez bizonyos extra szolgáltatásokat vásárol az ügyfél, mint pl. online segítségnyújtás, vagy a honlapon található fizetős tréninganyagok és/vagy egyéb multimédiás anyagok. Ezen megvásárolt extrákat a honlapon a bejelentkezés után érheti el a felhasználó, tehát rendelkeznie kell egy érvényes bejelentkezői azonosító/jelszó párossal. Ehhez a pároshoz rendeli hozzá a szolgáltató ezeket az extra hozzáféréseket. Ennek a letesztelését – hogy valóban megtörténik-e a jogosultságok kiosztása – csak egy megfelelő felhasználónévvel, azaz felhasználói fiókkal lehet véghezvinni. (Valójában nem egy, hanem több, a jogosultsági szintektől és azok variálási lehetőségétől függően.) Ennek az egyik lehetséges módja felhasználói fiókok generálása a

tesztrendszeren, majd ezekhez a fiókokhoz a megfelelő jogosultság – értsd: megvásárolt extra szolgáltatások – hozzárendelése. Elég gyakran ezt egy automatizált eszközzel valósítja meg a tesztmérnök, aminek több előnye is van – de ez már nem tartozik a témánkhoz.

Vannak azonban olyan esetek, amikor a feltétel nem engedi meg a tesztciklus előtti generálást, mert olyan feltételek teljesülését kellene biztosítani, amihez az adott felhasználói fióknak már évek óta a rendszerben kell lennie. Ilyen feltétel lehet például az, hogy 4-5, vagy több éves legyen az adott fiók. (Pl. hűségpontok, stb.). Ilyen esetben nincs lehetősség a generálásra, ilyenkor a rendszerben – az éles rendszerről másolt, de az ügyfelek valós adatait módosított – táblákból, adathalmokból kell megfelelőt előkeresni. Ezt nevezzük „adatbányászatnak", vagy „adatvadászatnak". Igen, kissé viccesnek hangzik, de néha tényleg szinte vadászni kell a megfelelő adatokra, vagy igen mélyről kell előszedni, szinte bányászni a megfelelő tesztadatot.

Az előbb említettem a tesztadat-generálás kapcsán a tesztautomatizálást, mind lehetséges megoldását a tesztesetek generálásának. Nagyon nem lennék igazságos, ha ennyiben hagynám, ugyanis a tesztmérnök egy igen fontos feladata az **automatizált tesztelés meghatározása** az adott projekten/tesztelési cikluson belül. Ez azt jelenti, hogy a tesztmérnök, miközben a tesztelési ciklusok terveit, teszteset-gyűjteményét készíti, már azt is figyeli, hogy hol és milyen módon lehetne a manuális teszt végrehajtást automatikussá tenni. Ez nem azt jelenti, hogy a tesztmérnök fogja automatizálni, azaz elkészíteni az automatikusan futtatható szkripteket, hanem azt jelenti, hogy a **lehetőségeket** gyűjti össze. Ezen lehetőségeket aztán később egy tesztautomatizálási specialistával megbeszéli és – amennyiben szükséges – további adatokat is gyűjt annak érdekében, hogy az automatizálási folyamat minél gördülékenyebben folytatódjon. Lehetséges, hogy némely egyszerűbb automatizálási feladatot egy tesztmérnök is végre tud hajtani, de a tesztautomatizálás egy különálló tudományág a tesztelésen belül. Ennek részleteit lásd Kundra Zoltán *Tesztautomatizálás* című tananyagában.

Mint ahogyan már említettem, a tesztmérnök legfőbb feladata a tervezés. Viszont vannak helyzetek, alkalmak és szituációk, amikor egy tesztmérnöknek is *tesztvégrehajtóként kell viselkednie*. Vagy éppen a két szerepet – emberi erőforrás hiányában – egy és ugyanaz a személy „játssza el". Bármelyik eset is áll fenn, a tesztmérnöknek néha tesztelnie is kell, azaz tesztvégrehajtóként is kell dolgoznia.

4.4 Tesztmenedzser – a főnök

Visszatérnék a „projekt a projektben" elméletemhez – vagy nevezzük tételnek – azért, hogy egy kicsit jobban megvilágítsam a tesztmenedzser helyzetét: Ahogyan a projekteknek is van vezetője (menedzsere), úgy a tesztelésnek is kell lennie egy vezetőjének. Ő az, aki a projekt életében vezetői

szinten vesz részt, azaz a szűk vezetői megbeszéléseken (core team meeting) képviseli a tesztelő csapat érdekeit. Ez nem azt jelenti, hogy megjelenik, néha szól egy-két jó szót, vagy éppen keresetlen mondatot és kész is. Nem, ennél sokkal többet jelent.

A **projekt kezdeti szakaszában** – a projekt tervezésénél – ő *már jelen van*. Az alapvető információk alapján (projekt felterjesztések, előzetes tervek) becsléseket készít: mind az erőforrások, mind pedig a várható tesztelési ablakok számának és hosszának meghatározása tekintetében. Ezeket a becsléseket felhasználják a teljes projekt tervezésekor, a mérföldkövek megállapításánál, az idővonal kialakításánál. A továbbiakban ezt az alapot felhasználva – viszonyítási alapként – fogja a megfelelő erőforrás-mennyiséget a teszteléshez rendelni. Kezeli az esetleges változásokat, biztosítja az információ áramlását minden irányba. (A projekttől a tesztelői csapat irányába és vissza.)

A következő fázisnál – a tervezés – ő az, aki elkészíti az adott projektre, termékre vonatkoztatott teszttervet. Ha eléggé komplex a projekt, esetleg egy programról beszélünk, akkor a fő teszttervet készíti el (Master Test Plan), ami a következőket tartalmazza:

- ***A teszterv azonosítója*** – fontos, főleg, ha több projekten is dolgozik párhuzamosan. Meg a visszakereshetőség szempontjából is igen hasznos tud lenni.
- ***A teszterv bemutatása*** *(introduction)* – röviden leírva, hogy mi is ez a dokumentum, miért, kik számára jött létre, milyen irányvonalakat fektet le.
- ***A tesztelés tárgya*** *(test items)* – a tesztelendő termék (szoftver, rendszer, stb.) megnevezése, rövid leírása. (pl. komplexitás, modul rendszere, célfelhasználók megnevezése, stb.)
- ***Tesztelendő funkciók*** *(in scope functions)* – azon funkciók bemutatása, amiket az adott csapat tesztelni fog. Ez a rész nem mindig határolható be pontosan. Ha ugyanis egy komplex rendszer egy kis részét kell tesztelni, ami egy folyamat alkotóeleme, akkor nehéz meghatározni azokat a pontokat, ahol még és ahol már nem az adott csapat feladata és felelőssége a tesztelés.
- ***Nem tesztelendő funkciók*** *(out of scope functions)* – azon funkciók felsorolása, amit biztosan nem ez a csapat fog tesztelni. Ilyen lehet esetleg egy webes alkalmazásnál a szerveroldali tesztelés, pl. teljesítmény-tesztek futtatása.
- ***Teszt megközelítés, teszt stratégia*** *(test approach, test strategy)* – erről a 3. Fejezetben már beszéltem.
- ***Belépési-kilépési feltételek*** *(entry and exit criteria)* – mik azok a feltételek, amiknek teljesülnie kell ahhoz, hogy a tesztelési fázis elkezdődhessen, valamint azok a feltételek, amiknek mindenképpen teljesülnie kell ahhoz, hogy a tesztelési ciklust lezárhassuk.
- ***Sikerességi-sikertelenségi feltételek*** *(passed/failed criteria)* – azok a feltételek, amik meghatározzák, hogy egy adott modul sikeresen „átment" a tesztelésen, avagy elbukott. (pl. a tervezett tesztesetek 80%-a sikeres, akkor a modul sikeresen átment a tesztelésen.[32])
- ***Felfüggesztési feltételek és visszatérési követelmények*** *(Suspension criteria and resumption requirements)* – mik azok az esetek, körülmények, amelyek nem bukottá, hanem csak felfüggesztetté minősítenek egy-egy tesztelést. Ilyen lehet pl. áramszünet, vagy szerverek nem

[32] Ez egy nagyon egyszerű feltétel, ami önmagában nem állja meg a helyét. Általában több feltétel együttes teljesülése adja meg a sikerességet.

tervezett leállása, stb. Ugyanígy meg kell adni azt is, hogy mik azok a körülmények, amiknek hatására a felfüggesztett tesztelést folytatni lehet.

- ***Tesztelési teljesítések/dokumentumok*** *(test deliverables)* – mindenképpen le kell írni, hogy milyen teljesítéseket vállal a tesztelői csapat. Azt is érdemes belefoglalni, hogy mi az, amit NEM vállal.
- ***Tesztelői feladatok*** *(testing tasks)* – úgy, ahogyan a teljesítéseket és dokumentációt leírtuk, azt is rögzíteni kell, hogy milyen feladatokat vállal a csapat. Van úgy, hogy csak a tervezést és a koordinálást kell elvégezni, mert a tesztek végrehajtását mások végzik: a végfelhasználók, vagy éppen külső tesztelők.
- ***Környezeti szükségletek*** *(environmental needs)* – azaz a tesztkörnyezet, amin tesztelni fog a csapat. Mikorra és milyen állapotba kell lennie. Melyik tesztelési ablakban milyen integrációs szint szükséges. Nagyon fontos ennek is a rögzítése a gördülékeny tesztelés érdekében.
- ***Szerepek és felelősségek*** *(Roles and Responsibilities)* – bármennyire is egyértelműnek tűnik, minden teszttervben újra és újra le kell írni. Nem szabad azt feltételezni, hogy mindenki tisztában van a lehetséges szerepekkel, azok jelentésével, valamint a szerepekhez tartozó felelősségi körökkel.
- ***Csapatösszetétel és esetleges képzési szükséglet*** *(staffing and training needs)* – mindenképpen rögzíteni érdemes (mondhatni szükséges) azt, hogy kik fognak tesztelni. Természetesen ez nem egy végleges állapot lesz, hiszen lehetnek változások, amiket a változásmenedzsment (change management) is rögzíteni fog, de egy alapnak lennie kell. Tulajdonképpen a „kezdő csapat" felállását kell rögzíteni. Ugyanakkor előfordulhat az is, hogy a tesztelőknek (akár mérnök, akár menedzser, vagy végrehajtó) szükséges valamilyen mértékű képzés az adott technológiából, vagy szoftveres megoldásból. Ezeket az igényeket is érdemes rögzíteni a tervben.
- ***Ütemterv*** *(schedule)* – a projekttervhez igazodva kell kidolgozni. Nagyon sok befolyásoló tényező lehet, mint például a fejlesztés ütemezése miatti szerverleállások, esetleg a beszerzések, de ugyanígy – főleg, ha multikulturális a tesztelői és/vagy a projekt csapat – a különböző kulturális szokásokból fakadó szabad- és ünnepnapok. Ez az ütemterv szintén módosulhat, hiszen nincs "kőbe vésve", de egy alapot mindenképpen meg kell fogalmazni. Legfőképpen azért, hogy mérni tudjuk az előrehaladást, vagy éppen a késedelmeket.
- ***Kockázatok és azok minimalizálása*** *(risks and contingencies)* – a projektmenedzser és csapata megcsinálta – optimális esetben – a projektre vonatkozó kockázat-elemzést. A tesztmenedzser feladata ugyanezt elvégezni a projekt tesztelésére vonatkozóan. Részletes leírását megtalálod a 3.5 fejezetben.
- ***Jóváhagyások*** *(approvals)* – ez az egyik legfontosabb rész: dokumentálni, hogy mindenki elfogadta és jóváhagyta a teszttervet. A mindenki alatt a projektmenedzsmentet értem. Ugyanitt kell jelölni a dokumentumban a későbbiekben végrehajtott főbb változásokat – illetve azok elfogadását – is.

Erre a teszttervre épít esetlegesen a tesztmérnök, aki a kisebb modulok, projektek teszterveit készít(het)i el – ahogyan azt az előző fejezetben már kifejtettem. (A 3.1.9-es fejezetben leírtak már a fő teszttervre épített dokumentum főbb pontjai.)

Innentől fogva minden, ami fontos és döntést igényel a tesztmenedzser „kezén” megy keresztül. Ez azt jelenti, hogy a tesztmenedzsernek tisztában kell lennie a projektek tesztelésének mindenkori állapotával, a tesztelés folyamatával, a dokumentáció készenléti fokával, stb.

A tesztelési ablakok alatt irányítja a munkálatokat, azaz kiosztja a tesztelési feladatokat, követi azok végrehajtását, figyeli a trend-adatokat, amikből következtetéseket von le. Ugyanakkor folyamatosan feltérképezi a meghatározott kockázati elemek előfordulási valószínűségének alakulását, esetleg azok hatásait a projektre. Amennyiben szükséges, megteszi a megfelelő lépéseket a kockázatok elkerülésére, vagy annak hatásának/hatásainak csökkentésére.

Folyamatosan tájékoztatja a projekt vezetését a tesztelésről, kapcsolatot tart a megrendelővel (amennyiben szükséges). Biztosítja az információ-áramlást mind a projekt résztvevői, mind pedig a tesztelő csapat irányába.

És még egy fontos dolog: minden felelősség, ami a tesztelést, annak folyamatáért, minőségéért, határidőinek betartását illeti – tehát tényleg MINDEN FELELŐSSÉG – a tesztmenedzseré.

5 Utószó

Kedves Kolléga. Remélem eljutottál eddig a bekezdésig, de úgy, hogy a köztes bekezdéseket is elolvastad. Bízom benne, hogy nem volt elvesztegetett idő számodra. Célom – mint azt a Bevezetésben is írtam – az volt, hogy a tesztmenedzsmentet a gyakorlati oldaláról mutassam be. Megmutassam, hogyan lehet a „Nagykönyv” szerinti elméleteket, modelleket a Gyakorlatban, az Életben is felhasználni. Minek mikor és hol lehet szerepe, hogyan lehet kihasználni és/vagy felhasználni azokat a technikákat, amiket leírtam. Tudom, ez a tananyag nem teljes – mindig lehet valami újat írni, mindig van valami, amit még be lehetne illeszteni. Ez a tanyag nem végleges – ez csak az első verzió. Talán, mire a végleges kiadását megérte, már nem is az első verzió, nem is az első lezárt kézirat lesz. A világon egyre több és egyre jobb szakkönyv jelenik meg, amik a tesztmenedzsment témakörét feszegetik. Új elméletek, módszerek jelennek meg, tételeket dolgoznak ki, majd döntenek meg. Szóval: a tesztmenedzsment is változik, akárcsak a világ körülöttünk. Ezeket a változásokat nagyon nehéz követni, főleg mivel egyre részletesebb, egyre szerteágazóbb lesz ez a tudomány is. De azért mi igyekszünk helytállni.

A végére még tartogattam egy-két apró jó tanácsot, ha elfogadod tőlem. Nem aranyigazságok, csak amolyan tippek, vagy trükkök, amit érdemes használni, vagy megfogadni:

- **Ne akard használni a különböző stratégiákat.** Úgy értem: nem minden projektben érdemes direkt módon használni. Van, ahol inkább csak, mint irányvonalat érdemes használni. Tehát: a kiválasztott stratégiák formai követelményeit, mint pl. a dokumentációk, nem mindig érdemes követelni.
- **Ne ragaszkodj a szabályokhoz.** Nem minden esetben lehet a szabályokat maradéktalanul betartani. Ennek oka lehet a projekt közege, kultúrája, esetleg a múltja (egy nagyon gyorsan fejlődő cég nehezen tér át a családias hangulatú projektekről, ahol mindenki mindent tudott, így nem kellett a szigorúan vett dokumentációval bíbelődni a komplex projektek közegébe), vagy más befolyásoló tényező. Ha „újonc” vagy egy projektben akkor érdemes előbb feltérképezni,

hogy „mit bír el” a projekt, a projektcsapat. Csak annyit érdemes rájuk ereszteni mindabból, amit ebben a tananyagban olvastál, amennyit meg tudnak emészteni. Majd idővel beléjük ivódik minden.

- **Mutass példát.** Sok helyen írtam, hogy ilyen, meg olyan dokumentumokat várunk a fejlesztőktől, az üzleti elemzőktől, a projektmenedzsertől, stb. Te, mint tesztmenedzser mindig járj elől jó példával: mindig legyen kész határidőre a minőségbiztosítási terv, a teszterv és a többi olyan dokumentum, amiért Te vagy a felelős. Nem biztos, hogy minden információ benne lesz, hiszen sok mindent a projekttől vársz, de így már meg tudod mutatni, hogy miért kell neked az a bizonyos dokumentum… Csak azt szabad másoktól elvárni, amit mi is teljesíteni tudunk.
- **Ne csak követelj, segíts is!** Előfordulhat, hogy egy olyan projektbe kerülsz, ahol tapasztaltabb vagy, mint a projektmenedzser. Ilyenkor érdemes inkább segíteni, mintegy mentorálni őt ahelyett, hogy folyamatosan csak az igényeinket (mondhatni követeléseinket) fogalmazzuk meg a megbeszéléseken. Sose mutass rá a hibáira – inkább segítsd annak kijavításában.
- **Minden projekt más.** Éppen ezért mindig elölről kell elkezdeni felépíteni a tesztmenedzsmentet is. Nem biztos, hogy az, ami az előző projektnél – bármennyire is hasonló – bevált, ennél az új projektnél is hatékony lesz.
- **A változás nem mindig rossz.** Már említettem, de nem győzöm hangsúlyozni: a terv sosem végleges. A tervet lehet, sőt, kell is változtatni, reagálva a környezeti hatásokra, változásokra. Alkalmazkodni a projekt nehézségeihez, körülményeihez nem szégyen, hanem ellenkezőleg: igen pozitív hozzáállás. Nem baj, ha változik a terv – a probléma ott kezdődik, ha a módosítások nem egyértelműek, nincs mindenki számára letisztázva, vagy esetleg nem hagyták jóvá a vezetők. Ez már lehet probléma – mégpedig igen nagy.

Volt Bevezetés, Tárgyalás, meg Befejezés (Utószó) is. Azt hiszem, megfelelek a formai követelményeknek. Remélem, hogy a Bevezetésben megígért dolgokat maradéktalanul teljesítettem, így nem okoztam Neked csalódást. Most már átadom a felelősséget Neked, Kolléga: használd ezt a net úgy, ahogyan a legjobb tudásod engedi. Kívánom, hogy egyszer, valahol, valamikor találkozhassunk, s kérdéseidre – ha vannak még – személyesen tudjak válaszolni. Addig is kívánok Neked minden jót az életben, profi projekteket és kellemes tesztelést.

A viszontlátásra!

6 Felhasznált szakirodalom

1. Rex Black: Pragmatic Software Testing, Indianapolis, Wiley Publishing Inc., 2007, ISBN: 978-0-470-12790-2
2. Rex Black: Managing the Testing Process, Second edition, Indianapolis, Wiley Publishing Inc., 2002, ISBN: 0-471-22398-0
3. Dorothy Graham, Erik Van Veenendaal, Isabel Evans, Rex Black: Foundation of software testing, ISTQB Certification, Cengage Learning EMEA, 2008, ISBN: 978-1-84480-989-9
4. Rex Black: Advanced Software Testing vol.2. , Santa Barbara, Rock Nook Inc., 2014, ISBN-13: 978-1-937538-50-7
5. Űreszközök a Marsnál 1960-2000, Távoli világok kutatói in http://www.urvilag.hu/ureszkozok_a_marsnal_1960_2000/20040101_a_mars_meghoditasa_6resz_ket_siker_negy_kudarc
6. Patrick Hendrickx, Chris Van Bael, Alain Bultink: Advanced Test Management – ps_testware nv, 2010 - ISBN: 978-90-9025727-3
7. HTB, Szoftvertesztelés egységesített kifejezéseinek gyűjteménye, 2013, ver. 3.13
8. Systemation: Fast Start in Project Management, 2014
9. Mark Utting, Alexander Pretschner and Bruno Legeard: A TAXONOMY OF MODEL-BASED TESTING, in Working Paper Series ISSN 1170-487X
10. Tóth Árpád: A modell alapú tesztelésről - http://www.tesztelesagyakorlatban.hu/keres_cikk.php?mit=3
11. Boda-Bodrogközi-Gál-Illés: Szoftvertesztelés a gyakorlatban, 2014.
12. A CRAMM-model - http://hu.wikipedia.org/wiki/Cramm-modell - 2015.03.30.