

Tutkiva testaus hyväksymistestauksen menetelmänä

Erkka Halme

Abstrakti—Asiakaskohtaisia järjestelmiä kehitettäessä järjestelmän laatuun liittyvät ongelmat ovat yleensä asiakkaan riski, ja siten pelkästään toimittajan tekemä testaus ei yleensä riitä asiakkaalle. Ketteriä ohjelmistokehitysmalleja käytettäessä perinteinen skenaarioperustainen tai mustalaatikko-perustainen hyväksymistestaus ei useinkaan ole toimiva vaihtoehto, vaan asiakkaan on varmistuttava järjestelmän laadusta jollakin toisella tavalla.

Tutkiva testaus on testaustapa, jossa testitapauksia ei suunnitella etukäteen, vaan testaaja luo uusia testejä aikaisempien testien tulosten perusteella. Väitetään, että tutkivalla testauksella voidaan virheitä löytää tehokkaammin kuin etukäteen suunniteltujen testitapausten avulla. Asiakkaan tekemä tutkiva testaus saattaa olla yksi mahdollisuus varmistua ohjelmiston laadusta ketteriä prosessimalleja käytettäessä.

Tässä tutkimuksessa esitellään tutkivan testauksen pääpiirteet ja selvitetään tutkivan testauksen soveltuvuutta hyväksymistestaukseen analysoimalla kahden eri opiskelijaryhmän sekä kahden todellisen asiakkaan suorittamaa sessioperustaista tutkivaa testausta. Tämän tutkimuksen perusteella tutkiva testaus vaatii erityisen kokeneen testaajan, ja on käyttökelpoinen hyväksymistestauksen menetelmä vain silloin, kun järjestelmän laatuvaatimukset eivät ole erityisen korkeat.

Hakusanat—testaus, tutkiva testaus, hyväksymistestaus, sessioperustainen testaus, ketterät prosessit

I. JOHDANTO

A. Tutkimuksen tausta

Ketterä ohjelmistokehitys edellyttää myös ketteriä testausmenetelmiä. Asiakaskohtaisen ohjelmiston ostajan on yleensä jollakin tavalla varmistuttava toimittajan tekemän ohjelmiston laadusta, sillä ohjelmiston mahdolliset laatuongelmat voivat aiheuttaa asiakkaalle hyvin suuria taloudellisia riskejä. Ohjelmistotuotteilla on neljä yhteistä tekijää, jotka vaikuttavat siihen, että ohjelmistoprojektin hallinta on vaikeaa ja siten myös laadun varmistus ja tarkkailu on hankalaa. Nämä neljä tekijää ovat (Brooks, 1987):

- *Näkymättömyys.* Fyysisten rakennusten ja laitteiden laatua ja toiminnallisuutta voidaan useimmiten havainnoida katsomalla, mutta ohjelmistojen ominaisuudet eivät yleensä ole suoraan nähtävissä.

- *Monimutkaisuus.* Käytettyä rahayksikköä kohden ohjelmistotuotteet ovat monimutkaisimpia ihmisen tekemiä rakennelmia.
- *Vaativuuden mukaisuus.* Ohjelmistojen pitää täyttää ihmisten ja organisaatioiden asettamat usein epä johdonmukaiset vaatimukset.
- *Joustavuus.* Ohjelmistojen vaaditaan muuttuvan ja sopeutuvan toimintaympäristön suuriinkin muutoksiin.

Ketterät prosessimallit antavat erittäin vähän konkreettista ohjeistusta siitä, kuinka hyväksymistestaus tulisi suorittaa (Abrahamsson, P., et al., 2002). Jos asiakkaalla ei ole käytössään ketteriin prosesseihin soveltuvia laadunvarmistusmenetelmiä, asiakas joutuu luottamaan toimittajaan perinteistä enemmän, tai toisaalta joutuu käyttämään testausmenetelmiä, jotka hidastavat ja kangistavat muuten ketterää kehitystä.

Hyväksymistestauksen epäonnistuminen siten, että tuotantoon päästetään virheellinen ohjelmisto saattaa monissa tapauksissa aiheuttaa asiakkaalle korvaamattomia tappioita. Hyväksymistestausta suunniteltaessa asiakkaan tulisikin aina miettiä, mitä riskejä ja seurauksia ohjelmiston virheillä voi olla. Tyypillisesti esimerkiksi tuotannonohjausjärjestelmän ongelmat voivat seisauttaa tehtaan tuotannon, tai kirjanpito-ohjelmiston virheet voivat pysäyttää yrityksen rahaliikenteen.

Perinteinen skenaarioperustainen hyväksymistestaus perustuu vaatimusmäärittelydokumenttiin ja laatukäsikirjaan (Leung, 1997), joita ketterissä prosesseissa ei tyypillisesti ole. Tämä asettaa testaukselle omat edellytyksensä. Vaikka ohjelmistokehityksessä ei käytettäisikään ketterää kehitysmallia, ovat puutteelliset ja ristiriitaiset vaatimusmäärittelyt silti yleinen ongelma.

Tutkiva testaus (James Bach 2003) on testaustapa, jossa testitapauksia ei suunnitella etukäteen, vaan testaaja ohjaa testin kulkua testin aikana tekemiensä havaintojen perusteella. Tässä tutkimuksessa selvitetään tutkivan testauksen soveltumista hyväksymistestaukseen, erityisesti käytettäessä ketteriä ohjelmistokehitysprosesseja.

B. Tutkimusongelmat

Tässä tutkimuksessa pyrin vastaamaan kysymykseen, min-käläisissä olosuhteissa asiakkaan tekemä raskas vaatimusmäärittelyihin nojaava hyväksymistestaus voidaan korvata tutkivalla testauksella. Aiheen tutkiminen on tärkeää, koska monet ohjelmistovalmistajat joko harkitsevat käyttävänsä tai ovat jo käyttäneet ketteriä menetelmiä ohjelmistokehityksessä (Abrahamsson, P., et al., 2002), jolloin vaatimusmäärittelydokumentteja ei välttämättä ole olemassa lainkaan. Lisäksi täydellisten ja kattavien vaatimusmäärittelyiden kirjoittaminen on osoittautunut käytännössä usein mahdottomaksi. (Kleinstejn 1988, Yu 1999) Tällöin myös asiakkaalla on oltava keinot varmistua

järjestelmän laadusta ketterän prosessin vaatimassa tahdissa. Testaus on kyettävä suorittamaan nopeasti ja riittävän kattavasti vaikka dokumentaatiota on tyyppillisesti varsin vähän. Ketterissä prosessimalleissa ohjelmistokehitystä tehdään inkrementaalisesti, ja asiakas on tiiviisti mukana kehitystyössä.

C. Tutkimuksen tavoitteet

Tutkimuksen tavoitteena on tutkia ketterissä menetelmissä tapahtuvaa testausta asiakkaan näkökulmasta. Tutkimus on rajattu koskemaan vain asiakaskohtaisia järjestelmiä, joissa toimittaja ja asiakas ovat eri organisaatioista. Tarkoitus on selvittää, missä olosuhteissa tutkiva testaus on riittävä menetelmä hyväksymistestaukseksi, ja mitä seikkoja tutkivassa testausta suoritettaessa tulee erityisesti huomioida. Lisäksi tutkitaan, mitä vaikutuksia testaajan teknisellä ja toisaalta sovellusalueen osaamisella on testauksen tuloksiin.

D. Tutkimuksen laajuus

Tämä tutkimus perustuu kirjallisuustutkimukseen sekä tapaustutkimukseen kahdesta opiskelijatyönä tehdystä ohjelmistosta. Opiskelijatyöt tilanneet asiakkaat testasivat ohjelmistoja käyttäen tutkivaa testausta. Lisäksi toiset opiskelijaryhmät testasivat tutkivaa testausta käyttäen ko. ohjelmistot.

E. Tutkimusmenetelmät

Tämä tutkimus sisältää kirjallisuusosan sekä käytännön osan. Teoriaosa perustuu ketteristä prosesseista ja testauksesta (erityisesti hyväksymistestauksesta sekä tutkivasta testauksesta) kertovaan tieteelliseen kirjallisuuteen. Tässä tutkimuksessa selvitetään ensin ketterien prosessien, hyväksymistestauksen sekä tutkivan testauksen tärkeimmät piirteet, rajoitteet ja vaatimukset. Tämän jälkeen pyrin selvittämään, kuinka hyväksymistestaukselle asetetut vaatimukset voidaan täyttää tutkivalla testauksella, kun ohjelmistoa kehitetään jonkin ketterän prosessimallin mukaan.

Tutkimuksen käytännön osa perustuu Teknillisen Korkeakoulun "Tietojenkäsittelyopin ohjelmatyö" -kurssin aikana tehtyjen ohjelmistojen testaamisen yhteydessä kerättyihin tietoihin. Kurssilla noin 7 hengen opiskelijaryhmät ovat tehneet yhden lukuvuoden aikana ohjelmistojärjestelmän jollekin asiakkaalle.

Tähän tutkimukseen otettiin mukaan kaksi projektia, joiden asiakkaat testasivat tuotetta kontrolloidusti yhden testisession ajan. Lisäksi tutkimuksessa on analysoitu testiloikeja, jotka ryhmät kirjoittivat vertaistestatessaan toistensa ohjelmistoja.

Ohjelmistoprojektit ovat olleet työmäärältään noin 1400 työtuntia. Projektin loppuvaiheessa ryhmät ovat testanneet toisten ryhmien tekemiä järjestelmiä käyttäen sessioperustaista tutkivaa testausta.

F. Raportin rakenne

Tämän raportin sisältö on järjestetty siten, että toisessa kappaleessa kerron, mitä eri ketterät prosessit opettavat asiakkaan tekemästä testauksesta ja erityisesti hyväksymistestauksesta. Kolmannessa kappaleessa kerron, mitä perinteinen skenaario-

perustainen hyväksymistestaus on, ja mitkä ovat hyväksymistestauksen tärkeimmät piirteet. Neljännessä kappaleessa kerron mitä tutkivalla testauksella ja erityisesti sessioperustaisella tutkivalla testauksella tarkoitetaan ja minkälaisia ominaisuuksia sillä on. Neljännessä kappaleessa tulen lisäksi pohtimaan, kuinka hyväksymistestaukselle kirjallisuudessa asetetut tavoitteet voidaan teoriassa täyttää tutkivalla testauksella. Viidennessä kappaleessa tulen esittelemään käytännön tutkimuksen tulokset. Viimeinen eli kuudes kappale sisältää yhteenvedon ja päätelmät tutkimuksen tuloksista.

II. LAADUNVARMISTUS KETTERISSÄ PROSESSEISSA

A. Ketterät menetelmät

Ketterän ohjelmistokehityksen voidaan katsoa alkaneen kun ryhmä ohjelmistoalan ammattilaisia julkaisi "Agile Software Development Manifesto":n (Beck 2001) vuonna 2001. Ketteriin menetelmiin kuuluu useita eri prosessimalleja, mutta tarkkoja rajoja joiden mukaan prosessi voidaan luokitella ketteräksi, ei ole (Boehm 2002). Yhteistä ketterille malleille on kuitenkin että ne ovat inkrementaalisia, asiakkaan ja kehittäjien yhteistyö on tiivistä, prosessi on suoraviivainen ja helppo oppia sekä prosessi on sopeutuvainen ympäristön muuttuviin vaatimuksiin. Näiden ominaisuuksien saavuttamiseksi prosessit eivät perustu raskaaseen dokumentointiin, vaan kommunikointi hoidetaan muilla tavoilla.

Agile Manifesto pitää sisällään neljä arvoväittämää:

- *Yksilöt ja yhteistyö* on arvokkaampaa kuin prosessit ja työkalut. Ketterissä menetelmissä korostetaan kehittäjien välisten suhteiden ja yhteistyön merkitystä, sekä pyritään luomaan mahdollisuudet tehokkaalle yhteistyölle erityisesti tiimitasolla.
- *Toimiva ohjelma* on arvokkaampaa kuin kattava dokumentaatio. Ohjelmasta pyritään julkaisemaan mahdollisimman usein toimivia ja testattuja versioita, ja koodi pyritään pitämään niin yksinkertaisena, että raskasta dokumentaatiota ei tarvita.
- *Yhteistyö asiakkaan kanssa* on arvokkaampaa kuin sopimuksista neuvottelu. Varsinaisen vaatimusmäärittelyvaiheen sijaan tärkeää on toimivan suhteen rakentaminen asiakkaan ja toimittajan välille.
- *Muutoksiin reagoiminen* on tärkeämpää kuin suunnitelmassa pysyminen. Sekä asiakkaalla että toimittajalla tulee olla tarvittavat tiedot, kyvyt ja lupa tarvittavien ohjaustoimenpiteiden tekemiseksi projektin aikana.

B. Mitä ketterät menetelmät kertovat testauksesta

Ketteriä menetelmiä vertailevan tutkimuksen (Abrahamsen, P., et al., 2002) mukaan yksikään ketterä prosessi ei tarjoa konkreettisia ohjeita hyväksymistestauksen suorittamiseen. Konkreettisia ohjeita integraatiotestauksen suorittamiseksi

tarjoavat Scrum, Pragmatic Programming ja Extreme Programming.

XP:ssä ohjelmistokehitys on testiperustaista, ja automatisoiduilla yksikkötesteillä on merkittävä rooli. Asiakas kirjoittaa toiminnalliset testit itse.

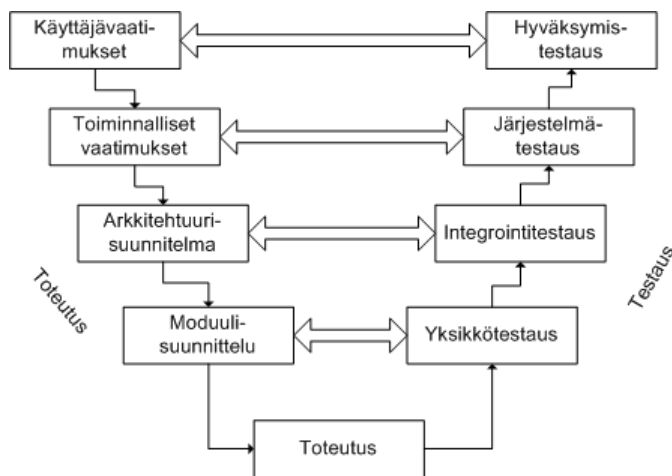
Myös Pragmatic Programming korostaa automatisoitujen yksikkötestien merkitystä.

XP:ssä testaus eroaa perinteisistä tavoista siinä, että koko kehitystyötä tekevä tiimi on vastuussa testauksesta ja järjestelmän laadusta. Hyväksymistestit määrittelevät laadun tason, jota asiakas haluaa, ja josta asiakas on valmis maksamaan. (Crispin 2003)

Ketterissä menetelmissä ja erityisesti XP:ssä on suuri paino automatisoiduilla yksikkötesteillä. (Crispin 2003) mukaan on erittäin hyödyllistä, jos myös hyväksymistestit voidaan automatisoida.

C. Mitä ketterien menetelmien käyttö edellyttää testaukselta

Ehkäpä tunnetuin perinteinen testausmalli on ns. V-malli. V-mallissa kutakin vesiputouksmallin vaihetta vastaa yksi testausvaihe, kuitenkin siten, että testaaminen aloitetaan vasta kun ohjelmisto on valmis. V-mallissa hyväksymistestaus perustuu ohjelmistoprosessin alussa tehtyyn vaatimusmäärittelydokumenttiin. (Hughes & Cotterell 2002)



Kuva 1 Testauksen V-malli

Ketterässä ohjelmistokehityksessä kattavaa vaatimusmäärittelydokumenttia ei välttämättä ole, joten hyväksymistestien tulee perustua johonkin muuhun.

Ketterissä prosesseissa kehitys on inkrementaalista, ja jokaisen vaiheen jälkeen asiakas voi varmistua tehdyn ohjelman osan toimivuudesta. Vaiheiden välissä suoritettava testaus pitää pystyä suorittamaan riittävän nopeasti, jotta prosessi ei kangistuisi hitaisiin testausjaksoihin vaiheiden välissä. Voidaan ajatella, että ketterässä kehityksessä hyväksymistestien ei tarvitse aina olla yhtä kattavia kuin vaikkapa vesiputouksmallissa.

Agile Manifeston määrittämät arvot pätevät myös testauksessa:

- Jokaisella on vastuu järjestelmän laadusta

- Ajettavat testitapaukset ovat tärkeämpiä kuin testidokumentit
- Testit suunnitellaan ja toteutetaan yhdessä asiakkaan kanssa
- Testien on mukauduttava helposti ohjelmiston muutoksiin.

Koska vastuu järjestelmän laadusta on jokaisella kehittäjällä, voidaan ajatella, että se pienentää erillisen hyväksymistestauksen merkitystä. Ketterä kehitys ei näe arvoa testisuunnitelmissa, vaan ainoastaan ajettavissa testitapauksissa tai muissa konkreettisissa testeissä. Asiakkaan on oltava aktiivisesti mukana hyväksymistestien suunnittelussa ja toteuttamisessa, koska vain asiakas voi tietää ja päättää minkälainen laatu on riittävää. Hyväksymistestien on oltava sellaisia, että ne voidaan mahdollisimman pienellä työmäärällä pitää ajan tasalla ohjelmiston ja vaatimusten muuttuessa.

III. MITÄ ON HYVÄKSYMISTESTAUS

A. Hyväksymistestauksen tavoitteet

Ohjelmiston laatu muodostuu ISO 9126-standardin (Hughes & Cotterell 2002) mukaan kuudesta tekijästä:

- Toiminnallisuus (functionality)
- Luotettavuus (reliability)
- Käytettävyys (usability)
- Tehokkuus (efficiency)
- Ylläpidettävyys (maintainability)
- Siirrettävyys (portability)

Jotta ohjelmisto voidaan ottaa käyttöön, tulee sen tyypillisesti täyttää jotkin asiakkaan määrittelemät kriteerit kunkin laadun tekijän osalta.

Hyväksymistestaus on ohjelmiston testaamisen viimeinen vaihe. Hyväksymistestauksen tavoitteena on evaluoida järjestelmän valmius tuotantokäyttöön (Leung 1997). Kun järjestelmä täyttää sille asetetut hyväksymiskriteerit, se voidaan ottaa käyttöön.

Perinteisen määrittelyn mukaan hyväksymistestauksessa pyritään todistamaan, että järjestelmä täyttää sille asetetut vaatimukset, jotka on kirjattu vaatimusmäärittelydokumenttiin. Käytännössä testauksesta on usein kuitenkin selvittävä puutteellisten ja ristiriitaisten vaatimusten perusteella. (Kleinstei 1988).

Puutteellisten vaatimusmäärittelyjen aiheuttamista ongelmista hyväksymistestauksessa on kirjallisuudessa paljon mainintoja (Yu 1999, Kleinstei 1988)

Hyväksymistestauksessa tarkastetaan, että järjestelmä täyttää sille asetetut liiketoimintavaatimukset. On tärkeää, että testaaja tuntee sovellusalueen (liiketoiminta) hyvin, jotta voidaan tietää miten sovelluksen tulee reagoida erilasiin syötteisiin. (Fournier 1998). Tyypillisesti testaajalla on parempi sovellusalueosaaminen kuin ohjelman kehittäjillä.

Hyväksymistestauksella on tärkeä rooli varmistettaessa, että on toteutettu oikea järjestelmä. Ei ole mitään järkeä etsiä ja korjata virheitä järjestelmästä, joka ei kuitenkaan ole hyödyllinen käyttäjälle.

Kirjallisuudessa on eitetty mm. seuraavia vaatimuksia ja ominaisuuksia hyväksymistestaukselle:

- Asiakkaan tulee olla tekemässä hyväksymistestausta
- Asiakas voi tehdä mitä testejä hän haluaa, yleensä kuitenkin liiketoimintaprosesseihin liittyen.
- Lähestymistapa on usein yhdistelmä skriptattua ja skriptaamatonta testausta.
- Testaus tulee suorittaa todellisessa käyttöympäristössä.
- Hyväksymistestit määrittelevät käyttäjävaatimukset suoraan verifioitavassa muodossa ja mittaavat kuinka hyvin nämä vaatimukset toteutuvat
- Hyväksymistestit paljastavat virheitä, joita yksikötesteissä ei havaita
- Hyväksymistestit tarjoavat valmiin määrittelyn ohjelman valmiusasteesta

Artikkelin (Miller 2001) mukaan kattavien ja täydellisten vaatimusmäärittelyiden tekeminen on usein mahdotonta, eikä täydellistenkään vaatimusten toteutumista voida suoraan verifioida. Artikkelissa ehdotetaan, että vaatimuksia ei kirjata erikseen vaatimusmäärittelyiksi, vaan suoraan hyväksymistesteiksi, jolloin asiakkaan ja toteuttajien väliset kommunikatiivirheet vähenevät. Hyväksymistestien olemassaolo parantaa jossakin määrin ohjelmiston laatua jo ennen testien suorittamista, sillä kehittäjä saattaa käyttää testejä ja niiden määrittelyjä apunaan jo ennen kuin luovuttavat koodin testattavaksi.

B. Mustalaatikko -perustainen hyväksymistestaus

Mustalaatikko-perustainen lähestymistapa on teollisuudessa yleinen hyväksymistestauksen menetelmä (Leung 1997). Tässä menetelmässä kehittäjä ja testaaja luovat testitapaukset usein yhdessä ja myös testit saatetaan ajaa yhdessä. Testitapauksen lähtökohtana ovat järjestelmän funktionaaliset tai ulkoiset vaatimukset. Mustalaatikko-perustaisessa testauksessa testaaja ei huomioi lainkaan järjestelmän sisäistä toteutusta, vaan testaaminen perustuu ohjelman käyttäytymisen seurantaan tietyillä syötteillä.

IV. MITÄ ON TUTKIVA TESTAUS

A. Tutkivan testauksen esittely

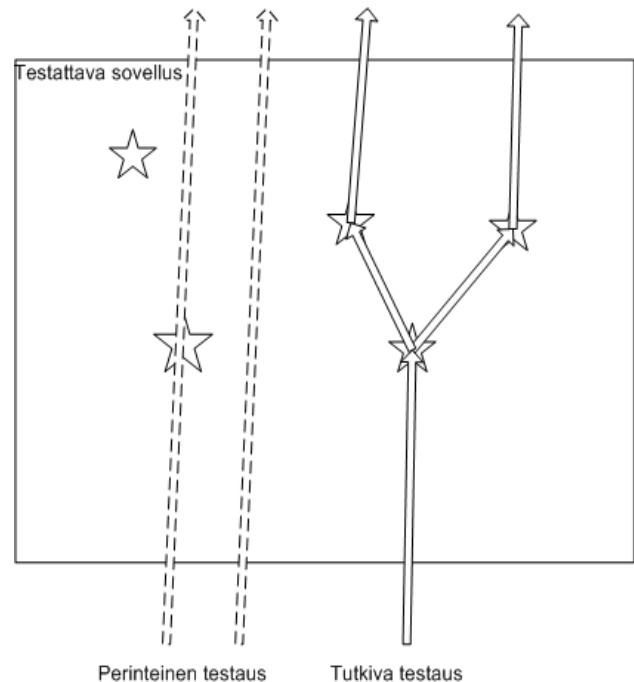
Termiä tutkiva testaus (Exploratory Testing) ryhdyttiin käyttämään 1990-luvun alussa rinnakkain ad hoc -testauksen kanssa.

Suosituin määritelmä tutkivalle testaukselle on Bachin "Exploratory testing is simultaneous learning, test design, and test execution". (Bach James 2003) Tutkivassa testauksessa testaaja ohjaa toimintaansa suorittamiensa testien perusteella. Testitapauksia ei siis suunnitella etukäteen, vaan testaaja pyrkii kokemuksensa ja jo suoritettujen testien perusteella löytämään uusia virheitä.

Lähes jokainen testaaja tekee ainakin toisinaan tutkivaa testausta esimerkiksi selvittäessään tarkemmin jonkin testitapauksen perusteella löytynyttä bugia. Tutkiva testaus yhdistetään joskus sattumanvaraiseen ja kurittomaan testaukseen ja haluan välttää dokumentointia. Tutkiva testaus on lähestymistapa, jossa testaaja voi käyttää useita erilaisia tekniikoita. (Tinkham 2003)

Kaner (2002) luettelee kahdeksan erilaista tapaa tehdä tutkivaa testausta:

- viimeisien löydettyjen bugien ja tehtyjen muutosten tutkiminen
- mallien (arkkitehtuurikaaviot, tilakaaviot jne.) tutkiminen
- esimerkkien läpikäyminen (käyttötapaukset, skenaariot)
- sellaisten asioiden muuttaminen, joiden ei pitäisi vaikuttaa testattavaan toimintoon
- ohjelman suorituspolun häirintä erilaisilla menetelmillä
- virhetilanteiden käsittelyn tutkiminen
- ryhmätyöskentely
- määrittelydokumenttien tarkastelu



Kuva 2 Tutkiva testaus verrattuna perinteiseen testaukseen

Testauksen aikana tietoisesti tapahtuvasta oppimisesta ja tiedon karttumisesta on kirjoitettu mielenkiintoinen artikkeli (Yu 1999). Artikkelissa tiedon karttuminen jaetaan vertikaaliseen ja horisontaaliseen karttumiseen. Vertikaalisessa tiedon karttumisessa on kyse pitkälti samasta ilmiöstä kuin Bachin tutkivassa testauksessa, eli yksittäisen testaajan tai ryhmän tiedot testattavasta kohteesta karttuvat ajan kuluessa. Horisontaalisessa tiedon karttumisessa tietoa levitetään aktiivisesti testaajien ja organisaatioiden välillä, jolloin karttuneen tiedon mää-

rää ei ratkaise kulunut aika vaan mukana olleiden henkilöiden lukumäärä. Vaikka Bach ei asiaa artikkeleissaan (Bach James 2003) asiaa erityisesti korosta, voidaan olettaa että tietoisesti tehtävällä horisontaalisella tietojen kartuttamisella voitaisiin huomattavasti tehostaa tutkivaa testausta tekevien organisaatioiden toimintaa.

Bach luettelee asioita, jotka vaikuttavat tutkivan testauksen suorittamiseen:

- testausprojektin missio
- testisession missio
- testaajan rooli projektissa
- testaajan taidot, kokemus ja mielipiteet
- käytettävät työkalut
- käytettävissä oleva aika
- käytettävissä oleva testidata
- käytettävissä oleva muiden ihmisten tuki
- testattava tuote
- testaajan aikaisemmat tiedot testattavasta tuotteesta
- mitä testaaaja haluaa tietää tuotteesta

Tutkivassa testauksessa testaaja pyrkii jatkuvasti miettimään, mikä on kullakin hetkellä paras testi seuraavaksi suoritettavaksi. Testaajan taidot ja kyvyt ovatkin tärkein yksittäinen testauksen onnistumiseen vaikuttava tekijä. Haasteellisuutensa ja vapautensa vuoksi tutkiva testaus voi olla testaajalle erittäin mielenkiintoista ja motivoivaa, varsinkin jos testaaja kokee löytämänsä virheet onnistumisina, eikä takaiskuina.

Tutkivassa testauksessa ei ole oleellista, että testit voidaan toistaa samanlaisina, vaan testien muuttuminen nimenomaisesti parantaa virheiden löytämismahdollisuuksia. Skriptattujen testien suuri ongelma on juuri se, että jos testiä ensimmäistä kertaa ajettaessa ei löydetä virheitä, ei virheitä todennäköisesti löydetä samaa testiä myöhemmin toistettaessa. Löydetyt virheet on toki tutkivassa testauksessakin kirjattava ylös riittävän tarkasti, jotta ne voidaan helposti toistaa. Testitapauksiin perustuvassa testauksessa kirjataan siis ylös nekin testit jotka eivät löydä virheitä, kun tutkivassa testauksessa tyypillisesti kirjataan ylös vain virheisiin johtaneet tilanteet.

B. Mitä on sessioperustainen tutkiva testaus

Jonathan Bachin(2000) mukaan testaamisen edistymisen seuranta saattaa olla tutkivassa testauksessa olla erittäin hankalaa. Tämän ongelman korjaamiseksi Jonathan ja James Bach ovat kehittäneet sessioperustaisen testauksenhallinnan menetelmän. Sessioperustaisessa testauksessa testausta tehdään noin 90 minuutin mittaisissa sessioissa. Sessio on keskeytymätön, sillä on selvä missio ja siitä tehdään raportti, jonka avulla kolmas osapuoli voi seurata testaamisen edistymistä. Jokainen testisessio perustuu toimeksiantoon (test charter), jossa määritellään session missio, eli mitä testataan ja minkälaisia ongelmia pyritään löytämään. Testisessiosta tehtävä raportti sisältää seuraavat tiedot:

- Session toimeksianto (missio ja testattavat kohteet)
- Testaajan nimi
- Päivä ja aloitusaika

- Tehtäviin kulunut aika(testien suunnittelu ja suoritus, bugien tutkiminen ja raportointi, muut järjestelyt)
- Käytetty testidata (tiedostot)
- Muut huomiot
- Testaamiseen tai koko projektiin liittyvät huomiot
- Bugit

C. Tutkivan testauksen soveltuvuus hyväksymistestaukseen (teoriassa)

Tutkivaa testausta käytettäessä ei testitapauksia suunnitella etukäteen, jolloin testejä ei voida käyttää ohjelman vaatimusten määrittelyyn. Tutkivan testauksen perusteella ei voida myöskään antaa kvantitatiivisia arvioida ohjelman valmiusasteesta. Hyväksymiskriteerien määrittely on myös hankalampaa kuin etukäteen suunniteltuja testitapauksia käytettäessä.

Tutkivaa testausta käytettäessä eivät puutteelliset vaatimusmäärittelyt aiheuta ongelmia. Koska testitapauksia ei tarvitse suunnitella eikä dokumentoida etukäteen, päästään tutkivassa testauksessa nopeammin tekemään varsinaista testausta, mikä saattaa nopeatahtisessa kehityksessä olla merkittävä etu.

Tutkiva testaus ja Agile Manifeston arvot sopivat hyvin yhteen:

- *Yksilöt ja yhteistyö* on arvokkaampaa kuin prosessit ja työkalut. Tutkiva testaus perustuu testaajan kykyyn tehdä havainnoja testattavasta järjestelmästä ja ohjata testausta havaintojen perusteella.
- *Toimiva ohjelma* on arvokkaampaa kuin kattava dokumentaatio. Tutkiva testaus ei vaadi eikä tuota paljoa dokumentaatiota, vaan testausta päästään suorittamaan nopeasti.
- *Yhteistyö asiakkaan kanssa* on arvokkaampaa kuin sopimuksista neuvottelu. Tutkivan testauksen käyttö hyväksymistestauksessa ei edellytä tarkkojen vaatimusmäärittelyjen ja sopimuksen tekemistä, vaan testit perustuvat asiakkaan käsitykseen omista tarpeistaan.
- *Muutoksiin reagoiminen* on tärkeämpää kuin suunnitelmassa pysyminen. Koska testausta ei suunnitella etukäteen, voidaan muutoksia tehdä nopeasti ohjelman tai vaatimusten muuttuessa sekä testatesa tehtyjen havaintojen perusteella.

Tutkivan testauksen tavoitteena on löytää ja kirjata ylös järjestelmän bugeja, kun hyväksymistestauksen tavoitteena on yleensä saada aikaiseksi dokumentti, joka osoittaa että järjestelmä täyttää sille asetetut vaatimukset. Tämä lähestymistapojen erilaisuus on selvä ongelma käytettäessä tutkivaa testausta hyväksymistestausmenetelmänä. Perinteinen hyväksymistesti kertoo suoraan, että määritellyillä testitapauksilla järjestelmä toimii oikein, kun tutkiva testaus kertoo ainoastaan, että tietty testaaja ei löytänyt virheitä järjestelmästä.

V. TUTKIMUKSEN TULOKSET

A. Tutkimusmateriaalin esittely,

Tutkimus perustuu kirjallisuustutkimukseen sekä kahdelle eri opiskelijaryhmien tekemille tuotteille tehtyihin testisessioihin. Opiskelijaryhmät muodostuivat 7:stä opiskelijasta loppuvaiheessa olevasta tietotekniikan opiskelijasta. Kummankin ryhmän testasi joku toinen opiskelijaryhmä käyttäen sessiope-rustaista tutkivaa testausta. Kukin ryhmä teki kolme eri testicharteriin perustuvaa testaussestiota. Yksi chartereista oli kaikilla ryhmillä sama, ja tässä tutkimuksessa analysoidaan ainoastaan tämän yhden charterin perustuneita sessioita. Kukin ryhmä kirjoitti jokaisesta testisessiosta testilokin.

Opiskelijaryhmien lisäksi ohjelmistot tilanneet asiakkaat testasivat tuotteet sessiope-rustaisella tutkivalla testauksella.

Testeistä tehtiin kappaleen IV B -mukainen testiloki.

B. Tutkimuksen rajoitusten selvitys

Tutkimuksessa analysoitujen testisessioiden testaajat ovat opintojensa loppupuolella olevia opiskelijoita, mutta eivät suinkaan välttämättä testauksen ammattilaisia. Opiskelijat testasivat toisten ryhmien tekemiä töitä, jolloin motivaatio sovellukseen/ sovellusalueeseen tutustumiselle ei välttämättä ole ollut kovin korkea.

Tutkiva testaus ei oletettavasti ollut tuttu monellekaan testaajista. Tutkivasta testauksesta oli annettu lyhyehköt kirjalliset ohjeet, sekä linkit kolmeen aihetta käsittelevään artikkeliin. Testilokeista ja -chartereista käy ilmi, että kaikki ryhmät eivät olleet ymmärtäneet tutkivan testauksen ideaa oikein. Kahta asiakkaan tekemää testisestiota lukuun ottamatta sessioita ei kontrolloitu, joten testilokien tietojen oikeellisuus perustuu vain testaajien omaan rehellisyyteen. Testien tuloksia ei kuitenkaan käytetty "Tietojenkäsittelyopin ohjelmatyö" -kurssilla arvioluperusteena, joten oletettavasti ryhmillä ei ole ollut motivaatiota tulosten tietoiseen vääristämiseen.

C. Tapaustutkimuksen kohteiden esittely

Kahden opiskelijaryhmän tekemien ohjelmistojen testaamisesta tehtiin tarkempi tapaustutkimus. Näissä ryhmissä myös järjestelmän tilannut asiakas testasi järjestelmän käyttäen samaa test charteria kuin opiskelijaryhmät. Näiden tulosten perusteella arvioidaan testaajan taustan vaikutusta testitulokseen. Oletuksena oli, että tekninen osaaminen (opiskelijatestaajilla) edesauttaa "teknisten" virheiden löytämistä, kun taas asiakastestaaja löytänee sovellusalueosaamisensa avulla enemmän ohjelman toiminnallisuuteen liittyviä virheitä sekä käyttöliittymä/käytettävyydevirheitä.

Ensimmäisessä tapauksessa testattava sovellus on Javalla toteutettu graafinen roadmapping-sovellus. Sovellus on erittäin käyttöliittymäintensiivinen, eikä sisällä erityisen mutkasta toiminnallisuutta. Käytännössä kyseessä on piirto-ohjelma, jolla voidaan piirtää muutamia erityyppisiä objekteja. Ohjelmistoprojekti oli luonteeltaan tutkiva, jossa tarkoituksena oli lähinnä tutkia, minkälainen roadmapping-sovellus saattaisi olla käyttökelpoinen. Sovellus ei ole sellaisenaan tarkoitettu

tuotantoon, mutta mahdollisesti käytettäväksi esittelytarkoituksissa asiakkaalle.

Toisen ryhmän tekemä sovellus oli Javalla ja JSP:llä toteutettu verkkokauppasovellus. Sovellus sisältää asiakkaille näkyvien toimintojen lisäksi liittymän, jonka avulla myytyjen tuotteiden toimituksia ja maksuja seurataan. Verkkokauppa-projektin tavoitteena oli saada järjestelmä tuotantoon asti, mutta tämän tutkimuksen aikana ei käyttöönotto vielä varmistunut.

D. Opiskelijaryhmien tekemien testilokien esittely

Roadmapping- työkalusta löytyi vertaistestauksessa 3 toiminnallista bugia ja 4 muuta havaintoa. Testisessio kesti 65 minuuttia.

Verkkokaupan vertaistestauksessa oli jonkin verran teknisiä ongelmia, jolloin kaikki ohjelman toiminnot eivät olleet testattavissa. Testisession aikana löytyi 2 toiminnallista bugia. Testisessio kesti 75 minuuttia.

E. Asiakkaiden suorittamien testisessioiden kuvaus

Molemmissa asiakkaiden tekemissä testisessioissa tutkimuksen tekijä esitteli testaajille lyhyesti sessiope-rustaisen tutkivan testauksen periaatteen, testilokin sekä testicharterin. Testauksen aikana testaajat täyttivät testilokin kohtia "Bugit", "testaamiseen liittyvät huomiot" ja "muut huomiot" erilliseen tekstitiedostoon. Tutkimuksen tekijä seurasi testisessioita ja kirjasi ylös testin kulkua ja ajankäyttöä.

Roadmapping-työkalun testaajana oli järjestelmän tilannut asiakas. Hänellä oli usean vuoden kokemus ohjelmistoprojek-teista, mutta testaamisesta hänellä ei ollut varsinaisesti kokemusta. Tutkiva testaus oli hänelle uusi käsite. Sovellusalue oli hänelle hyvin tuttu.

Testaajan lähestymistapana oli tehdä "ystävällismielistä" testausta, eli hän halusi selvittää, voiko ohjelmalla tehdä suunnitellut asiat, mutta hän ei erityisesti etsinyt virheitä.

Testaaja oli käyttänyt/testannut ohjelman aikaisempia versioita, eli hän pystyi ilman käyttöohjetta käyttämään ohjelmaa suunniteltuun tarkoitukseen. Hän aloitti testaamisen ja ryhtyi piirtämään ohjelmalla oikeaa dataan perustuvaa roadmappia. Session aikana testaaja löysi 5 bugia ja 6 muuta havaintoa. Havaintojen kirjaamiseen meni yhtä tiedoston tallentamiseen liittyvää vakavaa bugia lukuun ottamatta noin 30 sekuntia / havainto. Tiedoston tallentamiseen liittyvää vakavaa bugia testaaja selvitti n. 10 minuuttia löytämättä sille varsinaisesti syytä. Testisessio kesti kokonaisuudessaan 50 minuuttia, jossa ajassa testaaja sai omasta mielestään testattua sovelluksen kattavasti, eikä testaaja keksinyt enää enempää testattavaa. Löydettyistä bugeissa ei ollut yhtään vertaistestausryhmän löytämää bugia.

Testaaja ei ohjannut testin kulkua löytyneiden bugien perusteella, vaan toteutti aloittamaansa ohjelman käyttöskenaariota käytännössä koko testisession ajan.

Verkkokaupan testaajalla oli samantyyppinen lähestymistapa kuin roadmapping-työkalun testaajalla, eli hän ei varsinaisesti etsinyt virheitä, vaan pyrki ainoastaan käyttämään järjes-

telmää, kuten oletettu loppukäyttäjänkin sitä käyttäisi. Verkkokauppa-sovelluksen testaaja ehti testata noin puolet järjestelmän toiminnallisuudesta 45 minuutin session aikana.

Verkkokaupan testaajalla oli runsaasti kokemusta käytettävyyden arvioinnista, ja hän tekikin paljon nimenomaan käytettävyyteen liittyviä havaintoja. Erityisesti verkkokaupan testaamisesta voidaan havaita, että tutkivaa testausta käytettäessä keskitytään helposti vain käyttöliittymää lähellä oleviin asioihin toimintalogiikan jäädessä testaamatta.

Testaaja käytti havaintojen kirjaamiseen noin puolet koko sessioon kuluneesta ajasta, ja hän piti hyödyllisenä kirjata myös havaintoja, joissa ei välttämättä löydetty virheitä, vaan todettiin jonkin tietyn ominaisuuden toimivan oikein. Testaaja löysi 2 toiminnallista virhettä ja noin 10 käytettävyysongelmaa. Toinen asiakkaan löytämistä toiminnallisista bugeista oli sellainen, jonka myös vertaistestausryhmä oli löytänyt.

F. Testit suorittaneiden asiakkaiden haastattelut

Testauksen tehneille asiakkaille tehtiin heti testisession jälkeen haastattelu, jossa selvitettiin asiakkaan aikaisempi testauskokemus sekä tuntemuksia koskien tutkivaa testausta. Puolistrukturoidussa haastattelussa pyrittiin selvittämään testaajan tuntemuksia siitä, antaako suoritettu testisessio riittävän kuvan ohjelmiston laadusta käyttöönottoa ajatellen. Lisäksi testaajia pyydettiin arvioimaan, olisivatko he mieluummin suunnitelleet testitapaukset etukäteen. Testaajilta pyydettiin myös arvio siitä, miten paljon testitapauksien suunnittelu ja testien suorittaminen olisi vienyt aikaa, jos olisi käytetty mustalaatikko-perustaista testausta.

Roadmapping-työkalun testaajan mukaan testisession perusteella voidaan sanoa, että ohjelman peruskäyttö tuli testattua, mutta ei erityisen kattavasti. Suoritettu testaus olisi kuitenkin "ehkä riittänyt hyväksymistestaukseksi" ohjelman ottamiseksi esittelykäyttöön. Testaajan mielestä testaus painottui käyttöliittymän testaamiseen, koska kyseessä oli käyttöliittymäintensiivinen sovellus. Testaajan arvion mukaan testaaminen olisi vienyt huomattavasti enemmän aikaa, jos testitapaukset olisi pitänyt suunnitella ja dokumentoida etukäteen. Toisaalta testaaja uskoi, että formaalimmalla (suunnitelmallisemmalla) testaustavalla olisi löytynyt enemmän virheitä.

Verkkokaupan testaaja uskoi, että yhden testaajan suorittama testisessio ei ole missään tapauksessa riittävä hyväksymistesti, vaan tarvittaisiin vähintään 4-6 testaajaa, jotta testin kattavuus olisi riittävä. Testaaja totesi, että riittävän yleisellä tasolla oleva testicharter jättää tilaa testaajan persoonallisuudelle, eli järjestelmällisestä tavasta pitävä testaaja voi toteuttaa testin järjestelmällisesti, ja impulsiivisempi testaaja voi "poukkoilla" testin aikana toiminnallisuudesta toiseen. Testaajan mielestä testilokiin on tarpeellista tehdä merkintöjä niin, että myöhemmin voidaan todeta mitkä ominaisuudet on testattu. Pelkkä löydettyjen bugien korjaaminen ei siis riitä.

G. Tutkimustulosten analyysi

Haastattelujen perusteella näyttää siltä, että sessioperustainen tutkiva testaus voi olla riittävä menetelmä hyväksymistestauksen suorittamiseen, vain mikäli mahdolliset ohjelmistoon jäävät virheet eivät aiheuta asiakkaalle kovin suuria riskejä. Toisaalta tutkiva testaus mahdollistaa sen, että kokematon testaaja voi testisessioiden perusteella saada liian positiivisen kuvan ohjelmiston laadusta. Koska dokumentaation perusteella ei voida tarkkaan selvittää, mitä testaaja on testannut, ei organisaation muilla jäsenillä ole mahdollisuutta varmistaa, onko yhden testaajan järjestelmälle antama hyväksyntä luotettava. Verkkokauppa-sovelluksen testannut asiakas totesikin, että testaukseen pitäisi osallistua useita ihmisiä, jotta hyväksyntä voitaisiin tehdä. Tätä tukee myös havainto siitä, että saman testicharterin perusteella suoritetuissa vertaisryhmän ja asiakkaan tekemissä testeissä löytyi eri bugeja.

Jos testaajalla ei ole varsin paljon kokemusta testauksesta, on testaamisen ohjaaminen löytyneiden virheiden perusteella vaikeaa. Erityisesti havaittiin, että tutkivassa testauksessa testaajan asennoitumistavalla on suurempi vaikutus kuin etukäteen suunniteltuja testitapauksia käytettäessä. Mikäli tutkivaa testausta suorittava testaaja ei aktiivisesti halua löytää virheitä, ei tutkiva testaus ole toimiva menetelmä. Voitaneen ajatella, että tutkituissa tapauksissa testaus vastasi negatiivisessa mielessä ad hoc -testausta, eivätkä tutkivat testauksen hyödyt tulleet esille. Tutkimustapauksissa asiakkailla ei ollut myöskään käytettävissään listaa ohjelmistossa aikaisemmin havaituista bugeista, mitä olisi voinut käyttää apuna testin ohjaamisessa.

Sekä testattu verkkokauppa-sovellus että erityisesti roadmapping-sovellus olivat käyttöliittymäintensiivisiä, mikä vaikuttaa testaukseen luonteeseen selvästi. Tutkimuksen perusteella näyttää siltä, että käyttöliittymäintensiivisyydestä johtuen kokemattomampikin testaaja pystyy testaamaan järjestelmää, ja tekemään havaintoja nimenomaan käyttöliittymään ja käytettävyyteen liittyen. Verkkokaupan testaaja testasi kyllä, että tuotteiden hinnat näytetään käyttäjälle "oikean näköisesti", mutta se, laskiko ja käsittelikö järjestelmä tuotteiden hintoja ja veroja sisäisesti oikein jäi kokonaan testaamatta. Sovelluslogiikan testaus olisi kuitenkin tärkeää nimenomaan hyväksymistestauksessa, sillä luultavasti toimittaja kykenee itsekin havaitsemaan käyttöliittymässä olevia virheitä, mutta sovelluksen logiikassa saattaa olla määrittelyistä asti johtuvia virheitä, jotka vain sovellusalueen hyvin ymmärtävä asiakas voi havaita.

VI. YHTEENVETO JA PÄATELMÄT

A. Yleisiä havaintoja tutkivasta testauksesta

James Bachin (2003) mukaan ei ole mielekäästä määritellä tarkasti, onko testaus tutkivaa, vaan tärkeämpää on tietää, millä tavoilla ja missä määrin testaus on tutkivaa. Tässä tutkimuksessa molemmat asiakkaat tekivät testausta varsin samaan tapaan, mutta selvää on silti, että heidän tekemänsä testaus ei

suinkaan ole ainut tapa tehdä tutkivaa testausta. Bach myös korostaa sitä, että tutkivasta testauksesta tekee mielenkiintoista nimenomaan testaajan kyvyt ja taidot tehdä havaintoja ja reagoida niihin testauksen aikana. Testaajan täytyy olla asennoitunut siten, että hänen tehtävänä on etsiä mahdollisimman paljon virheitä, eikä vain pyrkiä käyttämään testattavaa tuotetta onnistuneesti. Tämä vaatimus saattaa olla ristiriitainen hyväksymistestauksen tavoitteen kanssa, mikäli vain halutaan varmistaa, että ohjelmalla voi tehdä tietyt asiat.

Vaikka tutkiva testaus vaikuttaa hyvin yksinkertaiselta ja tehokkaalta menetelmältä, ei se kuitenkaan ole mikään yleis-pätevä ratkaisu, jonka avulla kokemattomat testaajat pystyisivät vähällä vaivalla tekemään laadukasta testausta.

B. Tutkivan testauksen soveltuminen hyväksymistestaukseen.

Koska tutkivan testauksen testitapauksia ei suunnitella etukäteen, ei tutkivalla testauksella voida määrittellä käyttäjävaatimuksia suoraan verifioitavassa muodossa (Miller). Tutkivalla testauksella ei voida myöskään suoraan määrittää ohjelman valmiusastetta esimerkiksi laskemalla hyväksytyjä ja hylättyjä testitapauksia.

Jos testattavasta tuotteesta ei löydetä yhtään bugia, ei testausta voida käytännössä tällöin juurikaan ohjata havaintojen perusteella. Tällaisessa tapauksessa tutkiva testaus ei ole mielekäästä. Tällä perusteella voidaankin ajatella, että tutkiva testaus soveltuu paremmin käytettäväksi integrointi- ja järjestelmätestausvaiheissa, jolloin virheitä on järjestelmässä luultavasti enemmän, ja oleellista on mahdollisimman nopeasti löytää niistä mahdollisimman paljon.

Mikäli järjestelmän laatuvaatimukset eivät ole erityisen korkeat ja hyväksymistestaus pitää suorittaa mahdollisimman nopeasti, voidaan hyväksymistestaus suorittaa tutkivana testauksena.

C. Testaajan osaamistaustan vaikutus virheiden löytymiseen ja testauksen onnistumiseen:

Kirjallisuuden perusteella voidaan olettaa että testaajan kokemuksella on erittäin suuri vaikutus testauksen tehokkuuteen. Kokeneempi testaaja pystyy havaitsemaan pienempiä ja huomaamattomampia poikkeavuuksia ohjelman toiminnassa, ja kokenut testaaja kykenee myös paremmin suunnittelemaan testejä, jotka aiheuttavat ohjelman virheiden tapahtumisen. Toisaalta tutkivan testauksen varsin yksinkertainen idea saattaa antaa kokemattomalle testaajalle virheellisen kuvan testauksen helpoudesta.

Nyt analysoitujen testisessioiden perusteella näyttää siltä, että pelkästään vahva sovellusalueen osaaminen ei välttämättä riitä tekemään tutkivasta testaamisesta erityisen tehokasta. Testaaja löytää testien perusteella eniten virheitä sovelluksen alueista, joista häneltä on eniten kokemusta, ja käyttöliittymäintensiivistä sovellusta testattaessa kokematon testaaja saattaa keskittyä pelkästään käyttöliittymän testaamiseen niin että sovelluslogiikka jää kokonaan testaamatta.

Pelkkä sovellusalueen tunteminen ei antanut testaajalle riittävästi tietoja, jotta havaittujen virheiden perusteella olisi voitu löytää lisää virheitä. Jotta testauksen kulun ohjaaminen

havaintojen perusteella olisi mahdollista, on testaajan tunnettava myös käytettävää tekniikkaa, jolloin yhden virheen perusteella voidaan perustellusti lähteä etsimään lisää virheitä.

D. Mihin seikkoihin on erityisesti kiinnitettävä huomiota kun tehdään tutkivaa testausta

Mikäli järjestelmälle asetettavat laatuvaatimukset ovat erittäin korkeat, ei tämän tutkimuksen perusteella voida suositella tutkivaa testausta ainoaksi hyväksymistestauksen menetelmäksi.

Tutkiva testaus vaatii testaajalta runsaasti kokemusta sekä sovellusalueelta että käytettävien tekniikoiden osalta. Perinteisessä etukäteen suunniteltuihin testitapauksiin perustuvassa testauksessa voi kokenut testaaja suunnitella testitapaukset ja jakaa sitten aikaa vievät testitapauksien suorittamiset kokemattomimmille testaajille. Tutkivassa testauksessa tällaista työnjakoa ei kuitenkaan voida tehdä.

Mikäli tehdyt testit on dokumentoitava poikkeuksellisen tarkasti, asettaa tutkivan testauksen käyttö dokumentoinnille uusia haasteita. Jotta tutkiva testaus onnistuisi tehokkaasti, ei dokumentointi saa vaatia liikaa aikaa varsinkaan testisessioiden aikana.

Tutkivaa testausta tehtäessä on myös mietittävä, millä tavoilla organisaatio haluaa siirtää yksittäisille testaajille kertyvät tiedot ja kokemukset koko organisaation käyttöön.

Sessioperustaista tutkivaa testausta tehdessä on testien jakamisella eri sessioihin merkitystä testien onnistumiselle. Testisession tulee muodostaa järkevä kokonaisuus, mutta toisaalta tiettyjen ominaisuuksien testaamiseen kuluva aikaa on hyvin vaikea ennustaa etukäteen.

E. Lisätutkimuksen aiheita

Tutkivan testauksen tehokkuuteen vaikuttaa merkittävästi dokumentointiin kuluva aika, ja olisi hyödyllistä tutkia, miten tutkivaa testausta käytettäessä saadaan mahdollisimman tehokkaasti tallennettua ja toisaalta kommunikoitua testauksen aikana syntyneet havainnot koko organisaation käyttöön. Koska tutkiva testaus asettaa korkeat vaatimukset testaajan taidoille, olisi myös tärkeä tutkia, miten organisaatio voi helpottaa testaajien työtä koulutuksen ja tiedon levittämisen avulla. Tinkham (2003) käsittelee tutkimuksessaan yksittäisten testaajien oppimista, mutta seuraava luonnollinen askel olisi tutkia organisaation oppimista.

Olisi myös syytä tutkia, mitä tukea tutkiva testaus tarvitsee onnistuakseen. Järkevällä tavalla koostettu raportti ohjelmiston aikaisemmista bugeista ja korjauksista saattaisi olla hyödyllinen apu testaajalle testiä tehdessä.

LÄHTEET

Abrahamsson, P., et al. 2002, "Agile software development methods", VTT Publications 478 s. 95

Bach, James 2003, "Exploratory Testing Explained v.1.3 4/16/03", <http://www.satisfice.com/articles/et-article.pdf>

Bach, Jonathan 2000, "Session-Based Test Management", Software Testing and Quality Engineering, 11/00

- Beck, K., et al 2001, "Manifesto for Agile Software Development", <http://www.agilemanifesto.org/>
- Boehm, B. 2002, "Get Ready for the agile methods, With Care", Computer 35(1), 2002, s.64-69
- Brooks, F.P., 1987, "No Silver Bullet: essence and accidents of software engineering" The Mythical Man-Month, Anniversary Edition, Addison-Wesley, 1995
- Kaner, C. 2002, "A Course in Black Box Software Testing (Professional Version)",
http://www.testingeducation.org/coursenotes/kaner_cem/cm_200204_blackboxtesting/
- Crispin, L. 2003, "XP Testing Without XP: Taking Advantage of Agile Testing Practices", Methods & Tools, vol. 11, s. 2
- Fournier, R. 1998, "Testing, Testing... A rigorous Process", Information Week, 27.4.1998, s.8A
- Hughes, B., Cotterell. M. 2002, "Software Project Management", The McGraw-Hill Companies 2002, s. 234
- Kleinstei D., 1988, "System Acceptance Testing in a Real World: A Case Study", Computer Systems and Software Engineering, 1988. Proceedings., Third Israel Conference on , 6-7 June 1988
- Leung, H., Wong, P. 1997, "A study of user acceptance tests", Software Quality Journal 6, s. 137-149
- Miller, R. 2001, "Acceptance Testing", XP Universe 2001
- Tinkham 2003 "Learning Styles and Exploratory Testing"
- Yu Y 1999, "A Software Acceptance Testing TechniqueBased on Knowledge Accumulation", Ninth Great Lakes Symposium on VLSI, March 04 - 06, 1999