SYSTEEMITYÖ

Systeemityöyhdistys SYTYKE ry:n jäsenlehti N:o 4/2006

Testauksen arki Suomessa

SYTYKE ry on vuodesta 1979 toiminut valtakunnallinen systeemityöntekijöiden ammatillinen yhdistys, joka kehittää alan ammattilaisten välistä yhteistyötä ja tutkimustoimintaa.

Teemayhdistyksen jäseneksi voivat liittyä kaikki systeemityöstä kiinnostuneet yksityiset henkilöt, yhdistykset ja yritykset. SYTYKE ry:n toiminta-alueena on koko Suomi. SYTYKE on Tietotekniikan liitto Ry:n jäsenyhdistys.

Lisätietoja SYTYKE ry:stä: www.sytyke.org

TOIMISTO

Puhelinvastaus- ja sihteeripalvelu VT Oy/ Susanna Koskinen
Systeemityöyhdistys Sytyke ry
Henrikintie 7 A
00370 Helsinki
p. 09 56075363
f. 09 56075365
sytyke@hennax.fi

Johtokunta 2006

Puheenjohtaja
Helena Venäläinen
p. +358 10252 3690
+358 50 568 6690
helena.venalainen@op.fi

Paula Miinalainen
p. +358 50 500 2363
paula.miinalainen@arborvitae.fi

Markku Niemi
p. +358 50 512 4687
make.niemi@kolumbus.fi

Ilkka Pirttimaa
p. +358 50 389 0022
ilkka.pirttimaa@stockmann.fi

Tarja Raussi
p. +358 50 548 1823
tarja.raussi@tieturi.fi

Jori Räty
p. +358 50 551 5152
jori.raty@sysopendigia.com

Kari Uusi-Äijö
p. +358 40 835 6541
kari.uusi-aijo@pohjola.fi

Varajäsenet

Lauri Laitinen
lauri.laitinen@nokia.com

Lea Virtanen
lea.virtanen@jobit.fi

Liittokokous-edustajat

Silja Räisänen
silja.raisanen@op.fi

Jori Räty
jori.raty@sysopendigia.com

Helena Venäläinen
helena.venalainen@op.fi

Kati Viik
kati.viik@helsinkilainen.com

SYTYKE ry:n johtokunnan sähköpostilista:

hallitus@sytyke.ttlry.fi

Sytyttääkö? - Liity jäseneksi

Systeemityöyhdistyksen jäseneksi liitytään Tietotekniikan liiton kautta (http://www.ttlry.fi/, 020 741 9898, jasenasiat@ttlry.fi) valitsemalla jäsenyhdistykseksi Systeemityöyhdistys ry. Nykyinen Tietotekniikan liiton jäsen voi liittyä joko vaihtamalla jäsenyhdistystä tai liittymällä lisäjäseneksi.

Tietotekniikan liiton henkilöjäsenmaksu vuonna 2006 on alkaen 52€, erityisryhmien hinnoittelusta lisätietoja Tietotekniikan liitosta. Lisäjäsenyys maksaa 11€/yhdistys.

Osaamisyhteisöt

Systeemityöyhdistyksessä toimitaan niin yhdistystasolla kuin aihepiireittäin erikoistuneissa osaamisyhteisöissä. Monipuolisessa tarjonnassamme löytyy jokaiselle jotakin. Vaihtoehtona on myös perustaa omalle kiinnostukselleen uusi osaamisyhteisö - SYTYKE-hallitus toivottaa toimintaehdotukset tervetulleeksi. Osaamisyhteisön toimintaan pääset mukaan laittamalla postia vetäjälle.

RELA keskittyy relaatiotietokantoihin vetäjänään Lauri Pietarinen.
lauri.pietarinen
@relational-consulting.com

ProjektiOSY - PrOSY pyrkii yhdistämään Systeemityön projektitoiminnasta ja sen kehittämisestä kiinnostuneet, vetäjänä Markku Niemi.
make.niemi@kolumbus.fi

TestausOSY - FAST on testauksen keskustelu- ja yhteistyöverkosto, vetäjänä Maaret Pyhäjärvi.
maaret.pyhajarvi@iki.fi

DAMA Finland keskittyy tiedon, informaation ja tietämyksen hallintaan. Suomen osaston johtoryhmän (boardin) vetäjänä Pekka Valta, yhteyshenkilönä Minna Oksanen.
minna.oksanen@gmail.com

ViestintäOSY järjestää yhteistoimintaa viestintäsovellusten alueella, vetäjänä Tapani Ranta.
tapani.ranta@generum.fi

OlioOSY kehittää, tukee ja voimistaa oliomallilähtöistä sovelluskehittämistä, vetäjänä Jukka Tamminen.
jukka.tamminen@pp.inet.fi

JavaSIG - on Javan käyttäjien ja harrastajien intressiryhmä, vetäjänä Simo Vuorinen.
simo.vuorinen
@tietoenator.com

Julkaisija
Systeemityöyhdistys Sytyke ry
Puhelinvastaus- ja sihteeripalvelu VT Oy
Susanna Koskinen
Henrikintie 7 A, 00370 Helsinki
p. 09 5607 5363
f. 09 5607 5365
sytyke@hennax.fi

Päätoimittaja
Minna Oksanen
minna.oksanen@gmail.com
puh. 040 577 6640

Toimitussihteeri
Susanna Koskinen
sytyke@hennax.fi

Taitto
Speaking Bark Tmi
Katja Tamminen
sb@cabaroo.com

Toimituskunta 4/2006
Mitro Kivinen, FAST

Lisätietoja lehdestä
www.sytyke.org/lehti

Tilaukset
Systeemityölehti sisältyy yhdistyksen Tietotekniikan liiton suositusten mukaiseen yhdistyksen jäsenmaksuun.
Vuositilaus 20 €
Irtonumerot 5 €
Hyvissä ajoin ennen painatusta tehty vähintään 50 kpl lisätilaus 2 €/kpl.
Tilaukset yhdistyksen toimistosta.

Kansikuva
Matti Vuori, Plenware Group Oy

Seuraava numero
1/2007 Toimialakatsaus
"Mikä mahtaa olla IN?"
Toimituskunta: Tarja Raussi
Ilmestyy: to 22.2.2007

Painopaikka
T-Print
Ahokaari 1-3
05460 Hyvinkää
Puh. (019) 475 8500

Painos: 2500 kpl
ISSN 1237-0525
13 vuosikerta, nro. 4

Ilmoitushinnat
Takakansi A4 1200 €
Sisäkannet A4 1000 €
Sisäsivut 1/1 800 €
Sisäsivut 1/2 600 €
Sisäsivut 1/4 400 €

Arvonlisävero 0%
Vakiopaikan vähintään vuodeksi varanneille 20% alennus.

Teksti: Qentinelin Johtava Konsultti Mitro Kivinen, testausalan aktiivi, toimii TestausOSYn isännistössä ja Suomen ISTQB-johtoryhmässä.

Pääkirjoitus: Testaus kehittyy, missä nopeammin, missä hitaammin

Onko huono laatu ostajan oma vika? Asiakkaat haluavat halvalla sovelluksia omiin tarpeisiinsa mukautettuna ja sovelluksen ostohinta kilpailutetaan alas. Tällainen toiminta ei jätä myyjän takana olevalle organisaatiolle liikkumavaraa lisätä projektin kustannuksia ja testauskustannuksia on helppo leikata.

Jokaisessa organisaatiossa mietitään kuumeisesti, kuinka juuri me voimme testata enemmän vähemmällä. Tai ainakin kohdistaa testaus siihen, millä todella on merkitystä. Perinteiset organisaatiot toimivat kuten ennenkin: pitkissä ja monipolvisissa projekteissa testaajat tekevät parhaansa taistellessaan muuttuvia vaatimuksia vastaan yrittäen kattaa kaikki toiminnallisuudet. Markkinaorientoituneet organisaatiot vähintään harkitsevat, jolleivät jo kokeile ketterää kehitystä.

Avoimen sovelluskehityksen mukana on ihan salaa tullut olennaista paranemista testauksen alkupäähän. Laatupäälliköt ovat vuosia miettineet, kuinka saada kehittäjät innostumaan yksikkötestauksesta ja nyt kehittäjät ovat ihan itse rakentaneet itselleen automaattiset yksikkötestit xUnit–perheen työkaluilla ja ovat niistä innoissaan!

Tässä numerossa Systeemityö esittelee kokemuksia ja näkemyksiä testauksen hajanaisen kentän arjesta. Koko kenttää on hyvin vaikea kattaa, mutta toivon, että lukijat huomaavat testauksen arjen olevan kiehtovaa ja monipuolista. Sillä testaus ei elä tyhjiössä, vaan kaikki organisaation toiminnot vaikuttavat testaukseen ja sen onnistumiseen. Antoisia lukuhetkiä!

Sisällys

Teksti: Erkki Pöyhönen

Kehityksen testaus ja testaustiimi

ONGELMANA OHJELMISTOKEHITYKSEN TESTAUSOSAAMINEN

Erkki Pöyhönen, QA Business Development Manager TietoEnator, QACC Quality Assurance Competence Center

Nykyisin TietoEnatorilla vaikuttava Erkki Pöyhönen on aikaisemmin työskennellyt Nokiassa monissa eri testaus- ja kehittämisrooleissa yhteensä 13 vuoden ajan. Hänen tontillaan on ollut erityisesti testauksen menetelmien ja testauskoulutuksen kehittäminen eri yksiköiden tarpeisiin. Vuonna 2001 hän perusti suomalaisen testaajakerhon testaajien osaamisen ja ammatti-identiteetin kehittämiseksi ja vuonna 2003 Suomen testausraadin kansainvälisen testaajasertifikaatin valmisteluun ISTQB:n kautta (http://www.istqb.org). Erkki on toiminut myös mm. EuroSTAR2004-konferenssin puheenjohtajana ja on Tietotekniikan Liiton hallituksen jäsen. Nykyisin Erkin roolissa on testauksen palvelujen ja testausosaamisen kehittäminen QACC:ssä, TietoEnatorin testauksenasiantuntijaorganisaatiossa.

Yksi asia alkaa olla nykyään kaikkien testauspäälliköiden viesti, kun keskustellaan järjestelmän laadusta ja testauksen roolista: Ohjelmiston laatu tehdään oikeissa vaatimuksissa ja huolellisella suunnittelulla ja ohjelmoinnilla; testaus ei tee laatua. Testaus lähinnä tuottaa laatutietoa järjestelmistä ja osoittaa kohteita laadun parantamiselle (lue: ohjelmistovikoja). Kuitenkin on yleistä, että kehitystiimi delegoi ”laatuasiat” testaustiimille, etenkin jos asiakkaan kokema laatu ei ole kehitysorganisaatiolle konkreettisesti ymmärretty. Jos testaukseen tuleva ohjelmisto on kehnoa, ei hyväkään testaus tule menemään kovin sujuvasti ja myös lopputuotteen laatu tulee jäämään huonoksi.

Sana ”kehitys” tässä tarkoittaa sekä kehitysorganisaatiota (suunnittelu ja ohjelmointi) että myös niiden tekemää työtä, johon yleensä kuuluu suunnittelun ja ohjelmoinnin lisäksi jonkinlaista yksikkötestausta ja integrointia.

Sana ”testaus” tarkoittaa usein kahta eri asiaa:
a) Tekemistä eli Laadunvarmistuksen tehtäviä, jotka koskevat eri muodoissa melkein kaikkia kehitysprojektissa; ja
b) Ihmisiä eli Erillinen testausorganisaatio tai testaustiimi, ”testausfunktio”.

Tässä artikkelissa puhutaan erillisen testaustiimin (testauksen merkitys b) mahdollisuuksista ja tarpeista puuttua kehityksen tekemään testaukseen (merkityksessä a).

Varsinainen ongelma on, että kehityksen tuottaman järjestelmän laatu on huono. Ja usein vielä sillä tavalla, että projektissa harva sitä tiedostaa ennen kuin erillinen testaus alkaa. Syynä voivat olla esimerkiksi äsken viitattu, vanhentunut tapa delegoida kaikki laadunvarmistus testausammattilaisillekehittäjien puutteellinen testausosaaminen huono näkyvyys kehitteillä olevan tuotteen laatuun projektin sisällä vallitseva hajaannus projektin tavoitteista, tuotteen vaatimuksista ja prioriteeteista.

MITEN ASIOIDEN “TULISI YLEENSÄ MENNÄ”?

Katsotaanpa ”miten asioiden yleensä tulisi mennä”. Lainausmerkit ovat ihan tarpeen tässä, koska erilaisten käytäntöjen ja prosessien hyödyllisyys riippuu valtavasti projektin ja organisaation tilanteesta. Vaikka alla lueteltu prosessi on mahdollisimman abstrakti, niin helposti prosessin tehtävät tulkitaan jonkin kontekstin mukaan. Esimerkiksi jossain tilanteessa alla luetellut tehtävät tehdään ajallisesti peräkkäin, toisessa tilanteessa rinnakkain. Samoin toisaalla on tarpeen eriyttää rooleja enemmän ja toisaalla vähemmän. Kyse on siis yksinkertaistetusta näkemyksestä ja toistankin tässä itsestäänselvyyksiä, että on myöhemmin mihin viitata.

1. Asiakasta lähellä olevat tahot tuottavat järjestelmän rajauksen asiakasvaatimuksissa
2. Kehitys tuottaa tekniset vaatimukset ja arkkitehtuurin kokonaiskuvan
3. Varsinainen suunnittelu ja toteutus tapahtuvat kehitystiimeissä
4. Valmiit tuotokset julkaistaan integroitavaksi ja testattavaksi
5. Testaustiimi testaa kokonaisuuden

Usealla organisaatiolla on hyviä kokemuksia vaiheiden rinnakkaisuudesta ja roolien sekoittamisesta ja tämä on tärkeä näkökulma ketterissä menetelmissä ja muissa iteroivissa elinkaarimalleissa.

HALLITTAVUUSONGELMA

Jos halutaan tuottaa riittävän hyvää laatua, tarvitaan siis monta asiaa:

Kommunikoitavissa oleva näkemys tavoitteesta: olen tehnyt koko joukon prosessiarviointeja ja vaatimukset ovat varmaan yleisin kohtaamani ongelma. Joko niihin ei panosteta riittävästi, luotetaan sokeasti jonkun sidosryhmän tuottamaan dokumenttiin tai sekoitetaan eritasoiset asiat. Lopputuloksena on usein se, ettei projektissa synny yhteisymmärrystä projektin tavoitteista ja niiden

prioriteeteista. Laadulliset tavoitteet jäävät erityisen kehnolle tolalle, koska ei ole luotu yhteistä kieltä niistä puhumiseen ja asiakkaalle riittävä laatutaso on epäselvä.

Osaaminen tavoitteen saavuttamisen keinoista: voidaan asettaa kunnianhimoisia tavoitteita, mutta niiden saavuttaminen edellyttää sekä riittäviä resursseja että myös riittävää osaamista. Paljon hyviä menetelmiä ja työkaluja jää käyttämättä, kun kehityksen energia menee esimerkiksi sovellusalueen ymmärtämiseen. Tässä ei pidä unohtaa myös ei-teknisiä taitoja (kommunikointi, projektinhallinta, ongelmanratkaisu, …), joiden merkitys korostuu kun projekteja hajautetaan ja liiketoimintaympäristöt monimutkaistuvat.

Kyky havaita todellisuuden ja tavoitteen välinen ero: tämä on testauksen perinteinen tehtävä. Mutta tämä tarkoittaa myös organisaation kykyä puhua totta ja tehdä päätöksiä tosiasioiden pohjalta. Kovasti arvostamani Jim Highsmith puhuu usein projekti- ja laatutiedosta ja siitä, että tietoa ongelmista voidaan käyttää joko tekosyinä näkyvissä olevalle epäonnistumiselle tai sitten positiivisemmassa tapauksessa varhaisina varoituksina. Vanha projektinhallintasanonta kuuluu näin ”huonot uutiset menevät huonommiksi ikääntyessään”. Jos järjestelmän laadun systemaattisia ongelmia ei havaita kehityksen aikana, niiden korjaaminen on paljon hankalampaa. Samoin merkitsee valtavasti projektissa vallitseva kulttuuri huonoihin uutisiin: johdon käytös huonoja uutisia kuullessaan määrää sen, saadaanko tulevaisuudessa ongelmat tietoon ennen kuin niistä tulee kriisejä vai aletaanko tietoa pantata ja virheitä peitellä.

Sekä suunnittelijoiden ammattiylpeyttä että päälliköiden hyvää hallintoa, joka mahdollistaa järkevän toiminnan: on aivan perusteltu oletus, että projektissa kaikki haluavat tehdä hyvää työtä. Muistan itse miten suunnittelijana minua harmitti, kun jouduin luovuttamaan keskeneräisen sovellukseni paketointiin ja testattavaksi. Asiakas oli tyytyväinen, mutta itse en saanut tehdä niin hyvää työtä kuin olisin halunnut. Keskeinen projektinhallinnan haaste on moniulotteinen optimointi: sijoittaa resurssit ja mitoittaa kustannukset oikein, että tuetaan liiketoiminnan prioriteetteja. Testauksenkin tulisi olla koko ajan selvillä projektin tuote- ja liiketoimintariskeistä, että kaikki testaus kohdistuisi sinne missä sillä on eniten merkitystä.

Suuria kokonaisuuksia hallitaan monella tapaa. Yksinkertaisin tapa on tasapäistämispyrkimys, jossa koetetaan saattaa eri puolilla tehtävä kehitys ja testaus yhdenvertaiseksi. Hyödylliseksi on koettu vaikkapa laadunvarmistuksen minimitason asettaminen, eli sovitaan mitä katselmuksia ja testejä kaikki komponentit kokevat niitä muutettaessa. Sitten luonnollisesti riskien mukaan nostetaan kriittisimmän osajoukon testien vaatimustasoa, kattavuutta, jne.

Koodimassojen hallinta tapahtuu pääosin ”hajoita-ja-hallitse”, eli uustuotannossa iso järjestelmä on jo varhain projektissa jaettu osajärjestelmiin ja ne vielä komponentteihin, sovelluksiin, tms. Usein sopiva modularisoinnin taso löytyy sovellusarkkitehtuurista. Tämä on aivan hyvä tapa, mutta usein tämä jako tehdään etenkin vesiputous-mallisessa prosessissa liian varhain. Kokonaisuus on projektin alkuvaiheessa vielä liian epävarmalla pohjalla, jotta kaikkia osakokonaisuuksia voitaisiin alkaa kehittää itsenäisesti. Toisaalta harvassa projektissa on niin kehittyneet kommunikointimenetelmät, että voitaisiin moneen kertaan muuttaa kokonaisuuden osiinjakoa ilman merkittäviä kustannuksia.

NÄKYVYYSONGELMA TESTAUSTOIMESSA

Kun jouduin vähän aikaa sitten etsimään kokonaiskuvaa erillisen testauksen työmäärään ja aikatauluun vaikuttavista tekijöistä, esiin nousi kaksi asiaa: testauksen tavoitteet ja sisään tulevan ohjelmiston laatu. Testauksen tavoitteet tarkoittaa lyhyesti ainakin kahta asiaa: miten hyvää riskienhallintaa (~ miten tiheää seulaa) testauk-

selta edellytetään ja mitä muita tehtäviä varsinaisen testien suunnittelun ja suorituksen lisäksi testaustiimille asetetaan.

Tämä testattavaksi tulevan ohjelmiston laatu onkin hankalampi käsite. Testauksen sujuvuus riippuu eniten sisään tulevan ohjelmiston kypsyydestä ja eheydestä. Kypsyys tarkoittaa, että ohjelmisto on luotettava ja sisältää sovelluksen tavoitteisiin kuuluvat piirteet. Eheys tarkoittaa, että ohjelmiston toteutus pitää yhtä sisällön ja laadun lupauksen / dokumentaation kanssa.

Tietenkään ei ole realistista odottaa, että kaikki ohjelmisto on korkeaa laatua ja hyvin luotettavaa heti alkuunsa, mutta on testauksen aikeille tuhoisaa, jos järjestelmän komponentit valmistuvat merkittävästi odotettua hitaammin tai oleellisesti huonommassa kunnossa. Esimerkiksi testauskierrosten määrä riippuu paljon siitä, paljonko vikoja järjestelmässä on ja miten paljon jotkut viat peittävät muita vikoja näkymästä. Samoin integroinnin sujuvuus riippuu paljon komponenttien kypsyydestä.

Testaukselle ei ole mikään yllätys että järjestelmissä on vikoja. Mutta testauksella tulee olla näkyvyys järjestelmän laatuun jo kehityksen aikana. Muutoin testauksen panostus suunnittelussa, ympäristöissä ja resursoinnissa voi perustua vääriin oletuksiin. Tällöin hukataan projektin loppuvaiheen kallista aikaa sählääämiseen, pikaiseen uudelleensuunnitteluun ja esimerkiksi tehdään osa testauksesta epätarkoituksenmukaisessa ympäristössä. Testien suorittaminen on lähes joka projektissa sen kriittisellä polulla, eli testien myöhästyminen myöhästyttää koko projektia.

TESTAUS KEHITYKSEN KONSULTTINA

Testaukseen kertyy projektien kuluessa paljon tietoa, jota olisi sääli olla käyttämättä organisaation toiminnan kehittämiseksi ja tulevien projektien turvaamiseksi. Vain osa siitä opista ja tiedosta mitä testaajille kertyy, päätyy testaustiimin tai testaustoimen eduksi. Vaatii paljon energiaa, että tieto saadaan kulkemaan ja organisaatio oikeasti oppisi. Tästä löytyy tietoa aiemmin niin suositusta tietämyksenhallinnan (Knowledge Management) ajattelijoilta.

Keskeisin oppi tulee testattavien järjestelmien vahvuuksista ja heikkouksista sekä näiden kautta vastaavasti myös kehitysorganisaation toiminnasta. Paitsi konkreettista vika- ja muuta kirjattua tietoa, testaajat oppivat paljon MuTu-tietoa (”musta tuntuu”, ns. hiljainen tacit-tieto), joka harvoin kirjataan minnekään ja harvoin välittyy sinne missä siitä olisi hyötyä.

Toisaalta testaajat oppivat paljon myös asiakkaiden tarpeista ja liiketoiminnan luonteesta, koska kehityksen alun abstraktit tavoitteet ja toimitusajankohdan realismi kohtaavat juuri testauksessa kaikkein konkreettisimmin. Olisi iso etu, jos tämä kokemus saataisiin tehokkaasti jaettua kehittäjillekin. Hitain tapa opettaa kehitystä on kertoa asia vikaraporttien selityksissä; jokin aloitteellisempi keino olisi paljon halvempi ja tehokkaampi.

Luonnollisesti testauksen tärkein ero muun organisaation osaamiseen on tavallisesti erilaisten laadunvarmistuskäytäntöjen asiantuntijuus. Olen tavannut paljon testaajilla, jotka ovat opiskelleet itsekseen useita kikkoja ja keinoja oman roolinsa ulkopuolelta ollakseen avuksi kehittäjille ja vaatimusten kirjoittajille. Tämä on ilmeisen hyvä idea, koska on todennäköisempää saada viestinsä ”tälle tarvitsisi vielä tehdä jotain” läpi, jos on myös antaa ideoita mitä se jotain voisi esimerkiksi olla. Luonnollisesti parasta olisi, että kehittäjillä olisi jo perusopintojensa kautta kaikki tarvitsemansa laadunvarmistustaidot. Testauksen kuitenkin kannattaa panostaa tämänkin puutteen paikkaamiseen, niin kauan kuin näitä tietoja kehitykseltä puuttuu ja kiinnostusta kuitenkin löytyy.

NÄKYVYYS JA VAIKUTUSMAHDOLLISUUS KEHITYKSEN LAATUUN

Testaus on kehityksen lähin (yleensä sisäinen) asiakas, etenkin järjestelmätestauksessa ja järjestelmäintegraatiossa. Koska testauksen tärkeä velvollisuus on tunnistaa järjestelmän puutteita, kehitys asennoituu (toisinaan) testaukseen kuin asiakkaan edustajaan. Luonnollisesti tämä ei ole testauksen oma asema, vaan ainoastaan lainattua valtaa. Jos testaus ei ole palveluorganisaatio ja ole sidosryhmilleen aidosti hyödyksi, ei sitä myöskään arvosteta sisäisen asiakas-roolinsa takia.

Tässä asiakastoimessa testaus tarvitsisi eniten näkyvyyttä tekeillä olevien komponenttien laatuun, kuten jo mainittiin. Tämä auttaa valmistelemaan testausta ja keskittämään testauksen resurssit oikein.

Toinen tarve, ei niin ilmeinen, on ymmärtää kehityksen käytäntöjä. Tämä ei ole kovin suuri ongelma kehityksen ja testauksen ollessa saman organisaation eri puolia, mutta voi mennä todella hankalaksi, jos kehitys tai testaus on ulkoistettu. Ulkoistuksessa ei tietoa kehityksen menetelmistä luontevasti liiku ja asia pitää selvittää erikseen. Tällainen näkyvyys on tarpeen, jotta testaus voi oikein palvella kokonaisuutta, täydentää kehityksen puutteita ja tarvittaessa esimerkiksi auttaa puuttuvan käytännön liikkeelle saamiseen. Ilman aloitteellisuutta ja myöhässä tästä tiedosta on paljon vähemmän hyötyä.

Teksti: Tomi Kaleva

Hyväksymistestauksen anatomia

Tomi Kaleva toimii laadunvarmistusosaston ostopäällikkönä Tietokarhu OY:ssä. Hän on testauksen osaamisyhteisön ohjausryhmän jäsen.

Ohjelmistotuotanto sisältää monta vaihetta ja niistä viimeisin sekä kriittisin on asiakkaan hyväksyntä. Ilman sitä toimitus on epäonnistunut ja taloudellinen hyöty jää saamatta. Tästä syystä hyväksyntää pidetään hyvin tärkeänä vaiheena molemmille osapuolille. Muutamia vuosia sitten harvemmin asiakas osasi asettaa kriteerejä onnistuneelle toimivalle ohjelmalle, vaan luotti, että toimittaja on tehnyt toiveiden mukaisen järjestelmän. Useammissa tapauksissa asiakkaan tyytymättömyys ohjelmaan selvisi vasta tuotannon aikana. Tällöin ohjelmia oli vaikea enää muuttaa ja sopimusriitoja jouduttiin ratkaisemaan lakimiesten avulla.

Hyväksynnän helpottamiseksi lisättiin ohjelmistotuotantoon viimeiseksi vaiheeksi hyväksymistestaus, jossa asiakas yhteistyössä toimittajan kanssa tarkistaa, että toimitettu ohjelma on vaatimuksien, toiveiden ja sopimuksien mukainen. ”Hyväksymistestattu” leiman jälkeen asiakas on juridisesti todennut ohjelman valmiiksi ja velvollinen maksamaan siitä. Vaihe kuulostaa yksinkertaiselta, mutta on todellisuudessa hyvin haastava molempien näkökulmasta. Toimittajan tulee osoittaa, että ohjelma sisältää kaikki ne ominaisuudet, jotka asiakas on tilannut ja toivonut. Lähes poikkeuksetta ohjelmistotuotannon aikana tulee lukuisia muutoksia määrittelyihin ja niillä voi olla suuriakin vaikutuksia lopputulokseen. Pitkässä projektissa myös ulkoiset tekijät voivat muuttua ja ne tulee huomioida kehitystyön aikana. Harvemmin toimittaja voi todeta, että ohjelma on täydellisesti asiakkaan toiveiden mukainen, vaan se sisältää aina kompromisseja, joista on toivottavasti sovittu kirjallisesti asiakkaan kanssa. Hyväksymistestaus onkin siis toimittajan näkökulmasta ohjelmiston esittelytilaisuus ja mahdollisuus viimeisten virheiden korjaamiseen.

Tilaajan rooli hyväksymistestauksessa on vielä vaativampi. Sen tulee lyhyessä ajassa varmistua, että ohjelma on tilauksen mukainen. Ohjelmaa tulee tarkastella useasta näkökulmasta oikeiden asiantuntijoiden kanssa. Tilaajalla on myös suuri vastuu, koska hyväksymistestauksessa lopullisesti varmistetaan, ettei tuotantoon siirry virheellisiä ohjelmia. Haasteena on, että tilaajalla on harvemmin kokemusta ohjelmistotuotannosta ja erityisesti testauksesta. Hyväksymistestausta tulisi ajoissa valmistella ja suunnitella, jotta kokonaisuus saadaan kattavasti tarkistettua. Testauskäytännöt ja – prosessit ovat arkipäivää ohjelmistotaloille, joissa testausta tehdään useammassa projektissa päivittäin. Tilaajalla uusien ohjelmien käyttöönottojen välillä voi kulua aikaa useampia kuukausia jopa vuosia. Tässä ajassa testausrutiinit unohtuvat ja asiantuntijat vaihtuvat. Ilman asianmukaista dokumentaatiota hyväksymistestausta onkin mahdotonta uudelleen toteuttaa. Tilaajalla on kolme tapaa ratkaista ongelmakenttä: perustamalla oma hyväksymistestaustiimi, käyttämällä ulkopuolisia asiantuntijoita tai tekemällä tiivistä yhteistyö toimittajan kanssa.

Hyväksymistestaus sisältää myös eettisiä ja juridisia ongelmia. Jos toimittajan kanssa tehdään tiiviimpää yhteistyö ja heidän asiantuntijoita hyödynnetään hyväksymistestauksessa, niin saadaanko testaukseen riittävän ulkopuolinen näkökulma. Voidaanko toimittajan avulla tehtyä hyväksymistestausta pitää juridisesti pätevänä? Toisaalta onko eettisestä näkökulmasta oikein hyödyntää hyväksymistestauksessa järjestelmätestauksen testitapauksia ja – materiaalia. Tämä olisi helppoa, sillä molempien testauksien tavoite on samanlainen. Niissä varmistetaan, ettei tuotantoon siirry virheitä. Hyväksymistestauksessa tämä tulisi tehdä ulkopuolisesta näkökulmasta. Valitettavasti toimittaja on sokea omalle ohjelmalle eikä havaitse siinä kaikki virheitä ja ongelmakohtia. Aivan samaan tapaan ohjelmoija ei ole oikea henkilö tarkistamaan omaa koodiaan, vaan se tulee tehdä erillisen testaajan toimesta. Toimittajan omalla testaustiimillä tai ulkopuolisten asiantuntijoiden avulla voidaan hyväksymistestauksen suunnittelu aloittaa jo ohjelman kehitysvaiheessa. Lisäksi asianmukaisella dokumentaatiolla varmistetaan tiedon häviäminen hyväksymistestauksien välissä.

Ketkä ovat sitten oikeita henkilöitä suorittamaan

”ohjelmoija ei ole oikea henkilö tarkistamaan omaa koodiaan, vaan se tulee tehdä erillisen testaajan toimesta”

tilaajan hyväksymistestausta? Ovatko he testaajia, substanssin asiantuntijoita vai loppukäyttäjiä? Lähtökohtana on, että testaus tulee olla ammattilaisten suunnittelemaa ja sitä tulee asiantuntijoiden avulla peilata ohjelman määrittelyihin. Väärä asetelma on istuttaa kymmenen asiantuntijaa viikonlopuksi tietokoneiden ääreen tarkistamaan, että ohjelma toimii oikein. Tällöin testaus tulee vääjäämättä tarkistettua liian suppeasta näkökulmasta. Asiantuntijat tarkistavat vain ne kohdat, joka ovat heille tuttuja ja päivittäistyössä hyödyllisiä. Monimutkaisimpiin tapauksiin ei ehditä syventyä eikä niitä välttämättä osata testata. Lisäksi aineisto, jota vasten hyväksymistestausta suoritetaan voi olla liian suppea tuotantoon verrattuna. Pitää muistaa, että tuotannon tietokannasta otettu kopio ei sisällä uusimpia muutoksia ohjelmaan, vaan ne tulee sinne lisätä käsin. Tämä taas lisää hyväksymistestaukseen varattavaa työmäärää. Karkeasti voidaan todeta, että hyväksymistestauksen suunnittelun tulisi olla yhtä laajaa kuin järjestämätestauksen, mutta hyväksymiseen varattu aika on huomattavasti suppeampi. Näin ollen suunnittelun tulee olla kattavampaa ja huolellisempaa.

Kuten kaupankäynnin kultaisen säännön mukaisesti, onko asiakas aina oikeassa? Hyväksymistestauksen aikana tarkistetaan onko ohjelma määrittelyjen mukaisesti oikein toteutettu tai todetaan, että asiakkaan toiveet ovat muuttuneet kehityksen aikana eikä ohjelma vastaa tarvetta. Onnistuneen ja toimivan ohjelman todentaminen on useammin haastavaa tai lähes mahdotonta. Ongelmatilanteista tulisi sopia yhteistyössä toimittajan kanssa. Ihanteellisessa tilanteessa toimittaja aktiivisesti korjaa havaittuja virheitä hyväksymistestauksen aikana ja yhdessä tilaajan kanssa sopii mahdollisten muutoksien toteuttamisesta myöhäisemmissä versioissa. Tilaajan on välillä vaikea hyväksyä, ettei täydellistä ohjelma voida toteuttaa yhdellä kertaa, vaan kehitystyön tulee olla vaiheistettua ja pitkäjänteistä.

TESTAUKSEN AUTOMATISOINTI

Testauksen automatisointia käytetään hyväksymistestauksen apuna nopeuttamaan rutiinitehtävien tarkistamista. Hyväksymistestaus sisältää paljon triviaali tiedon tarkistamista, mikä on hidasta ja työlästä ihmiselle. Helposta testaamisesta tulee myös liian rutinoitua ja tehotonta. Testauksen automatisoinnilla tavoitellaan näiden tehtävien suorittamista ilman ihmistyötä. Automatisointi vaatii tilaajalta toimivia testauskäytäntöjä ja isoa rahallista panostusta työvälineiden hankintaan. Jälkimmäinen onkin yleisin este automatisointiprosessin käyttöönotolle. Onnistuneella hyväksymistestauksen automatisoinnilla asiantuntijat saadaan vapautettua vaativimpien testitapauksien tarkistamiseen. Ohjelmistotuotannon tehostuessa ja nopeutuessa myös hyväksymistestauksen aika supistuu.

Onneksi nykyään useammalle ohjelmistotalolle ja tilaajalle on selvää, ettei loppuasiakas eli ohjelman tuotannon aikainen käyttäjä voi olla hyväksymistestaaja. Silloin on liian myöhäistä ja kallista korjata virheellistä ohjelmistoa. Hyväksymisrutiineissa on vielä paljon kehitettävää ja parannettavaa. Valveutunut tilaaja tekee tilaamalleen ohjelmalle kattavan testauksen ennen sen hyväksymistä sekä vaatii toimittajaa korjaamaan havaitut puutteet. Tämä on kaikkien laatujärjestelmien mukaista toimintaa. Onnistunut hyväksymistestaus todentaa ohjelman ominaisuudet ja puutteet.

Teksti: Toni Huusko
Rajesh Singh

Toni Huusko ja Rajesh Singh toimivat molemmat testausasiantuntijoina Quality Consultant -nimikkeellä Qentinel Oy:ssä.

Uutena testaajana ketterässä projektissa

Astuimme vuoden alussa uuteen maailmaan, kun aloitimme ensimmäisessä ketterässä projektissamme; tehtäväksemme annettiin ohjelmiston toiminnallisuustestaus. Meillä ei itsellämme ollut aikaisempaa kokemusta ketteristä menetelmistä, olimme ainoastaan lukeneet muutamia artikkeleita aiheesta. Heti alusta alkaen huomasimme, että ketterät menetelmät vaativat testaajalta uuden ajatusmaailman omaksumista.

Vanha vesiputousmallin mukainen ajatus testauksen suunnittelun ja toteutuksen vaiheista sekä projektin aikataulusta oli heitettävä romukoppaan. Uusi ketterä maailma vaatii jatkuvaa testausta ja testaajien jatkuvaa kommunikointia kehittäjien ja määrittelijöiden kanssa. Lisäksi ketterä kehitys myös kaataa raja-aitoja eri osa-alueiden välillä, nimittäin määrittelijät, kehittäjät ja testaajat toimivat yhdessä ja kaikki osallistuvat jollain tavalla toistensa työhön, yhteisen päämäärän saavuttamiseksi.

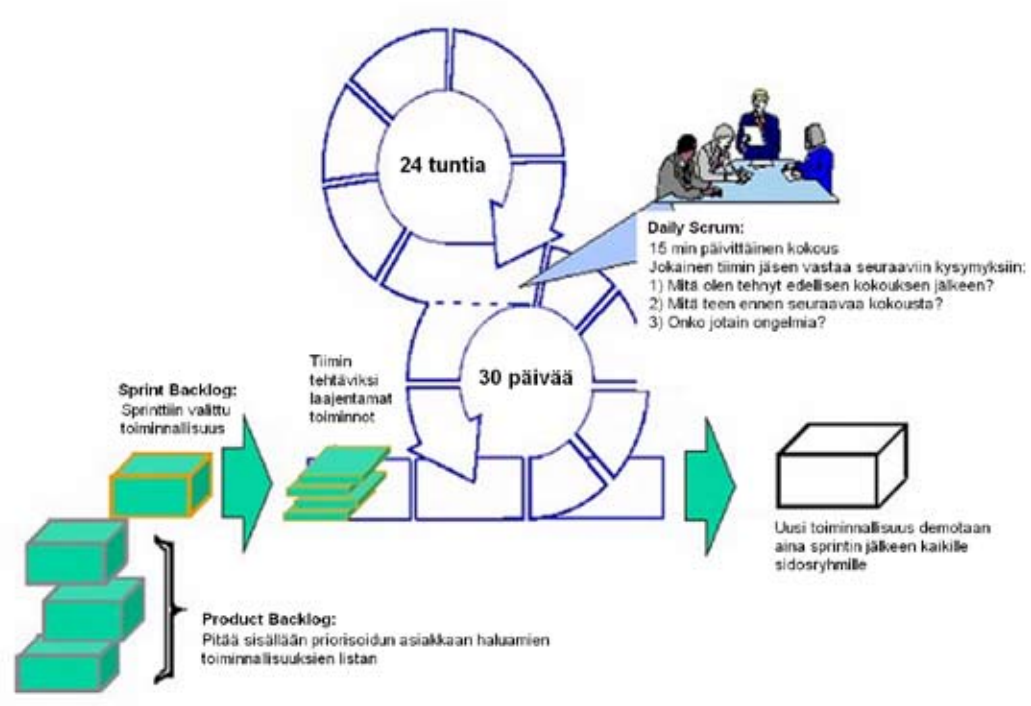
Projektissamme testaus etenee siten, että itse testaaminen pyritään aloittamaan heti alussa tutkivalla testauksella. Samaan aikaan määritellään toiminnallisuudet kattavia testitapauksia ja kun testitapauksia on saatu määriteltyä, siirrymme suunnitelmallisempaan testaukseen. Suunnitelmallisuuden tarkoituksena on tutkia järjestelmällisemmin ja syvällisemmin koko ohjelmiston toiminnallisuutta. Alun tutkiva testaus palvelee sitä, että testaajat oppivat ymmärtämään testauksen kohteena olevaa ohjelmaa ja suunnittelemaan parempia testitapauksia ja sitä että testaajat voivat löytää ilmiselviä virheitä jo heti projektin alkuvaiheessa ennen kuin testit ovat suunniteltu.

Asiakastyytyväisyys laadun mittarina

Ketterien menetelmien tärkein tavoite on asiakastyytyväisyys. Asiakastyytyväisyyden saavuttamiseksi asiakkaalle arvokas ohjelmisto toimitetaan usein jo aikaisessa vaiheessa. Projektissamme asiakastyytyväisyyteen pyrkiminen tarkoittaa sitä, että tuotetta pilotoidaan asiakkaalla ja piloteista kerätään arvokasta palautetta tuotteen tilasta. Meille pilotit ovat olleet hyviä oman työn laadun mittareita. Jos piloteissa ilmenee jotain ongelmakohtia, voidaan testausta tarvittaessa fokusoida näihin toiminnallisuuksiin (kuva 1).

Scrum

Projektissamme käytetään suosittua Scrum-menetelmää. Useiden ketterien menetelmien joukosta juuri Scrum valittiin sen takia, että se antaa rungon miten tehdään ja yksityiskohtaiset ketterät



Kuva1: Scrum prosessikaavio Lähde: http://www.scrum.dk/that_is_scrum.asp?sprog=engelsk

tekniikat voidaan päättää projekti- tai tiimikohtaisesti. Scrum on kevytrakenteinen ketterä prosessi, jota käytetään ohjelmisto- ja tuotekehityksen hallinnassa. Scrum on iteratiivinen, yleensä ajallisesti rajattu prosessi, jossa jokaisen iteraation jälkeen asiakkaalta saadaan palautetta ohjelmiston tilasta eli prosessi on myös evolutionäärinen.

Scrumin ydin on kommunikointi ja palaute kierros eli Sprint. Sprint on aikarajoitteinen, yleensä kuukauden pituinen, ohjelmistonkehitys pyristys, jonka jälkeen kehitys loppuu ja tietty osa ohjelmistosta on valmis. Kun useamman sprintin ohjelmiston osat kootaan yhteen, tuloksena on kokonainen tuote tai ohjelmisto. Jokaisen sprintin jälkeen järjestetään Sprint review, jossa kaikki projektin osapuolet, myös asiakas, katselmoivat toimitettavan kokonaisuuden ja mahdollisesti tekevät lisäyksiä Product Backlogiin eli tuotteen tehtäväluetteloon.

Ensimmäinen sprintti

Projektin ensimmäisen sprintin alettua aloimme pikkuhiljaa ymmärtää mitä ympärillämme tapahtui. Alussa muodostettiin sprintti-tiimit tekemään eri toiminnallisuuksia. Tyypillinen tiimi koostui 2-3 ohjelmoijasta, 1-2 testaajasta, sekä dokumentoijasta ja tiimipäälliköstä. Koko tiimi istui pyöreän pöydän ympärillä alueella, jota kutsuttiin projektialueeksi.

Päivittäisiä kokouksia

Projektissamme käytetään myös Daily Scrumia eli päivittäisiä lyhyitä kokouksia. Kokoukset kestävät yleensä noin 15 minuuttia ja niissä käsitellään henkilökohtaisesti mitä on tehty, mitä ollaan tekemässä ja onko jotain ongelmia, jotka häiritsevät työskentelyä. Kokouksia hallinnoi Scrum Master, joka myös huolehtii yleisten asioiden eteenpäin viemisestä.

Meille testaajina Daily Scrum kokoukset ovat olleet erittäin hyödyllisiä. Saimme päivittäin tietoa kehityksen tilasta ja pystyimme määrittelemään testitapaukset tehokkaammin ja tarkemmin. Myös tieto mahdollisista muutoksista oli helposti saatavilla näissä kokouksissa. Mielestämme kommunikointi eri osa-alueiden välillä oli antoisaa, saimme mahdollisuuden vaikuttaa siihen miten ohjelmistoa voidaan tehdä laadukkaammin ja toisaalta vinkkejä miten testata paremmin ja tarkemmin. Kokousten ansiosta pystyimme testaamaan ohjelmiston tehokkaammin ja mahdolliset riskit pystyttiin havaitsemaan jo aikaisessa vaiheessa kehitystä.

Tavoitteiden asettelua

Tuotteeseen tehtävät toiminnot on koottu asiakkaan vaatimuksista Product Backlogiin. Priorisoidut toiminnot valitaan product backlogista sprint backlogiin eli sprintin tehtäväluetteloon. Sprintin aloituspalaverissa määritellään aikataulu ja työaika-arviot. Toiminnot jaetaan tehtäviksi sprinttiin osallistuville henkilöille. Jos havaitaan, että sprint backlogissa on toimintoja, joita ei ehditä tekemään, tiimi voi valita ne poistettavaksi sprint backlogista.

Me testaajat käytimme Sprint Backlogia, kun laadimme alustavan testitapausluettelon. Seurasimme myös iteraation etenemistä Sprint backlogista ja siitä saimme selville myös kuka toteuttaa tietyn toiminnallisuuden. Pääsimme myös lisäämään Sprint backlogiin laatuun vaikuttavia tehtäviä.

Toinen sprintti

Toisessa sprintissä opimme lisää sprint-tason prosessista. Tässä projektissa sprintti oli 2-3 kuukauden pituinen ajanjakso, jossa tiimin tehtävänä oli toteuttaa juuri tähän sprinttiin valitut toiminnallisuudet. Tehtävänä oli myös korjata joitain edellisistä sprinteistä korjaamatta jääneitä bugeja.

Kolmas sprintti

Nyt kolmannen sprintin jälkeen olemme jo saaneet paljon kokemusta projektissa käytettävästä ketterästä prosessista. Sprintin aikana ymmärsimme, että järjestelmän hyväksyntätestien suorittaminen vei yli puolet testausajastamme. Huomasimme, että ilman testausautomaatiota ei järjestelmän testaaminen vain yksinkertaisesti onnistuisi järkevillä resursseilla ja halutussa ajassa. Tästä johtuen projektin johto päätti tutustuttaa meidät uuteen automaattiseen hyväksyntätestaustyökaluun.

Koulutuksen ja harjoittelun jälkeen olimme valmiita ottamaan uuden testausautomaatiotyökalun käyttöömme ja aloittamaan automaatiotestitapausten suunnittelun ja tekemisen seuraavaa sprinttiä varten.

Testausautomaation merkitys erityisesti regressiotestauksessa on suuri. Jokaisen muutoksen jälkeen koko järjestelmän regressiotestaus olisi liian suuri ponnistus. Nyt kun suuri osa regressiotestauksesta voidaan ajaa automaattisesti, voidaan regressiota tehdä esimerkiksi joka päivä tai joka kerta kun uusi paketti ohjelmistosta on saatavilla. Testien tilaa voidaan monitoroida eri tavoilla esimerkiksi html-pohjaisten raporttien avulla.

Testausautomaation työnjako

Projektissamme automaation tekeminen jaettiin niin, että automaattisista yksikkötesteistä vastaavat kehittäjät ja automaattisista hyväksyntätesteistä vastasimme me testaajat. Kehittäjät laativat yksikkötestit määrittelyiden pohjalta ja tämän jälkeen implementoivat uuden toiminnallisuuden järjestelmään. Automaattiset hyväksyntätestit suunnittelimme asiakasvaatimusten pohjalta, vastasimme myös niiden ylläpitämisestä.

Automaattisten hyväksyntätestien suunnittelu

oli helppoa, koska projektissa käytettiin avainsanaohjattua testausympäristöä, jossa testitapaukset määriteltiin sanallisesti. Emme juuri tarvinneet kehittäjien apua itse testitapausten tekemiseen. Kehittäjille automaation osalta suurin ponnistus oli uuden avainsanakirjaston tekeminen työkalua varten.

Kun testausautomaation taso oli korkea, pystyimme manuaalitestauksessa keskittymään tutkivaan testaukseen. Tällöin pääsimme suunnitelmallisesti testaamaan erilaisia erikoistapauksia ja virhetoimintoja. Tämän koimme hyödylliseksi koska helposti toistettavat testit suoritettiin automaattisesti ja pystyimme keskittymään enemmän poikkeustilanteiden ja asiakkaille kriittisten toimintojen testaamiseen.

Kommentteja ja kokemuksia

Kokonaisuudessa kokemuksemme ketterästä kehityksestä ovat hyviä. Toki aluksi oli vaikea muokata omia vanhakantaisia ajatuksiaan uusiksi. Kun aloitimme, projekti oli siirtymävaiheessa vesiputousmallista ketterään malliin. Projektin aikana myös projektiorganisaatio on kehittynyt valtavasti ketterien menetelmien käytössä.

Meillä henkilökohtaisesti siirtymävaihe vesiputousmallista ketterään malliin sujui aika helposti ja aikaa siihen kului varmaankin muutama viikko, mutta koko ison organisaation muuttaminen täysin ketterään maailmaan kestää varmasti kokonaisuudessaan vuosia.

Organisaatiolle siirtyminen ketteriin menetelmiin on suuri urakka, jossa kannattaa käyttää ulkopuolisia asiantuntijoita, jos organisaation sisältä ei löydy kivenkovaa kokemusta asiasta. On ollut hienoa seurata kuinka prosessit ja käytännöt ovat kehittyneet askeleittain sprinttien ja ohjelmiston kehittymisen ohella. Meille testaajille erityisesti keskittyminen regressiotestauksen sijasta tutkivaan testaukseen oli haastavaa ja mielenkiintoista. Projektin aikana meille on selvinnyt myös se, että testausautomaation rooli ketterissä menetelmissä on erittäin suuri.

Testaajilta vaaditaan
ketterässä projektissa:

1. Aktiivista kommunikointia,
2. Tietoa tutkivasta testauksesta,
3. Perustietoja ketterästä kehityksestä ja
4. Tietoa testattavasta tuotteesta.

Teksti: Juha Itkonen

Juha Itkonen, tutkija
Teknillinen korkeakoulu, SoberIT

Kirjoittaja toimii testauksen opettajana ja tutkijana Teknillisessä korkeakoulussa. Tutkimustyössään ohjelmistoprosessien tutkimusryhmässä häntä kiinnostavat erityisesti iteratiivisen ja inkrementaalisen ohjelmistokehityksen laadunvarmistus, tutkiva testaus sekä ketterät ohjelmistokehityksen menetelmät.

Tutkimuksen ja käytännön välisen kuilun partaalla

— Mitä oikeastaan tiedämme testauksesta tutkimustiedon perusteella?

Testauksen menetelmät ja toimintatavat ovat monille tämän lehden lukijoille käytännöstä tuttuja ja useimmilla on ymmärrystä ja ajatuksia hyvistä ja toimivista testausmenetelmistä. Monella alalla menetelmien ja mallien hyvyys ja toimivuus pyritään osoittamaan luotettavilla, riippumattomilla tutkimuksilla ja empiirisillä kokeilla Ohjelmistotuotannossa perinne on pitkään ollut toisenlainen. Tarjoan seuraavassa näkökulmia testauksen tutkimuksesta käytännön tekijöille.

Nykyään testaus tunnistetaan jo useimmiten elintärkeäksi osaksi ohjelmistokehitystä. Testauksen merkitys laadukkaan lopputuotteen aikaansaamisessa ymmärretään ja testaukseen halutaan panostaa monissa yrityksissä aiempaa enemmän. Testauksen ja testaajien arvostuskin näyttäisi olevan rivakassa nousukiidossa. Mutta miltä näyttää testaukseen liittyvän tutkimuksen tila?

Ohjelmistojen testausta on tutkittu paljon ja testaukseen sekä erityisesti sen automatisointiin ja optimointiin on kehitetty lukuisia menetelmiä, tekniikoita, algoritmeja, malleja ja työkaluja. Useimmat testauksen asiantuntijat ovat kuitenkin edelleen sillä kannalla, että yksi testauksen perustavoite, eli uusien vikojen löytäminen uudesta tai muokatusta koodista, on yhä vahvasti ihmisten tekemän manuaalisen testauksen varassa. Empiiriset tutkimukset teollisuuden testauskäytännöistä osoittavat, että ohjelmistoyrityksissä automatisointia hyödynnetään nykyäänkin varsin vähän, vaikka testauksen automatisointia tutkitaan paljon ja työkaluja on saatavilla runsaasti.

Testauksen automatisointi on työlästä ja haastavaa, eikä sen voida lähitulevaisuudessa olettaa syrjäyttävän tai oleellisesti vähentävän esimerkiksi manuaalisen toiminnallisen testauksen tarvetta. Tietotekniikan alan tutkijoille automatisointi tuntuu olevan varsin kiinnostava tutkimuskohde – onhan se mukava siinä mielessä, että voidaan keskittyä malleihin, algoritmeihin ja työkaluihin, sivuuttaen kätevästi sellaisia hankalia tekijöitä kuten ihmiset ja heidän toimintansa. Testauksen ammattilaiset kuitenkin tietävät, että testaajalla ja hänen osaamisellaan ja toiminnallaan on ratkaiseva merkitys testauksen lopputulokselle.

Testaus on merkittävä osa ohjelmistotuotannon prosesseja, ja ihmisten sekä tiimien toimintana siihen liittyy myös sosiaalipsykologisia ja sosiologisia ilmiöitä. Lisäksi testaus on kognitiivisesti vaativaa työtä, jossa testaajan tekemä aivotyö, ajattelu, oppiminen, muistaminen ja havaitseminen, on merkittävässä roolissa testauksen suorittamisen aikana. Kaikki nämä näkökulmat edustavat erilaisia tutkimusparadigmoja, joilla testauksen tutkimusta voidaan lähestyä. Ohjelmistotestausta käsittelevä tutkimus keskittyy kuitenkin varsin vahvasti testaukseen teknisenä automatisointiongelmana ja optimointihaasteena.

Testaus tutkimuksen kohteena voidaan nähdä:

- Teknisenä ongelmana
- Yksilön suorittamana kognitiivisesti haastavana tehtävänä
- Prosessina tai ohjelmistoprosessin osana
- Sosiologisena tai sosiaalipsykologisena ilmiönä

Ohjelmistotuotannon ja –prosessien tutkijana tutkin manuaalista testausta prosessina sekä testaajan työtä, siihen liittyviä toimintatapoja, strategioita ja tekniikoita. Olen kiinnostunut selvittämään, miten manuaalista testausta kannattaisi tehdä sekä mitkä tekijät vaikuttavat testauksen tehokkuuteen ja tuloksiin. Tutkija lähtee ensin selvittämään mitä testauksesta oikeasti jo tiedetään ja miten, luotettavien tutkimustulosten valossa, testausta pitäisi tehdä ja mitä menetelmiä käyttää. Hämmennys onkin suuri, kun huomaa olevansa kuilun partaalla – tieteellisen, tutkimukseen perustuvan tiedon ja käytännön testaustyön välisen kuilun partaalla.

Ohjelmistotuotanto on tutkimusalana nuori

Ohjelmistotuotannossa tutkitaan menetelmiä, toimintatapoja ja prosesseja, joita ohjelmistoprojekteissa käytetään. Ohjelmistotuotannon

tutkimusala on kovin nuori, moneen muuhun tieteenalaan verrattuna, ja vahvaa, vakiintunutta tutkimustraditiota ei ole ehtinyt muodostua. Tästä johtuen alamme käytettävissä oleva tieto on usein konsulttien ja käytännössä kokeneiden ihmisten tuottamaa kirjojen, ammattilehtien, kurssien ja seminaarien muodossa. Tämä tietämys on usein hyödyllistä ja arvokasta käytännön työssä. Tutkijan näkökulmasta tämänkaltainen tietämys on kuitenkin ongelmallista. Tutkimus ja tieteellinen tieto rakentuu aina olemassa olevan tiedon ja ymmärryksen varaan. Tieteellisestä näkökulmasta kaikki tieto, joka ei perustu luotettavaan tutkimukseen, on lähtökohtaisesti epäluotettavaa, eikä sen varaan voida rakentaa syventävää jatkotutkimusta.

Tietotekniikan alalla teknologia kehittyy erittäin nopeasti ja uusia innovaatioita syntyy vahvasta markkinavetoisesta käytännön tarpeesta eikä niinkään vuosikausien fokusoituneen tutkimuksen tuloksena. Tämä tietotekniikan alan yleinen piirre näkyy varsin vahvasti myös ohjelmistotuotannon alueella: uusia menetelmiä ja malleja kehitellään miltei jokaisessa alan tutkimuslaitoksessa ja yrityksessä. Kun näitä malleja ja menetelmiä ei ehditä koskaan kunnollisilla tutkimuksilla validoida (siis testata), on erittäin vaikea edistää tutkimusta ja syventää tietyn alueen ymmärrystä. Ohjelmistotuotannon alalla ei olekaan vielä kovin vakaata, laajoilla ja luotettavilla perustutkimuksilla luotua perustaa, jolle voitaisiin helposti rakentaa. Tutkimuksen edistämiseksi tarvitaan luotettavilla tieteellisillä foorumeilla julkaistuja tasokkaita tutkimuksia, jotka ovat läpikäyneet tiedeyhteisön kollegiaalisen vertaiskatselmoinnin. Tällainen tieto muodostaa vankan pohjan jatkotutkimukselle.

Esimerkiksi lääketieteessä edellytetään hyvin laajat ja useita kertoja eri tutkijoiden toimesta toistetut sokkotestit, ennen kuin tietyn lääkkeen tai hoitomenetelmän voidaan katsoa olevan tehokas, turvallinen ja vaikuttavan tiettyyn sairauteen tai oireisiin. Ohjelmistotuotannossa käytäntönä on ollut, että yksittäisen tutkijan, konsultin tai yrityksen ehdottama menetelmää tai työkalu otetaan hyväksyen käyttöön. Jos tämä tutkija on vieläpä kokeillut tuota menetelmäänsä hyvin tuloksin, niin tämä tuntuu riittävän osoitukseksi tehosta ja toimivuudesta. Ero on melko lailla vastaava kuin on apteekista lääkärin määräyksellä saatavilla lääkkeillä ja lehdissä julkkisten suosituksilla markkinoitavilla rohdosvalmisteilla: ”Minä sain avun ja paranin, osta sinäkin”. Jälkimmäiset voivat olla tehokkaita ja hyviä tiettyyn vaivaan tietyissä olosuhteissa, mutta toimivuudesta ja tehosta ei vain ole mitään luotettavia todisteita. Myös ymmärrys niistä olosuhteista joissa ne toimivat, ja mitkä ovat niiden ongelmat ja haittavaikutukset, puuttuu.

Testauksen perusteet ja tutkittu tieto

Palataanpa takaisin tutkimuksen ja käytännön kuilun äärelle. Olen tutkinut manuaalisten testien suorittamiseen liittyviä tekijöitä vikojen löytämisen tehokkuuden näkökulmasta. Tämä tutkimusaihe liittyy tiiviisti mm. viime vuosina jonkin verran huomiota saaneeseen tutkivan testauksen (exploratory testing) lähestymistapaan, jossa testaus ei perustu tarkasti etukäteen suunniteltuihin ja dokumentoituihin testitapauksiin. Koska tutkivan testauksen lähestymistavalla on väitetty löydettävän vikoja erittäin tehokkaasti, lähdin ensin käymään läpi olemassa olevan tutkimuksen testien suorittamiseen ja tehokkuuteen sekä testien dokumentointiin liittyen.

Olemassa olevan ymmärryksen pohjalta aioin rakentaa uusia hypoteeseja ja tutkimuskysymyksiä. Huomasin hyvin pian, että sellaiset testauksen peruspilarit kuin perustestaustekniikoiden tehokkuus sekä erityyppisen testidokumentaation hyödyt ja vaikutus testauksen lopputuloksiin ovat tieteellisessä kirjallisuudessa melko vähälle huomiolle jääneitä kysymyksiä. Oli mielenkiintoista, mutta toisaalta vähän pelottavaakin huomata, että esimerkiksi testitapausten suunnittelun ja dokumentoimisen hyödyt ja vaikutus testauksen tuloksiin ylipäänsä on täysin tutkimatonta aluetta. Tutkivan testauksen syvemmälle tutkimukselle tämä antaa tietysti varsin heiveröiset lähtökohdat. Erityisen kiinnostavaksi tämän havainnon tekee se, että lähes kaikki testauksen oppikirjat, koulutus ja tutkimus perustuvat testitapausten suunnittelulle ja dokumentoinnille. Tältä pohjalta olisi olettanut, että jo vuosikymmeniä sitten olisi tehty tutkimuksia, jotka osoittaisivat, että testauksen suunnittelu nimenomaan testitapauksina ja suorittaminen niitä tarkasti seuraamalla olisi selkeästi paras tapa tehdä testausta. Tällaisia tutkimuksia, saati tuloksia, ei kuitenkaan tunnu löytyvän.

”testaustekniikoiden tehokkuus sekä erityyppisen testidokumentaation hyödyt ja vaikutus testauksen lopputuloksiin ovat tieteellisessä kirjallisuudessa melko vähälle huomiolle jääneitä kysymyksiä”

Testaustekniikoita koskevat empiiriset tutkimukset vertailevat eri testitapausten suunnittelutekniikoita ja saavat varsin vaihtelevia ja osin ristiriitaisia tuloksia. Mikään tutkimus ei ole pystynyt osoittamaan, että testitapausten suunnittelutekniikalla olisi erityisen merkittävä vaikutus löydettyjen vikojen määrään. Aivan yhtä merkittävä vaikutus on esimerkiksi testausta suorittavalla henkilöllä ja testattavan ohjelmiston tyypillä, ja vieläpä niin, että useamman henkilön yhdistelmällä löydetään enemmän vikoja kuin useamman tekniikan yhdistelmällä. Siis vanha nyrkkisääntö: ”Eri tekniikoilla löydetään eri vikoja”, pätisikin paremmin muodossa: ”Eri testaajat löytävät samalla tekniikalla eri vikoja”. Jos testitapausten suunnittelutekniikoiden tavoite on tuoda systematiikkaa ja toistettavuutta testaukseen ja sen tuloksiin, niin tähänastiset tutkimustulokset eivät tue tätä käsitystä.

Itse asiassa muutamat empiiriset tutkimukset, joissa on kartoitettu teollisuudessa sovellettavia testauskäytäntöjä, ovat raportoineet, että testitapausten kurinalainen suunnittelu ja dokumentointi ei ole kovin yleistä ja sitä pidetään varsin vaikeana, työläänä ja jopa hyödyttömänä. Käytännössä näyttää, että on yleistä jättää yksityiskohtainen testitapausten valinta testauksen suorittajan vastuulle, eikä tätä edes koeta yrityksissä ongelmaksi. Tähän voisi tietysti sanoa, että ei ole menetelmän vika, jos sitä ei ymmärretä yrityksissä käyttää. Toisaalta tässä olisi mielestäni meidän tutkijoiden paikka osoittaa tutkimuksilla, mitä ja kuinka suuria ovat eri menetelmien hyödyt ja mitkä menetelmät ovat kustannustehokkaita. Lisäksi tilannetta voisi yrittää parantaa kehittämällä vähemmän työläitä ja helpommin käytännössä sovellettavia menetelmiä perustuen käytännössä todettuihin haasteisiin menetelmien soveltamisessa.

Sen lisäksi, että tutkittua tietoa testauksenkin alueelta tuntuu puuttuvan paljon, on tutkimusten ja käytännön välillä havaittavissa toisenkinlainen kuilu. Kun vertaan tutkimuskirjallisuudesta välittyvää kuvaa ja toisaalta yrityksissä tehtävää käytännön työtä, näyttäytyvät nämä kaksi maailmaa varsin erilaisina. Yrityksissä tehtävä testaustyö vaikuttaa käytännössä olevan varsin monipuolista ja erilaista eri konteksteissa. Tutkimus taas näyttää pyrkivän herkästi joko yleispäteviin menetelmiin tai sitten löytämään pitkälle vietyjä ratkaisuja yksittäisiin hyvinkin spesifeihin ongelmiin. Tuntuukin joskus siltä, että testaajien käytännön työtä seuraamalla on vaikea tunnistaa niitä tekniikoita ja menetelmiä, joita kirjallisuus kuvaa, tai niitä ongelmia, joita tutkimus tutkii.

Testauksen alueen tutkimattomia kysymyksiä:

Mitkä eri tekijät vaikuttavat manuaalisen testauksen tuloksiin ja miten paljon?

Miten testitapauksien käyttäminen vaikuttaa testauksen suorittamiseen ja tuloksiin?

Mitä eri tapoja testien suunniteluun, hallintaan ja dokumentointiin on olemassa? Miten nämä eri tavat toimivat erilaisissa tilanteissa?

Mitä tekniikoita testauksen suorituksen aikana voidaan soveltaa? Miten? Millaisin tuloksin?

Mitä ovat ne yksilölliset taidot, jotka aiheuttavat yksilöiden väliset suuret erot testaustyössä? Miten näitä taitoja voisi kouluttaa kokemattomille testaajille?

Miten ja minkälaisilla menetelmillä ja välineillä testausta tehdään teollisuudessa?

Mitkä ovat yritysten vaikeimmat ongelmat ja haasteet testauksen alueella tällä hetkellä?

Silta tutkimuksen ja käytännön kuilun yli

Koska ohjelmistotuotanto on luonteeltaan varsin erilaista kuin esimerkiksi lääketiede, tulisi tutkimuksen tällä alalla seurata ketterästi käytännön työstä kumpuavia loistavia ideoita, menetelmiä ja työkaluja. Tutkimus ei ole pelkästään uusien konstruktioiden kehittämistä teoreettisen laboratoriotyön kautta, vaan erityisesti olemassa olevien ilmiöiden tutkimista ja ymmärryksen lisäämistä. On olemassa monenlaista tutkimusta: uusien konstruktioiden kehittäminen (keksiminen) ja validointi, olemassa olevasta tiedosta ja teorioista johdettujen hypoteesien testaaminen sekä jonkin asian nykytilan kuvaaminen ja dokumentointi. Lisäksi on monenlaisia tutkimusmenetelmiä, joilla tutkimusta voidaan tehdä. Voidaan tehdä kontrolloituja tai osittain kontrolloituja kokeita, haastattelu- tai kyselytutkimuksia, tapaustutkimusta (case study) tai konstruktiivista tutkimusta. Näistä kaksi viimeisintä ovat tietotekniikan ja ohjelmistotuotannon alalla tyypillisiä, mutta tutkimusmielessä myös varsin haastavia menetelmiä.

Tutkimuksella on
erilaisia tavoitteita:

- Uusien menetelmien ja tekniikoiden keksiminen
- Hypoteesien testaaminen, eli olemassa olevan tiedon ja teorioiden tai olettamusten vahvistaminen tai hylkääminen.
- Nykytilan kuvaaminen ja dokumentointi, eli ymmärryksen lisääminen jostain asiasta tai ilmiöstä.

Teoreettista tutkimusta näkee varsin runsaasti esimerkiksi automatisoidun testauksen alueella. Automatisointi ja optimointi ovat tietysti tärkeitä ja puoleensavetäviä tutkimuskohteita, mutta käytännössä näyttäisi että suurimmalle osalle ohjelmistoyrityksistä testauksen automatisoinnin optimointi ei ole laadunvarmistuksen ongelma numero yksi. Soisi siis mielellään tutkimuspanostusta myös niille testauksen alueille joilla suurin osa yrityksistä painiskelee. Testauksen alalla tarvittaisiin kipeästi esimerkiksi olemassa olevien testausmenetelmien käytännön toimivuuden, tehokkuuden ja soveltuvuuden riippumatonta tutkimista empiirisesti todellisessa käytössä ja ympäristössä, siis kokeellisen ja tapaustutkimuksen menetelmillä. Olisi mukava nähdä tutkimusta, jotka keskittyisi enemmän testauksen tutkimiseen ja vähemmän uusien asioiden keksimiseen. Uusien menetelmien

keksimisen hyötykin lienee kyseenalainen, jos olemassa olevienkaan menetelmien toimivuudesta ja tehokkuudesta ei tiedetä paljoakaan, eikä siten näitä uusia menetelmiä voida helposti verrata olemassa oleviin.

Myös testauksen, kuten monen muunkin ohjelmistotuotannon alueen, osalta käytäntö on ajanut menetelmien ja tekniikoiden kehitystä siinä määrin, että tutkimustieto on jäänyt selkeästi jälkeen. Tässä olisi siis oivallinen tilaisuus kaikille suomalaisille testauksen asiantuntijoille kantaa oma kortensa kekoon tutkitun ja luotettavan testauksen alueen tiedon tuottamiseksi yhteistyössä eri yliopistojen tutkijoiden kanssa vaikkapa lisensiaatintutkintojen ja väitöskirjojen muodossa. Tarvitsisimme tutkimusta, joka kuvaisi, miten testaustyötä käytännössä tehdään, ja antaisi vastauksia sellaisiin kysymyksiin kuten: Mitä menetelmiä ja työkaluja testaajat käytännössä käyttävät? Kuinka hyvin nämä toimivat verrattuna oppikirjamenetelmiin? Mitkä ovat suurimmat haasteet ja ongelmat käytännön testauksen ja ohjelmistokehityksen alueella? Tämänkaltaisten tutkimusten tulokset auttaisivat myös muita tutkijoita fokusoitumaan relevantteihin tutkimuskysymyksiin.

Suomessa meillä on kansainvälisesti ajatellen poikkeuksellisen tiiviit välit yliopistojen ja yritysten välillä, mikä tarjoaa erinomaiset mahdollisuudet käytännöllisesti relevantin tutkimuksen tekemiselle. Myös sinä voit pyrkiä nostamaan testauksen alueen tutkimusta kaipaavia kysymyksiä esille. Ja vaikka juuri sinä et tuntisi tutkimustyötä ja jatkotutkinnon tekoa omaksesi, niin aina voit pyrkiä vaikuttamaan siihen, että kaikissa oman yrityksesi ja tutkimusmaailman välisissä yhteistyöhankkeissa keskityttäisiin relevantteihin ongelmiin ja hyvän tutkimuksen tekemiseen.

Teksti: Matti Vuori

Mistä TestausOSY:n jäsenistö puhuu?

Kyselytutkimuksen kertomaa

Testauksen maailma muuttuu. Koko professionaalisen testauksen paradigma on historiallisesti uusi ja samalla ohjelmistokehityksen ja systeemityön maailma on jatkuvassa murroksessa. Tilanne on haastava, sillä samalla kuin pyritään vakiinnuttamaan stabiileja perusmalleja, on testauksen kyettävä uudistumaan ja toimimaan tehokkaammin ja laadukkaammin. TestausOSY:n yksi keino uusien haasteiden ja kansallisten painopistealueiden tunnistamiseen on tehdä vuosittaisia jäsenkyselyjä, joissa kuunnellaan jäsenten suoraa puhetta. Tässä artikkelissa esitellään vuoden 2006 kyselyn keskeisiä tuloksia sekä tulkintoja ja johtopäätöksiä niistä.

Kyselyssä katse tulevaisuuteen

Kyselyn strategiana oli laadullinen tutkimus ja suuntautuminen tulevaisuuteen. Koska elämme muuttuvassa maailmassa, emme tarjonneet valmiiksi pureksittuja vastausvaihtoehtoja, vaan annoimme jäsenten äänen kuulua omanlaisenaan. TestausOSY:n sähköpostilistalla oli kyselyn ajankohtana 389 henkilöä. Kyselyyn saatiin 26 vastausta. Lisäksi listalla oli kaksi toimimatonta sähköpostiosoitetta. Vastausprosentti oli siten 6,7 %. Kysely perustui avoimiin kysymyksiin, joilla pyrittiin ilman ohjausta nostamaan esille sellaisia asioita, jotka aidosti ovat läsnä "kentällä" ja joita pitäisi yhdessä käsitellä, tutkia ja kehittää. Jämäköiden tilastojen sijaan etsimme hedelmällisiä ajatuksia ja tulkintoja tilanteesta, jotta niistä voidaan tunnistaa testauksen maailmaan vaikuttavia heikkoja ja vahvoja signaaleja.

Miten maailma muuttuu ohjelmistokehityksessä?

Vanha sananlasku sanoo, että "konteksti on kaikki". Siksi avainkysymys testauksen kehittämisessä on tunnistaa sen toimintaympäristössä tapahtuvia muutoksia, joihin testauksen - siis kulttuurisena ilmiönä ja osaamisalueena - pitää reagoida joko hyvissä ajoin proaktiivisesti tai ketterästi tilanteiden muuttuessa nopeasti. Tai mieluiten molemmilla tyyleillä! Tässä ajattelussa korostuu se, että mekanistisessa ajattelussa testaus on vain prosessitason asia, jonka ketteryys on testausprosessin ketteryyttä, mutta testaus on myös toimintajärjestelmä ja ajattelumallien avaruus, jonka toimivuus luo valmiudet prosessitason toiminnalle.

Testaus ei koskaan ole yksinäinen saareke. Keskeisin osa ohjelmistojen testauksesta tapahtuu ohjelmistokehityksen yhteydessä ja siksi onkin oleellista tunnistaa sellaisia ohjelmistokehityksen muutosilmiöitä, jotka vaikuttavat testaukseen.

Vastaajat nostivat esille jo tuttuja toimintaympäristön muutoksia, eli mm. ohjelmistokehityksen siirtämisen ulkomaille. Samalla lisääntyvät monitoimittajaprojektit, joissa tapahtuva testaus edellyttää monenlaisten toimintamallien ja kulttuurien yhteensovittamista. Mikään toimittajan CMMI-tasolupaus ei auta tällaisessa tilanteessa. Ongelma on suuri, koska testauksen lähtötilanne on harvalla toimijalla optimaalinen ja muutosten keskellä pitäisi myös testata paremmin ja tehokkaammin. Ja samalla myös osa testausta saatetaan siirtää ulkomaille! Testauksenhallinnan merkitys kasvaa valtavasti. Tehdyn testauksen todistaminen on yhä tärkeämpää - ohjelmistotoimittajan lupaus testauksesta ei enää riitä, vaan tarvitaan raportteja todisteena.

Ohjelmistokehitysprosessien alueella ketterien menetelmien soveltaminen etenee ja testauksen kehittäminen on myös osa ketterien menetelmien siirtämistä kohti professionaalista mainstreamia. Mutta harva organisaatio käyttää niitä sellaisenaan. Sen sijaan organisaatioissa tulee olemaan monia prosessimalleja, joiden rajat hämärtyvät. Tämä edellyttää testaukselta uudenlaista ohjelmistokehityksen ja oman toiminnan ymmärtämistä. Mallipohjainen kehittäminen tuo uusia haasteita ja edistää kehittämisen aikaista testausta, samoin kuin systemaattisen yksikkötestauksen yleistyminen. Työkalujen kehittyessä ei enää ole tekosyitä siitä tinkimiseen! Ohjelmistokehityksessä aletaan tiedostaa, että kehittäjien tekemä testaus on tärkein testaustaso ja avain robustiin ohjelmistoon ja myöhempien testaustasojen sujuvuuteen.

Keskeinen haaste testaukselle on järjestelmien kompleksisuus ja järjestelmien keskinäinen integroituminen. Samalla tulee käyttöön jatkuvasti uutta teknologiaa. Tämä lisää testaustarvetta, mutta yhtälö on ongelmallinen, koska samalla aikataulut lyhenevät ja osaamistakin pitäisi nostaa samassa tahdissa. Testauksen painopiste siirtyy järjestelmä-integrointitestaukseen ja regressiotestaukseen. Yhtenä tuoteteknologioiden muutosilmiönä on open source, jonka testaustarvetta ei ole aina oivallettu - kantapään kautta ale-

taan oppia, että muualla tehty ei ole testaamatta valmis käyttöön.

Eräitä ratkaisuja ovat testauksen strategioiden muutokset. Vähitellen aletaan oivaltaa, että nykytilanteessa prosessien lopussa oleva testausvaihe ei kerta kaikkiaan onnistu, koska testattavaa on niin paljon, viat pitäisi korjatakin - ja koska aikataulut loppuvat aina kesken. Testauksesta halutaan aikaisin alkavaa ja jatkuvaa toimintaa, jota ohjaavat tavoitteet.

Testausautomaatioon luotetaan edelleen, mutta kantapään kautta opittujen kokemusten ansiosta siihen suhtaudutaan realistisemmin kuin aikaisemmin. Systemaattisen yksikkötestauksen ja mallipohjaisen testauksen avulla siihen tulee uutta monimuotoisuutta. Savutestit ja perus-regressiotestit ovat realistisia perinteisiä testauskohteita.

Testauksen maailman muutoksia

Yleinen käsitys on, että testauksen arvostus kasvaa edelleen. Testauksen merkitys ymmärretään ja sen eri osa-alueet osataan paremmin jäsentää organisaatioissa. Hyvää testausta alkaa tapahtumaan pienemmissäkin yrityksissä.

Testaajien työnkuva muuttuu ohjelmistokehityksen muutosten myötä. Enää ei olla prosessin loppupäässä testaustyötä odottavia henkilöitä, vaan päästään mukaan toimintaan varhaisemmassa vaiheessa. Testauksen näkökulma laajenee virheiden metsästyksestä laadunvarmistukseen. Tämä edellyttää organisaatioissa uudenlaisia roolikäsityksiä. Muutosta on nähtävissä ensisijaisesti ketteriä prosesseja käyttävissä organisaatioissa.

Olennaista onkin priorisoinnin taito. Se korostuu erityisesti systeemitestauksessa. Tämä merkitsee sitä, että testaussuunnittelussa on ymmärrettävä korkean tason bisnesvaatimuksia ja riskejä. Priorisoinnin tavoite on saada tärkeimpien asioiden testaus alkamaan mahdollisimman nopeasti. Tämä edellyttää aikaista, integroitua osallistumista projekteihin ja korkean tason varhaista testaussuunnittelua, jolla voidaan myös vaikuttaa ohjelmistokehittäjien näkemyksiin siitä, mitä pitää saada aikaiseksi missä järjestyksessä. Testauslähtöinen kehittäminen makrotasolla on tärkeä lähtökohta projektien pitämiseksi kurissa.

Testauksen osaamistarpeet kasvavat. Jo mainitun teknologiaosaamisen lisäksi on ymmärrettävä liiketoimintaa ja asiakastarpeita. Muuten ei tehokas, priorisoitu ja riskiperusteinen toiminta onnistu. Uutena haasteena ovat uudet testityypit ja niiden tehokas toteuttaminen. Monissa organisaatioissa testaus on ollut pääosin toiminnallisuustestausta, mutta järjestelmien vaatimusten - sekä objektiivisten, että kulttuurin ja järjestelmäymmärryksen kehittyessä odotetun vaatimustason - kasvaessa nousee esille esimerkiksi suorituskyvyn, tietoturvallisuuden ja käytettävyyden varmistaminen. Nämä ovat monille organisaatioille sokeita pisteitä, mutta vähitellen käsitykset ja osaaminen

Testauksen lähitulevaisuuden haasteet ovat monenlaisia.

kehittyvät. Haasteena on se, että testauksen erityisalueet edellyttävät erikoistuneita ammattilaisia. Testausosaamisen kehittäminen onkin monen organisaation toimenpidelistalla korkealla.

Testauksen näkeminen palvelutoimintana kehittyy. Koska testausta tehdään heterogeenisissä olosuhteissa, sen aiempaa parempi määrittäminen ja jäsentäminen on tärkeää. Rajapinnat tiedonkulussa ja toimituksissa ovat entistä olennaisimpia. Testauspalvelut voivat olla ulkoisia palveluita tai sisäisiä palveluita. On olennaista, että tarjolla ei ole yhtä testauspalvelua, vaan sortimentista löytyy joustavia ratkaisuja organisaation kaikkiin testaustarpeisiin. Palvelukehityksessä on prosessien määrittelyn lisäksi olennaista ajattelumallien kehittäminen asiakaslähtöisiksi - maailma ei pyöri testauksen ympärillä. Tästä teemasta on toistaiseksi vähän jäsennettyä tietoa, mutta työtä asian parissa on Suomessakin alettu tehdä (mm. kirjoittaja ja Erkki Pöyhönen kehittävät asiaa jatkuvasti).

Testausohjelmistojen keskeinen ongelma on ollut niiden korkea hinta. Open source -työkalujen valikoima alkaa kasvaa. Ne ovat jo mullistaneet yksikkö- ja integrointitestausta, suorituskykytestausta ja vianhallintaa. Myös testauksenhallinnan ohjelmistot alkavat kypsyä. Enää ei ole yrityksen budjetista kiinni se, että käytössä on asianmukaiset välineet.

Organisaatiot muuttuvat

Ulkoistusten lisäksi organisaatioissa tapahtuu muunkinlaisia rakenteellisia muutoksia. Ei ole tämän vuosikymmenen ilmiö, että yritykset yhdistyvät tai niistä erotetaan uusia yrityksiä tai että ulkomaisten yksiköiden kanssa pitäisi luoda sujuvaa yhteistyötä. Asian tekee testauksen osalta akuutiksi se, että kun testausta on systematisoitu ja sille luotu toimintaympäristö ja infrastruktuuri,

organisaatiomuutokset ovatkin aiempaa vaikeampia! Siinä, missä epäkypsä adhoc-toiminta voidaan käynnistää hetkessä uudestaan, on systemaattisten prosessien harmonisointi uusien kumppanien kanssa kertaluokkaa vaikeampaa. Uusi testauksen kehittämisen tarvealue onkin selvästi se, miten yhdistyneiden toimijoiden testauskäytännöt saadaan toimimaan yhteen ja miten uudelle organisaatiolle luodaan nopeasti uusi testaustoiminta. Koska tuotteiden pitää olla huippuluokkaa välittömästi, ei ole aikaa kehittää toimintaa orgaanisesti vuosien ajan - riittävän kehityskaaren pitää tapahtua muutamissa kuukausissa. Perinteinen ongelma on edelleen se, miten kasvavissa organisaatioissa harmonisoidaan testaustavat ja kehitetään niitä yhtenäisesti, kuitenkin tukien yksiköiden tarpeita ja mahdollisuuksia.

Toiminta yhteisöissä — TestausOSY:n toiminnan laatu

Vähitellen on alettu oivaltaa, että dynaamisessa ja kompleksisessa toimintaympäristössä on tärkeää osallistua oman organisaation ulkoisiin yhteisöihin. Tämä ajatus esiintyy jo tietoturvastandardissa ISO/IEC 17799 Information technology - Security techniques - Code of practice for information security management, jossa esitetään, että "appropriate contacts with special interest associations should be maintained", jotta mm. saadaan tietoa parhaista käytännöistä, asiantuntijaneuvoja ja vaihdetaan tietoja ja kokemuksia muiden kanssa. Moderni testaus on samantyyppinen aihepiiri ja siksi voidaankin hieman provosoivasti sanoa, että organisaation testaustoiminta ei ole riittävällä tasolla, jos organisaatiosta ei ole jäseniä TestausOSY:ssä!

Ajatuksesta voidaan johtaa se, että TestausOSY:n toiminta on oikeasti tärkeää ja toimintamallin on oltava tarkoituksenmukaisia. Nykyinen toiminta on pitkälti keskittynyt seminaareihin, jotka ovatkin saaneet kiitosta ja antaneet merkittävää arvoa osallistujille. Seminaaripohjainen toiminta ei kuitenkaan ole kovin vuorovaikutteista, ja siksi jäsenet toivovatkin uusia yhteistoiminnan muotoja, kuten vuorovaikutteiset työpajat, Open Space -sessiot, vierailut toisten työpaikoilla, projektit, ryhmätyöt, yhteinen sisällöntuotanto jne.

Seminaarien konkreettinen ongelma on ollut toiminnan pääkaupunkiseutupainotteisuus. On turha kuvitella kovin monen lähtevän Keski-Suomesta iltaseminaariin Espooseen. Nyt onkin esim. Oulussa ollut jo paikallista aktiviteettia. Iltaseminaareja on muutenkin kyseenalaistettu, sillä osana testauksen professionaalisuuden kasvua ei testausyhteisössä toimiminen ole enää harrastustoimintaa, vaan työhön integroitu kiinteä, työnantajalle lisäarvoa tuottava osa.

Mutta keskeistä on luoda sähköisiä vuorovaikutuskanavia, virtuaalisia kokouksia, verkkokeskustelufoorumeja jne. Tarjolla on nykyisin hyviä välineitä tämän alueen kehittämiseen.

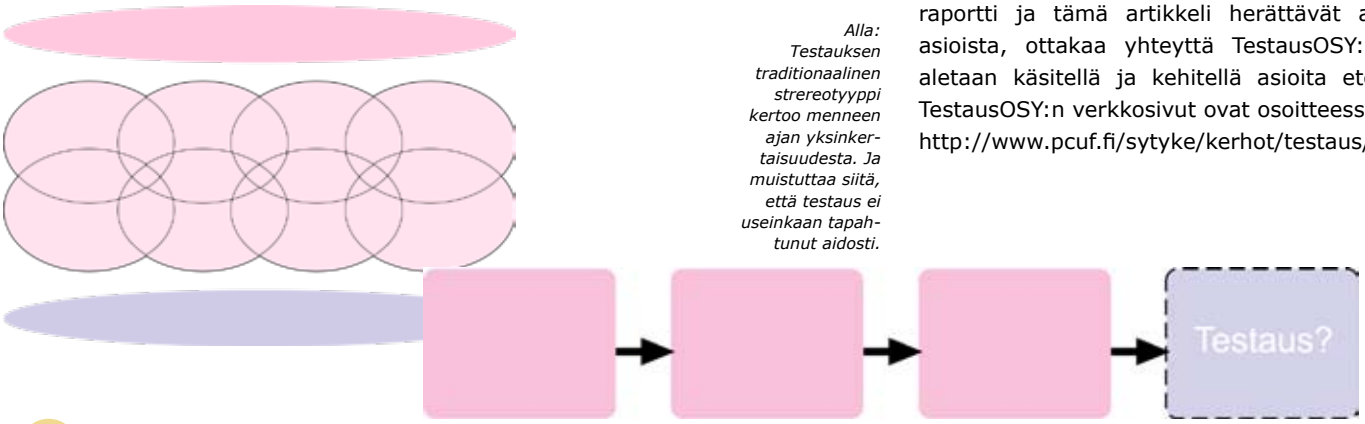
On lisäksi muistettava se, että kuten testaus ei ole oma saarekkeensa organisaatioissa, se ei ole oma saarekkeensa osaamisalueiden verkostossa. TestausOSY:n yhteistyö muiden OSY:jen kanssa on tärkeää. Yhteisseminaarit ja muu yhteistoiminta auttavat luomaan uutta ymmärrystä, uusia ideoita ja uusia synergioita.

Valoisa tulevaisuus edessä

Testauksella on monia haasteita, mutta yleisesti ottaen tulevaisuus on hyvin valoisa. Testauksen toimijoiden parissa on saavutettu hyvä kokonaisvaltainen ymmärrys omasta toiminnasta, sen roolista ja mahdollisuuksista. Samalla toimintaympäristö antaa yhä paremmat mahdollisuudet tehdä hyvää testausta, vaikka ollaankin aiempaa vaativammassa ympäristössä. Mutta se on yleinen maailmaamme liittyvä ilmiö, eikä testaus ole siinä suhteessa missään erityisasemassa. TestausOSY:n koetaan selkeästi auttavan kansallista kehittymistä energisenä osaamisyhteisönä, joka pyrkii kehittämään toimintaansa.

Tässä artikkelissa ei ole raportoitu kaikkia kyselyssä esille nousseita teemoja. Kyselyn raportti löytyy TestausOSY:n sivustolta, suorasta osoitteesta http://www.pcuf.fi/sytyke/kerhot/testaus/arkisto/testaus-osy_jasenkysely_2006_raportti.pdf. Raportti on jätetty puhtaan dokumentaariseksi, jotta johtopäätösten tekoon olisi kaikilla osapuolilla hyvät mahdollisuudet. Osa nykypäivän osallistuvaa ja verkottunutta maailmaa onkin se, että kaikilla on mahdollisuudet tehdä tulkintoja nykytilasta ja tulevaisuuden haasteista. Jos raportti ja tämä artikkeli herättävät ajatuksia asioista, ottakaa yhteyttä TestausOSY:yn, niin aletaan käsitellä ja kehitellä asioita eteenpäin. TestausOSY:n verkkosivut ovat osoitteessa http://www.pcuf.fi/sytyke/kerhot/testaus/.

Moderni testaus integroituu ohjelmistokehitykseen, on jatkuvaa, syklistä, hallittua. Samaan aikaan se on kompleksista ja vaativaa.



Alla: Testauksen traditionaalinen strereotyyppi kertoo menneen ajan yksinkertaisuudesta. Ja muistuttaa siitä, että testaus ei useinkaan tapahtunut aidosti.

Teksti: Maaret Pyhäjärvi

Testauksen onni ja haasteet osana tuotekehitystä

Maaret Pyhäjärvi, Senior Quality Engineer, F-Secure Oyj

Maaret Pyhäjärvi toimii testauksellisissa tehtävissä F-Securella ja osallistuu tuotekehitys- ja toimitusprojekteihin, joissa toimitetaan tietoturvatuotteita Windows työasema- ja palvelinympäristöihin. Hänen työhönsä kuuluu testausta ja testauksen opastamista projekteissa, sekä yleistä testaustoiminnan ja -käytäntöjen kehittämistä F-Securella.

Testaus on haastavaa ja hauskaa - ainakin kun tuotekehitysporukassa on yhteistyössä tekemisen meininki. Tämä artikkeli on tarina siitä miltä testaus meillä F-Securen Windows työasema- ja palvelintuotteiden tekemisessä näyttää, kun suuntana kehitys on vienyt ketterästä testauksesta ketterään ohjelmistokehitykseen.

Ketterästä testauksesta ketterään kehitykseen

Testausporukka - tai laatuinsinöörit, niinkuin meitä F-Securella kutsutaan - on jo jonkin aikaa omannut varsin kohtuullisen ja keskeisen roolin osana tuotekehitystä. Testaukseen keskittyviä asiantuntijoita on yksi kahta ohjelmistokehittäjää kohden, ja olen kahden F-Securella viettämäni vuoden aikana oppinut tuntemaan itseni etuoikeutetuksi saadessani tehdä töitä F-Securen Windows työasema- ja palvelintietoturvatuotteiden parissa osana osaavaa ja oppivaa porukkaa. Osana tuotekehitystä testausta on kehitetty eteenpäin riskeihin ja tiedon arvoon perustuen. Resurssit ovat aina rajalliset, aikataulut haastavat ja valitut käytännöt haetaan toiminnasta ja sen tavoitteista, ei testauksen teoriakirjoista. Testauksessa on mietitty vuosien varrella tarkkaan testausdokumentaation arvoa ja roolia, testaustyön jakamista ohjelmistokehittäjien kanssa sekä läheistä yhteistyötä liiketoiminnan kanssa päätöksenteon osalta. Jo reilu vuosi sitten olin valmis luonnehtimaan F-Securen testaustoimintaa ketteräksi. Toimimme mukavasti yhdessä eri alueiden testaajien kesken, jaoimme töitä ja tavoitteita suhteessa siihen mikä oli kokonaisuudessamme tärkeintä ja neuvottelimme asioista erilaisten sidosryhmien kanssa. Haimme aktiivisesti soveltuvia käytäntöjä ja kehitimme toimintaamme eteenpäin.

Viime loppusyksy toi toimintaamme uutta lisäväriä ja haastetta kahden suuren ohjelmistokehitykseen kohdistuneen muutoksen myötä: vaihdoimme koko ohjelmistokehitysrakenteemme perustumaan ketteriin menetelmiin sekä toimme samanaikaisesti käytäntöön tuotelinjan korvaamaan aiemmat erilliset tuoteprojektit. Ketterään kehitykseen siirtyminen on pakottanut meidät oppimaan paljon ja muuttamaan toimintatapojamme. Tuotelinja-ajattelu on kuitenkin monella tapaa ollut muutoksena testauksen näkökulmasta vielä haastavampi. Tuotelinja vaihtoi tavoitteen yhden tuotteen tekemisestä kaikkien tuotteiden tekemiseen, ja siinä missä aiemmin haasteenamme oli käydä läpi yksittäinen tuote ja järjestelmä, nyt lisäksi haemme testaustulosten uudelleenkäyttöä ja sopivaa täydentämistä samoihin teknologioihin perustuvien varsin erilaisten tuotteiden toimittamiseen. Kun tämän vielä yhdistää 30-päivän inkrementaalisiin toimituskierroksiin, opittavaa on ollut paljon varsin vauhdikkaan vuoden aikana.

Testauksen organisointi suhteessa ohjelmistokehitykseen

Testaus ei ole itsenäinen toiminne, vaan palveluntarjoaja varsin moneen suuntaan ohjelmistokehityksessä. Testausta tekevät meillä niin erikoistuneet testausrooleissa toimivat laatuinsinöörit kuin ohjelmistokehittäjämmekin, ja vastaavasti testaustyötä ja tavoitteita jaetaan yhtä lailla liiketoiminnan kuin tuen edustajien kanssa. Testaus on investointia laatutietoon, ja organisoinnin keskeisenä tavoitteena on saada oikeat asiat tehtyä oikeaan aikaan oikeiden ihmisten toimesta kokonaisuuden optimoimiseksi. Perusjakona testaajien ja kehittäjien roolituksessa on että siinä missä kehittäjät toimittavat mahdollisimman hyviä osia järjestelmiin, testaajat lähtevät pilkkomaan riskejä järjestelmän ja käyttäjän näkökulmasta.

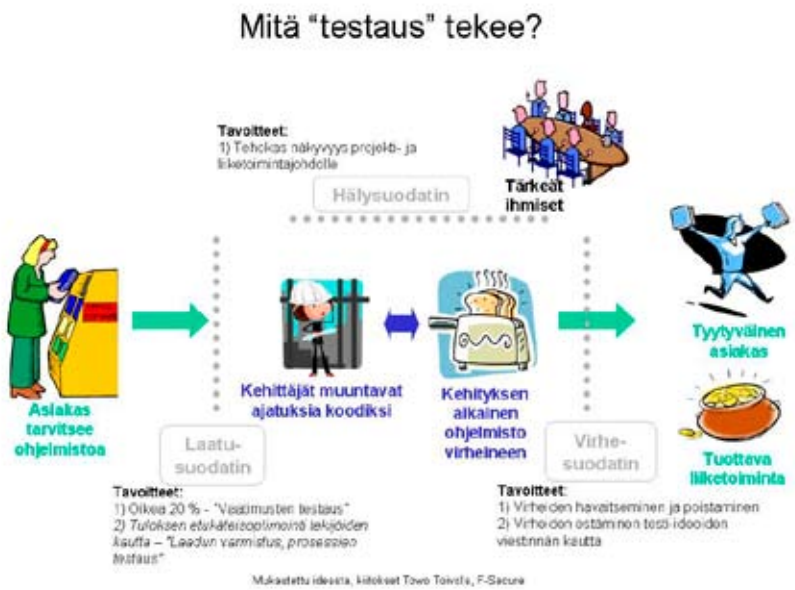
Ketterän ohjelmistokehityksen myötä olemme siirtyneet pienien tiimien toimintamalliin, jossa testaajat ovat osana ohjelmistokehitystiimiä, joista kukin toimittaa omia muutoksiaan päivittäisen integroinnin kautta osaksi kokonaisjärjestelmää. Viimeisen vuoden aikana olemme kokeilleet niin erillistä järjestelmätestausryhmää osana ketterää ohjelmistokehitystä kuin nyt valittua pienten tiimien polkua. Käytännön kokemukset viittaavat että nyt valitsemamme polku vie oikeaan suuntaan: tiimien kyky nähdä muutokset osana kokonaisuutta ja toimittaa asioita jotka ovat lähempänä asiakkaan kokemaa valmista järjestelmää on

parantunut. Työn jakamista on helpottanut että ei ole ryhmää jolle asioita voisi siirtää, vaan asiat ovat oman porukan listoilla - porukan kokoonpanoa säädellään sopimaan tavoitteisiin siirtämällä osaa testaajista tiimeistä toisiin eri kuukausina.

Testaajien yhteinen suunta ja tavoite

Testaajien liittyminen ja jakaantuminen ohjelmistokehitystiimeihin on otettu varsin hyvin vastaan. Jo ennen ketterää ohjelmistokehitystä toimimme varsin tiiviisti yhdessä ja yhteistyö on entisestään tiivistynyt. Ehdotus erillisen testausporukan perustamisesta saa nykyään jo varsin yleisen vastarinnan aikaan, menettäisimme sisäpiirin tiedon siitä millaisia muutoksia ohjelmistoon on tullut ja tulossa. Testausnäkökulman kärjistyminen liian kehittäjäsuuntautuneeksi on ollut huolenaiheena, joka tosin F-Securen kokeneen testausporukan kanssa ei ole muodostunut ongelmaksi. Olemme tiimeissä mukana olemisen lisäksi ylläpitäneet käytäntöä testausnäkökulman viikkokohtaamisista, jossa vaihdamme kunkin edustajan keräämää sisäpiiritietoa muutoksista ja riskeistä, ja etsimme mahdollisuutta optimoida toimintaamme yli tiimirajojen.

Yhteinen suunta ja tavoite on parantunut myös testauksen ja hallinnollisen puolen osalta. Testauksen haasteita osana toimitusta ratkotaan yhteistyössä resurssiomistajien kanssa, etsin luovia ratkaisuja niin oman porukan tekemisten ja tavoitteiden kuin ulkoisten toimittajien käyttämisen osalta. Testaus tuottaa hyödyllistä ja käyttökelpoista tietoa, ja saa vastineeksi yhä paremmat mahdollisuudet toimia tämän tiedon toimittajana.



Vaatimukset, toimitukset ja suunnittelun tärkeys

Ketterä ohjelmistokehitys on muuttanut suunnittelun käytäntöjä ja roolia. Etenkin kokonaisuuden suunnittelu ja yhtä tiimiä suuremman porukan työn synkronointi on opettanut vuoden aikana paljon ketterän ohjelmistokehityksen luonteesta. Toimintaa ohjaa varsin konkreettisesti kuukausittaiset toimitukset ja sen miettiminen mitä seuraavassa sprintissä toimituksen tavoitteeksi otetaan. Pidemmän aikavälin suunnittelun mekanismit perustuvat tuotebacklogiin ja kunkin kuukauden karkean toimitussuunnitelman tarkasteluun. Tuotebacklog korvaa projektisuunnitelman, ei niinkään vaatimusmäärittelyä. Vaatimusten ymmärtämiseksi perinteiset menetelmät käyttötapausten läpikäymiseksi ovat osoittautuneet edelleen toimiviksi.

Iso muutos virheraporttien käsittelyssä on ollut muutoshallinnan mekanismin jakaminen tiimeissä toimiviin muutosraateihin (change control board) ja yli tiimirajojen toimivaan muutosneuvostoon (change advisory board). Tiimeissä tiedetään muutoksen yksityiskohdat, kustannukset ja riskit, kun taas ulkopuolelta neuvotaan lähinnä riittävän laadun konkretisoinnissa. Muutosneuvosto auttaa oppimaan konkreettisemmin millaista on hyvä laatu meidän toimitustemme osalta, jotta tiimien muutosraati osaa itseohjautuvasti tehdä oikeansuuntaiset kompromissit. Pienentynyt lisäkustannus on auttanut meitä parantamaan laatua merkittävästi, kun usein parannus tehdään vastaavassa ajassa kuin perinteisin prosessein on saatu neuvoteltua josko muutosta yleensäkään voi tehdä.

Testausdokumentaatio

Testauksen tarkoituksena on tuottaa tarpeellista tietoa, ja dokumentaatiolla on rooli siltä osin kuin se on hyödyllistä. Käytännön kautta on opittu tekemään testitapauksia varsin toisella tapaa kuin testauskirjallisuus opastavaa, käyttäen testitapausmäärittelyjä testaustoiminnan runkona ja muistilistana, mutta ei yksityiskohtien dokumentoijana ja puolustusmekanismina. Ei ole itsestäänselvää että saman testin suorittaminen uudelleen olisi parasta ajankäyttöä, joskin tärkeistä näkökulmista pidetään toistuvasti huolta.

Nykyinen sovittu toimintatapamme hyödyntää käyttötapauksia testauksen lähtökohtana. Testitapausmäärittelyihin kirjataan käyttötapauksia kahdella tapaa: hyötynäkökulmasta (mitä käyttäjä haluaa) ja toteutusnäkökulmasta (mitä toteutamme käyttäjälle). Näiden käyttötapausten ympärillä teemme tutkivaa testausta, rajaten kunkin testin laajuutta budjetoidulla ajalla. Sama tavoite kahden tunnin aikarajalla ja kahden viikon aikarajalla löytää varsin erilaisia asioita, ja sopiikin erilaisten muutosten riskien analysointiin.

Testauksen tilanteen raportointi perustuu yhä suuremmassa määrin porukan näkemysten yhteen vetämiseen ja perinteiset testauksen mittarit toimivat lähinnä muistuttajina, eivät raportoinnin lähtökohtana.

Ketterän kehityksen myötä testaussuunnitelmien asemesta muokataan tuotebacklogia sisältämään toimituskohteita jotka sisältävät kohtuullisen testauksellisen tavoitteen, sekä luodaan suoraan tuotedokumentaatiota loppukäyttäjälle testausraporttien asemesta.

Opittua: mitä jäi käteen?

Toimiminen varsin haastavassa ympäristössä pakottaa oppimaan joka päivä. Yhteistyössä tekeminen haastaa itse kunkin auttamaan kaveria ratkomaan esteitä. Oppiminen vaatii melkoista luovuutta ja jatkuvaa halua etsiä parasta tapaa kulkea kohti tavoitetta - kunhan ensin huolehditaan että tavoite on ymmärretty puolin ja toisin.

Opit yhteistyöstä pienissä tiimeissä

Testaajan ammattitaito ja eheys pannaan koetukselle kehittäjien lähellä

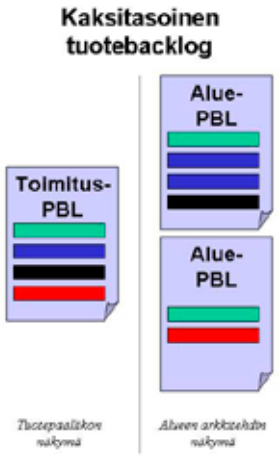
Etenkin uudet testaajat saattavat suostua vääriin kompromisseihin tärkeiden näkökulmien osalta kehittäjien lähellä toimiessaan, ja unohtaa muita palveltavia sidosryhmiä. Testaajaverkoston ylläpitäminen tuntuu auttavan säilyttämään oikeita painotuksia ja kehittymään eteenpäin, niin että läheisyyden hyödyt ylittävät mahdolliset haitat.

Kehittäjät osaavat testata

Ohjelmistokehittäjät osaavat testata ja analysoida omien osiensa toimintaa osana järjestelmää, kunhan vain aikaa analysointiin on. Tästä huolimatta työnjakomielessä toimii varsin mainiosti nopeamman aikataulun saavuttamiseksi se että kehittäjät testaavat osan ja erilliset testaajat osan. Ketterien tiimien resursoinnissa olemme joutuneet joinain kuukausina luomaan myös porukoita joissa on vain kehittäjiä, jotka osaavat varsin mainiosti kantaa vastuun niin tekemisen kuin toimituksen laadun osalta. Läheinen yhteistyö on opettanut arvostamaan osaamista puolin ja toisin, ja etenkin automatisoinnin osalta yhteistyö tuottaa mainiota tuloksia tavoitteen ja toteutuksen kohtaamisen kautta.

Testaus tarvitsee joustoa myös kuukausitoimitusten sisällä

Kuukausitoimituksen suunnittelu perustuu kiinteisiin työmääriin, joiden käyttö suunnitellaan kuukausi kerrallaan tarkemmin. Tämä voi väärin sovellettuna johtaa tarpeettomaan jäykkyyteen testauksen työmäärien suhteen, kun viikon testauksen jälkeen opitaan että työmäärät toisella tapaa kohdistettuna toisivat paremman lopputuloksen. Testaus on jatkuvaa oppimista riskeistä, ja kukin päivä testausta oleellisesti muuttaa suuntaa ja tarvetta loppuinvestoinnille. Tämä oppi on ollut erityisen haastava kun testaustyö jakaantuu useille tiimeille. Uusi suunnittelukäytäntö alkuun vähensi testauksemme ketteryyttä, kunnes opimme suunnittelemaan järkevällä yksityiskohtien tasolla.



Testauspäällikön rooli murroksessa

Testauspäällikön toimenkuva ja tarpeellisuus on etsinyt käytännön toteutustaan uudessa toimintamallissa, ja on muuttunut varsin paljon. Aiemman kokonaisuuden testauksellisen koordinoinnin sijasta testauspäällikön työ on keskittynyt tavoiteasentantaan osana tuotebacklogien tekemistä, pitäen huolta siitä että valmiuden käsite konkretisoituu yhä paremmin. Suoraa käskyvaltaa itseohjautuviin tiimeihin ei voi olla, tiimit suunnittelevat testaustyönsä ja testauspäällikön rooliksi jää osoittaa mahdollisia huolenaiheita vastuiden päällekäisyyden tai puutteen osalta useiden tiimien kuviossa.

Testausyhteistyö useiden tiimien kesken

Päällekäisyyksien ja aukkojen välttäminen samaa kokonaisuutta muuttavien pienten tiimien kesken vaatii yhteistyötä ja koordinointia useiden tiimien kesken. Testausnäkökulman synkronointi viikottain ja aktiivinen yritys optimoida kokonaisuutta on tuntunut toimivan, etenkin kun sitä täydentää toisinaan testaussuunnittelun työpajoilla, joissa jaetaan kokonaisuuden vastuita yhteistyössä. Tiedon siirto tiimien välillä oikealla yksityiskohtien tasolla on edelleen suurimpia haasteita, vaikka olemme sen osalta työtä paljon tehneet.

Opit järjestelmätestauksen järjestämisestä

Stabilointisprintit ovat tarpeellisia

Vaikka yritämme huolehtia että työtä ei vuoda ulos kuukausitoimituksista, toimituskokonaisuudet laajuudessaan vaativat enemmän aikaa. Toisinaan on tarpeen suunnitella kokonaisia kuukausia selvitys- ja stabilointitavoitteen ympärille, parantaen laatua myös aiemmista tuotejulkaisuista, sallien pidemmän aikavälin katsoa järjestelmää kokonaisuutena yli tiimirajojen.

Oppimisen aikaväli on suurempi kuin sen työmäärä

Usein asiaan omaksuminen vaatii aikaa kypsyttelyyn. Ihmiset tarvitsevat usein aikaa haudutteluun ja uusien ideoihin syntymiseen. Oppimisen mahdollistava aikamäärä tuntuu usein olevan suurempi kuin oppimisen työmäärä. Yksilökohtaiset erot ovat merkittäviä, toisilla oppiminen oivallukseen saakka tuntuu olevan nopeampaa. Ketterän kehityksen lyhyet toimitusajat vaativat nopeaksi oppijaksi oppimista.

Valmiuden määrittely tuotebacklogin kautta

Testaussuunnitelmien sijasta huomio on siirtynyt tuotebacklogin toimitussisältöjen testaukselliseen näkökulmaan. Isoimpia oivalluksia meille oli tajuta että tuotebacklog ei korvaakaan vaatimusmäärittelyä, vaan projekti- ja testaussuunnitelmat, ja sitä kautta auttaa meitä keskittymään jatkuviin toimituksiin.

Sitoutumisesta ja laadusta raportointi

Eri osapuolten yhteisen ymmärryksen työstäminen siitä mitä tullaan saamaan valmiiksi on edelleen päivittäisen oppimisen kohde. Eri osapuolet ajattelevat eri abstraktiotasoilla, ja kompromissina olemme hakeneet kaksitasoisen tuotebacklogin keskeisimpien näkökulmien mukaan tuomiseksi. Laadun kontrolli on siirretty tiimeihin, muutoksista ja korjauksista päättää tiimien sisäinen muutosraati ulkoisen muutosohjauksen perusteella - ulkopuolelta kerrotaan sääntöjä tärkeydelle ja tuetaan jatkuvaa oppimista siitä mikä laadulle on tärkeää.

Jatkuvan beta-ohjelman organisointi

Ulkoisiin toimituksiin liittyy lisäkustannus, ja meidän oli tarpeen minimoida ulkoisten toimitusten määrää kuukausitoimitussyklin aikaan saamiseksi. Teimme tämän yhdistämällä eri tuotteiden betaohjelmia yhteen yhdeksi jatkuvaksi betaksi, johon toimitetaan kuukausittain. Isoja oppeja on saatu beta-ohjelmien tavoitteista ja hyödyistä, ja niiden luovasta yhdistelemisestä oikean asiakaspalautteen saavuttamiseksi oikeaan aikaan.

Laaturaportti

	Julkaisuhaaran koonnit	Kehityshaaran koonnit	Muutostahti
KULLEKIN ALUEELLE			
Toiminnallisuusalue 1	1	1	↑
Toiminnallisuusalue 2	3	1	↑
Toiminnallisuusalue 3	3	2	→
Toiminnallisuusalue 4	2	2	↑
Toiminnallisuusalue 5	2	2	↓
Toiminnallisuusalue 6	1	2	↓
JÄRJESTELMÄLLE			
F - toiminnallisuus	2	2	↑
U - käytettävyys	1	1	↓
R - luotettavuus	1	1	↑
P - suorituskyky	0	0	↑
S - tietoturva	1	1	↓
S - tuettavuus	0	0	↓

Tulkinta:

Numerot - testauksen kattavuus; testaajien subjektiivinen arvio ryhmänä
- 0 - emme ole testanneet
- 1 - hiukan katsottu, mutta ammattimainen testaaja ei voi väittää testanneensa
- 2 - peruskattavuus, perustilanteet
- 3 - aidosti testattu

Värit - ohjelmiston laatu; testaajien subjektiivinen arvio ryhmänä
- vihreä - toimii, ei isoja ongelmia
- keltainen - jotain ongelmia, mutta toimii
- punainen - isoja ongelmia, ei toimi

Nuolet - muutosmäärä ohjelmistossa; yhteinen arvio kehittäjien kanssa
- nuoli ylös - paljon muutoksia, mitätöi tietotason, testattava uudelleen
- nuoli sivulle - joitain muutoksia, hallittava testaus tuottaa merkittävää tietoa
- nuoli alas - vakiintuva aluea, vain pieniä muutoksia, testaus etenee eikä toista

Opit suunnittelun tärkeydestä

Tuotebacklogit ovat yhteistyön tulos

Olemme huomanneet parhaiden tuotebacklogien syntyvän yhteistyössä niiden osapuolien kesken jotka sitä yhteisenä mekanisminaan käyttävät: luonnollinen yhdistelmä on projektipäällikön, testauspäällikön ja tuotepäällikön yhteistyö, niin että tuoteomistaja (tuotepäällikkö) omistaa priorisointinäkökulman.

Etukäteissuunnittelu sisältyy tuoteomistajan rooliin

Alkuun keskityimme oppimaan kuinka toimitetaan kuukauden osissa, kiinnittäen vähemmän huomiota kokonaisuuden suunnitteluun pidemmällä aikavälillä. Sittemmin olemme huomanneet roolien muutoksen siltä osin että pidemmän aikavälin suunnittelu tehdään tuotebacklogin kautta osana tuoteomistaja-näkökulmaa, joka meidän rakenteissamme jakaantuu kolmelle roolille: tuotepäällikölle (prioriteetit), program managerille (resurssit) ja projektipäällikölle (viestintä). Etukäteissuunnittelu on keskeistä, jopa suuremmassa määrin ketterässä kehityksessä kuin perinteisessä, joskin varsin toisenlaisin mekanismein.

Järjestelmäajattelu ja arkkitehtuurit ovat tärkeitä

Järjestelmän ja sen arkkitehtuurin ymmärtäminen on perusta jolla tiimimme toimittavat. Sen merkitys ei ole lainkaan vähentynyt ketteryyden myötä, päin vastoin arkkitehtuurien merkitys on kasvanut tuotelinjan myötä. Arkkitehtuurit ovat kriittinen onnistumistekijä kehityksen menetykselle.

Haastetta tulevaan

Jokainen päivä opettaa meille edelleen uutta. Jatkamme samojen pohdintojen kanssa ja haemme päivittäin ratkaisuja, joilla työn sisältöä ja kykyämme toimittaa voidaan parantaa. Tekemistä riittää edelleen: haluaisimme toki yhä enemmän vastaavalla panostuksella, aina hommista ja haasteista nauttien.

Teksti: Kari Kakkonen

Projektiliiketoiminnan vaikutukset ja vaatimukset testaukselle

Kari Kakkonen, Johtaja, Testaus ja menetelmät, Endero Oy

Kirjoittaja toimii Testaus ja menetelmät –osaamiskeskuksen johtajana sekä konsulttina ja kouluttajana Endero Oy:ssä, TestausOSY:n isännistön jäsenenä sekä ISTQB Expert Level Working Groupin jäsenenä.

Testausta tarvitaan joka paikassa. Vaikka testausmenetelmät ja taidot ovat periaatteessa, ja usein käytännössäkin, siirrettävissä eri tyyppisiin projekteihin, testaus on silti hyvinkin erilaista erilaisissa ympäristöissä. Projektiliiketoiminnassa testaus on erittäin tärkeässä osassa. Usein testaus on normaalin laadunvarmistuksen lisäksi myös varmistamassa toimituksen laatua. Toisaalta testaukseen ei välttämättä ole mahdollistaa panostaa vastaavalla tavalla kuin tuoteliiketoiminnassa.

Testauksen rooli projektiliiketoiminnassa vs. tuotekehitysliiketoiminnassa

Projektiliiketoiminnalla tarkoitetaan tässä mitä tahansa ohjelmistokehitystä millä tahansa yritystoiminnan tai julkisen sektorin alalla, mikä toteutetaan projektiluontoisesti. Pääpaino on räätälöidyllä järjestelmällä, vaikka toki myös tuotekehitysprojekteja toteutetaan projektiluontoisesti. Yleistykset perustuvat kokemuksiimme suomalaisissa yrityksissä. Taulukossa (kuva 1) on muutamia yleistyksiä, jotka olemme havainneet usein paikkansapitäviksi.

Projektimuotoisen ohjelmistokehityksen ominaispiirteitä;

- määrittely-, ohjelmistokehitys- ja testausvaiheet on projektoitu
- projekti on luonteeltaan kertaluonteinen, jatkokehitykselle ei ole välttämättä näkyvyyttä
- lopputuloksena on usein tietojärjestelmä, joskus toki ohjelmistotuotekin
- lopputulokselle järjestetään ylläpito, mutta se on luonteeltaan reaktiivinen
- koodaus ja määrittelyvaiheet on usein ulkoistettu tai tilattu joltakin alan toimijalta

Testauksen vaiheet projektin aikana

Projektimuotoisessa ohjelmistokehityksessä erottuvat usein toisistaan selvästi aika ennen ja jälkeen projektin. Projektia edeltää määrittelyvaihe ja sitä seuraa ylläpitovaihe. Näihin vaiheisiin on joskus vaikea saada mukaan testausnäkökulmia, mutta toisaalta se olisi erittäin olennaista laadunvarmistuksen kokonaisonnistumisen kannalta.

Huomioitavaa testausnäkökulmien osallistuessa määrittelyvaiheeseen:

- Tuotteen määrittelyvaiheeseen osallistuminen vie testaajien aikaa pois mahdollisesta toisesta aktiivisesta projektista
- Testaajan käytännön maailman tunteminen tuo uuden näkökulman tuotekehitykseen ja tuotteen arkkitehtuuri rakennetaan automaattisesti helposti ylläpidettäväksi ja testattavaksi
- Testaajat täytyy olla aikaisin tiedossa, jotta määrittelyvaiheessa saatavat opit voidaan tehokkaasti hyödyntää

Ylläpidon aikaisen testauksen järjestäminen on usein unohtuva, mutta olennainen tehtävä. Varsinainen testausprojekti täytyy ensinnäkin usein venyttää hieman pidemmäksi kuin varsinainen projekti – sekä testausroolien, että ohjelmistokehitysroolien osalta. Näin julkaisun jälkeen saada nopea vasteaika kriitisimpien virheiden korjaukseen. Projektin aikana valmistellaan ylläpitovaiheen testaus myös laajemmin. Kun projekti on päätetty ja ylläpitovaihe on alkanut sataprosenttisesti – usein kuukausia tuotantoonoton jälkeen – täytyy edelleen päättää mitä tapahtuu testaukselle jatkossa.

Kun tulee ylläpidon aikaisia virheitä tai muita muutostarpeita, kuka tekee korjaukset, miten testit toistetaan? Mikä on testausautomaation rooli testien toistossa. Tässä törmätään usein ongelmaan, että järjestelmän ylläpito-organisaatio ei osaa mitään näistä tehtävistä niin hyvin kuin projektiorganisaatio. Projektin jälkeen projektihenkilöt siirtyvät uusiin projekteihin.

	Pitkäaikainen tuotekehitys	Kertaluontoinen projekti
Projektiryhmä	Vakituinen organisaation osa	Koottu tätä tapausta varten esim. alihankkijoilta
Vaatimukset	Kehittyvät työn edetessä	Tulevat annettuina tilaajalta
Testaustasot	Tehdään itse, tyyppihyväksyntä viranomaisten toimesta	Tilaaja tekee yleensä itse hyväksymistestauksen
Oppimisaika	Mahdollistaa hyvän sisäänajon	Pari päivää ja sitten töihin
Investointi-mahdollisuudet	Automaatioon ja koulutukseen ollaan valmiita sijoittamaan	Mahdollisimman edullisesti

Kuva 1.

Erityisesti lyhyissä projekteissa ei testauksella ole enää virheidenkorjaajia. Tekninen tuki voi olla vaikeasti saatavissa. Käytännössä usein päädytään tilanteeseen, että ylläpito lainaa projektiorganisaatiolta kriittisiä osaamisia – valitettavan usein asiaa etukäteen sopimatta. Osalle henkilöistä jää vastuita vanhoista projekteista. Nämä vastuut jäävät häntäkuormaksi henkilöille, jotka uusissa projekteissa eivät siten voi olla enää niin tehokkaita.

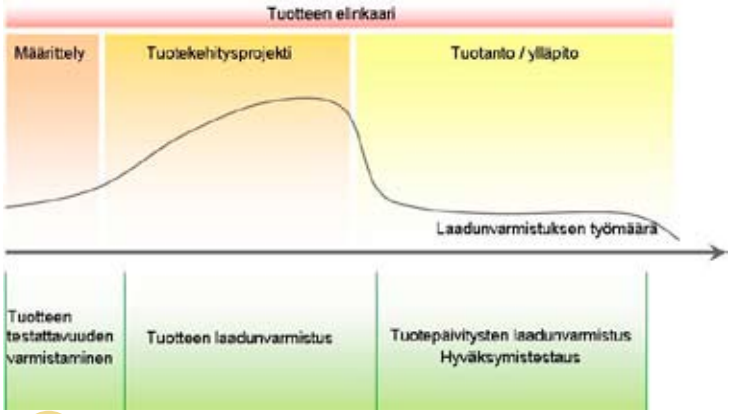
Prosessimallit

Maailmalla on kaikenlaista ohjelmistokehitys- ja testausmallia suunnitelmavetoisista hyvin ketteriin malleihin. Projektimuotoisessa toiminnassa usein korostuu vahva suunnitelmavetoisuus. Lisähaasteen tuo projektin jakaminen osaprojekteihin, joista osan toteuttavat talon ulkopuolisesta, tilaajasta riippumattomat alihankkijat. Tämä tosiasia itsessään pakottaa suurempaan suunnitelmallisuuteen, ainakin työnjaon ja kommunikaation suunnittelun suhteen.

Testauksen etenemän seurannan ja tekemisen kannalta tulisi testauksen tasot ja integrointimalli suunnitella ajoissa. Kun mukana on alihankkijoita, on entistä tärkeämpää suunnitella millaisissa osissa niin tuotekehitys kuin testauskin tehdään ja kenen toimesta. Oma kehitys on helppo jakaa yksikkö-, integrointi- ja järjestelmätestaukseen, mutta useilta yhtäaikaisilta alihankkijoilta tulevat ohjelmat vaativat lisäksi osajärjestelmien integrointia ja testausta. Mukaan tulee siis mahdollisesti 1-2 testauksen tasoa lisää.

Pelkistettynä projektiliiketoiminnan vaiheita kuvaa usein V-malli tai jotkut sen johdokset. Varsinkin vastuunjako testauksen eri tasojen välillä on varsin selkeä kommunikoida V-mallin johdannaisten välillä. Käytännön toiminta täytyy toki suunnitella inkrementaalisemmaksi, jolloin toimittajalta saadaan tilaajalle ja laajempiin testausvaiheisiin valmista tavaraa säännöllisesti eikä vasta kerralla. Tähän käytetään usein apuna laadunvarmistuksen tarkastuspisteitä, joka inkrementin on läpäistävä päästäkseen prosessissa eteenpäin.

Kuva 2. Testauksen vaiheet projektin aikana. Lähde: Enderon koulutusmateriaalit



Testauksen rooli kohoaa usein hyvin keskeiseksi, irti yksittäisistä testaustasoista, ohjaamaan tilaajan ja toimittajien välistä kommunikaatiota sekä työn että laadun etenemisen suhteen. Ohessa esimerkki testauksen keskeisestä asemasta, jonka tyyppisellä konseptilla olemme toimineet useilla asiakkailla vuosien ajan.

Suunnittelu

Testauksen suunnittelu itsessään ei paljon poikkea suunnittelusta muunkaan tyyppisissä projekteissa. Testauksen vaiheet ja tasot täytyy suunnitella sekä korkealla tasolla (resurssit, työmäärät, tehtävät, painotukset jne.) että tarkalla tasolla (testitapaukset, testiaineistot jne.). Merkittäväksi piirteeksi nousee kaikkein eri testaustasojen ja –vaiheiden välistä työnjakoa koskeva yhteinen suunnittelu. Tämä suunnitelma on usein nimeltään kokonaistestaussuunnitelma. Se on erityisen tärkeä suunnittelun vaihe silloin kuin toimittaja on erillinen organisaatio. Päällekkäistä testaustyötä kannattaa tehdä vain rajallinen määrä.

Tarkasta työnjaosta huolimatta päällekkäisyyksiäkin voi toki tulla juuri eri organisaatioiden mukanaolon vuoksi - kunkin osapuolen täytyy itse olla riittävän vakuuttunut projektin kokonaislaadusta. Eri osapuolten oma testaussuunnittelu korostuukin koko projektin kokonaistestaussuunnittelun lisäksi. Kun on paljon yhteentuotavia osatekijöitä, jotka vielä toimivat organisaatiorajojen sekä sopimusrajojen yli, täytyy kunkin osapuolen tehdä kattava testaus.

Ohjelmistoprojektin tilaajan testaus- ja laadunvarmistustoimet on suunniteltava sekä ohjaamaan ja tukemaan kehitysorganisaation – usein siis alihankkija – toimia,että tekemään tarvittavat hyväksymistestit kokonaislaadun varmistamiseksi. Lisähaasteena tilaajalla ei usein ole käytössään suurta määrää testauksen ammattilaisia, jolloin vähillä omilla ammattilaisilla ja konsulteilla täytyy saada oman toimen ohella testaavat henkilöt toimimaan tehokkaasti. Tähän tarpeeseen täytyy yleensä suunnitella lisää koulutusta ja tukitoimia.

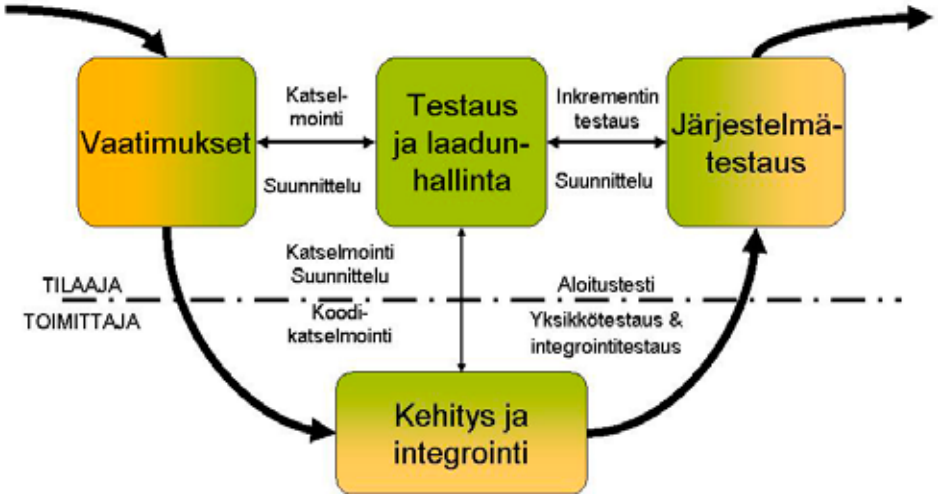
Aikataulut ja budjetit

Tuotekehitysorganisaatiolla on usein seuraava julkaisu, johon voidaan lykätä vähemmän kriittisten piirteiden toteutus ja testaus, kun aikataulu ja budjetti alkavat paukkua. Projektiliiketoiminnassa tätä luksusta ei ole. Kun projektin päätös koittaa, ei edessä ole kuin kevyt ylläpitovaihe, jonne ei ilman uuden projektin luomista voi mitään lykätä. Kaikki täytyy siis saada aikaan projektin aikana tai tyytyä ”lopullisesti” vähempään määrään toiminnallisuutta. Tätä ongelmaa on toki yritetty

ratkaista useilla prosessimalleilla, mutta valitettavan usein ylläpitovaiheen jälkeinen kehitys- ja testaustyö jää ”ilmaan”.

Aikataulupaineet toki vaihtelevat projektista toiseen, eivätkä välttämättä ole yhtä kovia kuin jotkin tuotekehityksen julkaisupäivät, mutta varsin usein projektilla on ulkoinen rajoite, joka pakottaa pitämään aikataulun. Moni yritys ja julkinen organisaatio pitää itse kovaa budjettikausikuria tai on riippuvainen esim. lainmuutoksista.

Moni projekti ei näistä syistä voi tinkiä projektin aikana julkaistavista järjestelmän tai ohjelmiston toiminnallisuuksista. Tämä aiheuttaa sekä ohjelmistokehitykselle että testaukselle entistä suuremmat paineet. Asia ratkeaa usein rankalla testauksen priorisoinnilla kriittisimpiin järjestelmän toiminnallisuuksiin, esim. riskipohjaisesti. Käytetään käytettävissä oleva aika niin hyvin kuin voidaan.



Kuva 3. Testauksen asema prosessissa. Lähde: Enderon koulutusmateriaalit.

Yhteenveto

Projektiliiketoiminnassa korostuvat tuoteliiketoimintaan verrattuna testauksen kokonaisvaltainen suunnittelu, suurempi haastellisuus järjestää projektia edeltävä ja seuraava laadunvarmistustoiminta, todennäköisen alihankkijan ohjaaminen testaustoiminnan avulla, oman testauksen jakautuminen testausammattilaisille ja oman toimen ohessa testaaville sekä aikataulujen tiukkuus ja priorisoinnin välttämättömyys. Useimmat asiat tulevat esiin kaikessa testauksessa, mutta projektiliiketoiminnassa nämä piirteet korostuvat ja niille täytyy huolehtia erikseen riittävä huomio.

Teksti: Tarja Raussi
Kuvat: Ilkka Pirttimaa

Tarja Raussi on systeemityön asiantuntijana Tieturissa ja toimii myös Sytykeen johtokunnassa.

Tuottavat ja laadukkaat prosessit

Sytykeen 9. laivaseminaari pidettiin 6. – 8.9.2006 Viking Mariellalla. Paikalla oli hyvän kokoinen joukko aktiivisia sytykeläisiä kuulemassa monipuolisia puheenvuoroja, keskustelemassa ja verkostoitumassa.

Tänä vuonna laivaseminaarin aihe keskittyi prosesseihin. Prosessi pidettiin tarkoituksella väljänä terminä, sillä meillähän on monenlaisia prosesseja: liiketoimintaprosesseja, systeemityöprosesseja, testausprosesseja jne. Näin ohjelmasta saatiin monipuolinen ja erityyppisissä tehtävissä toimivia henkilöitä kiinnostava.

Sytyke ry:n puheenjohtaja Helena Venäläinen avasi seminaarin ja esitteli millaisia asioita tarvitaan tuottaviin ja laadukkaisiin oppimisprosesseihin seminaarin aikana. Ilkka Pirttimaa Stockmann Oyj:stä toimi seminaarin puheenjohtajana ansiokkaasti ja teki esiintyjille hyviä kysymyksiä.

Määrittely kunniaan!

Sakari Lehtonen SysOpen Digiasta painotti, että Service Oriented Architecture (SOA) on 2/3 liiketoimintaa ja 1/3 tekniikkaa, ja että siihen lähteminen on aina liiketoimintastrateginen päätös. Huonoon lopputulokseen voidaan päätyä, jos lähdetään liikkeelle pelkästään tekniikan ehdoilla. SOA tulee selkeimmin esille B2B-verkostoliiketoiminnassa. SOA lisää jo tehtyjen IT-järjestelmäinvestointien elinkaarta jopa 25 vuotta.

SOA korostaa entisestään hyvien UML-määrittelyjen merkitystä. Jos hyvä määrittely unohdetaan, syntyy palvelujen kaaos. Pahimmassa tapauksessa luodaan palveluja, joita ei tarvita missään. UML-kaavioista tilakaavion merkitys on kasvanut paljon palvelujen suunnittelussa. Palvelujen elinkaaren hallinnassa auttavat mm. ITIL ja erilaiset repositoryt eli palvelujen säilytyspaikat hallintatyökaluineen.

Jyrki Lahnalahti Inspecta Sertifiointi Oy:stä pohti saataisiinko prosesseista apua asiakasvaatimuksien määrittelyyn ja hallintaan. Hän kertoi prosessien kypsyysarvioinnin CMMI:n eri tasojen vaatimuksista. Esim. Requirements Management CMM-2-tason saavuttamiseksi vaaditaan mm. kaksisuuntaisen vaatimusten jäljitettävyyden ylläpitoa (bi-directional traceability).

Jyrki kertoi myös kokemuksia kypsyystasojen arvioinneista. Suurimmat ongelmat ovat itse vaatimusmäärittelyssä: vaatimuksia ei analysoida mitenkään tai koko vaatimusmäärittelyä ei osata tehdä, ts. tekijöiltä puuttuu koulutus asiaan. Yllättävintä oli se, että vaatimusten hallintaa kyllä tehdään.

Prosesseista koodia!

IBM:n edustajat Esa Törölä ja Antti Lundsröm kertoivat, miten liiketoimintaprosesseista voidaan saada toimivaa ohjelmakoodia. Esa käytti alustuksessaan omien sanojensa mukaan ”aina toimivaa alustaa” eli fläppitaulua, mikä herätti positiivista hilpeyttä. Ensin prosessit pitää mallintaa ja analysoida (Model), sitten kootaan tarvittavat komponentit prosessin toteuttamiseksi (Assemble), sijoitellaan optimoidut prosessit toimintaympäristöön (Deploy) ja sen jälkeen hallitaan prosesseja (Manage). Tässä kaikessa voidaan käyttää esim. IBM:n WebSphere-välineitä apuna.

Antti demosi IBM:n välineillä sen mitä Esa lyhyesti esitti. Hän mallinsi ensin prosessin ja määritteli attribuutit prosessin askelille. Kun prosessi oli mallinnettu, liiketoimintasäännöt kuvattu ja mittarit asetettu, hän muunsi ne suoritettavaksi koodiksi (BPEL) toiselle välineelle. Näin syntynyttä koodia voidaan ajaa ja siten simuloida prosessin toimivuutta erilaisilla parametreilla.

Esitys oli mielenkiintoinen, mutta yleisökeskustelussa yhteisesti totesimme, että toistaiseksi IBM:n välineet eivät mahdollista suoritettavan koodin muuttamista lennosta prosessin muuttuessa. Jos prosessi muuttuu, siitä pitää generoida uusi versio ohjelmistoa. Mutta saa nähdä, mitä tulevaisuus tuo tullessaan!

Parhaiden käytäntöjen kokoelmat avuksi

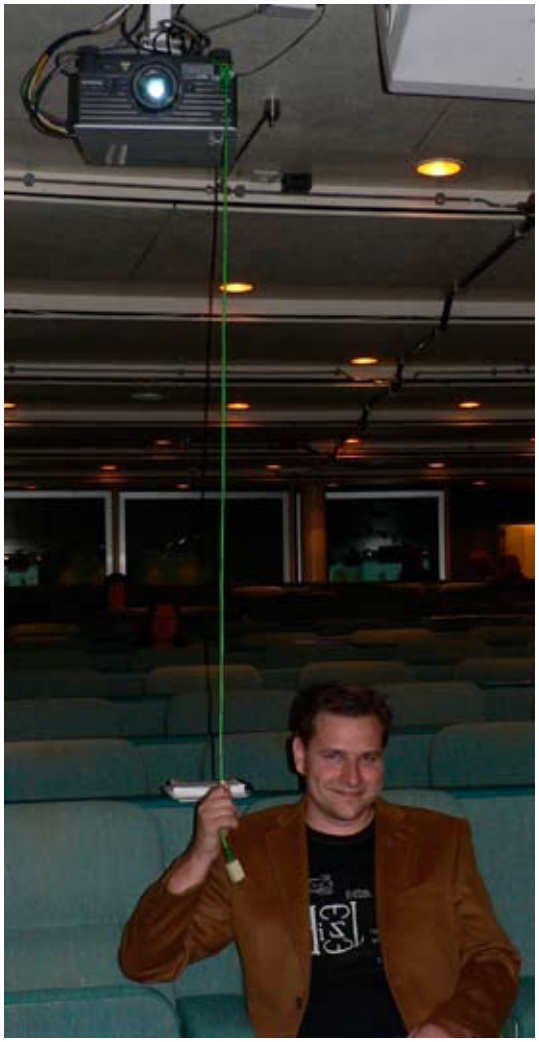
Taina Lintilä Tieturista kertoi parhaista käytännöistä laadukkaassa ohjelmistotuotannossa. Ohjelmistotuotannossa on siirrytty yhä enemmän kehittämisestä palvelujen elinkaaren hallintaan. Vaikka parhaat käytännöt olisivatkin käytössä, se ei riitä, sillä niitä pitää kehittää jatkuvasti. Itse ei tarvitse kuitenkaan keksiä kaikkea, vaan maailmalla on joukko parhaitten käytäntöjen kokoelmia.

ITIL (IT Infrastructure Library) tarjoaa yhteisen kielen liiketoiminnan ja IT:n välille. Sen yksi osa on ITIL Application Managment, joka tarkastelee sovellusten hallintaa yrityksen omaisuutena. Sovellusta käsitellään ”mustana laatikkona”, ts. pääasia on, että se toimii. ASL (Application Services Library) on viitekehys sovellushallinnan prosesseille. Se käsittelee sovellushallintaa huomattavasti monipuolisemmin kuin ITIL. Myös IT-ostamiseen on joukko parhaita käytäntöjä, mm. kansainvälinen ISPL ja Tietotekniikan liiton kehittämä 4V-malli.

Sakari Lehtonen, SysOpen Digia

Inspecta tekee mm. CMMI-kypsyysarviointeja.

Esa ja varmasti toimiva alusta.



Ilen tärinänvaimennin.

Tiedosta sudenkuopat!

Ensimmäisen päivän päätteeksi Risto Nevalainen STTF:stä kertoi vuosikymmenten kokemuksella, mitkä asiat tuovat nostetta tai toisaalta aiheuttavat notkahduksia projekteissa. Alustukseksi hän pohti, miksei hommaa ole jo opittu, vaikka näitä on tehty vuosikymmeniä. Yhtenä syykokonaisuutena lienee se, että ohjelmiston kehittämisestä on siirrytty systeemien rakentamiseen, jolloin vastaan tulevat mm. konfiguroinnin haasteet, hajautettujen järjestelmien laajuus ja palveluiden vuorovaikutuksen ajattelutapa.

Viime aikojen alaspainavia voimia ovat lisäksi olleet kvartaalitalous ja hajautettu kehittäminen. Pahimmassa tapauksessa silloin kukaan ei ajattele pitkällä tähtäimellä tai kokonaisuuksia. Toisaalta nostavana voimana hän näkee liiketoimintaprosessien hallinnan.

Sudenkuoppina voivat olla mm. jatkuvat muutokset ja uudet asiat, ja ketterät prosessit (mutta näissä on myös positiiviset puolensa).

Toimittajan kannalta asiakkaan osaaminen, jos asiakas on kypsyystasossa kovin eri tasolla (ylempänä tai alempana) kuin toimittaja.

Verkostoitumista ja kiinnostavia keskusteluja

Ensimmäisen päivän päätteeksi nautimme buffet-illallisen hyvän seuran kera. Täytyy sanoa, että Vikingin keittiö oli uusinut pöydän antimia monipuoliseen suuntaan, samoin ruokajuomat olivat varsin hyviä. Ainakin meidän pöydässä virisi monipuolinen ja antoisa keskustelu niin päivän aiheista kuin muistakin kiinnostavista aiheista. Lopulta tarjoilijat tulivat suorastaan huomauttamaan, että ruokailuaika oli jo päättynyt :-)

Tukholmassa torstaina kaikilla oli vapaata aikaa. Jotkut lähtivät kosteaan säähän kävelylle ja toiset ostoksille. Mutta laivalle kaikki palasivat ajoissa jatkamaan seminaaria klo 15.30. Laivan lähtiessä moottorien tärinä aiheutti videotykin tärinää, mutta sitten puheenjohtajamme Ile keksi, miten tykin saa stabiloitua. Hän saikin keksinnöstään aplodit.

Hyödyllisyys, käytettävyys ja vetovoima

Sami Wilkman Adage Oy:stä kertoi käyttäjäkeskeisestä suunnittelusta. Hän totesi, että hyödyllisyys, käytettävyys ja vetovoima ovat aina läsnä kaikissa laitteissa tai tuotteissa, mutta niiden painotus vaihtelee.

Käyttäjäkeskeisen tuotekehityksen keskeisiä asioita ovat:

- Käyttäjien aktiivinen osallistuminen
- Tarpeiden selkeä ymmärtäminen
- Toimintojen luonteva jako käyttäjän ja teknologian välille
- Iteratiiviset suunnittelumenetelmät
- Eri alojen asiantuntijoiden yhteistyö

Hän painotti, että on hukkaan heitettyä rahaa, jos halutaan vasta lopussa tehdä käytettävyystestaus, mutta ei ole aikaa tai halua korjata testauksessa havaittuja ongelmia.

Testausprosessi kuntoon!

Mitro Kivinen Qentinel Oy:stä kertoi, että testaukseen menee 30 – 70 % koko projektin työmäärästä. Testauksen kehittäminen tähtää laadukkaaseen ohjelmiston tuottamiseen nopeammin ja halvemmalla.

Hyvä testausprosessi
- Auttaa kohdistamaan kalliit testausresurssit oikein
- Tuottaa tietoa ohjelmiston laadullisesta tilasta
- Muuttaa testauksen laadukkaan tekemisen kulttuuriksi

Ei riitä, että pelkkää testausprosessia kehitetään. On kehitettävä myös muita työkokonaisuuksia, jotta laadukas ja tuottava kehitys olisi mahdollista.

Prosessit osana kokonaisarkkitehtuuria

Heini Holopainen ja Leena Valtonen Tieto-Enatorista kertoivat, miten prosessimallin avulla saadaan liiketoiminta vastaamaan paremmin asiakkaan tarpeeseen. Prosessijohtamiseen siirtyminen ja liiketoiminta-arkkitehtuurin kehittäminen edellyttää muutosjohtajuutta.

Liiketoiminta-arkkitehtuuri (Business Architecture) lähtee palvelu- ja taustaprosessien kehittämisestä asiakkaan tarpeen pohjalle läpi toimialojen. Uudet sähköiset kanavat avaavat uusia tapoja tarjota ja tuottaa palveluja. Toisaalta kytköksellisyys ja riippuvuus toisista lisääntyvät. Muutoksen läpivienti edellyttää kokonaisuuden hallitsemista. Prosessien hallinnalla onkin palveluja kokoava rooli.

Arviointi on osa prosessia

Helena Venäläinen ja Ilkka Pirttimaa vetivät yhteen seminaarin antia. Seminaarin mahdollistaneet sponsorit saivat kukin 2 minuutin puheenvuoron, joissa onnistuttiin välttämään liika mainostaminen.

Toisen päivän ilta huipentui a la carte –illalliseen, jonka aikana testattiin osallistujien muistia pöytäkunnittain leikkimielisen visailun avulla. IBM lahjoitti visailun palkinnot.

Osallistujat antoivat myös palautetta seminaarin järjestäjille. Pääosin kiiteltiin monipuolista ohjelmaa ja aikataulussa pysymistä. Moitteita sai lähinnä auditorion kylmyys – lämpöä ei saatu lisää, vaikka sitä henkilökunnalta pyydettiin. Onneksi seminaarin henki oli kuitenkin lämmin ja aktiivinen, mistä suuri kiitos runsaasti keskusteleville osallistujille!

Seminaarin esitykset ovat nähtävissä websivuillamme www.sytyke.org valikosta seminaarit ja tapahtumat/laivaseminaarit.

Laivaseminaaritoimikunta 2006

Pirkko Leivo, OPK
Ilkka Pirttimaa, Stockmann
Tarja Raussi, Tieturi
Kati Viik, Nokia

Seminaarin sponsorit

Adage Usability research expertise

IBM

Qentinel SOFTWARE QUALITY ASSURANCE

SYSOPENDIGIA

TietoEnator Building the Information Society

Tieturi

Kiitos seminaarin mahdollistamisesta!

Seminaarin päätösillallinen.

KUUTAMOLLA

Testauksen arki Suomessa

Tämä artikkeli sisältää voimakkaan asenteellista materiaalia ja saattaa aiheuttaa traumatisoivia painajaisia, yökastelua ja pitkäaikaista ahdistusta arkaluontoisille ihmisille, jotka eivät ole samaa mieltä kirjoittajan kanssa tai muuten vaan ovat alemmalla älyllisellä tasolla tai mahdollisesti yliherkkiä noin yleisesti. Se on myös saattanut jo aiheuttaa yllämainittuja oireita päätoimittajalle, jolloin artikkelin sisältö on punnittu liian de(kon)struktiiviseksi ja sen julkistaminen on kielletty ja tämä sivu on tyhjä. Siinä tapauksessa suosittelen, että ette jatka enää lukemista.

Näen aina punaista, kun ajattelen testauksen asemaa ja ihmisten yleistä suhtautumista siihen. Kaiken nähneenä koodauksesta aloittaneena pitkän linjan tietotekniikka-alan teknokraattina, joka on työskennellyt tavalla tai toisella bittimaailmassa jo tietokone-ajoista lähtien (maailmanhistoria tuntee jura- ja liitukauden jälkeen esimerkiksi tietokone-, atk-, it- ja ict-kaudet) en voi muuta kuin kummastella sitä väheksyvää asennetta, jota niin monessa tilanteessa on havaittavissa testausta kohtaan.

Viime vuosituhannella kun kirjoitettiin COBOL-koodia, niin eihän sitä testattu muuten kuin lukemalla koodia uudestaan ja uudestaan erittäin syvällä ajatuksella. Silloin ei oikein edes tiedetty testauksesta. Kun siitä alettiin jotain tietää, niin koodarikunnassa muodostui käsitys, että yksikkötestaus on itsetunnon puutetta. Sitten elettiin muutama uneton yö, kun ohjelmisto meni tuotantoon. Tuo ajattelutapa on monin paikoin vallalla edelleenkin. Unettomat yöt kyllä ovat koodareitten osalta suurimmalta osin taaksejäänyttä elämää, koska nykyäänhän koodari heittää puolivalmiin tekeleensä (unettomista öistä kärsivälle) testaajaparalle, joka yrittää saada todennettua edes jonkinlaista toimintaa testattavassa kohteessa. Mahdollisesti - siis mahdollisesti vika on siinä, että koodaria ei huvita tehdä edes alkeellista yksikkötestiä, koska testaus on alempiarvoista puuhaa. Ei vaikka nykyään testaajan työnhakuilmoitukset alkavat olla vaativuudeltaan jo sellaisia, että koodareiden esimiehet yskähtelevät lukiessaan testauskollegoidensa ilmoituksia: "Eihän tollasia superhessuja löydy mistään, nää vaatimuksethan on kovempia kun mitä me halutaan koodareilta..".
Niinpä. Testaajan työ nyt vaan on vaativampaa kuin koodaajan.

Vaikka turhaanpa minä tässä pelkästään koodaajia osoittelen syyhyttävällä sormikkaallani. Aivan yhtä syyllisiä ovat määrittelijät ja vielä enemmän yhtä syyllisiä ovat asiakkaan tai toimeksiantajan edustajat. Edelliset aloittavat hyvin aikaisessa vaiheessa eiku-iteroinnin ja oho-tää-unohtu-päivitykset, jälkimmäiset taas saattavat odotella siihen asti, että toteutusvaihe on aloitettu ja sitten käyttävät kaiken olemassa olevan energiansa yrittäessään keksiä erittäin tärkeitä ja strategisesti ykköksprioriteetilla painotettuja uusia ominaisuuksia, joita ilman koko järjestelmä onkin itse asiassa aivan turha. Mutta testaajat, nuo tuskassaan väkevät! Heidän äititeresamaisen lojaliteettinsa ja anttirokkamaisen tunnollisuutensa avulla on pelastunut moni projekti ja erityisesti projektipäällikkö, isommista pampuista nyt puhumattakaan.

Elokuvia testataan, kosmetiikkaa, kodin elektroniikkaa ja lääkkeitä testataan oikein urakalla, kuten myös talvirenkaita. Monta kertaa tosin testaaminen tarkoittaa vertailua muihin rinnakkaisiin tuotteisiin, mehän tämän(kin) alan asiantuntijoina tarkoitamme testauksella testattavan tuotteen omien ominaisuuksien verifiointia ja validointia. Avioliittoa voi testata avoliitossa, lyhytaikaisempia parisuhteita voi testata niin maksullisesti - kuulemma, isot pojat ovat kertoneet - kuin vähän vähemmilläkin kustannuksilla - tämänkin ovat isot pojat kertoneet. Varsinkin ihmisiä testataan. Syksyn aikana julkisuudessa ollut älykkäiden järjestö esimerkiksi harjoittaa pientä korvausta vastaan lyhytaikaista kuormitustestiä ihmisten aivokoneistoa kohtaan. Onkohan älykkyystesteissä koskaan tapahtunut kuormitustesteissä aika useinkin vastaantulevaa ongelmaa, että koko järjestelmä olisi kaatunut? Testattava henkilö olisi joko noussut talitinttiä matkien ylös pulpetista ja vienosti hypähdellen poistunut testitilaisuudesta tai sitten jäänyt sumusilmäisenä tuijottamaan vastapäistä seinää, kuolan pikku hiljaa valuessa suupielestä.

Mahtavatkohan Mensan testaajat tietää, miten aivojen resetointi oikein tapahtuukaan? Pakkohan niiden on, huippuälykkäitä kun ovat.

No, itse asiassa, tästä jutusta ei tainnut tullakaan ensimmäisen kappaleen mukaista. Ehkä sitä ensimmäistä kappaletta voisikin pitää määrittelydokumenttina, jota vastaan joku nyt sitten voisi toteutuksen jälkeen validoida tätä tekstiä ja havaita, että tekstistä puuttuu hyvin suuri määrä mainittuja ominaisuuksia. Tosin muutamat ominaisuudetkin ovat hyvin epämääräisesti ilmaistuja, mutta kun en nyt vaan yksinkertaisesti saanut ilmaistua itseäni implisiittisen eksaktisti tai edes repimättä suuria määriä jo entuudestaan hyvin harvoja hiuksiani.

Mutta toisaalta, sellaista sattuu niin kovin usein. Siis sitä, että lopputulos on ihan vaan pikkuisen jotain muuta kuin oli alkuperäinen tarkoitus. Olemmehan ihmisiä.

Kriittiset järjestelmät vaativat kriittistä testaamista

Tietokarhu on TietoEnatorin ja Suomen valtion yhteinen yhtiö, jonka 230 ammattilaisella on merkittävä rooli verohallinnon tietojärjestelmien kehittämisessä ja ylläpitämisessä. Verohallinto on yksi valtiohallinnon suurimmista ja moderneimmista tietotekniikan hyödyntäjistä ja se on ottanut ennakkoluulottomasti käyttöön uusia tekniikoita ja ratkaisuja.

Jos olet laadunvarmituksesta ja testaamisesta kiinnostunut kehittämishaluinen ja organisointikykyinen systeemityön ammattilainen ja suhtaudut työhösi **vakavissaan, mutta iloisesti,** Tietokarhu on sinulle harvinaisen antoisa ympäristö.

Järjestelmät joita operoimme ovat kriittisiä, suuria ja vaativia – ja testaamisen arvostus organisaatiossamme on sen mukainen.

Laadunvarmistuksestamme vastaa 40 hengen osasto, jossa yhteistyö ja kollegoiden tuki toimivat hyvin. Olemme aktiivisesti mukana TietoEnatorin testauksen kehittämisverkostossa ja ihmistemme ammatillista kasvua tukee kehittämämme “testaajan kasvupolku”.

Koska me panostamme poikkeuksellisen paljon laadunvarmistukseen ja osaamiseen sekä näiden kehittämiseen, voimme luvata sinulle paitsi poikkeuksellisen antoisia haasteita, myös kunnon eväät niissä onnistumiseen.

Tietoa meistä ja testauspäällikön työstä sekä hakemuksen (**ilmoitusnumero 4660**) löydät osoitteesta **www.tietoenator.fi/tietokarhu/rekrytointi**

Lisätietoja: Tomi Kaleva, p. 040 344 6721, tomi.kaleva@tietoenator.com

TietoEnator on yksi tehokkaan tietoyhteiskunnan pääarkkitehdeistä ja yksi Euroopan suurimmista tietotekniikan palveluyrityksistä. TietoEnator konsultoi, kehittää ja hoitaa asiakkaidensa liiketoiminnan ydinjärjestelmiä verkottuvassa maailmassa. Palvelut perustuvat syvään asiakastoimialojen ja viimeisimmän tietotekniikan osaamiseen. Yhtiöllä on 15 000 asiantuntijaa ja toimipaikat yli 25 maassa. **www.tietoenator.com**

TietoEnator

Building the Information Society

www.tieturi.fi/ohjelmistotuotanto

MIHIN TESTAUS PYSTYY?

Ohjelmistotestauksen arvo on kiistaton, mutta sen suoraa arvoa on vaikea laskea. Erilaiset arviointimenetelmät antavat yleensä vain kuvan testauksen ulkoisesta laadusta, siitä miten testauksen tulokset välittyvät suoraan loppukäyttäjälle. Usein unohdetaan kartoittaa miten tuotetta voi muokata myöhemmin ja voidaanko päivitys tehdä muilla resursseilla. Testausta ei arvosteta silloin kun pitäisi ja siltä odotetaan vääriä asioita.

Tule Tieturin testauskursseille päivittämään tietosi laadusta ja testauksesta sen osana. Mihin testaus pystyy ja milloin testaus ei enää maksa itseään takaisin?

Web-sovellusten testaus	12.2.2007
C++-testaus	27.-28.2.2007
Ohjelmiston käytettävyyden testaus ja arviointi	5.3.2007
Testauksen valmennusohjelma	29.-31.1.2007
ITIL Foundations	9.-11.1.2007
ASL (Application Services Library) Foundations	26.-28.2.2007
ISPL Foundations	24.-26.1.2007
Tietojärjestelmän hallittu hankintaprosessi	15.2.2007
Tietojärjestelmän käyttöönotto	23.1.2007
Tietojärjestelmäprojektin suunnittelu ja läpivienti	24.-26.1.2007
Prosessien mallintaminen	22.-23.1.2007
Nykyaikainen systeemityö	1.2.2007
Ketterä ohjelmistokehitys	23.-24.1.2007
Järjestelmävaatimusten määrittely ja hallinta	29.-31.1.2007
Sulautetun ohjelmiston määrittely ja suunnittelu	29.-30.1.2007
SOA ja Web Services -yleiskatsaus	2.2.2007
SOA-palveluiden määrittely ja suunnittelu	16.2.2007
Web-Sivuston suunnittelu - uudistumisprosessi	5.2.2007
Oliot ja UML määrittelijälle	5.-6.2.2007
Arkkitehtuurin suunnittelu (UML)	8.-9.2.2007
Käyttöliittymän suunnittelu	1.-2.3.2007
Järjestelmäintegroinnin valmennusohjelma	alk. 1.3.2007
Java-järjestelmän projektipäällikkövalmennus	29.-31.1.2007
Java- ja .NET-arkkitehtuurien vertailu	26.2.2007
.NET-järjestelmän suunnittelu	8.-9.2.2007
.NET-arkkitehtuuri	12.2.2007

Ilmoittautumiset ja lisätiedot: puh. (09) 4315 5333 | kurssit@tieturi.fi | www.tieturi.fi/syty