Matti Vuori

125 pointtia testaustyökalun hankintaan

Testauksessa käytettäviä työkaluja hankintaan usein ajopuuperiaatteella tai sattuman kauppaa. Niinpä lopputulos ei aina olekaan yhtä tehokas organisaation kannalta kuin systemaattisemmin tehty hankinta ja käyttöönotto. Tässä tekstissä on 125 ”pointtia” hankinnan tueksi.(Suurin osa pointeista muuten pätee kaikkiin muihinkin ohjelmistotyön välineisiin…)

Oikealla jalalla liikkeelle

1. Yleistieto ohjelmista auttaa aina. Riippuen organisaatiosta, myös testausohjelmistotarjonta voi olla yksi klassisen systemaattisen teknologianseurannan aiheita ja tuottaa sen, että omassa talossa on joku, joka osaa kertoa, mitä nykyään on tarjolla, mitä uusia mahdollisuuksia on, mihin kehitys on menossa ja miten kiinnostavat ohelmistot suhtautuvat toisiinsa.
2. Ohjelma on vain väline, eikä toimi yksinään. Hyväkään ohjelma ei sovi jokaiseen paikkaan, vaan edellyttää sopivia olosuhteita, ajattelumalleja, osaamista, pelisääntöjä jne…
3. Kokonaisuuden on oltava sopusoinnussa. Jos työtä muutetaan uudella välineellä, on väistämättä muutettava ainakin kahta muutakin asiaa – pitää selvittää, mitkä ne kulloinkin ovat.
4. Automaatioon pätee vanha viisaus: jotta jonkin prosessin voisi automatisoida, sen pitää olla hyvässä kunnossa manuaalisesti tehtynä.
5. Siksi testausohjelmien hankinta voi olla mielekkäintä kokonaisvaltaisen testauksen kehittämisprojektin osana.
6. Anna tilaa luovuudelle, uusille mahdollisuuksille ja sattumalle. Epätodennäköinen ohjelmisto voi ollakin juuri se puuttuva palanen loistavasta kokonaisuudesta!

Tiedätkö todelliset tarpeet?

7. Mistä impulssi hankintaan on peräisin? Haluatteko oikeasti parantaa jotain testauksen piirrettä vai luoda vakuuttavuutta hankinnoilla? Vai onko kauppias hurmannut jonkun sisäisen vaikuttajan?
8. Selvitä tarpeet ja vaatimukset. Keskustele kaikkien sidosryhmien ja ennenkaikkea testaajien kanssa, mitä oikeasti tarvitaan ja mitä vaatimuksia ohjelmalle on.

9. Erilaiset testausstrategiat ja paradigmat edellyttävät erilaisia työkaluja. Oletteko miettineet, millaista testauksenne on luonteeltaan ja millaisia välineitä se edellyttää?
10. Jos luovuus on tärkein menestystekijänne, ideointityökalu voi antaa enemmän etua kuin testitapausten automaattinen generoija.

11. Katso jo tarpeissa tulevaisuuteen – miltä maailma näyttää vuoden tai kahden päästä? Mihin menee organisaatio? Mihin tuoteteknologia? Mihin kehittämisteknologia?

12. Jos on kyse laajasta työhön paljon vaikuttavasta järjestelmästä, analysoi tarkasti kaikki sen vaikutukset toimintaan, ihmisiin, tiimeihin. Ota huomioon myös kaikenlaiset yhteisöpsykologiset tekijät.

Valinta useasta vaihtoehdosta

13. Anna välineen käyttäjien tehdä valinta. Mitä osallistuvampi prosessi on, sitä todennäköisemmin se onnistuu.
14. Tiedota, tiedota ja tiedota. Kaikenlaisessa kehittämisessä on tiedottaminen haukutuin asia. Se koskee myös välinehankintoja.
15. Selvitä vaihtoehdot. Ilmiselvin ja ”jonkun kehuma” tai mukanaan muualta tuoma ehdokas ei ole aina paras mahdollinen.
16. Vaihtoehdot kannattaa arvioida ja pisteyttää, jotta esimerkiksi valintaryhmän ryhmädynamiikka ei ohjaa ajatuksia ja osataan ottaa kaikki asiat huomioon niiden oikealla vakavuudella.
17. On tärkeää tunnistaa kaikki pienetkin pakolliset ohjelman ominaisuudet, joista ei voida tinkiä.
18. Erota toisistaan pakolliset vaatimukset, joiden on pakko täyttyä, olivatpa ne miten pieniä tahansa, ja lisäetua ja vaihtoehtojen eroja tuottavat vapaaehtoiset asiat.
19. Pisteytysmenetelmän pitää ennenkaikkea näyttää olennaisuudet, eikä kadottaa niitä kriteerien metsään. Pisteytä, vertaile ja keskustele korkeintaan 10 isoa kriteeriä, mutta anna ihmisten tsekata, että kaikki välttämättömät nippelitkin toteutuvat.
20. Tiedonhaku ohjelmaan liittyvistä kokemuksista on oleellista. Google on tässä perustyökalu, mutta sellaista ohjelmaa ei kannata edes harkita, jolla ei ole kohtuullisen aktiivista foorumia tai edes sähköpostilistaa tiedusteluihin ja keskusteluarkiston tutkimiseen.
21. Kumppaniverkosto on olemassa juuri tällaista yhteistyötä ja kokemustiedon jakamista varten.
22. Puhumattakaan esimerkiksi testausalan foorumeista ja järjestöistä.
23. Teetä jo alustavista arvioinneista tiiviit raportit, koska tällaiset faktat kiinnostavat monia.

Ohjelman soveltuvuus tärkeimpään tarkoitukseensa

24. Tunnista softan rajoitukset – mitä se tekee ja mitä ei; mitä työkaluja se korvaa ja mitä ei; mitä manuaalisia työvaiheita se korvaa ja mitä ei.

25. Virheitä etsivien ohjelmien riittävä kyvykkyys on tarpeellista varmistaa testaamalla – silloin tarvitaan toinen referenssiohjelma, jonka kyvyt tunnetaan – ja testauksen kohde, jossa olevat bugit tunnetaan riittävästi.
26. Ohjelmien validoinnille voi olla toimialakohtaisia vaatimuksia, jotka on syytä selvittää huolella.

Ohjelman yleisistä piirteistä

27. Suhtaudu epäillen kaikkiin toimittajan väitteisiin. Ei ole mitään syytä olettaa, että testausohjelmia myytäisiin yhtään rehellisemmin kuin mitään muitakaan ohjelmia.
28. Hinta ei aina ole hyödyn tae. Joskus saman tai paremman tuloksen saa murto-osalla toisen hinnasta.
29. Älä osta koko gorillaa, jos pelkkä banaani riittää. Ohjelmia myydään usein pitkän ominaisuuslistan avulla. Jos tarvitset niistä vain yhtä, muista on vain haittaa – ja kustannuksia. Täsmäohjelmat ovat usein paremmin tehtyjä ja paljon halvempia.
30. Onko ohjelma luonteeltaan softaa validoiva vai sitä kyseenalaistava. Tämä piirre on äärimmäisen tärkeä ja ratkaisee ohjelman avulla tehdyn testauksen laatua.
31. Testaajat ovat testauksen ytimessä. Hyvä työkalu tukee heidän työtään, rikastaa ja energisoi. Huono työkalu koetaan riippakiveksi.
32. Ohjelmilla on tapana sitoa merkityssisältöjä ja olla asioiden symboleja. Sen vuoksi ohjelmat ovat tärkeämpiä muutenkin kuin vain välineenä tai prosessien elementteinä. Esimerkiksi ”markkinajohtajan” tai ”avoimen lähdekoodin” tai ”mallipohjaisen testauksen” käyttäminen edustaa tietynlaisia arvoja ja niihin sitoutumista.
33. Väline on käyttäjille organisaation symboli: se voi edustaa heille sekä visioita ja unelmia että organisaation rappiota. Siksi on parasta tehdä välinevalinta hyvin.

34. Eri yksiköissä ja lokaatioissa sama väline voi edustaa aivan eri asioita!

Sovittaminen toimintaan ja arkeen

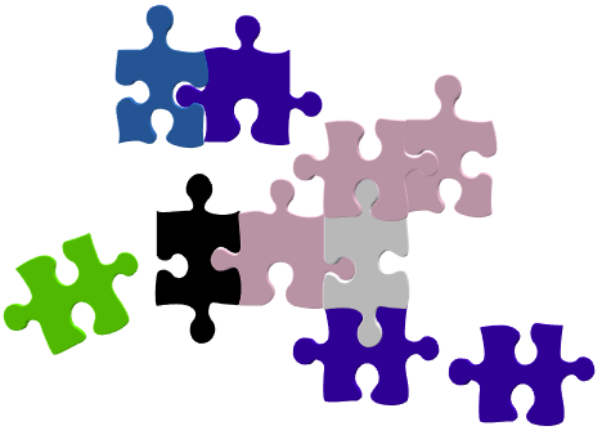
35. Sovita väline mahdollisimman luontevasti työnkulkuihin.
36. Selvitä, vaatiiko väline ihmisten työnkuvien, roolien ja tehtävien muutoksia. Jos vaatii, asia hankaloituu.
37. Mieti, sopiiko ohjelman edellytämä osaamistaso sen tuleville käyttäjille.
38. Ota kulttuurierot huomioon selvittäessäsi sopivuutta.
39. Ei ole synti olla ottamatta käyttöön välinettä, jolle ei olla vielä valmiita, vaan se on viisauttaa!
40. Jos ohjelma lisää kokonaistyömäärää, sen ennuste on erittäin huono.
41. Miten nopea ohjelma on konfiguroida projektiin – henkilölle, joka tietää, mitä tekee? Jos käyttöönotto edellyttää kahden viikon konsultointia, kaikki on liian hidasta.
42. Älä osta uusinta muotia, vaan koeteltu ratkaisu.
43. Suosi vakiintuneita paradigmoja, kuten xUnit tai testauksenhallinnan ”tavalliset” ratkaisut, koska niistä on kokemuksia ja ne ymmärretään.
44. Hyvä ohjelma on käytännöllinen, mutta sille on eduksi perustuminen johonkin teoriaan. Näin sen olemus voidaan ymmärtää paremmin.
45. Mieti, miten ratkaisu saadaan siirrettyä vaikkapa toiselle ohjelmointikielelle ja toiseen kehitysympäristöön. xUnit:n kaltaisilla ratkaisuilla on etua tässäkin – ajattelutavaltaan, toiminnaltaan ja arkkitehtuuriltaan vastaava löytyy jokaiseen ympäristöön.
46. Suosi avoimen lähdekoodin välineitä, koska niitä voit jakaa täysin vapaasti kenelle haluat, myös alihankkijoille ja asiakkaille.

Varmista tulevaisuus

47. Varmista softan ylläpito – sekä toimittajalla että oma. 80 % softan elinkaaresta on ylläpitotyötä, eli jossain pitää olla tiukka ylläpitotiimi. Jos softalle ei ole tehty vuoteen mitään, miksi tehtäisiin jatkossa?
48. Varmista softan muu tulevaisuus mahdollisimman hyvin. Missä kohden elinkaartaan ohjelma on? Mitä toimittaja lupaa? Onko kehitykselle roadmap? Mitä toimittajan historia kertoo – onko siellä haaksirikkoja?
49. Suosi avoimen lähdekoodin ratkaisuja, koska niiden kanssa voit itse hallita ohjelman tulevaisuutta, jos toimittaja pettää tai yhteisö riitaantuu.
50. Varmista ohjelman laadunvarmistus, ettei tule yllätyksiä. Stabiilien tuotantoversioiden pitää olla erossa kehitysversioista, betaversioista ja julkaisukanditaateista. Välineen systemaattinen testausprosessi ja testisetit ovat plussaa.
51. Jokainen päivitysversiokin pitää testata. Testaus pitää tehdä erityisen hyvin, jos ohjelman laadunvarmistus toimittajalla / yhteisössä ei ole kunnossa.
52. Päivitysten käyttöönottoon pitää olla jokin sovittu prosessi.
53. On hyvin harvinaista, että päivityksissä ei mene jotain rikki. *Onko sinun käytössäsi jotain toimintoja tai ominaisuuksia, joilla ei ole väliä?*

Ohjelman keskeisistä toiminnallisuuksista ja laatuominaisuuksista

54. Varmista yhteentoimivuus. Jokaisessa organisaatiossa on kauhea määrä työkaluja, joiden pitää toimia yhteen. Puoltakaan niistä et edes tiedä – eikä kukaan muukaan! Ja monilla tiimeillä on kriittisiä pikkutyökaluja, joita eivät huomaa mainita edes kysyttäessä, mutta joiden sovittaminen uuteen kuvioon voi olla työlästä.
55. Kaikenlainen inventaario on aina hyvä tapa aloittaa kehittäminen.



56. Avoimet rajapinnat ovat yhteensovittamisessa merkittävä etu.
57. Selvitä, voiko ohjelmaa laajentaa omilla rutiineilla.
58. Ei ole hyvä, jos ohjelma edellyttää taas yhden uuden ohjelmointikielen opettelun. Yleisillä kielillä on ”olematon” oppimiskynnys, hyvä laajennettavuus ja varmempi tulevaisuus.

59. Selvitä, miten sen saa liitettyä keskeisiin olemassaoleviin järjestelmiin (testauksenhallinta, jatkuvan integroinnin ohjelmat, yksikkötestausohjelmat jne…).
60. Toinen puoli asiaa on erotettavuus: miten väline toimii läppärissä ilman verkkoyhteyksiä? (Esim. organisaation Wikissä olevat asiat on kiva saada exportattua mobiilikäyttöön tai testausohjelman on joskus kyettävä toimimaan ilman verkkoyhteyttä.)
61. Testausohjelmissa pitää aina jäädä jotain jälkiä siitä, mitä on tehty ja mitä on saatu aikaan. Varmista kaiken logitiedon syntyminen ja hyödynnettävyys.
62. Varmista, että raportit voidaan sovittaa organisaation tarpeisiin ja tapoihin.
63. Varmista, että raportoinnille ja kyselyjen tekemiselle on kaikkia sidosryhmiä palveleva helppo käyttöliittymä – joka ei tuota lisäkustannuksia.
64. Testausmateriaalien version- ja konfiguraationhallinta on yhä tärkeämpää. Varmista, että ohjelma sovittuu työkaluunne.
65. Projektikohtainen räätälöitävyys. Joissakin ohjelmissa on mahdollista vain yksi konfigurointi esimerkiksi työnkuluille per installaatio, mikä haittaa pahasti toimintaa tilanteissa, joissa projektien tyypeissä on vaihtelua.
66. Diagnosoitavuus. Testausvälineen toimimattomuus voi aiheuttaa isoja ongelmia. Varmistakaa, että ohjelmasta näkee helposti, tekeekö se oikeasti jotain.
67. Auditoitavuus. On kyettävä saamaan selville, mitä kaikkea tietoa välineellä on käsitelty.
68. Jos ohjelman tekemä testaus vie paljon aikaa, selvitä, miten ajantarvetta voi säätää (asetukset, strategiat, lopetuskriteerit, testaustietokoneiden määrä jne…).
69. Varmista vaihdettavuus. Jokaisella softalla on elinkaarensa. On kiva, jos ohjelma voidaan paketoida prosesseihin ja työnkulkuihin modulaarisesti, hyvillä rajapinnoilla siten, että se voidaan joskus vaihtaa toiseen helposti.
70. Tuki organisaatiomuutoksille ja ketterälle organisaatiolle. Organisaatiomuutokset ovat arkipäivää. Mieti jo nyt, miten suuri urakka on konfiguroida ohjelma ja sen tietokannat vastaamaan seuraavaa organisaatiokaaviota.
71. Tuki adhoc-käytölle. Voiko ohjelmaa käyttää satunnaiskäytössä portable-moodissa? Voiko palvelinohjelmiston asentaa helposti tavalliselle läppärille?
72. Softan suorituskyky ja skaalautuvuus – kaikenlainen sellainen! Miten se suhtautuu käyttäjien määrä, datan määrän jne. kasvamiseen
73. Miten ohjelma toimii eri lokaatioissa? Miten eri lokaatioissa tuotettu data toimii yhteen?
74. Hyvä tietoturvallisuus on nykyään pakollisia vaatimuksia. Miten turvallisesti palvelintyökalussa saa erotettua projektit, yksiköt, asiakkaat yms. toisistaan? Tukeeko työkalu virtualisointia? Onko tietoliikenne suojattu riittävästi?
75. Käytettävyys on olennainen asia, jos ohjelman halutaan olevan tehokas työn rutiineissa ja virheiden löytämisessä. Hankala ohjelma suuntaa testaajien henkiset voimavarat vääriin asioihin. Tärkeä elementti tässä on ohjelman käsitteistö ja sellaiset mallit, jotka vastaavat testaajien sisäisiä malleja.
76. Kaikki muukin käyttäjäkokemus on oleellista. Subjektiivinen työkalun laatukokemus, arvostuksen kokeminen välineen kautta, tuki identiteetille jne… Nämä ovat kollektiivisia asioita, joten kyse on työpaikan ”yhteisökokemuksesta”.
77. Yhteisölliset toiminnot ovat yhä tärkeämpiä. On hyvä arvioida, olisiko hyvä saada organisaation ääni ja kasvot esille myös välineen kautta – ja jos niin on, miten väline tukee sitä.

Uuden ohjelman myyminen yksiköille ja projekteille

78. Kypsässä organisaatiossa ei tarvita ”myymistä” manipuloivassa mielessä. Kun esimerkiksi Leanin stereotyyppisessä organisaatiossa on nähtävissä kehitysmahdollisuus, ja johto ja työntekijät jakavat saman filosofian, oikean parannuskeinon käyttöönotto on itsestäänselvää.
79. Yleensä laadun parantamisen myyminen organisaatiolle ei onnistu, ellei ole koettua tarvetta eli puutetta. Testaustoiminnan arviointi on hyvä keino (ei siis prosessin, vaan koko toiminnan!). Mittarit ja benchmarkkaus ovat aina hyviä. Kaikki, mikä vähentää työtä eli säästää rahaa on helppo myydä.
80. Jos ohjelma tuottaa vain vähän etua, se kannattaa unohtaa toistaiseksi. Edun pitää olla selkeä, että kaikki sitoutuvat.
81. Jos väline on strateginen, riittävän korkean johdon tuki on kriittinen.

Testausohjelmakin pitää testata

82. Testaa hyvin. Tee kunnollinen hyväksymistestaus työkalulle.
83. Testaa ja analysoi kaikki vaatimusmäärittelyn asiat.
84. Muista testatessa kaikkien lokaatioidenne näkökulma.
85. Pilotoi ohjelmaa jossain projektissa. Mielellään sellaisessa, jossa ohjelman puutteet eivät lisää riskiä projektin epäonnistumiselle. Hanki pilotissa raakaa faktaa ja mittaustietoa perustelujen tueksi.
86. Muista: pilotointi ei ole valinnan validointi, vaan oppimisprosessi, jossa käsitykset muuttuvat, kun vain pitää silmät auki.

Ohjelman sovittaminen omaan ympäristöön

87. Tee suunnitelma siirtymiselle vanhasta ratkaisusta.
88. Älä mieti vain toimintaa, vaan esimerkiksi vuosikymmenten kuluessa syntynyttä testitapausten valtavaa massaa, jota ei konvertoida hetkessä eikä vuodessa.
89. Ota suunnitelmassa huomioon koko yhteistyöverkostosi.
90. **Huomaa, että siirtymisprosessi on aina vähintään kaksi kertaa vaikeampi, hitaampi ja kalliimpi kuin worst case -arviosi.**
91. Älä tee testausohjelmasta konsernin omistamaa ja hallinnoimaa, koska se ei koskaan toimi. Anna projekteille mahdollisuudet hallita työkalujaan ja niiden ympäristöjä.
92. Jos tietohallinto valitsee ohjelmistotuotannon tai laatuorganisaation välineitä, jossain on pahasti vikaa.
93. Jos organisaatio ei tue projekteja, ne improvisoivat, ja tuloksena on adhoc-tiedonhallintaa, joka on läppäreillä ilman varmuuskopioita.
94. Ylläpitoprosessissa on olennaista saada aikaan nopea käyttäjienhallinta, projektien itsepalveluna.
95. Erilaiset tietoverkon ongelmat ovat edelleen keskeinen hankaluuden lähde. Se puoltaa ohjelmistoinstallaatioiden paikallisuutta – mutta tiedot pitää sitten synkronisoida.
96. Suunnittele lisenssien hallinta, jos se edellyttää seurantajärjestelmää. On kovin ikävää, jos lisenssit loppuvat kesken, kun pitäisi tehdä töitä tai käynnistää projekti!
97. Mieti, miten välineen käyttöä voisi seurata etänä. Esimerkiksi projektien auditointi on helpompaa, jos näkymät ovat aina olemassa. Muista tässä sisäinen luottamus (”miksi joku seuraa meitä…”) ja tietenkin tietoturvallisuus.

Käyttöönoton tuki

98. Luo ohjelmalle sisäiset konsultit antamaan tukea ja siirtämään siihen liittyvää osaamista.
99. Ohjelmilla pitää olla nimetty yhteyshenkilö / pääkäyttäjä. Pk-yrityksissä saa edelleenkin joskus etsiä, kuka vastaa jostain sovelluksesta.
100. Tukipyynnöille ja virheilmoituksille pitää luoda järjestelmä.
101. Palvelutasoille pitää luoda kuvaukset, vaatimukset ja ne toteuttava järjestelmä. Jos esimerkiksi uuden käyttäjän luominen kestää muutaman viikon, voi olla, että tarvetta ei enää silloin ole. Aika *ei hoida* asioita!
102. Luo rutiinit, pohjat yms., joilla ohjelma saadaan mahdollisimman helposti käyttöön uusissa projekeissa.
103. Kaikki valmisteleva räätälöinti vähentää kynnystä, muutosvastarintaa ja helpottaa ihmisten elämää. Ja säästää aikaa ja rahaa.
104. Teetä skriptejä, joilla projektien konffausta voi auttaa.
105. Laita kaikki apuvälineet ja tiedot jakoon Wikissä.
106. Luo sisäinen yhteisö käyttäjistä ja konsulteista. Tee yhteisölle yhteistyövälineet, erityisesti keskustelufoorumi. Tee yhteisöstä aktiivinen!
107. Tee Wikiin tai vastaavaan kokemustietopankki.
108. Luo tavat jakaa välinetietämystä yksiköiden ja projektien välillä. Tässä on kyse ihmisten aktivoinnista.
109. Tuo välineen tarvitsemaa tiedonhallintaa. Tee rutiinit, apuohjelmat, skriptit, hakemistorakennemallit, tietokannat yms.
110. Suunnittele välineen konfigraationhallinta. Tee välineet, joilla välineen tila ja konfiguraatio voidaan tallentaa ja palauttaa tarvittaessa käden käänteessä tietty versio.
111. Huolehdi kaikkien välineen tarvitsemien komponenttien konfiguraationhallinnasta – esimerkiksi Pythonin tai Perlin versio voi olla pakko jäädyttää, mikä vaikuttaa muihinkin asioihin, mitä niillä tehdään.

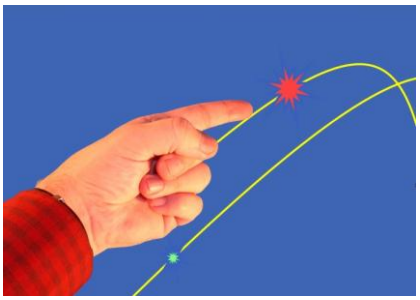
112. Ajokonfiguraation ongelmat ovat usein kaikkein tuskallisimpia tunnistaa. Tee konfiguraation mahdollisia ongelmia tunnistavia apuohjelmia (esim: kun kerran tiedät, että ohjelma vaatii tietyn Python-version, tee rutiini, joka kertoo, että käytössä on se oikea.)
113. Kaikki versioiden jäädyttäminen edellyttää sopimuksia, kompromisseja ja tuottaa myös joskus ongelmia.
114. Taas yksi avoimen lähdekoodin etu: voit tarvittaessa kääntää ja sovittaa välineen vaikkapa mainittujen ohjelmointikielten uusille versioille ja tehdä sen myös silloin, jos toimittajan / yhteisön porttaustyö kestää kohtuuttoman kauan – tai sitä ei enää aiota tehdä.
115. Katselmoi kaikki tiedonhallintaan liittyvät asiat testausmateriaalien ylläpidon ja tulevaisuuden näkökulmasta
116. Mieti, miten ohjelma saadaan hajautetuissa projekteissa kaikkien käyttöön. Globaalisti hajautettu toiminta, jossa on monta osapuolta, voi olla haastava. Mutta senkin pitää onnistua.

Auta käyttäjiä käyttöönotossa

117. Kouluta ja auta. Ohjelmia ei saa vain työntää tarjolle, vaan pitää aktiivisesti antaa koulutusta ja auttaa sen käyttöönotossa ja käytössä.
118. Vaadi ja valvo. Projekteja pitää vaatia käyttämään ohjelmaa, jos sen käytöstä on sovittu.
119. Juuri tällaisten asioiden seuraamiseksi laatupäälliköt tsekkaavat projektisuunnitelmat ja laadunvarmistussuunnitelmat.
120. Käytön seuranta auditoimalla on ihan mukava tapa, kun ne ovat kehittämisauditointeja, joilla autetaan projekteja, eikä poliisitoimintaa.
121. Uuden välineen käyttöön on heti ulotettava organisaation normaali katselmointikäytäntö.
122. Vaikka sitä ei olisikaan edellytetty organisatorisesti, on aluksi syytä katselmoida esimerkiksi välineen testausmateriaalien hallinta ja skriptien ja muiden testien formaalien kuvausten laatu. Ihan vain ihmisten auttamiseksi.
123. Älä ota välinettä ja sen käyttöä liian vakavasti. Joskus väline on vain väline.

Mitä tämän jälkeen?

124. Kaikella on elinkaarensa. Varaudu alusta alkaen tämänkin ohjelman poistumiseen.



Extra-pointit!

125. Mieti, mitä geneerisesti oleellista on jäänyt mainitsematta! Mieti, mitä sinun kontekstissasi oleellista on jäänyt mainitsematta!