113

OHJELMISTOTESTAUS JA KETTERÄT MENETELMÄT

Vesa Kettunen

Tutkimusraportti 113
Research Report 113

OHJELMISTOTESTAUS JA KETTERÄT MENETELMÄT

Vesa Kettunen

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan osasto
PL 20
53851 Lappeenranta

ISBN 978-952-214-875-9
ISSN 0783-8069

Lappeenranta 2009

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan osasto

Vesa Kettunen

Ohjelmistotestaus ja ketterät menetelmät

Diplomityö

2009

115 sivua, 19 kuvaa, 6 taulukkoa ja 4 liitettä

Tarkastajat: Professori Kari Smolander
DI Jussi Kasurinen

Hakusanat: ohjelmistotestaus, ketterät menetelmät, laadullinen tutkimus, aineistopohjainen menetelmä
Keywords: software testing, agile methods, qualitative research, grounded theory

Tämän tutkimuksen tavoitteena on selvittää, miten erityyppisissä organisaatioissa ohjelmistotestaus on organisoitu, sekä mitä ongelmia ja etuja testauksen toimenpiteissä on käytännössä havaittu. Tutkimuksessa kiinnitetään huomiota myös testausresurssien määrään ja asiakkaan toimintaan ohjelmistokehitysprojekteissa.

Tässä tutkimuksessa keskityttiin selvittämään ketterien menetelmien vaikutusta ohjelmistotestauksen toteuttamiseen, sekä miten ketterät menetelmät vaikuttavat asiakkaiden toimintaan ohjelmistokehitysprojekteissa. Tutkimus toteutettiin laadullisena tutkimuksena, jossa tutkimusmenetelmänä käytettiin aineistopohjaista menetelmää. Tutkimusaineisto on kerätty haastattelemalla 12 organisaatioyksikön edustajia.

Tutkimuksessa havaittiin, että ketterien menetelmien käytöllä voidaan järjestää lisää aikaa ohjelmistotestauksen toteuttamiseen. Ketterissä menetelmissä testaus sidotaan kehitysprosessiin tiiviisti, jolloin testaustoimenpiteet tulee huomioida jo kehitystyön alkaessa. Tällainen lähtökohta tasaa testausresurssien tarvetta, koska testaustoimenpiteitä voidaan suorittaa projektin alusta lähtien.

Ketterien menetelmien havaittiin vaikuttavan myös asiakkaan toimintaan. Ketteriä menetelmiä varten toimittajaorganisaation on lisättävä yhteistyön ja kommunikoinnin määrää asiakkaan kanssa. Lisäksi asiakkaalta vaaditaan jatkuvaa läsnäoloa sekä ymmärrystä ketterästä kehityksestä, jotta kehittäjät saavat jatkuvasti palautetta nopean ja joustavan kehityksen takaamiseksi.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology

Vesa Kettunen

Software Testing and Agile Methods

Master's Thesis

2009

115 pages, 19 figures, 6 tables and 4 appendices

Supervisors: Professor Kari Smolander
M.Sc. Jussi Kasurinen

Keywords: software testing, agile methods, qualitative research, grounded theory

The purpose of this study is to determine how different types of organizations structure their software testing and what advantages or disadvantages in their approach have been detected in practice. Resources for testing and the actions of customers in software development projects are also considered.

This study focuses on researching how agile methods influence to the software testing activities and to activities of customers in software development projects. The study is a qualitative study and the research method used is grounded theory. The research data was gathered by interviewing representatives of 12 organization units.

It was identified that in agile methods there can be more time to carry out software testing activities. In agile methods software testing is closely attached to development work, so testing activities should be considered as soon as the projects begins. This approach divides the need for testing resources in a way that resources are needed throughout the project, not only at the end of the project.

Agile methods seem to affect the operation of customers also. For the use of agile methods, the vendor organization must increase the amount of collaboration and communication with the customer. In addition, active and continuous customer involvement is necessity for guaranteeing the quick feedback to developers. The customer feedback guarantees the fast and flexible software development.

SISÄLLYSLUETTELO

LYHENTEET

AM	Agile Modeling
ASD	Adaptive Software Development
CMMI	Capability Maturity Model Integration
CoC	Cycles of Control - framework
DSDM	Dynamic Systems Development Method
FDD	Feature Driven Development
ICT	Information and Communications Technology
ISO	International Organization for Standardization
MASTO	Adaptive Reference Model for Software Testing Organizations
OSS	Open Source Software (Development)
PK-yritys	Pieni ja Keskisuuri yritys
PP	Pragmatic Programming
RUP	Rational Unified Process
TDD	Test Driven Development
XP	eXtreme Programming

1 JOHDANTO

Tämä diplomityö on tehty osana MASTO-projektia. MASTO-projektin tavoitteena on kehittää ohjelmistotestaukselle adaptiivinen referenssimalli. Mallin tarkoituksena on antaa organisaatioille hyödyllistä tietoa testausprosesseista, tietämyksenhallinnasta sekä testausautomaatiosta, ja siten auttaa organisaatioita kehittämään näitä ohjelmistotestauksen osa-alueita.

Tämän diplomityön tavoitteena on selvittää miten erityyppisissä organisaatioissa ohjelmistotestaus on organisoitu, sekä mitä ongelmia ja etuja testauksen toimenpiteissä on havaittu. Organisaatiot voivat käyttää ketterää menetelmää, suunnitelmalähtöistä menetelmää tai niiden yhdistelmää. On selvää, että ohjelmistotestaus suoritetaan käytössä olevan ohjelmistokehitysmenetelmän vaatimalla tavalla, joten yleiskatsaus organisaatioiden tuotantomenetelmiin on tarpeen.

Tutkimuksen tarkoituksena on tutkia eroja ja yhtäläisyyksiä suunnitelmalähtöisten ja ketterien menetelmien välillä ohjelmistotestauksen näkökulmasta. Tavoitteena on löytää hyväksi havaittuja keinoja, miten organisaatiot ovat käytännössä selvittäneet testauksessa kohdatut ongelmat. Tutkimukselle löytyy perusteita, sillä esimerkiksi Itkonen et al. (2005) linjaavat tutkimuksessaan, että jatkotutkimusta tarvitaan ketterissä menetelmissä tarvittavien ohjelmistotestaustoimenpiteiden tunnistamiseksi. Työssä huomioidaan myös organisaatioilla käytössään olevat resurssit ja se, miten ne vaikuttavat ohjelmistotestaukseen. Työlle asetettu tutkimuskysymys kuuluu: *Kuinka ketterät tuotantomenetelmät vaikuttavat ohjelmistotestauksen toteuttamiseen?*

Tutkimus tehdään aineistopohjaisen menetelmän avulla, joten tutkimuksen kohteena ei ole koko ohjelmistotuotannon toimiala, vaan valittu otos toimialan edustajista. Tutkimus rajoittuu 12 organisaatioyksikön tutkimiseen, joten tulosten soveltaminen käytännössä vaatii tulkintaa.

Työ etenee siten, että luvussa 2 esitellään ohjelmistotuotannon menetelmät yleisesti. Luvussa 3 käsitellään tarkemmin ohjelmistotestausta, joka on tutkimuksen kannalta tärkeä ohjelmistotuotannon osa-alue. Luvussa 4 esitellään tutkimuksessa käytetty tutkimusmenetelmä sekä vaihtoehtoisia menetelmiä. Tutkimuskohteet on esitelty luvussa 5, jossa on lisäksi selvitetty, miten tutkimuksen aineisto on kerätty ja miten aineisto on analysoitu. Organisaatioyksiköiden tuotantomenetelmät on käsitelty luvussa 6. Luvussa 7 käsitellään aineistopohjaisen menetelmän avulla luodut kategoriat ja kategorioiden pohjalta määritellyt hypoteesit. Luku 8 koostuu tulosten pohdinnasta ja luvussa 9 on työn yhteenveto.

2 OHJELMISTOTUOTANNON MENETELMÄT

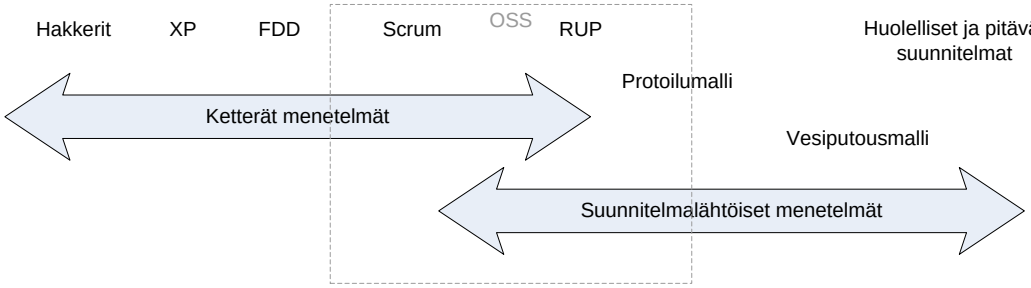
Ohjelmistotuotannossa on vuosien saatossa käytetty useita menetelmiä ohjelmistojen kehitykseen. Perinteisesti ohjelmistokehittäjät ovat suosineet menetelmiä, joissa kehittyneeseen ratkaisuun pyritään kattavien suunnitelmien ja kodifioitujen prosessien avulla (Boehm, 2002). Kirjallisuudessa tällaisista menetelmistä käytetään termiä suunnitelmalähtöiset menetelmät.

Suunnitelmalähtöisiä menetelmiä on kritisoitu, koska ne vaativat kattavaa dokumentointia, ja koska menetelmiin kuuluu uskomus, että ohjelmiston vaatimukset voidaan tunnistaa täydellisesti ohjelmistokehitysprojektin alussa (Sutherland, 2001). Vaatimusten on kuitenkin todettu muuttuvan lähes jokaisessa projektissa, koko projektin ajan (Haikala ja Märijärvi, 2004). Vaatimusmuutosten hallinnalla pyritään vaikuttamaan tähän ongelmaan, mutta menetelmien vaatimien dokumenttien muokkaus on työlästä, ja se luonnollisesti vie resursseja varsinaiselta kehitystyöltä.

Ketterät tuotantomenetelmät iskevät juuri suunnitelmalähtöisten menetelmien kritisoituihin kohtiin: hyväksytään muutokset projektien aikana ja dokumentoidaan vain välttämättömimmät asiat. Menetelmien välillä on tietenkin myös muita eroja. Yleisesti ajatellaan, että ketterät menetelmät ovat kevyitä ja nopeita, kun suunnitelmalähtöiset menetelmät ovat raskaita, hitaita ja byrokraattisia ohjelmistokehitysmenetelmiä. Kuvassa 1 menetelmiä on vertailtu niiden ketteryyden mukaan. Vasemmalla laidalla on niin kutsutut hakkerit, jotka tuottavat ohjelmistoa suunnittelemattomasti ja ilman järjestystä. Oikealla laidalla on täsmälliset ja sitoutumista vaativat suunnitelmat. Evoluutiomallin mukaiset menetelmät löytyvät kahden ääripään välimaastosta.

OSS tarkoittaa avoimen lähdekoodin ohjelmistoa, jonka lähdekoodi on vapaasti käytettävissä ja muokattavissa. Lähdekoodin veloitukseton uudelleenjakelu on myös sallittua. OSS-ohjelmiston kehitystä ei voi sijoittaa vertailuun täsmällisesti, koska kehitysprosessi voi vaihdella projektista riippuen. OSS-kehitys perustuu yhteisön toimin-

taan ja myötävaikutukseen, sillä yhteisön jäsenet kehittävät ohjelmistoa vapaaehtoisesti.. OSS-kehitysprosessi sisältää yhtäläisyyksiä ketteriin menetelmiin esimerkiksi pienten ja nopeiden kehityskierrosten osalta, mutta eroaa esimerkiksi kommunikoinnin järjestelyssä vapaaehtoisten kehittäjien hajautuneisuuden vuoksi. (Abrahamsson et al., 2002)



Kuva 1. Ohjelmistokehitysmenetelmät niiden ketteryyden mukaan, täydennetty Boehmin (2002, s. 65) kuvasta.

Scrum, XP ja FDD ovat ketteriä tuotantomenetelmiä. XP on yksi ensimmäisistä suunnitelluista ketteristä menetelmistä ja se on Scrum-menetelmän kanssa suosituimpia käyttöönotetuista ketteristä menetelmistä. XP ja Scrum on tarkemmin esitelty luvuissa 2.2.1 ja 2.2.2. FDD keskittyy ohjelmistokehityksen suunnittelu- ja toteutusvaiheisiin, joten se ei yksinään riitä kattamaan koko ohjelmistokehitysprosessia. FDD on kuitenkin suunniteltu yhteensopivaksi siten, että kehitysprosessia voidaan täydentää toisten kehitysmenetelmien käytännöillä. RUP on kehitysmenetelmien taitekohdassa, koska se sisältää ominaisuuksia ketterästä sekä suunnitelmalähtöisestä ohjelmistokehityksestä. (Abrahamsson et al., 2002)

Kuva 1 voi antaa käsityksen, että ketterissä menetelmissä ei suunnitella valmistettavaa ohjelmistoa. Mutta kyse on siitä, että suunnitelmat tehdään ja välitetään työryhmän sisäisissä keskusteluissa eikä luettavina dokumentteina. Näiden dokumenttien puutetta pidetään suunnittelemattomuutena. (Highsmith ja Cockburn, 2001; Boehm, 2002)

Ketterissä menetelmissä luotetaan siihen, että kehittäjien hiljainen tietämys on riittävällä tasolla, jolloin työryhmän jäsenet ymmärtävät mitä valmistettava sovellus vaatii. Puutteellinen tietämys voi kuitenkin johtaa vakavaan tai korjauskelvottomaan virheeseen esimerkiksi sovelluksen arkkitehtuurissa. Hiljaisella tietämyksellä tarkoitetaan ihmisten vuorovaikutuksissa välittyvää tietoa, jota ei ole dokumentoitu. Suunnitelmalähtöisissä menetelmissä hiljaisen tietämyksen välittymiseen liittyvät riskit minimoidaan huolellisten ja dokumentoitujen suunnitelmien avulla. Vaatimusmuutokset voivat kuitenkin tehdä suunnitelmista kalliita ylläpidettäviä tai jopa hyödyttömiä. (Boehm, 2002)

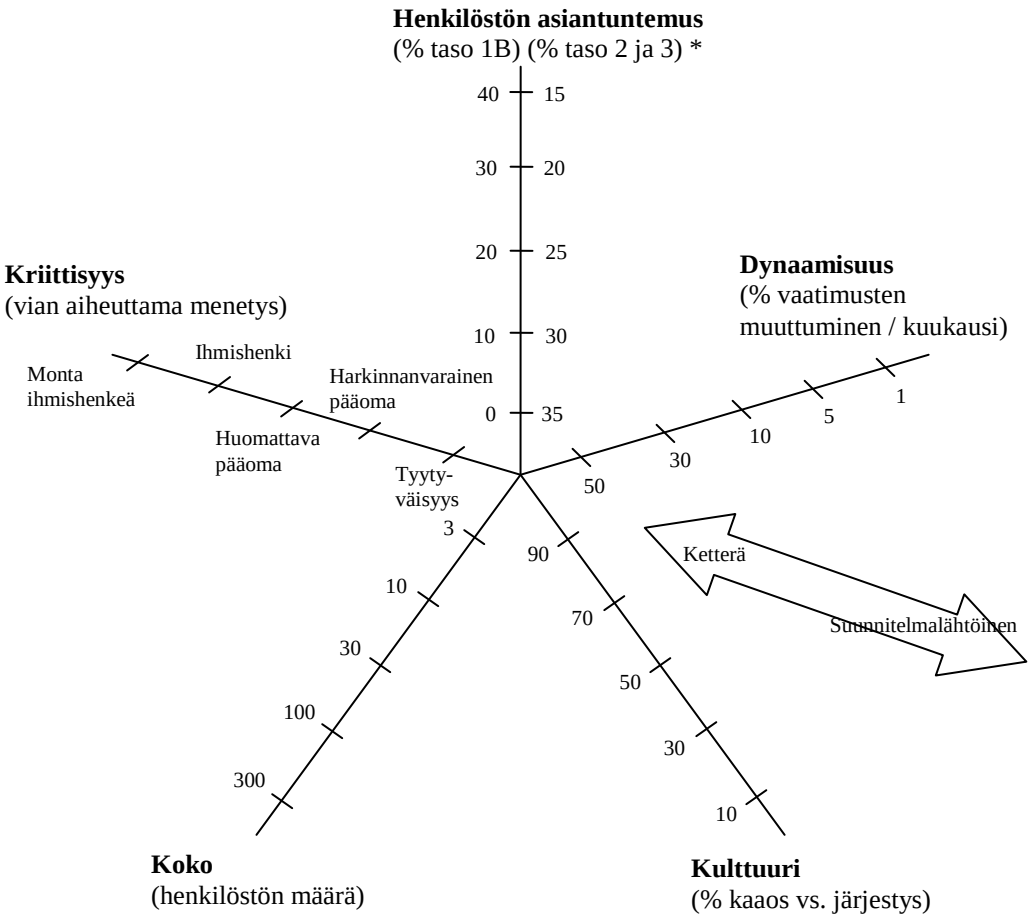
Sen lisäksi, että ketterissä menetelmissä kehittäjiltä vaaditaan hieman enemmän lahjakkuutta sekä taitoa, menetelmien ensisijaiset tavoitteetkin eroavat. Ketterien menetelmien ensisijainen tavoite on tuottaa toimiva sovellus nopeasti, joka on siten arvokas asiakkaalle (Fowler ja Highsmith, 2001). Suunnitelmalähtöisissä menetelmissä ensisijainen tavoite on tuottaa vakaa ja luotettava sovellus (Boehm, 2004). Menetelmien eroja on tarkemmin esitelty taulukossa 1.

Taulukko 1. Suunnitelmalähtöisten ja ketterien menetelmien eroja (Nerur et al., 2005).

	Suunnitelmalähtöinen	**Ketterä**
Perusolettamukset	Järjestelmät ovat spesifioitavissa ja ne voidaan rakentaa erittäin tarkalla ja laajalla suunnittelulla	Pienillä työryhmillä voidaan kehittää korkealaatuinen ja räätälöity ohjelmisto, hyödyntäen periaatetta: jatkuva suunnittelun ja testauksen parantaminen nopean palautteen ja muutosten ehdoilla
Kontrolli	Prosessikeskeinen	Ihmiskeskeinen
Johtamistyyli	Komenna ja kontrolloi	Johtajuus ja yhteistyö
Tietämyksenhallinta	Eksplisiittinen, täsmällinen	Hiljainen tietämys
Roolien määrääminen	Yksilöllinen – suosii erikoistumista	Itseorganisoituvat ryhmät – kannustaa roolien vaihtoihin
Kommunikaatio	Muodollinen	Epämuodollinen
Asiakkaan rooli	Tärkeä	Kriittinen
Projektin sykli	Tehtävien tai toimintojen ohjaama	Tuotteen ominaisuuksien ohjaama
Kehitysmalli	Elinkaarimalli (vesiputous, spiraali tai variaatio)	Evoluutiomalli
Toivottu organisaatiorakenne	Mekaaninen (byrokraattinen ja muodollinen)	Orgaaninen (joustava ja osallistuva, kannustaen yhteistyöhön ja sosiaaliseen toimintaan)
Teknologia	Ei rajoituksia	Suosii olio-orientoitunutta teknologiaa

Menetelmiä ei voi kuitenkaan pitää toistensa kilpailijoina, sillä yksikään menetelmä ei voi kattaa kaikkia erilaisia ohjelmistoprojekteja (Abrahamsson et al., 2002). Ketterien menetelmien on katsottu soveltuvan pienille riskittömille projekteille paremmin kuin suunnitelmalähtöiset menetelmät, jotka vastaavasti soveltuvat paremmin isoille riskialttiille projekteille (Boehm, 2006). Amblerin (2008) mukaan ketterien menetelmien menestys pienissä projekteissa on kuitenkin herättänyt organisaatioiden kiinnostuksen menetelmien käyttöönottamiseksi suuremmissakin projekteissa.

Kuvassa 2 on Boehmin (2004) malli siitä millaisiin projekteihin menetelmät soveltuvat. Kuvion keskelle sijoittuvat projektit soveltuvat paremmin ketterille menetelmille ja kuvion reunoille sijoittuvat projektit soveltuvat paremmin suunnitelmalähtöisille menetelmille. Reunan ja keskustan väliin sijoittuvissa projekteissa pitäisi käyttää räätälöityä menetelmien yhdistelmää eli hybridimallia, jossa yhdistellään molempien menetelmien ominaispiirteitä vastaamaan kunkin projektin haasteita. Hybridimallissa sovelluksen runko voidaan esimerkiksi tehdä käyttämällä suunnitelmalähtöistä menetelmää, ja käyttöliittymä tehdä ketterän menetelmän avulla.



* Tasot 1B, 2 ja 3 kuvaavat Cockburnin ohjelmiston ymmärtämisen kolmea tasoa. Tasot 2 ja 3 kuvaavat asiantuntijoita. Cockburnin asteikko on verrannollinen sovelluksen kompleksisuuteen. Yksinkertaisia sovelluksia tuottavassa organisaatiossa kehittäjä voi olla tasolla 2, mutta monimutkaisia sovelluksia kehittävässä organisaatiossa sama kehittäjä voi olla tasolla 1A.

Kuva 2. Menetelmien käyttökohteet (Boehm ja Turner, 2004, s. 56).

Hybridimallin käyttöä tukee esimerkiksi Salon ja Abrahamssonin (2008) tutkimuksessa tehty havainto, jossa huomattiin organisaatioiden käyttävän ketterien menetelmien ominaisuuksia täydentämään alkuperäistä kehitysprosessia. Samassa tutkimuksessa käytettiin Boehmin kaaviota (Kuva 2.) projektien erottelemiseen ja havaittiin, että yksikään projekti ei selkeästi sijoittunut kaavion keskelle tai reunalle. Myös Cusumanon et al. (2003) tutkimuksessa on viitteitä siitä, että organisaatiot käyttävät projekteissaan molempien menetelmien ominaisuuksia yhdessä.

Hybridimenetelmiksi voidaan luokitella esimerkiksi Scrum ja RUP. Scrum käsitetään ketteränä menetelmänä, mutta siihen sisältyy kuitenkin perinteiset ohjelmistokehityksen vaiheet: analyysi, suunnittelu, toteutus ja toimitus. RUP on suunnitelmalähtöinen menetelmä, joka on kuitenkin muokattavissa ketterän menetelmän mukaiseksi. RUP sisältää ketterille menetelmille tyypillisiä ominaisuuksia, kuten iteratiivisen kehityksen ja testauksen korostamisen iteraatioiden aikana.

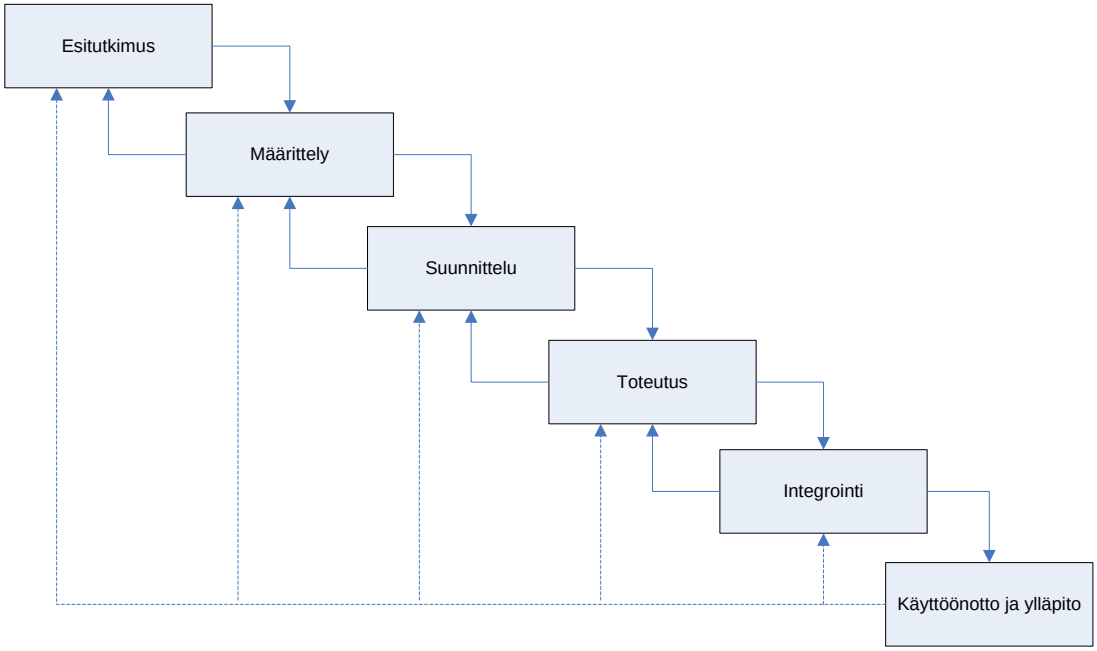
2.1 Suunnitelmalähtöiset menetelmät

Suunnitelmalähtöisistä menetelmistä on esitelty vesiputous- sekä protoilumalli. Vesiputousmalli on pitkään ollut ohjelmistokehityksen perustana. Sommervillen (1995) mukaan vesiputousmalli esiteltiin ohjelmistokehityksen prosessina ensimmäisen kerran 1970-luvulla. Protoilumallissa on samankaltaisia piirteitä ketterien menetelmien kanssa, joten se on esitelty erojen selventämiseksi.

2.1.1 Vesiputousmalli

Perinteisesti suunnitelmalähtöiset menetelmät ovat noudattaneet vesiputousmallin mukaista kehitystä, jossa määritellään jokainen kehitysvaihe erikseen. Vesiputousta mukaillen, vaiheet suoritetaan järjestyksessä ylhäältä alas. Jokainen vaihe tuottaa dokumentin tai dokumentteja, joiden perusteella seuraavassa vaiheessa jatketaan kehi-

tystyötä. Vesiputousmallista on tehty useita eri versiota, joille yhteisiä piirteitä ovat ainakin määrittely-, suunnittelu- ja toteutusvaiheet (Haikala ja Märijärvi, 2004). Kuvassa 3 on Haikalan ja Märijärven (2004) teoksessa esiintyvä versio vesiputousmallista.



Kuva 3. Vesiputousmalli (Haikala ja Märijärvi, 2004).

Käytännössä vesiputouksen vaiheita ei suoriteta täysin lineaarisesti aloittamalla seuraava vaihe edellisen päätyttyä. Vaiheita suoritetaan päällekkäin, koska seuraava vaihe paljastaa usein ongelmia edellisestä vaiheesta. (Sommerville, 1995)

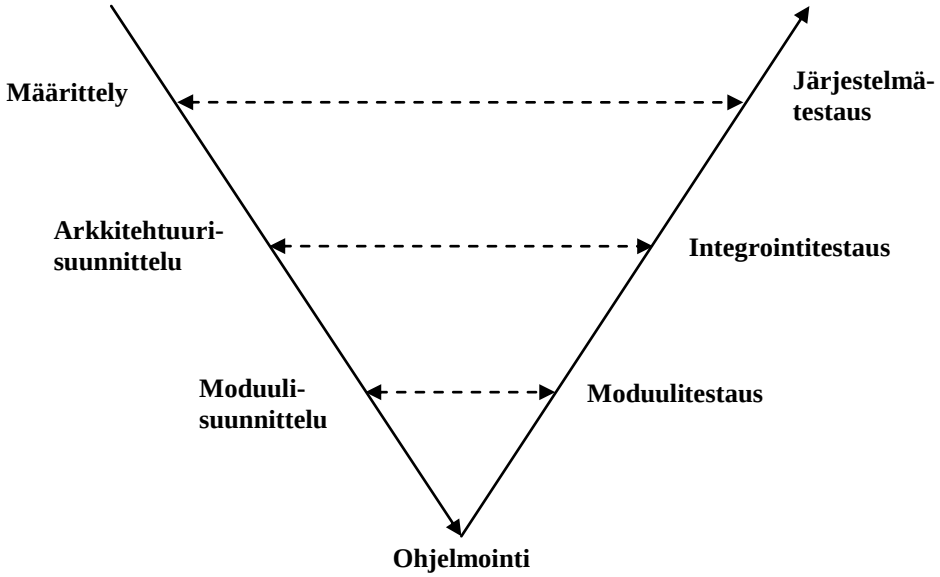
Esitutkimusvaiheessa laaditaan valmistettavalle järjestelmälle yleiset vaatimukset ja selvitetään miksi järjestelmää ollaan tekemässä. Yleiset vaatimukset käsittävät myös asiakasvaatimukset, jotka ovat kriittisiä projektin onnistumiselle. Asiakasvaatimukset pitää ymmärtää oikein, jotta tuotettu järjestelmä täyttää asiakkaan odotukset. Asiakasvaatimusten analysointi jatkuu määrittelyvaiheessa, jossa niistä muodostetaan järjestelmävaatimukset. Järjestelmävaatimuksiin sisältyy ohjelmiston toiminnot, ominaisuudet sekä ei-toiminnalliset vaatimukset. Ei-toiminnallisia vaatimuksia ovat esimerkiksi suorituskyky ja käytettävyys. (Haikala ja Märijärvi, 2004)

Suunnitteluvaiheessa suunnitellaan järjestelmävaatimuksissa määriteltyjen toimintojen toteutus. Suunnitteluvaihe koostuu järjestelmän arkkitehtuurin sekä yksittäisten moduulien suunnittelusta. Moduulilla tarkoitetaan mahdollisimman itsenäistä järjestelmän osaa. Toteutusvaiheessa ohjelmoidaan suunnitelmien mukaiset moduulit. Toteutusvaiheeseen kuuluu testaustoimenpiteistä moduuli- eli yksikkötestaus, jossa verifioidaan, että moduulit vastaavat niiden määrittelyjä. (Sommerville, 1995; Haikala ja Märijärvi, 2004)

Integrointivaiheessa moduulit liitetään yhteen kokonaiseksi järjestelmäksi ja suoritetaan integrointi- sekä järjestelmätestaus. Integrointitestauksessa tutkitaan moduulien rajapintojen toimivuutta, ja järjestelmätestauksessa tarkastellaan koko järjestelmää. Järjestelmätestauksen tarkoitus on tutkia, täyttääkö järjestelmä sille määritellyt vaatimukset. Ei-toiminnallisten ominaisuuksien testaus tehdään järjestelmätestauksen aikana. Myös hyväksymistestaus voi sisältyä järjestelmätestaukseen, mutta se voidaan toteuttaa erikseenkin viimeisenä testausvaiheena järjestelmätestauksen jälkeen. Hyväksymistestauksessa ohjelmistotuotetta verrataan sen alkuperäisiin vaatimuksiin sekä loppukäyttäjän tarpeisiin. (Myers, 2004; Haikala ja Märijärvi, 2004)

Testauksen jälkeen järjestelmä toimitetaan asiakkaalle, jolloin alkaa viimeinen vaihe: käyttöönotto ja ylläpito. Sommervillen (1995) mukaan tämä on pisin vaihe ohjelmistokehityksessä. Ylläpidon aikana korjataan aiemmin löytymättömiä virheitä, parannetaan järjestelmän osia ja lisätään toimintoja järjestelmään, jos uusia vaatimuksia havaitaan.

Ohjelmistotestauksen V-malli täydentää vesiputousmallia lisäämällä siihen testauksen tasoja sekä sitoo testauksen suunnittelun paremmin kehitystyöhön. V-malli on esitetty kuvassa 4. V-mallissa jokainen testauksen taso suunnitellaan sitä vastaavalla järjestelmän suunnittelutasolla (Haikala ja Märijärvi, 2004). Bertolinon (2007) mukaan V-malli on suosittu menetelmä ohjelmistoteollisuudessa, mutta mallia on kuitenkin kritisoitu tehottomaksi ja tarpeettoman byrokraattiseksi.

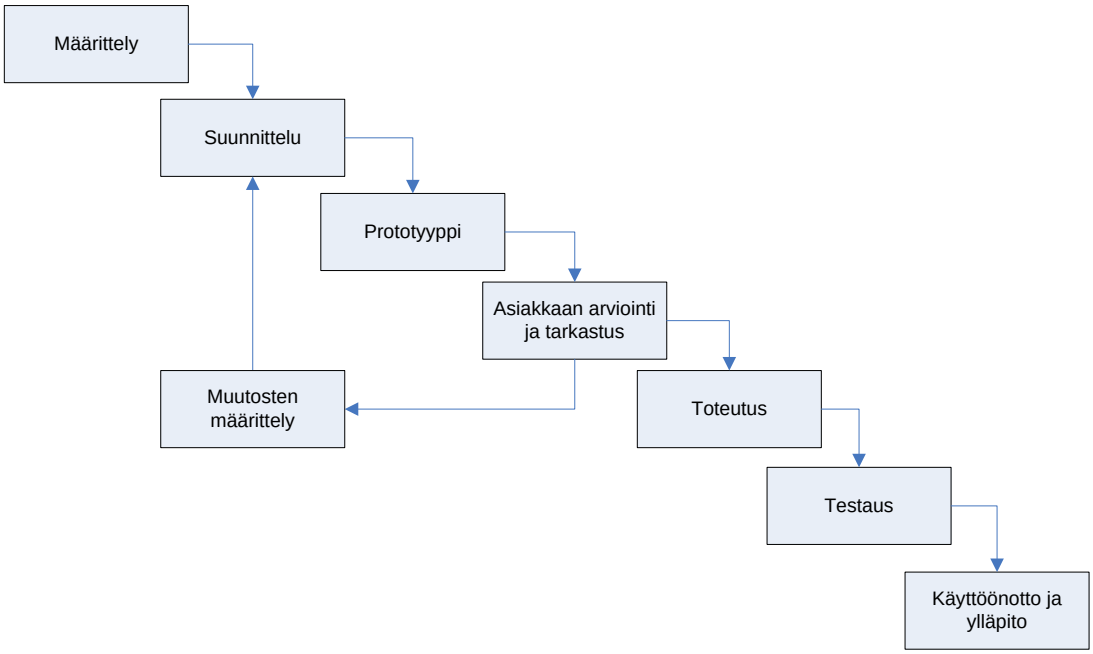


Kuva 4. Ohjelmistotestauksen V-malli (Haikala ja Märijärvi, 2004).

Vesiputous- ja V-mallin mukaisesti testaus suoritetaan ohjelmistokehityksen viimeisenä vaiheena. On havaittu, että tuotekehityksen venyessä tarvittava lisäaika otetaan testausajasta (Kit, 1995; Cohen et al., 2004; Taipale, 2007). Syitä testausajan lyhentämiseen ovat esimerkiksi testauksen arvostuksen puute sekä testauksessa tarvittavien resurssien aliarvioiminen (Cohen et al., 2004; Taipale, 2007).

2.1.2 Protoilumalli

Vesiputousmallin lisäksi protoilumalli on suunnitelmalähtöinen menetelmä. Protoilumallin avulla pyritään reagoimaan paremmin muuttuviin asiakasvaatimuksiin tuottamalla ensin prototyyppi valmistettavasta järjestelmästä. Tällöin kaikkia ohjelmiston vaatimuksia ei tarvitse määritellä heti projektin alussa. Kuvassa 5 on esitelty protoilumallin mukainen tuotekehitys.



Kuva 5. Protoilumalli (Haikala ja Märijärvi, 2004)

Protoilumalli on iteratiivinen prosessi, jossa tuotetta demonstroidaan, arvioidaan, jalostetaan ja laajennetaan. Tyypillisesti yksi iteraatio kestää 1-3 kuukautta, mutta se voi kestää jopa kuusi kuukautta. Protoilumalli perustuu siihen, että asiakas tietää mitä haluaa järjestelmältä vasta sen jälkeen, kun hän on nähnyt ja kokeillut sitä. Prototyypin avulla asiakas pääsee kokeilemaan järjestelmää ja muokkaamaan järjestelmän kehityssuuntaa projektin aikana. Tällöin prototyypissä havaitut virheet eivät siirry varsinaiseen järjestelmään. Protoilumallin prosessi vaatii sitoutumista asiakkaalta, koska asiakkaan on säännöllisesti annettava palautetta prototyypeistä. (Fitzgerald et al., 2002)

2.2 Ketterät menetelmät

Ketterillä menetelmillä tarkoitetaan keveitä ja nopeasti muutoksiin reagoivia ohjelmistokehitysmenetelmiä. Menetelmille yhteiset perusarvot ja periaatteet ovat määritelty Agile Manifestissa (Fowler & Highsmith, 2001). Manifestissa määritellyt neljä perusarvoa ovat:

- yksilöt ja vuorovaikutus ovat tärkeämpiä kuin prosessit ja työkalut,
- toimiva ohjelmisto on tärkeämpää kuin kattava dokumentointi,
- yhteistyö asiakkaan kanssa on tärkeämpää kuin sopimusneuvottelut ja
- muutoksiin sopeutuminen on tärkeämpää kuin suunnitelman noudattaminen.

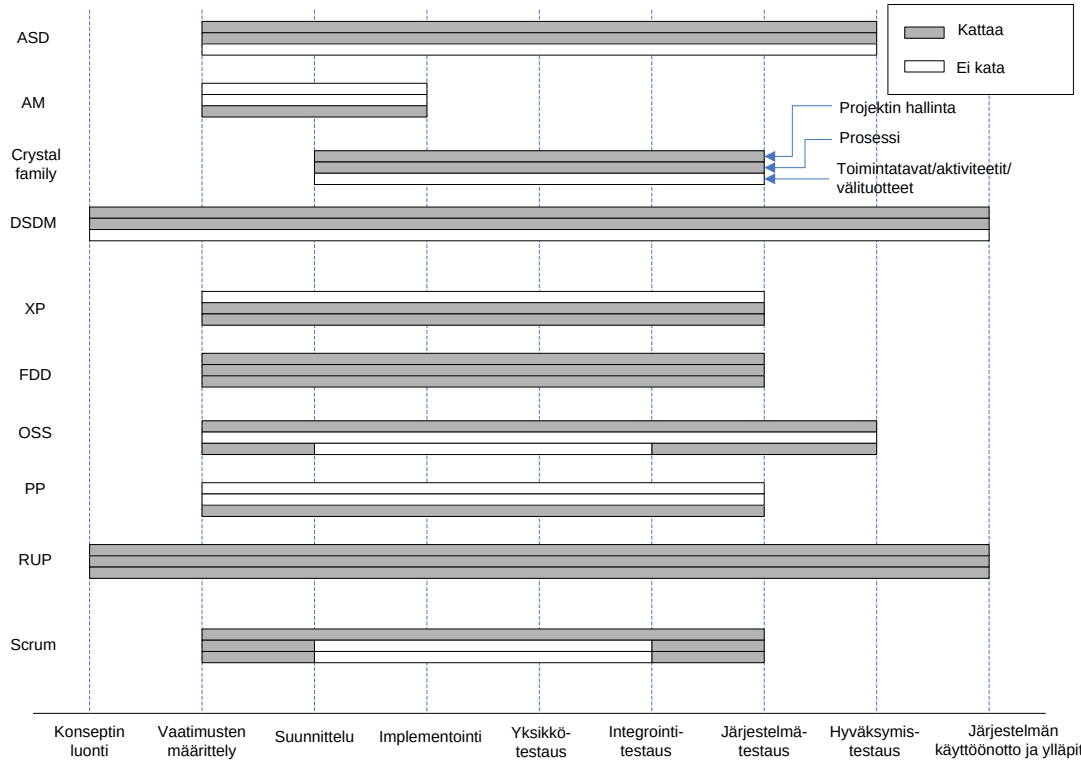
Fowlerin ja Highsmithin (2001) mukaan perusarvoissa on huomioitava, että jokaisessa väittämässä oikealle puolelle jäävä arvo on tärkeä, mutta kyse on siitä mikä on vielä tärkeämpää. Tärkeysjärjestys perustuu siihen, kumpi arvoista on ketterämpi.

Ensimmäinen väittämä korostaa taitavien yksilöiden ja heidän välisten vuorovaikutusten tärkeyttä. Käytännössä tämä arvo konkretisoituu esimerkiksi tiiviinä työympäristönä, joka kannustaa keskusteluihin ja siten lähentää työryhmän sisäisiä ihmissuhteita. Toisen väittämän tarkoitus on helpottaa kehittäjien dokumentointityötä, koska ketterissä menetelmissä tyypillisesti julkaistaan ohjelmistosta useita versioita tiheään tahtiin. Versioiden tarkka dokumentointi vaatisi kohtuuttoman määrän työtä, joten työryhmille annetaan päätösvalta riittävän dokumentoinnin tuottamiseksi. Kolmas väittämä perustuu siihen, että asiakkaan vaatimukset ymmärretään paremmin jos asiakas tekee yhteistyötä ohjelmistokehittäjien kanssa. Tiivis yhteistyö auttaa molempien osapuolien edustajia ymmärtämään paremmin valmistettavaa ohjelmistoa, jolloin on hyvin todennäköistä, että toimitettu ohjelmisto vastaa asiakkaan odotuksia. Neljännen väittämän avulla ketterät menetelmät sopeutuvat ympäristöön, jossa on odotettavissa paljon vaatimusmuutoksia. Monet projektit ovat onnistuneet vain siksi, että niissä on reagoitu nopeasti muuttuviin vaatimuksiin. Tämä edellyttää, että kehittäjillä on valta tehdä muutosten aiheuttamat korjaukset ja että kehittäjät ovat myös valmiita tekemään tarvittavat korjaustoimenpiteet. (Fowler ja Highsmith, 2001; Abrahamsson et al., 2002)

Ketterissä menetelmissä kehitysprosessi toteutetaan lyhyinä iteratiivisina ja inkrementaalisina, laajennuksin kasvatettavina, kehityskierroksina. Prosessissa vaaditaan, että asiakas on aktiivisesti mukana määrittelemässä, priorisoimassa ja verifioimassa vaatimuksia. Olennainen osa kehitysprosessia on itseorganisoituvien (self-

organizing) työryhmien käyttäminen, työryhmän annetaan itse päättää miten työ tehdään. Kehitysprosessia ei varsinaisesti lukita noudattamaan tiettyä kaavaa, vaan prosessin pitäisi antaa muodostua ja täydentyä projektin edetessä. Kehitysprosessille on kuitenkin tyypillistä, että toiminnallisuudet priorisoidaan ja tärkeimmät toteutetaan ensimmäisinä. (Boehm, 2004)

Abrahamssonin (2002) mukaan ketterien menetelmien variaatiot keskittyvät ohjelmistokehityksen eri osa-alueisiin. Esimerkiksi XP ja AM keskittyvät toimintatapoihin, Scrum keskittyy ohjelmistoprojektien hallintaan, kun DSDM kattaa koko ohjelmistokehityksen elinkaaren. Kuvassa 6 on esitetty mihin osa-alueisiin eri menetelmät ovat keskittyneet ja mihin ohjelmistokehityksen vaiheeseen menetelmä sopii. Harmaa väri kertoo, mitkä osa-alueet menetelmä kattaa ja valkoinen on merkkinä siitä, jos menetelmä ei kata jotain osa-aluetta. Niissä kohdissa, joita menetelmä ei kata, voidaan soveltaa organisaatiossa aiemmin käytettyjä menettelyjä tai lainata menettelyjä toiselta ketterältä menetelmältä. Esimerkiksi Scrum- ja XP-menetelmien yhdistelmä on todettu yhteensopivaksi Vriensin (2003) sekä Judyn ja Krumins-Beensin (2007) tutkimuksissa. Näissä tutkimuksissa Scrum-menetelmää käytettiin projektin hallintaan ja XP-menetelmästä sovellettiin toimintatapoja, kuten pariohjelmointia sekä testilähtöistä kehitystä (TDD).



Kuva 6. Ketterät menetelmät ja ohjelmistokehityksen vaiheet (Abrahamsson et al., 2002).

Vaikka kuvasta 6 voisi päätellä, että testaus on kohtuullisen hyvin katettu ketterissä menetelmissä, on ongelmia menetelmien käytössä esiintynyt juuri testauksen osalta. Ketterät menetelmät tuovat haasteita ohjelmistotestaukseen Agile Manifestissa määriteltyjen periaatteiden vuoksi. Säännöllisesti toimitettavien ohjelmistoversioiden vuoksi testausaika on lyhyt, eivätkä testaustoimenpiteet saa ylittää toimituksen määräaikaa. Muutoksien hyväksyminen myöhäisessäkin projektin vaiheessa aiheuttaa sen, että testausta ei voida suorittaa valmiiden määrittelyjen rajoissa. Lisäksi haasteena on saada kehittäjät testaamaan tuotetta, koska monissa ketterissä menetelmissä testaus on myös kehittäjien vastuulla. Tämä synnyttää ristiriidan perinteisen testauksen kanssa, sillä testaus on ammattiala, joka vaatii tuhoamisasennetta, erityistä taitoa ja kokemusta. Näitä ominaisuuksia ei omaa sovellusta testaavalla kehittäjillä välttämättä ole. (Itkonen et al., 2005)

Itkosen et al. (2005) mukaan suurimmassa osassa ketteristä menetelmistä ei ole riittävästi ohjeita testauksen suorittamiseen, poikkeuksena kuitenkin XP-menetelmä. Useissa menetelmissä on katettu yksikkö- ja integrointitestauksen toteutus, mutta esimerkiksi järjestelmätestauksen toteutus on jätetty kehittäjien päätettäväksi (Itkonen et al., 2005). V-mallin mukainen testaus ei sellaisenaan sovellu käytettäväksi ketterissä menetelmissä, koska se on vaiheittain etenevä malli. Ketterissä menetelmissä testaustoimenpiteiltä vaaditaan palautetta jatkuvasti ja varhain projektin aikana, koska testaus pitää tehdä kehityskierroksen aikana.

Aikaisemmissa tutkimuksissa on todettu, että testauksen automatisointi on lähes välttämätöntä ketterissä menetelmissä testauksen kattavuuden varmistamiseksi (Puleio, 2006; Sumrell, 2007; Shaye, 2008). Iteratiivisessa prosessissa ei ole aikaa tehdä kaikkea testausta manuaalisesti. Testausautomaation avulla kehittäjät saavat palautetta nopeasti, jolloin toiminnallisuuksien kehittämiseen jää enemmän aikaa (Puleio, 2006).

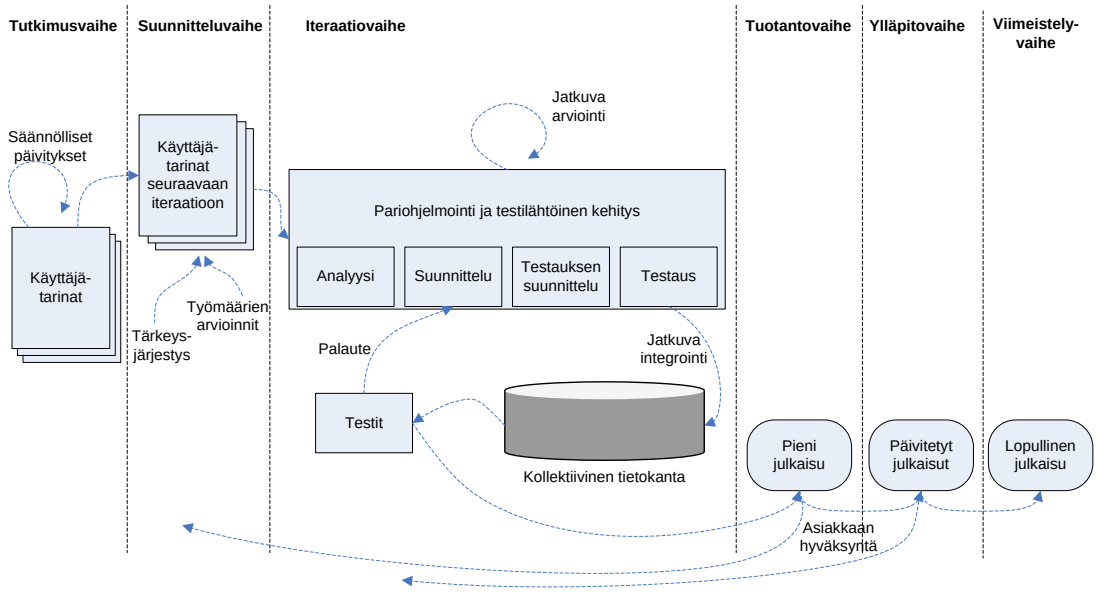
Esimerkiksi regressiotestauksen automatisointi on hyödyllistä, koska inkrementaalisessa kehityksessä aiemmin suunniteltuja testejä tarvitaan seuraavan laajennuksen testaamisessa. Testausautomaatio asettaa kuitenkin haasteita henkilöstölle, sillä tietämys testauksen automatisoinnista voi olla vähäistä (Shaye, 2008). Shayen (2008) tapauksessa puutteet henkilöstön tietämyksessä testausautomaation osalta korjattiin koulutuksen sekä mentorointiohjelmien avulla.

Toimivan testausautomaation avulla projektin työryhmään mahdollisesti kuuluvat testaajat voidaan keskittää tekemään tutkivaa (exploratory) testausta sovelluksen haasteellisempiin osiin (Shaye, 2008). Tutkivassa testauksessa suoritettavat testit eivät ole valmiiksi määriteltyjä, vaan testaaja muokkaa edellisiä ja kehittää uusia testejä kokemuksensa, tietojensa sekä aiemmin suoritettujen testien tulosten perusteella (Itkonen ja Rautiainen, 2005).

Seuraavassa on ketterien menetelmien kirjosta esimerkkimenetelminä lyhyesti esitelty XP ja Scrum. Käyttöönotetuista menetelmistä XP ja Scrum ovat olleet suosituimpia ja ne ovat myös hyvin dokumentoituja menetelmiä (Salo ja Abrahamsson, 2008).

2.2.1 Extreme Programming

XP-menetelmä on kehitetty pieniä ja keskisuuria työryhmiä varten, jotka kehittävät ohjelmistoa jatkuvasti muuttuvien vaatimusten keskellä (Beck, 1999). Menetelmä ei itsessään tuo uusia käytäntöjä ohjelmistotuotantoon, vaan se muodostaa aikaisempia käytäntöjä yhdistelemällä uuden tavan kehittää ohjelmistoja (Abrahamsson et al., 2002). Menetelmässä hyväksi havaitut käytännöt ja periaatteet viedään äärimmäisyyksiin (extreme). XP-menetelmä sisältää 12 toimintatapaa, joiden yhteiskäytöllä pyritään saavuttamaan ketterämpi kehitys. Toimintatapoja ovat esimerkiksi testilähtöinen kehitys (TDD), pariohjelmointi, jatkuva integrointi sekä pienet julkaisut. Kuvassa 7 on esitetty XP-menetelmän kehitysprosessi.

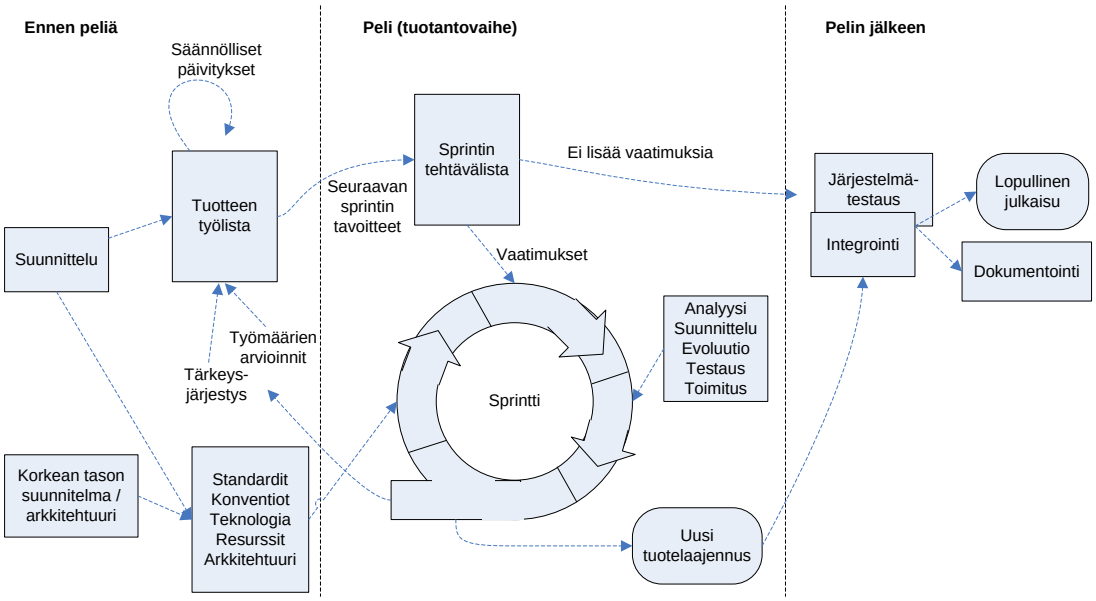


Kuva 7. XP-menetelmän kehitysprosessi (Abrahamsson et al., 2002).

Ketterille menetelmille ominainen inkrementaalinen kehitys tapahtuu prosessin iteraatiovaiheessa. Iteraation kesto on yhdestä neljään viikkoa, jonka aikana työryhmä pyrkii saamaan valmiiksi yhden tai useamman tavoitteeksi asetetun, ja asiakkaan valitseman käyttäjätarinan. Asiakas kirjoittaa käyttäjätarinat, jotka kuvaavat ohjelmistoon lisättävää toiminnallisuutta (Abrahamsson et al., 2002). Asiakkaan pitää kirjoittaa myös hyväksymistestit, jotka suoritetaan iteraatioden päätteeksi yksikkötestien kanssa (Beck, 1999). Kehitystyö päättyy, kun asiakas ei enää tuo uusia toiminnallisuuksia toteutettaviksi.

2.2.2 Scrum

Scrum-menetelmän lähtökohtana on ohjelmistokehitysprosessin hallinta. Scrum ei määrittele kehitystyölle tiettyjä käytäntöjä tai tekniikoita noudatettavaksi, vaan se keskittyy työryhmän tuottavuuden takaamiseen muutosalttiissa ympäristössä (Abrahamsson et al., 2002). Usein työryhmä aloittaa Scrum-menetelmän käytön soveltamalla yrityksessä aiemmin käytettyjä tekniikoita. Prosessin edetessä Scrum puuttuu käytettyihin tekniikoihin, jos niistä koituu häiriöitä työryhmälle tai jos ne vähentävät työryhmän tuottavuutta (Schwaber ja Beedle, 2002). Scrum edistää kehitysprosessia kehittämällä käytettyjä tekniikoita sekä poistamalla työryhmän kohtaamat esteet. Scrum-menetelmän prosessi on esitetty kuvassa 8. Termi *scrum* pohjautuu rugby-peliin, jonka vuoksi prosessin vaiheet ovat nimetty peliin liittyviksi.



Kuva 8. Scrum-menetelmän prosessi (Abrahamsson et al., 2002)

Scrum-menetelmän ketterä osuus on tuotantovaihe, jossa laajennukset tehdään pyrähdyksissä. Scrum-menetelmässä pyrähdyksiä kutsutaan sprinteiksi, joten jatkossa tässä työssä pyrähdyksiä kutsutaan termillä sprintti. Sprinteissä suoritetaan perinteiset ohjelmistokehityksen vaiheet: analyysi, suunnittelu, toteutus ja toimitus. Scrum pyrkii hallitsemaan muuttujia kuten aikaa, vaatimuksia, laatua, resursseja, ja implementointitekniikoita sprinttien aikana, jotta prosessissa voidaan joustavasti sopeutua esiintyviin muutoksiin. (Abrahamsson et al., 2002)

3 OHJELMISTOTESTAUS

Virheet ovat väistämättömiä, ihmiset tekevät virheitä luonnostaan. Ohjelmistotuotannossa asiakkaalle asti pääsevät virheet voivat kuitenkin aiheuttaa vakavia seuraamuksia, kriittisessä ohjelmistossa jopa ihmishengen menetyksen. Ohjelmistotuotannossa virheisiin pätee myös se, että mitä aikaisemmin ne löydetään, sitä huokeammin ne ovat korjattavissa. Ohjelmistotestauksen tavoite on löytää tuotteesta mahdollisimman paljon virheitä annetuissa rajoissa. Rajoissa siksi, että ohjelmistotestaus on usein tasapainoilua resurssien ja ajan kanssa. (ISO/IEC 29119)

Ohjelmistotestauksen osuus koko ohjelmistokehityksen kuluista voi nousta jopa 50 prosenttiin (Kit, 1995; Bertolino, 2007), joten testauksen tärkeyttä ei voi sivuuttaa. Kitin (1995) mukaan testaus muodostuu verifioinnista ja validoinnista. Verifiointi ja validointi ovat prosesseja, joilla varmistetaan, että tuote täyttää spesifikaatiot ja asiakkaan odotukset (Sommerville, 1995). Verifioinnilla tarkastetaan, tehdäänkö tuotetta oikein ja validoinnilla tarkistetaan, tehdäänkö oikeaa tuotetta. Verifiointia ja validointia suoritetaan koko ohjelmistokehityksen ajan. Lisäksi on havaittu (Kit, 1995), että vaatimusmäärittelyvaiheessa syntyy yli 50 prosenttia ohjelmiston virheistä. Vaatimusten verifiointi mahdollistaa virheiden löytämisen aikaisessa vaiheessa, jolloin niiden korjaaminen on helpompaa ja halvempaa. Testaus pitää aloittaa ajoissa, jotta säästytään myöhemmin löytyvien virheiden aiheuttamilta kuluilta.

Testauksen kohteena voi olla dokumentit, ohjelmiston komponentti, täydet ohjelmistovaatimukset, suunnitelmat, koodin osat tai valmis järjestelmä. Testaus tehdään kullekin testaustoimenpiteelle valitun vaatimuksen mukaisesti, jolloin testaustoimenpiteelle muodostuu tavoite. Testattavana voi olla esimerkiksi ei-toiminnallinen vaatimus tai toiminnallisuuden vaatimus. Kohteen vertaamista sille määriteltyihin vaatimuksiin kutsutaan testauksen suorittamiseksi. Testauksen suorittamisen pitää kuitenkin olla suunniteltu ja valmisteltu toimenpide. Tämä edellyttää, että testausta hallin-

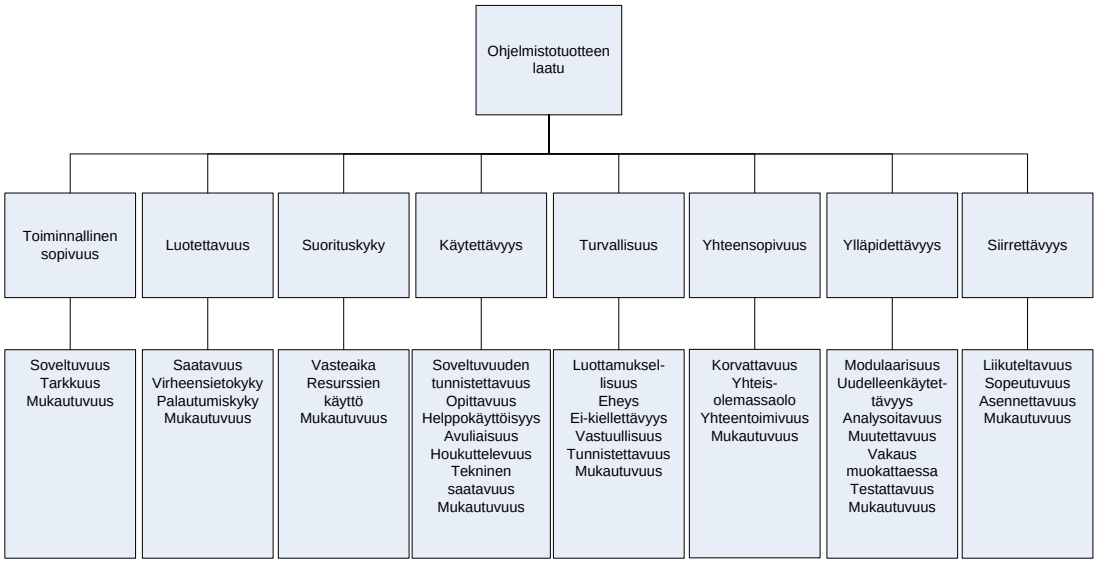
noidaan ja sen etenemisestä kerätään tietoa analysoitavaksi ja raportoitavaksi. Kerätyn tiedon avulla saadaan käsitystä testattavan kohteen laadusta. (ISO/IEC 29119)

3.1 Testauksen tavoitteet ja laatu

Testauksella täytyy olla syy tai tavoite. Ohjelmistotestauksen tavoitteita ovat esimerkiksi riskien vähentäminen, asiakasvaatimusten täyttymisen arviointi, laadun arviointi sekä tehdyn virheenkorjauksen tai muutoksen oikeellisuuden arviointi. Ohjelmistotuotteen perusteellinen testaus on lähes mahdotonta, koska sen suorittaminen vaatisi kohtuuttoman määrän aikaa. Testaus on siten otoskeskeistä, jossa testausotokseen pitää tunnistaa tärkeimmät ja riskialttiimmat kohteet. Tällaista testauksen lähestymistapaa kutsutaan riskipohjaiseksi testaamiseksi. (ISO/IEC 29119)

Virheenkorjaus tai järjestelmään muusta syystä kohdistuva muutos on myös testattava. Virheenkorjaus voi aiheuttaa muutoksia järjestelmän toimintaan jossain toisessa järjestelmän osassa, kuin missä alkuperäinen virhe havaittiin (ISO/IEC 29119). Regressiotestauksella pyritään varmistamaan tehdyn muutoksen oikeellisuus, eli ettei järjestelmän toiminta ole muuttunut esimerkiksi virheenkorjauksen jälkeen.

ISO/IEC 25010 -standardi luokittelee ohjelmistotuotteen laadun kahdeksaan laatuattribuuttiin, jotka jakautuvat edelleen useampiin aliattribuutteihin. Kuvassa 9 ohjelmistotuotteen laatumalli on esitelty ISO/IEC 25010 -standardin mukaisesti. Ohjelmistotuotteen laatu pitäisi rakentaa ohjelmiston kehitysvaiheessa. Testaus on tukiprosessi, jonka avulla laatua on helppo mitata ja varmentaa, muttei rakentaa.



Kuva 9. Ohjelmistotuotteen laatumalli (ISO/IEC 25010).

Ohjelmistotuotteen laadun arvioimiseksi ei ole tarkoituksenmukaista testata kaikkia laatuattribuutteja. Tarkoitus on, että attribuuteista valitaan valmistettavalle ohjelmistolle tärkeimmät tai ohjelmistoon olennaisesti liittyvät attribuutit testattaviksi. Ohjelmiston ominaisuuksien testaus suoritetaan testaamalla toiminnallisuutta, sekä ei-toiminnallisuutta valittujen laatuattribuuttien validoimiseksi ja verifioimiseksi.

3.2 Testausprosessi

Kehitteillä olevassa ohjelmistotestauksen standardissa, ISO/IEC 29119, ohjelmistotestaus on määritelty seuraavasti: Prosessi, jossa analysoidaan välituotetta (esimerkiksi dokumentti, suunnitelma tai vaatimusmäärittely), ohjelmiston osaa tai koko ohjelmistoa, jotta siitä löydetään erot toteutuksen ja vaatimuksien välillä, ja jossa osoitetaan, että järjestelmä soveltuu käyttötarkoitukseensa. Standardissa määritelty testausprosessimalli jakautuu kolmeen kerrokseen: testausprosessiin, joka määrittelee testauspolitiikan ja testausstrategian, testauksen hallintaan ja testauksen tasoihin.

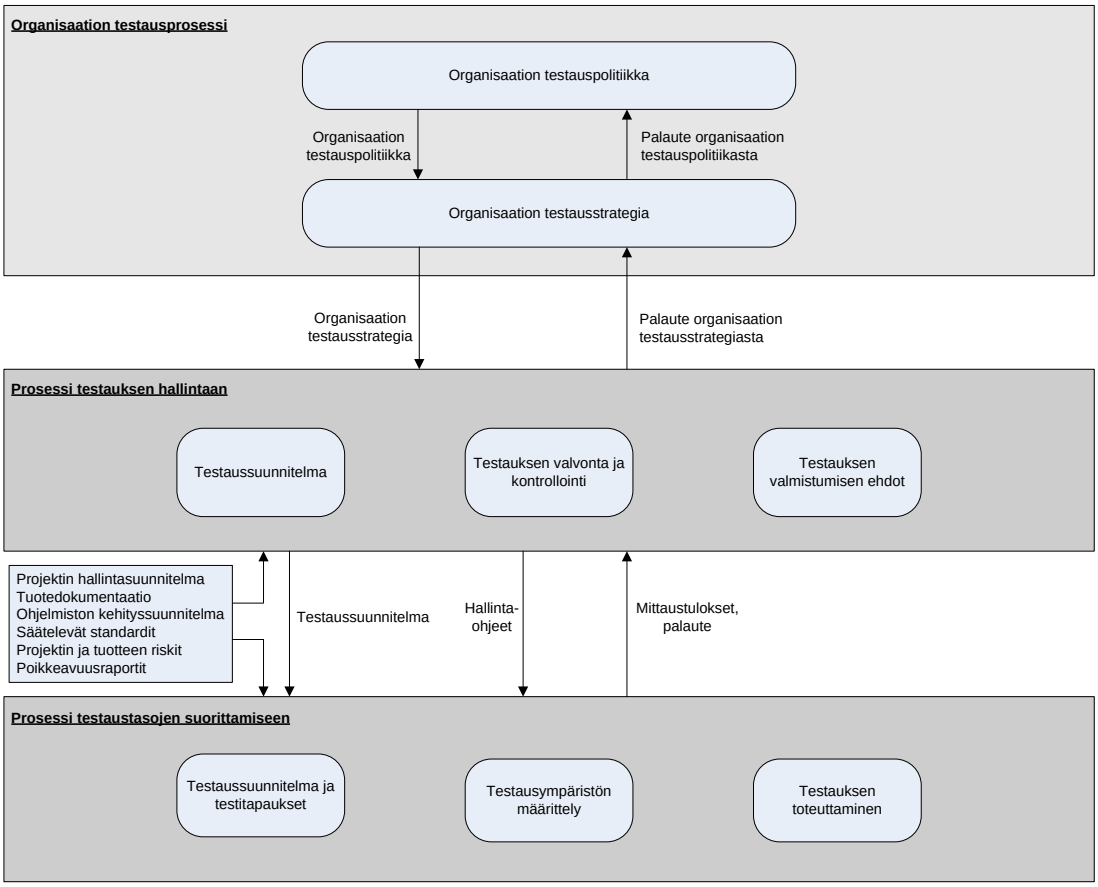
Ylin kerros määrittelee prosessit organisaation testauspolitiikan ja testausstrategian luomiseksi ja ylläpitämiseksi. Testauspolitiikka määrittelee koko organisaation suh-

tautumisen ohjelmistotestaukseen. Testauspolitiikka konkretisoituu lyhyenä ja suurpiirteisenä johtotason dokumenttina, joka määrittelee johdon roolin testauksessa. Testauspolitiikassa määritellään mitä testausta organisaatiossa tehdään, mutta ei sitä, miten testaus suoritetaan. Ylin kerros sisältää myös puitteet testauspolitiikan, testausstrategian sekä projektien testauksen hallinnan vakiinnuttamiseen, arvioimiseen ja kehittämiseen. (ISO/IEC 29119)

Testausstrategiassa selvennetään miten testaus suoritetaan määrittelemällä testauskäytännöt ja uudelleenkäytettävät testausohjeet. Testausstrategia perustuu testauspolitiikkaan ja se on tarkoitettu organisatoriselle tasolle, joten sitä käytetään kaikissa organisaation projekteissa. Tämä mahdollistaa yhdenmukaisen menettelyn ohjelmistotestaukseen koko organisaatiossa. (ISO/IEC 29119)

Testauksen hallinnan prosessit määritellään prosessimallin toisella kerroksella. Testauksen hallinnan prosessia voidaan soveltaa projektien testauksen hallintaan, eri testausvaiheiden hallintaan sekä testaustyyppien hallintaan. Eri hallintaprosessit voivat olla yhteydessä toisiinsa, koska esimerkiksi järjestelmätestauksen hallinta voi perustua projektin testaushallintaprosessin tuottamaan testaussuunnitelmaan. Prosessit käsittävät testauksen suunnittelun, tarkkailun, hallinnan ja raportoinnin. (ISO/IEC 29119)

Viimeinen kerros määrittelee prosessit testaustasojen ja testaustyyppien suorittamiseksi projektissa. Testaustasoja ovat esimerkiksi yksikkö-, integrointi-, järjestelmä- ja hyväksymistestaus. Testaustyyppejä ovat esimerkiksi suorituskyky-, turvallisuus- ja toiminnallisuustestaus. Testitapausten analysointi, suunnittelu ja toteutus tehdään alimmalla prosessimallin kerroksella testauksen hallintaprosessin valvonnassa. Testauksen prosessimalli on kokonaisuudessaan havainnollistettu kuvassa 10. Kuvassa 10 havainnollistuu myös eri prosessien väliset suhteet. Ylempiä kerroksia pitää voida kehittää, jos alla olevissa kerroksissa havaitaan virheitä tai ongelmia prosesseihin liittyen. (ISO/IEC 29119)



Kuva 10. Testauksen prosessimalli (ISO/IEC 29119).

Testauksen prosessimalli on suunniteltu kattamaan myös ketterät menetelmät. Esimerkiksi Roosmalen (2007) on huomioinut, että testauspolitiikan ja testausstrategian määrittelystä on hyötyä Scrum-projektin testauksen suunnittelussa ja suorittamisessa.

3.3 Testauksen organisointi

Yksinkertaisesti määriteltynä, testausorganisaatio on ne resurssit, jotka on omistettu vain testaustoimenpiteitä varten. Mitä suurempi organisaatio on kyseessä, sitä enemmän resursseja tarvitaan myös testaukseen. Testauksen on siten kasvettava organisaation mukana. Kit (1995) on esitellyt seitsemän vaihtoehtoista mallia testauksen organisointiin. Mallit on esitelty taulukossa 2, jossa on huomioitu myös niiden hyödyt ja haitat. Mallien voidaan nähdä soveltuvan organisaation koon suhteen siten,

että ensimmäinen malli sopii pienelle organisaatiolle ja seuraavat mallit soveltuvat kokoajan suuremmalle organisaatiolle.

Taulukko 2. Testauksen organisointimallit (Kit, 1995).

Organisointimalli	Hyödyt	Haitat
1) Testaus on kehittäjien vastuulla	Luonnollinen ratkaisu testauksen toteuttamiseksi	Kehittäjät ovat sokeita omille virheilleen, testaus on tehotonta
2) Testaus on yksikön vastuulla,	Ratkaisee 1. mallin ongelman	Kehittäjillä ei ole aikaa tehdä ja omaksua molempia töitä ja kehittäjien testausosaaminen on rajallista
3) Testausvastuu erillisellä testaajalla	Ratkaisee 2. mallin ongelman, kehittäjällä ei ole kahta työtä Edelleen vain yksi työryhmä	Tuotekehityspäälliköllä liikaa vastuuta, onko tuotekehityspäälliköllä aikaa ohjeistaa kehitystyön lisäksi testaustoimenpiteet?
4) Testausorganisaatio laadunvarmistuksen osana	Ratkaisee 3. mallin hallintoongelmat	Yhteistyöongelmia kehittäjien ja testaajien välillä, tuotekehitys ei ole yksin vastuussa laadusta ja testaustehtävistä voi tulla toisarvoisia laadunvarmistustehtävien rinnalla
5) Testausorganisaatio tuotekehityksen osana	Ratkaisee 3. mallin hallintoongelmat ja mahdollisesti 4. mallin yhteistyöongelmat	Tuotekehityspäällikön hoidettava myös testauspäällikön tehtävät
6) Keskitetty testausorganisaatio	Ratkaisee 5. mallin ongelman Luo urapolun testauspäällikölle	Testausorganisaation toiminta on riippuvainen johdosta. Yksilöiden ja projektien tasolla voi ilmetä yhteistyöongelmia Yhtenäisten testauskäytäntöjen puute eri yksiköissä
7) Keskitetty testausorganisaatio erillisellä asiantuntijaryhmällä	Ratkaisee 6. mallin yhtenäisten testauskäytäntöjen puutteen	Testausorganisaation toiminta on riippuvainen johdosta. Yksilöiden ja projektien tasolla voi ilmetä yhteistyöongelmia

Ensimmäiset kolme Kitin mallia soveltuvat pienten yritysten käyttöön. Ensimmäinen malli on tyypillisin lähtökohta testauksen liittämiseksi kehitystyöhön. Mallin ongelmaksi koituu se, että ihminen on sokea omille virheilleen, joten testaus on tehotonta. Toisessa mallissa tämä haitta estetään siten, että kehittäjät testaavat toistensa tuotos-

ta. Kehittäjien aika ja omaksumiskyky on kuitenkin rajallista, jolloin usein toissijaista työtehtävää, tässä tapauksessa testausta, tehdään jos ehditään.

Kolmannessa mallissa testausvastuu on siirretty erilliselle testaajalle, jolloin kehittäjät voivat keskittyä omaan työhönsä. Erillinen testaaja luo kuitenkin paineita tuotekehityspäällikölle, sillä tuotekehityspäällikön on nyt ohjattava kehitystyön lisäksi testaustoimenpiteitä. Kit (1995) epäilee tuotekehityspäällikön kykyä tarjota testaajan käyttöön testausstandardit, -prosessit, -politiikan, -työkalut ja -koulutuksen.

Mallissa neljä testaus on kokonaan eriytetty kehittäjistä ja se on laadunvarmistuksen osana. Tuotekehityspäällikön paine helpottuu, mutta testaustehtävät voivat jäädä laadunvarmistustehtävien rinnalla toisarvoisiksi. Lisäksi kehittäjien ja testaajien yhteistyössä voi esiintyä ongelmia.

Viidennessä mallissa testaukselle on muodostettu oma testausorganisaatio, jolla voidaan estää yhteistyöongelmia sekä helpottaa hallinto-ongelmia. Tuotekehityspäällikkö on edelleen vastuussa testauksen hallitsemisesta mutta hänellä on kuitenkin apunaan testausryhmän vetäjä.

Keskitetyn testausorganisaation avulla kuudes malli ratkaisee viidennen mallin ongelmat. Samalla se luo urapolun testauspäällikölle, koska keskitetyllä testausorganisaatiolla on erillinen testauspäällikkö. Seitsemäs malli on hieman kehittyneempi muoto kuudennesta mallista, sillä asiantuntijaryhmä takaa yhtenäiset testauskäytännöt, jos organisaation toiminta hajautuu useisiin osastoihin. Kuudennelle ja seitsemännelle mallille yhteiset ongelmat ovat, että testausorganisaation olemassa olo on riippuvainen johdosta sekä kehittäjien ja testaajien yhteistyössä voi esiintyä ongelmia.

4 TUTKIMUSMENETELMÄ

Tässä työssä käytetään tutkimusmenetelmänä aineistopohjaista menetelmää (grounded theory), josta käytetään myös nimiä ankkuroitu teoria sekä grounded-teoria. Aineistopohjainen menetelmä on laadullisen tutkimuksen laji, jossa pyritään luomaan uutta teoriaa havainnoimalla tutkittavasta kohteesta kerättyä aineistoa. Laadullisen tutkimuksen lisäksi tässä luvussa on käsitelty tilastollinen eli kvantitatiivinen tutkimus sekä toimintatutkimus vertailupohjan luomiseksi.

4.1 Laadullinen tutkimus

Laadullisen, eli kvalitatiivisen, tutkimuksen lähtökohtana on todellisen elämän kuvaaminen. Todellisuudessa tapahtumat liittyvät toisiinsa eri tavoin ja usein monimuotoisesti, joten tutkittavaa kohdetta ei voi mielivaltaisesti jakaa osiin. Tapahtumien välille muodostuu suhteita, jolloin tapahtumat voivat muokkautua toistensa vaikutuksista. Laadullisen tutkimuksen tarkoituksena on tutkia kohdetta mahdollisimman kokonaisvaltaisesti. (Hirsjärvi et al., 2009)

Strauss ja Corbin (1990) määrittelevät, että laadullinen tutkimus on mitä tahansa tutkimusta, jossa havaintoja ei tehdä tilastollisia tai määrällisiä toimenpiteitä käyttäen. Laadullinen tutkimus on analyyttinen ja ei-matemaattinen prosessi, jonka tulokset ovat peräisin tutkittavasta kohteesta kerätystä aineistosta. Laadullista tutkimusta tekevältä tutkijalta vaaditaan teoreettista herkkyyttä (theoretical sensitivy), joka tarkoittaa tutkijan kykyä löytää ja tulkita aineistosta löytyviä ilmiöitä omaan kokemukseen ja tietoon nojautuen, mutta samalla säilyttäen tutkimuksen objektiivisuuden. (Strauss ja Corbin, 1990)

Laadullista tutkimusta käytetään yleensä silloin, kun tutkittava ongelma on luonteeltaan sellainen, että sitä ei voi tutkia tilastollisesti (Strauss ja Crobin, 1990). Tällaiset

ongelmat ovat usein monimutkaisia ja niissä käsitellään tilastoitavien tietojen sijaan laadullista sekä merkityksellistä tietoa (Hirsjärvi et al., 2009). Esimerkiksi ihmisen motivaation, kommunikoinnin ja ymmärtämisen tutkiminen on edellä mainitun mukainen ongelma. Seamanin (1999) mukaan ohjelmistotuotannossa tekniset ja inhimilliset näkökulmat yhdistyvät, jolloin tutkimuksissa tarvitaan sekä tilastollisia että laadullisia tutkimusmenetelmiä.

Laadullisten tutkimusmenetelmien etuna on, että ne pakottavat tutkijan perehtymään tutkittavaan ongelmaan syvällisesti. Laadulliselle tutkimukselle on tyypillistä, että tutkimussuunnitelman annetaan muotoutua tutkimuksen edetessä olosuhteiden muuttuessa. Tutkimuksen tarkoituksena on paljastaa odottamattomia asioita, minkä vuoksi aineistoa on tarkasteltava yksityiskohtaisesti. Laadullisten tutkimusten tuloksia pidetään usein monipuolisempina sekä informatiivisempina, mutta ei yhtä täsmällisinä, kuin tilastollisten tutkimusten tuloksia. (Seaman, 1999; Hirsjärvi et al., 2009)

4.1.1 Aineistonkeruu laadullisessa tutkimuksessa

Aineistonkeruun perusmenetelmät ovat kysely, haastattelu, havainnointi sekä dokumentit. Perusmenetelmät ovat yhteisiä eri tutkimustyypeille, toiset menetelmät sopivat tietenkin paremmin eri tutkimuksiin. Laadullisessa tutkimuksessa aineiston hankinnassa suositaan menetelmiä, joissa tutkittavien näkökulmat ja eleet pääsevät esille. Tällaisia menetelmiä ovat esimerkiksi haastattelut. Laadullisessa tutkimuksessa juuri haastattelu on yksi aineistonkeruun päämenetelmistä. (Hirsjärvi et al., 2009)

Haastattelun etuna on, että se on varsin joustava aineistonkeruumenetelmä. Aineistonkeruu on säädeltävissä haastattelun aikana, jolloin kysymyksiä voidaan tarkentaa tai muokata tilanteen ja vastaajan mukaan. Lisäksi haastattelijalla on mahdollisuus tulkita vastaajan käyttäytymistä sekä vastauksia keskustelun aikana. Haastattelun haittapuolena on, että haastateltava voi kokea haastattelutilanteen uhkana itselleen tai haastateltava voi antaa valheellista tietoa, jotta tutkimuksen kohde näyttäisi parem-

malta kuin mitä se oikeasti on. Haastattelija voi vaikuttaa tilanteeseen yrittämällä tehdä siitä vähemmän uhkaavan, sekä havainnoida haastateltavan käyttäytymistä huomatakseen mahdolliset totuutta venyttävät vastaukset. (Hirsjärvi et al., 2009)

Tutkimushaastattelut jakautuvat kolmeen ryhmään: strukturoitu, avoin ja teemahaastattelu. Strukturoidussa haastattelussa edetään lomakkeen avulla. Esitettävät kysymykset ovat ennakkoon laadittu lomakkeeseen ja ne esitetään ennalta määritetyssä järjestyksessä. Strukturoidun haastattelun tekijällä tavoite on usein selkeä, eli tiedetään millaista tietoa etsitään (Seaman, 1999). Avoimessa haastattelussa selvitetään haastateltavan ajatuksia, mielipiteitä, tunteita ja käsityksiä ilman johdatusta. Haastattelun muodoista avoin haastattelu on lähimpänä keskustelua. Teemahaastattelu on strukturoidun ja avoimen haastattelun välimuoto. Haastattelussa esillä olevat aihepiirit on valmiiksi määritelty, mutta haastattelun eteneminen ei noudata tiettyä järjestystä. Toisin kuin strukturoidussa haastattelussa, teemahaastattelussa kysymysten tarkkoja muotoja ei ole määritelty valmiiksi. (Hirsjärvi et al., 2009)

Laadulliselle tutkimukselle on tyypillistä, että tutkittava kohdejoukko valitaan tarkoituksenmukaisesti. On tärkeää, että aineisto kerätään todellisissa ja luonnollisissa tilanteissa. Tapauksia käsitellään yksittäisinä ja tiettyihin olosuhteisiin sidottuina, joten tutkimustuloksia ei voi yleistää muihin tutkimukseen liittymättömiin tapauksiin. (Hirsjärvi et al., 2009)

4.1.2 Aineistopohjainen menetelmä

Glaser ja Strauss (1967) esittelivät aineistopohjaisen menetelmän ensimmäisen kerran vuonna 1967. Nykyinen käsitys menetelmästä on jakautunut Glaserin (1992) sekä Straussin ja Corbinin (1990) näkemyksiin. Tässä työssä noudatetaan Straussin ja Corbinin lähestymistapaa, johtuen rajoitetusta vapaudesta havainnoida tutkittavia organisaatioita. Glaserin lähestymistavassa havainnoinnin vapaus on keskeinen vaatimus.

Aineistopohjaisessa menetelmässä jonkin ilmiön teoria johdetaan sitä käsittelevästä tutkimuksesta. Aineistopohjaisessa menetelmässä ei muodosteta testattavia hypoteeseja vaan teorian annetaan muodostua ja kehittyä aineiston pohjalta. Teoria verifioidaan järjestelmällisen aineistonkeruun ja aineiston analysoinnin avulla. Aineistopohjaisen tutkimuksen tuloksena on todellisuudessa tapahtuvan ilmiön teoreettinen muoto. (Strauss ja Corbin, 1990)

Kuten kaikissa muissakin tutkimusmenetelmissä, myös aineistopohjaisessa menetelmässä ensimmäinen askel on tutkimusongelman ja -kysymyksen määrittely. Tutkimuskysymys ohjaa tutkimusta oikeaan suuntaan tutkimusta tehtäessä sekä auttaa tutkimusongelman rajaamisessa. Aineistopohjaisessa menetelmässä tutkimuskysymystä jalostetaan ja tarkennetaan tutkimuksen aikana aineiston analysoinnin myötä (Strauss ja Corbin, 1990).

Straussin ja Corbinin (1990) mukaan aineiston analysointi tehdään kolmessa vaiheessa, nämä vaiheet ovat avoin, aksiaalinen ja selektiivinen koodaus. Koodauksen tarkoituksena on ensin hajauttaa aineistossa oleva tieto osiin, ja sen jälkeen liittää hajautetut osat loogisesti takaisin yhteen, samalla muodostaen uutta tietoa ja teoriaa aineistosta. Koodauksen vaiheita ei välttämättä suoriteta järjestyksessä vaan niissä voidaan liikkua edestakaisin, varsinkin avointa ja aksiaalista koodausta tehdään usein rinnakkain.

Analysoinnin ensimmäinen vaihe on avoin koodaus. Avoimessa koodauksessa aineisto hajautetaan osiin, joista etsitään ilmiöitä ja käsitteitä. Ilmiöt sekä niihin liittyvät käsitteet nimetään ja samankaltaiset tai toisiinsa liittyvät ilmiöt luokitellaan kategorioihin. Kategorioilla on ominaisuuksia ja ominaispiirteitä, jotka pitää tunnistaa, koska ne muodostavat perustan kategorioiden yhteen liittämiseksi koodauksen seuraavassa vaiheessa. Avoin koodaus alkaa jo aineistonkeruun alussa, sillä analysoinnin avulla aineistonkeruuta voidaan ohjata tutkimuksen edetessä. (Strauss ja Corbin, 1990)

Aksiaalisessa koodauksessa hajautettu aineisto kootaan takaisin yhteen muodostamalla suhteita kategorioiden välille. Kategorioista voi tulla alakategorioita, jos niiden havaitaan liittyvän isompaan kokonaisuuteen. Aksiaalinen koodaus keskittyy kategorioiden tarkentamiseen tutkimalla, missä olosuhteissa ja kontekstissa kategoria eli ilmiö esiintyy. Olosuhteiden tutkimiseen kuuluu myös havainnointi, miten ilmiötä on pyritty hallitsemaan ja mitä seurauksia ilmiön hallinnalla on ollut. Avointa ja aksiaalista koodausta tehdään kunnes saavutetaan teoreettinen saturaatio, eli aineistosta ei löydy enää uusia kategorioita tai käsitteitä. (Strauss ja Corbin, 1990)

Aineistopohjaisen menetelmän viimeisessä analysoinnin vaiheessa, selektiivisessä koodauksessa, tunnistetuista kategorioista valitaan ydinkategoria vertaamalla sitä järjestelmällisesti muihin kategorioihin ja kategorioiden välisiin suhteisiin. Ydinkategoria kuvaa tutkimuksen kannalta keskeisintä ilmiötä. Selektiivisessä koodauksessa ydinkategorian ja muiden kategorioiden välille luodaan suhteet. Ydinkategoriaksi voi valikoitua myös useampi kategoria, jos yksi kategoria ei riitä kuvaamaan tutkimuksen keskeisintä ilmiötä. Kun kategoriat ovat yhdistetty ydinkategorian ympärille, mallista alkaa hahmottua muodostuva teoria. Teoria ankkuroidaan eli validoidaan testaamalla sitä aineistoa vastaan. (Strauss ja Corbin, 1990)

4.1.3 Tutkimustulosten luotettavuus ja pätevyys

Tutkimustulosten arviointi on tärkeä toimenpide, sillä sen avulla varmennetaan tutkimuksen luotettavuus ja pätevyys. Luotettavuudella tarkoitetaan tässä tapauksessa sitä, että tutkimustulokset eivät ole sattumanvaraisia. Pätevyydellä arvioidaan tutkimuksen validiutta, eli kuinka hyvin tutkimusmenetelmä soveltuu tutkimusongelman ratkaisemiseen.

Hirsjärven et al. (2009) mukaan sattumanvaraisia tuloksia voidaan eliminoida käyttämällä useampaa tutkijaa, tai tutkimalla kohdetta toistuvasti. Tutkimuksen pätevyyttä voidaan varmentaa käyttämällä useampia tutkimusmenetelmiä yhdessä. Laadulli-

sessa tutkimuksessa luotettavuutta lisää yksityiskohtainen selostus tutkimuksen toteutuksesta ja miten tutkimuksessa on päädytty saatuihin tuloksiin. Laadullisen tutkimuksen pätevyyttä voidaan arvioida siten, kuinka hyvin tulos sopii selostuksessa kerrottuihin olosuhteisiin ja suoritettuihin toimenpiteisiin. (Hirsjärvi et al., 2009)

Straussin ja Corbinin (1990) mukaan aineistopohjainen tutkimusmenetelmä pitäisi arvioida kolmen osa-alueen mukaisesti. Ensimmäinen arvioinnin kohde on kerätty aineisto, miten luotettava ja uskottava se on. Toinen kohde on tutkimusprosessi, eli miten aineistopohjaisen menetelmän vaiheet on suoritettu ja onko niissä tehdyt päätelmät oikeellisia. Viimeiseksi arvioidaan, miten hyvin menetelmän tulos (teoria) on ankkuroitu aineistoon. Menetelmässä on huomioitu aineiston vääristyminen, ja vääristymien estämiseen on olemassa toimenpiteitä.

4.2 Vaihtoehtoiset tutkimusmenetelmät

Seuraavassa on lyhyesti esitelty tilastollinen tutkimus sekä toimintatutkimus vaihtoehtoisina tutkimusmenetelminä laadulliselle tutkimukselle. Tilastollista ja laadullista tutkimusta pidetään usein toistensa vastakohtina, mutta niitä voidaan käyttää täydentämään toinen toistaan. Hirsjärven et al. (2009) sanoin: ”Numerot perustuvat merkityksiä sisältävään käsitteellistämiseen, ja merkitystä sisältäviä käsitteellisiä ilmiöitä voidaan ilmaista numeroin”.

4.2.1 Tilastollinen tutkimus

Tilastollinen tutkimus perustuu määrälliseen ja numeeriseen mittaamiseen. Täten havaintoaineisto pyritään keräämään siten, että se voidaan esittää numeerisesti. Numeerinen aineisto muutetaan tilastolliseen muotoon, kuten taulukoiksi, joista analysoinnin avulla tehdään loogisia johtopäätöksiä aiempiin teorioihin nojautuen. (Hirsjärvi et al., 2009)

Tilastollisen tutkimuksen taustalla on ajatus siitä, että todellisuus koostuu tosiasioista, joita voidaan havainnoida objektiivisesti. Tilastolliselle tutkimukselle tyypillisiä piirteitä ovat esimerkiksi hypoteesien määrittely sekä tutkittavien kohteiden suunnitelmallinen valinta. Tutkimukselle määritellään perusjoukko, josta valitaan kattava otos tutkimuksen kohteiksi. Otos tulee valita siten, että tutkimustulokset ovat päteviä määritellylle perusjoukolle. (Hirsjärvi et al., 2009)

4.2.2 Toimintatutkimus

Toimintatutkimuksessa tutkimusta tehdään toiminnan keskipisteessä. Tutkimuskohteena olevaa toimintaa ei havainnoida objektiivisesti etäältä, vaan tutkija on mukana toiminnassa, tehden yhteistyötä toimintaa normaalisti suorittavien henkilöiden kanssa. Toimintatutkimuksen prosessi koostuu iteratiivisesti suoritettavista vaiheista: suunnittelu, toimenpiteen suorittaminen ja toimenpiteen arviointi. Toiminnassa suoritettujen toimenpiteiden arvioinnin avulla suunnitellaan seuraavat toimenpiteet. Tutkimusprosessin tarkoituksena on kehittää tutkittavaa toimintaa tai järjestelmää kokeilemalla ja kehittämällä siinä suoritettavia toimenpiteitä. Tavoitteena on muuttaa tutkittavaa toimintaa tehden siitä tehokkaampaa ja samalla muodostaa kohteesta tutkimustietoa. (Coghlan ja Brannick, 2005)

Toiminnan kehittämisen lisäksi toimintatutkimus on myös lähestymistapa ongelmanratkaisuun, jossa kokeilemalla ja faktojen etsinnällä pyritään löytämään ratkaisu johonkin käytännön ongelmaan. Toimintatutkimus edellyttää tutkijoiden ja tutkittavaa toimintaa suorittavien henkilöiden yhteistyötä, koska toisin kuin perinteisessä tutkimuksessa, toimintatutkimuksessa nämä henkilöt eivät ole tutkimuksen kohteita tai koehenkilöitä. Ratkaisun löytymisen lisäksi toimintatutkimuksen tarkoituksena on olla opettavainen prosessi sekä tuottaa tutkimustietoa ja uutta käytännönläheistä teoriaa. (Coghlan ja Brannick, 2005)

4.3 Tutkimusmenetelmän valinta

Aineistopohjainen menetelmä valittiin työn tutkimusmenetelmäksi, koska ohjelmistotuotanto tutkimuksen kohteena on monimuotoinen. Monimuotoisuus perustuu siihen, että ihmiset ja ihmisten välinen vuorovaikutus on keskeinen osa ohjelmistotuotantoa. Seamanin (1999) mukaan ihmisen käytös on ilmiönä niin monimuotoinen, että sitä pitää tutkia laadullisen tutkimusmenetelmän avulla. Tilastolliset tutkimusmenetelmät eivät välttämättä riitä kuvaamaan ja selittämään ihmisen käytöstä.

Strauss ja Corbinin (1990) näkemystä aineistopohjaisesta menetelmästä käytettiin, koska siinä korostetaan havaintojen kategorioimista ja loogista päättelyä. Glaserin (1992) näkemyksessä tutkijan pitäisi seurata tutkimuksen kohdetta sen luonnollisessa ympäristössä ja tehdä havaintoja samanaikaisesti. Tällaista lähestymistapaa ei tämän työn rajoissa nähty hyödylliseksi. Toimintatutkimuksen kaltaistakaan lähestymistapaa ei voitu käyttää, koska tutkimuksen tekijöillä ei ollut siihen vaadittavia vaikutusmahdollisuuksia tutkimukseen osallistuneissa organisaatioissa.

5 TUTKIMUKSEN TOTEUTUS

Tässä luvussa käsitellään tämän tutkimuksen toteutusta. Esitellään tutkimuksen kohteet sekä kerrotaan miten aineisto on kerätty ja analysoitu. Lisäksi luvun lopussa käsitellään tutkimuksen luotettavuuteen liittyviä seikkoja.

5.1 Tutkimuskohteiden esittely ja aineistonkeruu

Tutkimuskohteina toimivat organisaatioyksiköt. ISO/IEC 15504-1 -standardissa on määritelty, että organisaatioyksikkö on tutkimuksen kohteena oleva organisaation osa. Organisaatioyksikkö koostuu yhdenmukaisissa konteksteissa suoritettavista prosesseista, joilla on yhtenevät liiketoiminnalliset tavoitteet. Tyypillisesti organisaatioyksikkö on osa suuresta yrityksestä, mutta pienemmissä organisaatioissa se voi kattaa koko organisaation. Organisaatioyksikkö-termiä käytetään siksi, että saadaan vertailukelpoista tietoa yrityksistä ilman, että niiden vaihtelevat koot vaikuttavat tutkimukseen.

Tässä tutkimuksessa käytetty aineisto kerättiin haastattelemalla organisaatioyksiköiden edustajia valmiiksi määriteltyjen kysymyksien avulla. Haastattelut tehtiin kolmessa kierroksessa, joista ensimmäisellä ja kolmannella kierroksella haastateltiin vain tutkimuksen fokusryhmää, joka koostui 12 organisaatioyksiköstä. Toisella kierroksella käytettiin laajennettua otosta, jolloin haastateltiin yhteensä 31 organisaatioyksikköä. Kaikilla haastatelluilla organisaatioyksiköillä oli käytössä vertailukelpoiset prosessit.

Fokusryhmän organisaatioyksiköt tulevat erilaisista Suomessa toimivista organisaatiotyypeistä, joiden liiketoiminta-alueet, organisaation koot ja toimialueet vaihtelevat. Yhteistä fokusryhmän organisaatioyksiköille on kuitenkin se, että niiden pääteh-

tävänä tai -prosessina on ohjelmistokehitys. Fokusryhmä on esitelty kokonaisuudessaan taulukossa 3.

Taulukko 3. Haastatellut fokusryhmän organisaatioyksiköt.

Organisaatioyksikkö	Liiketoiminta	Yrityksen koko / Toimialue
A	Tuotannonohjausohjelmistojen tuottaminen sekä elektroniikan valmistus	Pieni / Kansallinen
B	Internet-palvelujen kehitys ja konsultointi	Pieni / Kansallinen
C	Logistiikkaohjelmistojen kehitys	Suuri / Kansallinen
D	ICT-konsultointi	Pieni / Kansallinen
E	Turvallisuus- ja logistiikkajärjestelmien kehitys	Keskikokoinen / Kansallinen
F	Laivaliikenteen ohjelmistojärjestelmien kehitys	Keskikokoinen / Kansainvälinen
G	Taloushallinto-ohjelmistojen kehitys	Suuri / Kansallinen
H	Tuotannonohjausohjelmistojen tuottaminen sekä logistiikkapalvelujen tarjoaja	Keskikokoinen / Kansainvälinen
I	PK-yritysten ja maatalouden ICT-palvelujen tarjoaja	Pieni / Kansallinen
J	Mallinnusohjelmistojen kehitys	Suuri / Kansainvälinen
K	ICT-kehitys ja konsultointi	Suuri / Kansainvälinen
L	Taloushallinto-ohjelmistojen kehitys	Suuri / Kansainvälinen

Kuten edellä mainittiin, haastattelut tehtiin kolmessa kierroksessa. Jokaisella kierroksella organisaatioyksikköä edustava haastateltava henkilö vaihtui. Fokusryhmän osalta, jokaisesta organisaatioyksiköstä haastateltiin samassa projektissa toimiva suunnittelija, projekti- tai testauspäällikkö sekä testaaja. Toiseen haastattelukierrokseen fokusryhmän lisäksi valituissa organisaatioyksiköissä haastateltiin vain projekti- tai testauspäällikköä.

Ensimmäisellä kierroksella haastateltiin valmiiden kysymysten avulla ohjelmiston arkkitehtuurista vastaavaa suunnittelijaa tai suunnittelusta vastaavaa ohjelmoijaa, jos

erillistä suunnittelijaa ei ollut. Kysymykset käsittelivät aiheita, kuten tuotantomenetelmiä, testauskäytäntöjä, standardeja, testausautomaatiota, ulkoistusta, laatua sekä asiakassuhteita. Ensimmäisen kierroksen kysymykset ovat kokonaisuudessaan liitteessä 1.

Toisella kierroksella haastattelussa käytettiin kyselylomaketta, jonka avulla kerättiin laadullisen aineiston lisäksi tilastollista tietoa yhteensä 31 organisaatioyksiköstä. Haastateltavat henkilöt olivat projekti- tai testauspäälliköitä. Päällikkötason henkilöt valittiin, koska heidän nähtiin ymmärtävän organisaation toimintaa ja ohjelmistoprosesseja laajemmin kuin vain projektikohtaisesti. Kyselylomakkeen aiheet pysyivät pääpiirteittäin samoina kuin ensimmäisellä kierroksella, mutta vastaukset kerättiin tilastolliseen muotoon Likert-asteikkoa käyttäen. Toisen kierroksen kysymykset ovat kokonaisuudessaan liitteessä 2.

Kolmannella kierroksella palattiin haastattelemaan vain fokusryhmää. Tällä kierroksella haastateltavana oli testaaja tai henkilö, joka teki testausta oman työn ohella. Kysymykset kohdistuivat pääasiassa ohjelmistotestaukseen liittyviin aiheisiin, kuten testausmenetelmiin, testausstrategioihin, testausautomaatioon, testausstandardeihin, testauksen ulkoistamiseen sekä ketterien menetelmien vaikutuksista testaukseen. Kolmannen kierroksen kysymykset ovat kokonaisuudessaan liitteessä 3. Haastattelukierrokset ovat tiivistetysti esitetty alla olevassa taulukossa 4.

Taulukko 4. Haastattelukierrokset.

Kierros	**Haastateltavien organisaatioyksiköiden määrä**	**Haastateltavan rooli**
1) Strukturoitu haastattelu	12 (fokusryhmä)	Suunnittelija tai suunnittelusta vastaava ohjelmoija
2) Strukturoitu haastattelu ja kyselylomake	31 (sisältäen fokusryhmän)	Projekti- tai testauspäällikkö
3) Strukturoitu haastattelu	12 (fokusryhmä)	Testaaja tai testauksesta vastaava ohjelmoija

Kaikki MASTO-hankkeessa suoritetut haastattelut nauhoitettiin laadullista tutkimusta varten. Haastatteluissa käytettiin strukturoitua haastattelumenetelmää, jossa kysymykset suunnitellaan valmiiksi haastattelutilannetta varten. Kahden ensimmäisen kierroksen jälkeen kysymyksiä kehitettiin seuraavaa kierrosta varten, jos siihen oli tarvetta. Toisen haastattelukierroksen aikana kerätty tilastollinen aineisto on käsitelty Luomansuun (2009) opinnäytetyössä.

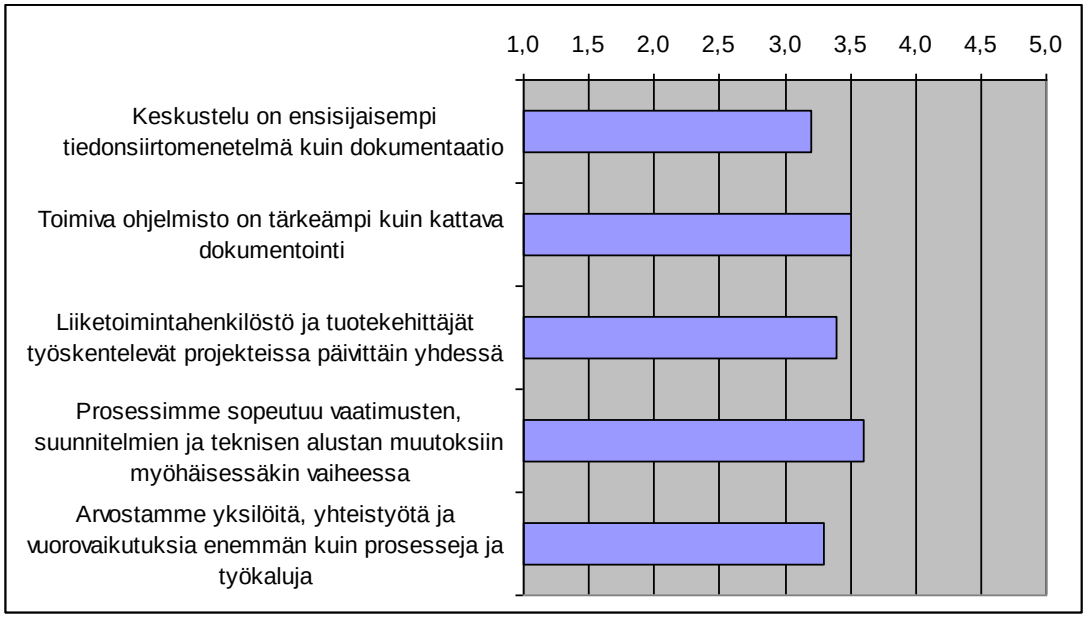
5.2 Yleistä tietoa tutkimukseen osallistuneista organisaatioyksiköistä

Tutkimukseen osallistuneiden organisaatioiden koko vaihteli 350 000 henkilön yrityksestä muutaman henkilön yritykseen. Ohjelmistokehittäjien ja -testaajien määrässä esiintyi siten myös paljon vaihtelua organisaatioyksiköiden välillä. Toisissa yksiköissä ei ollut yhtään omaa ohjelmistokehittäjää, jolloin niissä käytettiin kolmannen osapuolen ohjelmistokehittäjiä. Ketterien menetelmien osuus organisaatioyksiköissä käytetyistä tuotantomenetelmistä vaihteli 10 ja 100 prosentin välillä, mediaanin ollessa 30 prosenttia. Testausresursseja organisaatioyksiköillä oli käytössä pääsääntöisesti kohtuullisen hyvin, mutta mediaanin mukaan ohjelmiston kehitysajasta testaukseen käytettiin vain 25 prosenttia. Joissain organisaatioyksiköissä testaukseen ei varattu aikaa lainkaan. Organisaatioyksiköihin liittyvät havainnot on esitelty taulukossa 5.

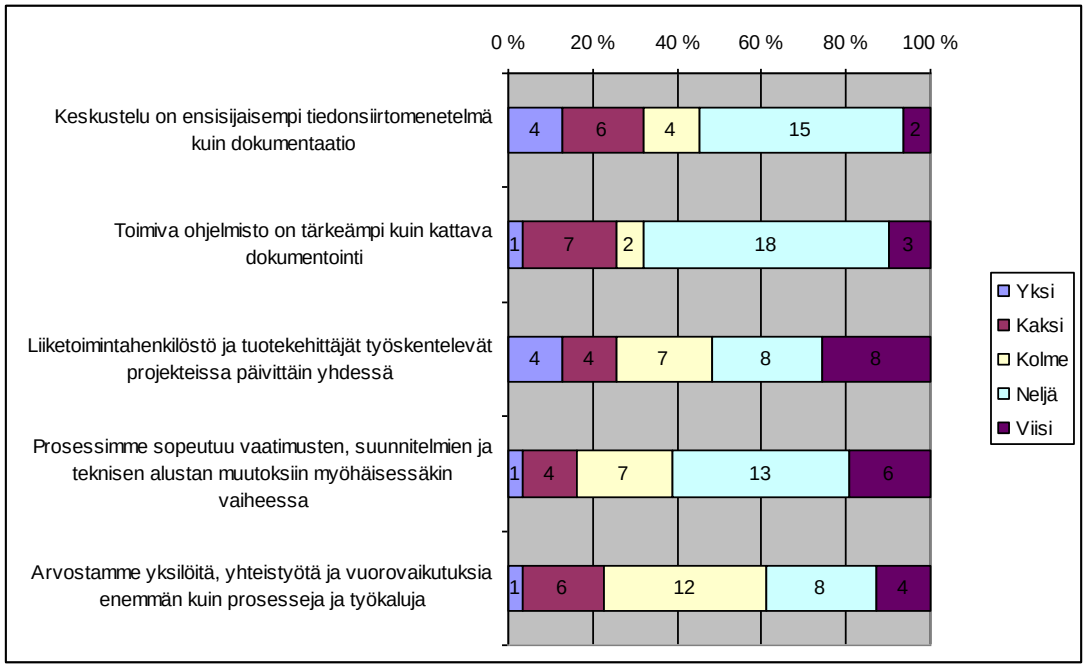
Taulukko 5. Haastatellut organisaatioyksiköt.

	Maksimi	Minimi	Mediaani
Koko yrityksen henkilöstön lukumäärä	350 000	4	315
Ohjelmistokehittäjien ja -testaajien lukumäärä organisaatioyksikössä	600	0	30
Testausautomaation osuus prosentteina (%)	90	0	10
Ketterien menetelmien prosentuaalinen osuus käytetyistä tuotantomenetelmistä (%)	100	0	30
Testausresurssien määrä suhteessa tarpeeseen (%)	100	10	75
Kuinka suuri osa kehitysajasta käytetään testaukseen (%)	70	0	25

Toisella haastattelukierroksella organisaatioyksiköiden edustajilta kysyttiin muun muassa tuotantomenetelmiin, ohjelmistotestaukseen, asiakkaan osallistumiseen ja ISO/IEC 29119 -standardiin liittyviä kysymyksiä. Haastateltavat arvioivat lomakkeessa esitettyjä väittämiä 5-portaisen Likert-asteikon mukaisesti, jossa vastaus 1 tarkoitti, että vastaaja oli täysin eri mieltä ja vastaus 5 tarkoitti, että vastaaja oli täysin samaa mieltä. Tuotantomenetelmiin liittyneet väittämät muotoiltiin ketterien menetelmien perusarvojen ja periaatteiden mukaisesti. Alla olevassa kuvassa 11 on esitelty kysymyksiin saatujen vastausten keskiarvot ja kuvassa 12 on vastausten jakaumat.



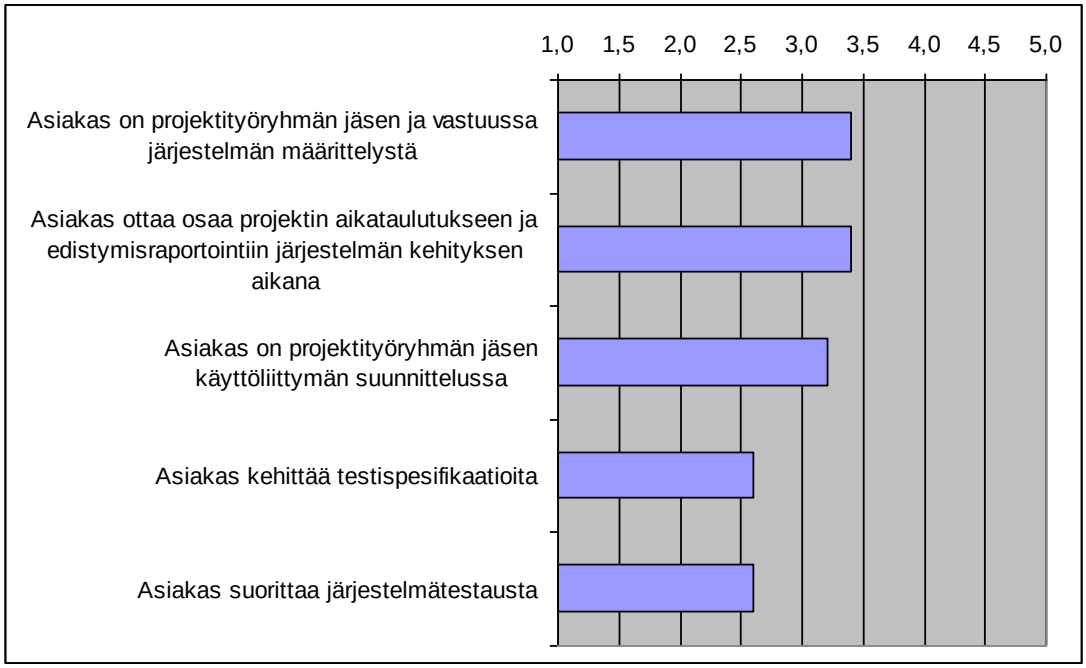
Kuva 11. Vastausten keskiarvo ohjelmistokehitysmenetelmiin liittyneistä kysymyksistä.



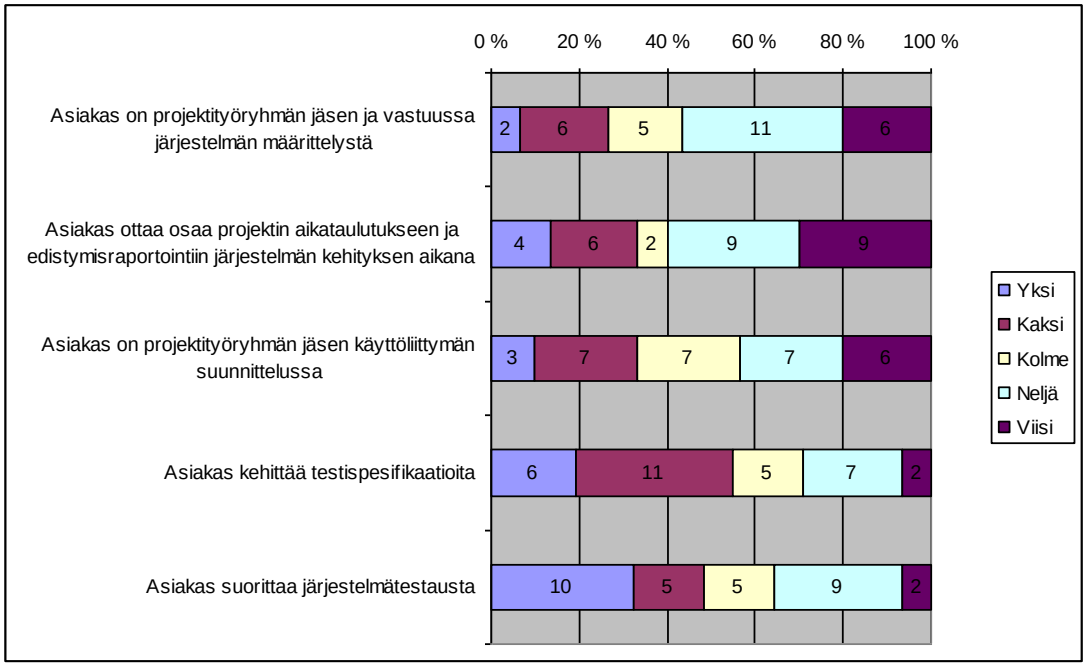
Kuva 12. Vastausten jakauma ohjelmistokehitysmenetelmiin liittyneistä kysymyksistä.

Kuten kuvista 11 ja 12 voi nähdä, vastaukset painottuvat ketterien menetelmien suuntaan, vaikka organisaatioyksiköt kertoivat käyttävänsä enemmän suunnitelmalähtöisiä tuotantomenetelmiä. Nämä vastaukset tukevat Nandhakumarin ja Avisonin (1999) väitettä, että käytännössä ohjelmistokehitystä ei tehdä orjallisesti perinteistä tuotantomenetelmää noudattaen, vaan perinteiset tuotantomenetelmät tarjoavat vain pakollisen kulissin kontrolloidusta ohjelmistokehityksestä. Nandhakumarin ja Avisonin (1999) mukaan käytännössä ohjelmistokehitys tehdään tilanteen vaatimalla tavalla. Abrahamssonin (2002, s. 93) mukaan Nandhakumarin ja Avisonin havaitsemat todelliset työtavat muistuttavat hyvin läheisesti ketterille menetelmille tyypillisiä työtapoja.

Kun tuotantomenetelmiin liittyvien kysymysten vastausten keskiarvot eivät painotu selkeästi kumpaankaan äärilaitaan, tulos voidaan tulkita myös siten, että organisaatioyksiköissä käytetään tuotantomenetelmänä hybridimallia. Kuvassa 13 on asiakkaan osallistumiseen liittyvien vastausten keskiarvoja ja kuvassa 14 vastausten jakauma. Vastaukset tukevat hybridimallitulkintaa. Asiakas on selkeästi paremmin osallistunut vaatimusmäärittelyosuuteen sekä käyttöliittymän suunnitteluun, kuin testauksessa suoritettaviin toimenpiteisiin. Tulokset antavat viitteitä siitä, että eri ohjelmistokehityksen vaiheissa käytetään mahdollisesti erilaisia työtapoja tai jopa kokonaan eri tuotantomenetelmiä.

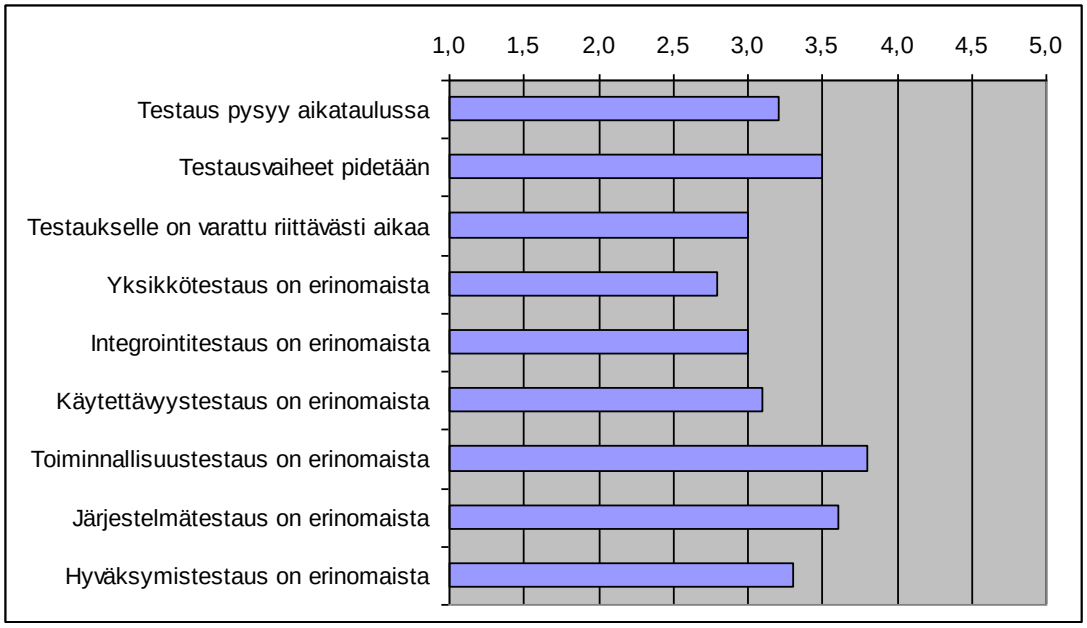


Kuva 13. Vastausten keskiarvo asiakkaan osallistumiseen liittyneistä kysymyksistä.

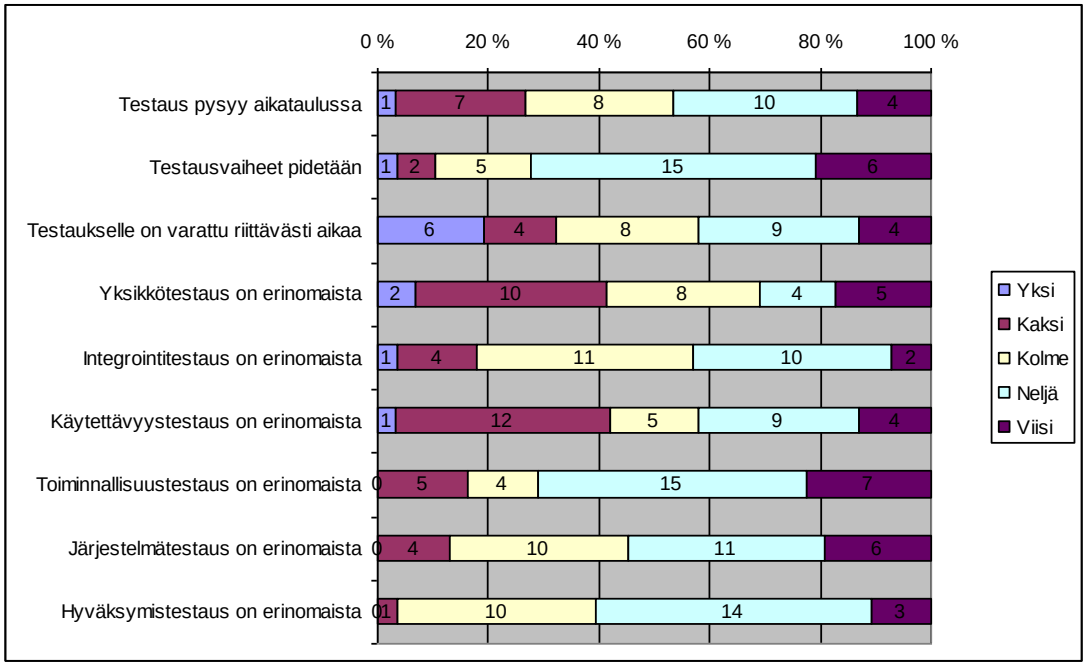


Kuva 14. Vastausten jakauma asiakkaan osallistumiseen liittyneistä kysymyksistä.

Kuvassa 15 on organisaatioyksiköiden ohjelmistotestaukseen liittyneiden kysymysten vastausten keskiarvoja ja kuvassa 16 vastausten jakaumat. Kaiken kaikkiaan ohjelmistotestaus on kohtuullisen hyvällä tasolla vastausten perusteella. Testaus suoritetaan aikataulussa eikä testausvaiheita jätetä tekemättä, vaikka testaukselle varattu aika ei riittäisi. Testauksen tasoista yksikkö- ja integrointitestaus olivat heikoimmin suoritetut testausvaiheet. Varsinkin yksikkötestauksessa organisaatioyksiköillä on selvästi parannettavaa. Järjestelmä- ja toiminnallisuustestaus tehtiin organisaatioyksiköissä huolellisimmin. Vaikuttaisi siis siltä, että ohjelmistokehityksen alussa testaustoimenpiteitä ei suoriteta yhtä huolellisesti kuin kehityksen lopussa. Tämä tulos viittaa myös siihen, että testaus tehdään suunnitelmalähtöisen menetelmän mukaisesti ohjelmistokehitysprosessin viimeisenä vaiheena.

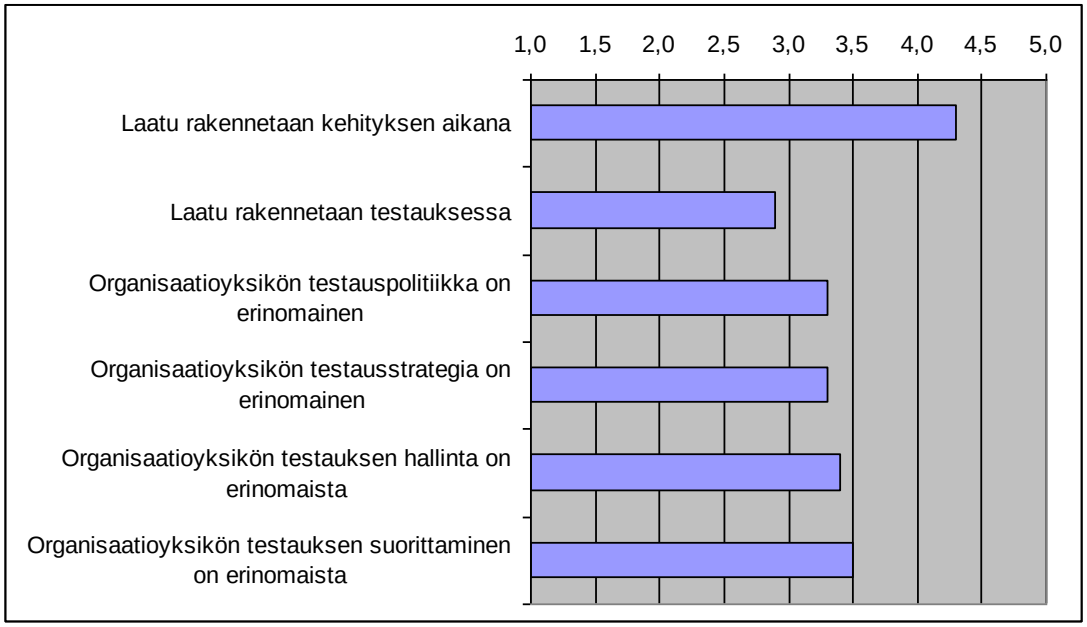


Kuva 15. Vastausten keskiarvo ohjelmistotestaukseen liittyneistä kysymyksistä.

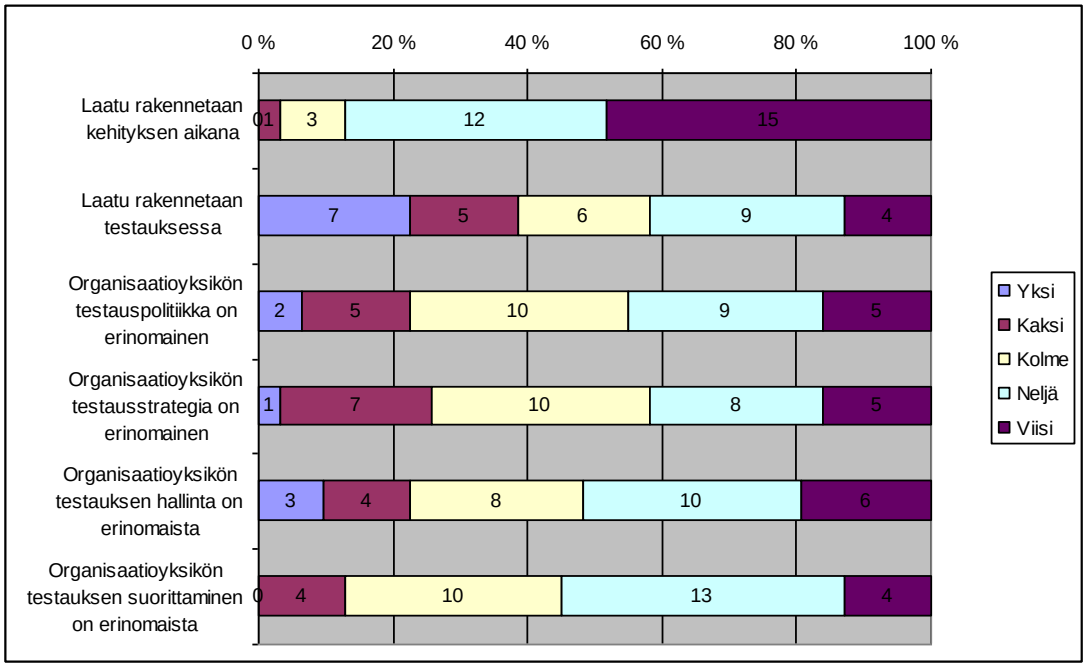


Kuva 16. Vastausten jakauma ohjelmistotestaukseen liittyneistä kysymyksistä.

Organisaatioyksiköitä pyydettiin arvioimaan myös ISO/IEC 29119 -ohjelmistotestausstandardin mukaisesti muotoiltuja väittämiä. Vastausten keskiarvot ovat kuvassa 17 ja jakaumat kuvassa 18. On kuitenkin huomioitava, että ISO/IEC 29119 -standardi ei ole vielä valmis. Vastaajien mielestä tuotteen laatu rakennetaan pääasiassa kehityksen aikana, mutta testauksellakin on osuutensa laadun rakentamisessa. ISO/IEC 29119 -standardissa esitetyn testauksen prosessimallin eri kerrokset ovat vastausten mukaan kohtuullisen hyvin organisoitu.



Kuva 17. Vastausten keskiarvo ISO/IEC 29119 -standardiin liittyneistä kysymyksistä.



Kuva 18. Vastausten jakauma ISO/IEC 29119 -standardiin liittyneistä kysymyksistä.

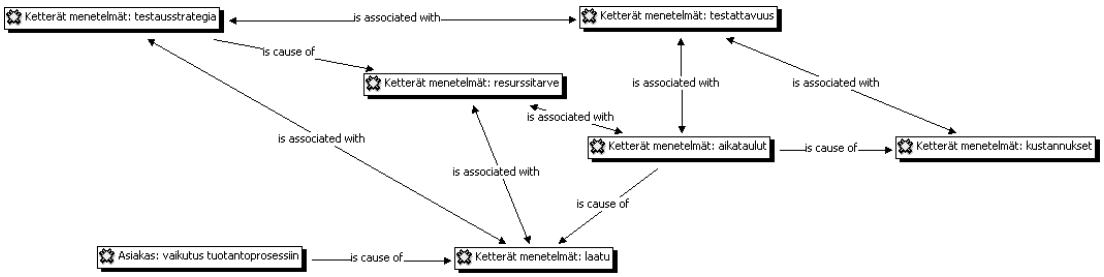
Edellä esitettyjen tilastojen mukaan organisaatioyksiköiden ohjelmistotestaus on kohtuullisen hyvin katettu testaustasojen suorittamisesta testauksen hallinnointiin. Organisaatioyksiköt käyttävät myös tuotantomenetelminä jonkin verran ketteriä menetelmiä, joten tämän työn tutkimuskysymykselle löytyy perusteita. Työn seuraavassa vaiheessa aineisto analysoidaan aineistopohjaisen menetelmän avulla.

5.3 Aineiston analysoinnin vaiheet

Aineistopohjaisen menetelmän mukaisesti tämän työn ensimmäinen vaihe oli avoin koodaus. Seamannin (1999) mukaan ohjelmistotuotantoa tutkiessa tutkimuskysymyksestä johdettuja niin kutsuttuja siemenkategorioita pitäisi muodostaa ennen koodauksen aloittamista. Työn alussa näitä siemenkategorioita olivat:

- ketterät menetelmät,
- ongelmat,
- testausstrategia,
- testausresurssit ja
- asiakas.

Avoin koodaus tehdään usein rinnakkaisesti aksiaalisen koodauksen kanssa, eikä tämä työ ole poikkeus. Rinnakkaisesti suoritettuina vaiheet täydentävät toisiaan, sillä esimerkiksi kategorioiden suhteita muodostaessa syntyy uusia ideoita avoimeen koodaukseen. Koodausvaiheissa käytettiin apuna Atlas.ti-ohjelmistoa (2005). Kuvassa 19. on esitelty kuvankaappauksen avulla hahmottelua aksiaalisesta koodauksesta eli kategorioiden yhdistämisestä ja niiden suhteiden määrityksestä, Atlas.ti-ohjelmistossa.



Kuva 19. Hahmotelma aksiaalisessa koodauksessa tehtävästä ryhmittelystä.

Aksiaalisen ja myöhemmin selektiivisen koodauksen aikana ketteriin menetelmiin liittyvien havaintojen kokoavaksi ydinkategoriaksi muodostui testausstrategia. Tämän havainnon avulla työstettiin havaintomatriisi (Liite 4.), jossa nähdään organisaatioyksiköiden nykytilat tutkimukselle oleellisten kategorioiden osalta. Tämän havaintomatriisin avulla pyritään myöhemmin esittämään hypoteeseja ketterien menetelmien käyttöönottoon liittyen.

Kaiken kaikkiaan aineistopohjaisen menetelmän käyttäminen ohjelmistotuotannossa vaati soveltamista, koska projektit ja organisaatiot ovat hyvin erilaisia. Organisaatioiden käytännöt muotoutuvat ajan kuluessa vastaamaan kyseisen organisaation tarpeita, joten kunkin organisaation omien käytäntöjen vertaileminen on haastavaa.

5.4 Luotettavuuden varmistaminen

Tämän tutkimuksen luotettavuus pyrittiin varmistamaan käyttämällä useampaa tutkijaa. Tämän työn tekijä ei osallistunut aineistonkeruuseen, vaan aineistonkeruun tekivät eri henkilöt, jotka toimivat MASTO-hankkeessa. Haastattelijoina toimi yhteensä kolme eri henkilöä. Lisäksi avoimeen koodaukseen osallistui tämän työn tekijän lisäksi toinen tutkija. Havaintojen varmentamisessa käytettiin tekijän lisäksi kahden projektiin osallistuneen tutkijan näkökulmia.

Aineiston monipuolisuutta ja luotettavuutta lisättiin siten, että eri haastattelukierroksilla haastateltiin suunnittelijaa, projekti- tai testauspäällikköä sekä testaajaa. Näin aiheisiin saatiin kolme eri näkökulmaa projekteja toteuttavien henkilöiden kesken.

6 ORGANISAATIOYKSIKÖIDEN TUOTANTOMENETELMÄT

Suurin osa 12 fokusryhmän organisaatioyksiköstä käytti suunnitelmalähtöisiä menetelmiä. Ketterää menetelmää lähes jokaisessa projektissa tietoisesti käyttäviä organisaatioyksiköitä oli vain yksi. Ketterään siirtyviä tai siirtymistä harkitsevia organisaatioyksiköitä oli useita. Organisaatioyksiköissä, joissa ketterä menetelmä ei ollut käytössä, miellettiin, että ketterissä menetelmissä testaus on luontevammin mukana projekteissa. Ketterien menetelmien etuna pidettiin myös nopeampaa kehitysvauhtia.

> ”Mitä se [ketterät menetelmät] tekee, se niinku luontevasti ohjaa siihen että se testaus tapahtuu, et se ei ole vaan projektin päätteessä tapahtuva asia että siihen se tietysti pyrkii vaikuttamaan, mut ehkä se niinku luontevasti jakaa sitä sitten paremmin, paremmin koko projektin kestolle.” (Organisaatioyksikkö B, suunnittelija)

> ”Siit on ihan hyviä kokemuksia ollut, että testaus pääsee mukaan mahollisimman aikaisessa vaiheessa, koska testauksen ongelmahan on se just, että jos sen järjestelmän toteutus myöhästyy, ja sit sen pitää olla tuotannossa tiettyyn aikaan, niin sithän se vaan supistaa aina sitä testausaikaa ja tavallaan testaajat joutuu puristamaan sitten sen. Niin, ketterässä menetelmässähän se sit jakautuu se testauksen osuus, niin mun mielestä se kuulostaa huomattavasti paremmalta ja järkevämmältä.” (Organisaatioyksikkö G, testaaja)

Toisaalta ketteriä menetelmiä pidettiin epävarmoina menetelminä, jotka eivät sovellu laajojen kokonaisuuksien hallintaan. Lisäksi ketteristä menetelmistä oli huonoja kokemuksia, jos sitä oli käytetty projekteissa, joiden lopputulos ei ollut selkeä.

> ”Se oli niinku hyvin ikävä kokemus et se projekti meni aika hankalaks et siinä sit leivottiin ja hierottiin ja tehtiin uudestaan ja muutettiin viikottain. Et eli puuttu se maali täysin siitä hommasta.” (Organisaatioysikkö B, suunnittelija)

Vesiputousmallia pidettiin siten varmempana ja helpommin hallittavissa olevana kehitysmallina. Aikataulujen koettiin olevan paremmin arvioitavissa, kun käytettiin vesiputousmallia.

> ”Me ollaan koettu näin, että tää vesiputousmalli on jotenkin helpompi pitää hanskassa, kun meillä on selkee tämmönen päämäärä, johon pyritään. Me pystytään aikatauluttamaan suunnittelut, designit, toteutus ja testaus ja pyritään tietenki pysymään näissä aikatauluissa.” (Organisaatioyksikkö J, suunnittelija)

Ketteriin menetelmiin siirtyminen koettiin myös haastavana silloin, kun alkuperäisessä kehitystyössä dokumentit olivat tärkeässä asemassa tai kehitysprosessi noudatti tiukasti virallista tai sisäistä standardia. Tällöin muutoksen nähtiin vaikuttavan myös liiketoimintamalliin asti. Dokumentoinnin puute aiheutti myös epäilyksen tai pelon mahdollisista vaikeistakin riitatilanteista asiakkaan kanssa määrittelyjen vuoksi.

6.1 Organisaatioyksikkö A

Organisaatioyksikkö A:n suunnittelija ja testaaja eivät tunteneet termiä ketterät menetelmät lainkaan. Organisaatio on aikoinaan valmistanut tuotteen suunnitelmalähtöisesti ja jatkokehityksessä on voitu käyttää ketterille menetelmille tyypillisiä piirteitä tiedostamatta. Projektipäällikön mukaan organisaatioyksikön kehitysprosessi on suunniteltu tukemaan organisaation toimintaa ja joustamaan tarpeen vaatiessa. Kehitysprosessissa on nähtävissä selkeitä yhtäläisyyksiä ketteriin menetelmiin.

> ”Mutta semmosessa iteratiivisessa luupissa. Suhteessa nyt tommoseen niinku suunnitelmalliseen testaukseen nähden niin, kyllä tuontyyppinen on käytännössä niinku se. Kyllä jotain muutoksia tulee, vois sanoa, että se on meiän ehkä vahvuus, pystytään (-) asiakkaan muuttuviin tarpeisiin kesken projektin. Asiakas ei tarkkaan itsekään tiedä, mitä se tahtoo siinä vaiheessa kun se lähtee projektiin, se luulee tietävänsä. Mut sit siinä paljastuu, että eihän tää näin toimikaan tämä asia.” (Organisaatioyksikkö A, projektipäällikkö)

> ”Eli me luodaan kyllä hyvinkin paljon asioita sinne niinkun asiakaslähtösesti. Et heiltä tulee palautetta, me kirjataan ne tänne ja sit katotaan et mihin versioon ne tehdään.” (Organisaatioyksikkö A, testaaja)

6.2 Organisaatioyksikkö B

Organisaatioyksikkö B:n suunnittelija oli sitä mieltä, että organisaatioyksikön pääsääntöinen kehitysmalli noudattaa vesiputousmallia. Projektien ei kuitenkaan anneta laajentua niin suuriksi, että niiden hallinta olisi vaikeaa. Suunnitelmalähtöisen menetelmän käytön taustalla oli suunnittelijan mukaan se, että organisaatiossa halutaan tehdä määrittelyt, joiden avulla projektit etenevät. Joissakin projekteissa organisaatio voi kuitenkin käyttää ketteriä menetelmiä, mutta vain jos organisaatiolla on hyvä suhde asiakkaan kanssa.

> ”Meillä nyt niinku tyypillisesti mennään vesiputousmallimaisesti eniten että yhdistelemme näitä siten että projektit pyritään osittamaan järkeviks kokonaisuusiks ettei niinku liian suuria vesiputouksia rakenneta. Pyritään kuitenkin tekemään määritykset, jotka toimii siinä jonkinlaisena pohjana et minkä kans mennään. On sit tiettyjä asiakkaita, joiden kans on nyt päästy hyvään suhteeseen et voidaan mennä myös ketterillä, mut kyllä mun mielestä se vesiputousmalli ja se nimenomaan ositettuna muutamiin osatoimituksiin on ehkä se pääsääntönen malli.” (Organisaatioyksikkö B, suunnittelija)

6.3 Organisaatioyksikkö C

Organisaatioyksikkö C:n edustajien mukaan organisaatiossa käytetään ketterää menetelmää, Scrumia, lähes jokaisessa projektissa. Vain yksi projekti noudatti perinteistä vesiputousmallia haastatteluajankohtana. Testaajan mukaan organisaatioyksikkö käyttää Scrum-menetelmää, koska sen avulla on saavutettu hyviä tuloksia. Testaajan näkökulmasta positiivisesti vaikuttava tekijä oli myös testauksen aikainen kytkentä kehitystyöhön.

> “Well, the reason that this one project that’s using the traditional method isn’t using SCRUM is because it is being implemented by external consultants who are Germany, and there’s too many external and off-site groups, to actually integrate well into a SCRUM environment, so that’s why they’re not using SCRUM. Everyone else is using SCRUM because we have had quite good results with it.” (Organisaatioyksikkö C, testaaja)

6.4 Organisaatioyksikkö D

Organisaatioyksikkö D:n suunnittelija kertoi, että organisaation kehitysmenetelmä on vaihdellut ajan, resurssien ja henkilöstön myötä. Suunnittelijan mukaan suunnitelmalähtöinen menetelmä vaati kunnollisen rahoituksen toimiakseen. Haastatteluajankohtana heillä oli kuitenkin käytössä suunnitelmalähtöinen menetelmä, koska suunnittelijan mukaan sen avulla saadaan varmempia tuloksia kuin ketterällä menetelmällä.

> ”No kyl mä oon joutunu myöntämään tosiasiat, että perinteinen menetelmä on kuitenkin se, vaik se on hidasta ja kallista, se tuottaa varmempia tuloksia.” (Organisaatioyksikkö D, suunnittelija)

6.5 Organisaatioyksikkö E

Organisaatioyksikkö E:n edustajien mukaan organisaatiossa käytetään suunnitelmalähtöistä kehitysmenetelmää, koska organisaatio valmistaa reaaliaikaista ja kriittistä tuotetta. Kolmannella kierroksella haastatellun suunnittelijan mielestä tuotteen kriittisyyden ja vähäisen suunnittelun vuoksi ketterien menetelmien käyttö ei ole suotavaa. Ensimmäisellä kierroksella haastatellun suunnittelijan näkökulma oli, että ketterät menetelmät saattaisivat häiritä kehitystyön alkuperäistä tarkoitusta.

> ”Sanosin, et ehkä enemmän suunnittelulähtöstä kyllä. Ettei oo nähty tarpeelliseks ruveta niitä [ketteriä menetelmiä], niinku itsetarkotuksena viemään ja sit taas toisaalta tuote, mikä on tämmönen hyvin reaaliaikanen tuote, niin siinä meillä täytyy olla aika suuri hallinta siihen ohjelmistoon. Ja nyt jos käytettäs jotain tällasia ketteriä menetelmiä, niin siinä itse asiassa saattas sit monin osin vähän hämärtyä se prosessi, mitä ollaan tekemässä.” (Organisaatioyksikkö E, suunnittelija)

> ”Sitä ei oikein hirveen hyvin tässä pysty käyttämään. Tässä pittää tietää ensin että mitä järjestelmä tekee ja mihin se vaikuttaa ja, ettei tuu sitten semmosta, että no eipä tullut kukkaan ajatelleeks tuota asiaa, että koodataan se nyt sitten ensimmäisen onnettomuuen jälkeen.” (Organisaatioyksikkö E, suunnittelija)

6.6 Organisaatioyksikkö F

Organisaatioyksikkö F:ssä käytetään tai ollaan siirtymässä käyttämään kehityksessä ketterää menetelmää, mutta käytännössä iteratiivinen prosessi heillä venyi ja lopulta muistutti enemmän vesiputousmallia. Organisaatioyksiköllä on edelleen siirtymävaihe meneillään.

> ”Mä haluaisin mennä lähemmäs semmosta, jatkuvaa iteratiivista, testataan, kehitetään, testataan, kehitetään. Nyt tehdään enemminkin sillä tavalla, että kehitetään tosi paljon ja sit testataan intensiivisesti, korjataan intensiivisesti. Ja tää ei mun mielest sovi hyvin tämmöseen ketterämpään ajatusmalliin, vaan se ehkä on enemmän tämmönen vanha vesiputousmallin mukainen tapa toimia.” (Organisaatioyksikkö F, suunnittelija)

6.7 Organisaatioyksikkö G

Organisaatioyksikkö G:n edustajien mukaan organisaatiossa on varhaismuotoisista ketteristä menetelmistä siirrytty suunnitelmalähtöiseen malliin. Organisaatioyksikössä noudatetaan itse muodostettua kehitysmallia, joka perustuu suunnitelmalähtöisyyteen mutta sopeutuu tuotteeseen. Organisaatioyksikön hankejohtajan mukaan he ovat tutkimassa ketteriä menetelmiä ja harkitsevat kannattaako niihin siirtyä. Toimialasta johtuen, organisaation tuotantomenetelmässä dokumentointi tulee ottaa huomioon.

> ”Meillä on ohjelmistosuunnittelun tai ohjelmistototeutukseen tällä hetkellä käytössä tämmönen oma kehitysprosessimalli. Suunnittelulähtöinen, se on perinteinen. Siittä se etenee, siit on määrittely, tekniseen suunnitteluun, toteutukseen, testaussuunnittelu testaukseen, ja tuote-, tuotantoonottoon, mahollisesti pilotointi ja tämmöset siinä välissä.” (Organisaatioyksikkö G, suunnittelija)

6.8 Organisaatioyksikkö H

Organisaatioyksikkö H noudattaa suunnitelmalähtöistä menetelmää. Projektipäällikön mukaan organisaatio pyrkii kehittämään tuotetta suunnitelmalähtöisesti, mutta projektien aikataulujen vuoksi organisaatiolta vaaditaan usein nopeampaa lähestymistapaa kehitystyöhön. Suunnittelijan mukaan varsinaisia ketteriä menetelmiä ei voida käyttää, koska organisaatioyksikön asiakkaat eivät halua osallistua iteratiiviseen kehitysmalliin.

> ”Me pyritään kehittämään tuotetta silleen suunnitelmallisesti tuotenäkökulmasta. Mut sitten taas meidän liiketoiminta tehdään pitkälle projekteissa, joissa tehdään sitten paljon tämmöstä hyvinkin kireellä aikataululla, lyhyen tähtäimen kehitystä. Ja nää on aika usein ristiriidassa keskenään. Mut, sanosko että, että suunnitelmallisesti tehtäs 50 prossaa ja loput tehdään sitten, silleen niinkun lennossa.” (Organisaatioyksikkö H, projektipäällikkö)

> ”Varsinaisesti, ihan niin älyttömän ketteriä menetelmiä me ei käytetä ja se on hyvin pitkälti meidän niinkun asiakasrajapinnan sanelema elikä on vaikea löytää sellasia asiakkaita, jotka haluis lähtee tällaseen sykliseen malliin ja tarkennettavaan malliin mukaan, vaan hyvin pitkälti tilaukset on paketti ja sille joku pakettihinta.” (Organisaatioyksikkö H, suunnittelija)

6.9 Organisaatioyksikkö I

Organisaatioyksikkö I:n suunnittelijan ja testaajan mukaan organisaation kehitysprosessi sopeutuu tuotteeseen, jolloin prosessi muokkautuu projektin koon mukaisesti. Prosessin on oltava joustava, koska projektit vaihtelevat pienistä ja nopeista projekteista isoihin ja suunnittelua vaativiin projekteihin. Testaaja kertoi myös, että kehitystyö ja suunnittelu jätetään usein kehittäjän vastuulle, koska tarkkoja suunnitelmia ei ehditä tekemään kiireisten aikataulujen vuoksi. Toisaalta organisaatioyksikön projektipäällikkö kertoi, että organisaation prosessi on vaiheittain etenevä ja vaiheiden valmistumisehdot on ohjeistettu kaikille.

”Menetelmäpuolella on sillä lailla että, emme ehkä ole tiedostaneet että mitä menetelmää nuista käytettäs mutta ketterät menetelmätkin on aika lähellä sitä mitä ajoittain se, projekti toimii että. Ei varsinaista ketterää menetelmää mutta jotaki piirteitä sieltä löytyy. Riippuu aina projektista vähä että täällä, meillä henkilöityy että osa haluaa tehä sillä lailla että ne tekkee pitkiä, pitkiä muutosprosesseja ja osa koittaa tehä sitte lyhyempiä.” (Organisaatioyksikkö I, suunnittelija)

6.10 Organisaatioyksikkö J

Organisaatioyksikkö J:n edustajat kertoivat, että organisaation tuotantomenetelmänä käytetään suunnitelmalähtöistä menetelmää ja sitä noudatetaan tarkasti. Lisäksi prosessiin on liitetty virallinen standardi ja siten dokumentointia pidetään tärkeänä toimenpiteenä. Suunnittelijan mukaan vesiputousmalli on helpommin hallittavissa, kuin ketterä tuotantomenetelmä.

”Kyllä meil pyritään varmaan enemmän tähän perinteiseen, vesiputousmalliin enemmän, se oikeestaan ny riippuu aika lailla siitä projektista mitä tehdään, että meillähän, tehdään kaks ohjelmistoversioo vuodessa ja pyritään siihen, että ensin on suunnitteluvaihe ja sitten on tää design ja sitten, toi ite toteutusvaihe ja sitten pyritään ajottamaan se testaus sit sinne ihan loppupäähän. Yleensä menee niin kun, prosessikaaviossa meillä on silleen, että menee toteutus ja testaus jonkun verran limittäin, mutta kyl se todellisuudessa käy usein niin, että se toteutus saattaa venyä aika lähelle sitä itse ohjelmiston valmistumisajankohtaa, jollon testaus ja nää menee hyvin pitkälle niinku käsi kädessä.” (Organisaatioyksikkö J, suunnittelija)

6.11 Organisaatioyksikkö K

Organisaatioyksikkö K:n edustajat kertoivat yksimielisesti, että organisaatiossa käytetään suunnitelmalähtöistä menetelmää. Kukin heistä oli ollut mukana ketterän menetelmän käyttöönottokokeilussa ja he olivat saaneet koulutusta ketteristä menetelmistä. Suunnittelijan mielestä ketterä menetelmä ei sopinut organisaation monien

pienten ja rinnakkaisten projektien kehitysmenetelmäksi. Myös testaajalla oli huono kokemus Scrum-menetelmän käyttöönottokokeilusta.

> ”Tällä hetkellä on ton vesiputousmallin tyylinen käytössä et ketterii menetelmii ollaan kokeiltu jonkun verran, et se on ilmeisesti niinkun vasta tulossa meille. Ollu jotain semmosia pilottiprojekteja missä ollaan näitä ketterii menetelmii käytetty. Mä en tiedä se [ketterä menetelmä] ei hirveen hyvin ainakaan niinkun sovi tohon meijän rytmiin, kun meillä mennään aika nopeesti et monia projekteja tai silleen lyhyitä projekteja on paljon ja nopeet aikataulut.” (Organisaatioyksikkö K, suunnittelija)

6.12 Organisaatioyksikkö L

Organisaatioyksikkö L:n edustajien mukaan organisaatiossa käytetään suunnitelmalähtöistä menetelmää, mutta jossa voi esiintyä myös ketterille menetelmille tyypillisiä piirteitä. Suunnittelijan ja testaajan mukaan ketterien menetelmien piirteitä on omaksuttu kehitysprosessissa mutta virallisesti organisaatioyksikössä ketteriä menetelmiä ei käytetä.

> ”Pelkästään, minun ymmärryksen mukaan se on suunnittelulähtöistä, ehkä siihen ollaan pikkuhiljaa niin kun menossa siihen että on niin kun, on tehty yks osio kerrallaan, mutta tällä hetkellä mun mielestä se on niin ku sitä suunnittelulähtöstä.” (Organisaatioyksikkö L, testaaja)

7 LAADULLISEN ANALYYSIN TULOKSET

Seuraavassa on esitelty tutkimuksen tulokset. Ensin esitellään analyysin aikana aineistosta havaittujen ilmiöiden perusteella luodut kategoriat. Sen jälkeen on esitelty havaintojen ja kategorioiden perusteella muodostetut hypoteesit.

7.1 Kategorioiden esittely

Kategoriat valittiin tutkimuskysymykseen ja aineistosta tehtyihin havaintoihin perustuen. Kategorioiksi muodostuivat testauksen organisointi, testausosaaminen, standardoinnin taso, asiakas, testausstrategia sekä testauksen ongelmat. Kategoriat toisiinsa liittävänä ydinkategoriana on testausstrategia. Kategoriat ja niiden dimensiot organisaatioyksiköittäin on esitelty taulukkona liitteessä 4. Seuraavassa kategoriat on esitelty yksityiskohtaisemmin.

7.1.1 Testauksen organisointi

Tutkimukseen osallistuneissa organisaatioyksiköissä testaus oli organisoitu vaihtelevasti. Organisaatioyksiköinä tutkimisesta huolimatta, organisaation koosta johtuvat erot olivat havaittavissa. Pienempiin organisaatioihin kuuluvissa organisaatioyksiköissä A, B, D ja I testausta tehtiin pääsääntöisesti oman työn ohella, jolloin erillisiä testaajia ei ollut käytössä. Organisaatioyksiköiden B ja I tapauksissa testausvastuuta siirrettiin asiakastuessa toimiville henkilöille. Lisäksi organisaatioyksikkö B:n tapauksessa asiakkaalle annettiin alennusta, jos he suorittavat testauksen.

> ”Ja asiakastuelle kuuluu meillä, tuotteiden testaus. Eli kun tulee uus versio niin, me testataan se loppukäyttäjän näkökulmasta.” (Organisaatioyksikkö B, testaaja)

Organisaatioyksiköissä A ja D testaus oli heikoimmin organisoitunutta. Molemmissa organisaatioyksiköissä kehittäjä testasi tuotetta itsenäisesti omien näkemyksiensä mukaisesti. Organisaatioyksikkö D oli kehittäjien testauksen lisäksi suorittanut kenttätestauksen tuotteelleen, mikä sisälsi 22 000 käyttäjien raportoinnit tuotteen ongelmista.

> ”Koska varsinaist testiorganisaatiota ei ole, ei ole erikseen nimettyjä testaajia. Se testataan itse mikä tehdään, ei kovin kattavasti, laajasti.” (Organisaatioyksikkö A, suunnittelija)

Keskikokoisiin organisaatioihin kuuluvissa organisaatioyksiköissä E, F ja H testaus oli huomioitu hieman paremmin kuin pienissä organisaatioissa. Kriittistä tuotetta valmistavassa organisaatioyksikkö E:ssä testaus oli suunnitelmallista ja testaus oli suunnittelijoiden ja kehittäjien vastuulla.

> ”Eli työtehtävä täällä on ohjelmistosuunnittelija ja teen yleensä testaussuunnitelmat myös suunnittelemiini ohjelmistoihin ja sitten myös testaan osittain niiltä osin, mikä mulle kuuluu siinä että se tietysti osittain se ja kokonaisuus siinä integraatiotestauksessa kuuluu mulle sitten.” (Organisaatioyksikkö E, testaaja)

Organisaatioyksikkö H:ssa testauksesta vastasi erillinen testaustyöryhmä, mutta kehittäjät tekivät yksikkö- ja regressiotestausta. Organisaatioyksikössä F testausvastuu oli siirretty sisäiselle asiakkaalle. Organisaatioyksikkö F:n testausta tekevä henkilö oli huolissaan tilanteesta, koska hänen mielestään heille ei testauksen arvostuksen puutteen vuoksi palkata vain testaustyötä tekeviä henkilöitä.

> ”Jostain syystä meiän firmas on sellanen asenne joillain henkilöillä että, testaajia ei palkata vaan että pitää olla joku muu ihminen joil on muu tehtävä yritykses joka suorittaa testausta, eli esimerkiks myyjillä saattaa olla testausvastuuta, tai niin et ne joutuu suorittaa testausta ja se on niitten toissijanen tehtävä ja se ei oikeen toimi.” (Organisaatioyksikkö F, testaaja)

Organisaatioyksiköt C, G, J, K ja L kuuluvat isoihin organisaatioihin. Näistä Organisaatioyksiköistä L oli ainoa, jolla ei ollut erillisiä testaajia käytössä vaan he käyttivät testauksessa testausvastaavaa, joka ohjasi kehittäjät testaamaan tuotetta. Organisaa-

tioyksiköissä G, J ja K oli kokonaan erilliset testausyksiköt, joissa noudatettiin testauspolitiikka tai vastaavanlaista ohjeistusta. Organisaatioyksikössä C oli erilliset testaajat, jotka toimivat Scrum-työryhmien jäseninä.

Organisaatioyksikkö C:ssä toimi yhteensä seitsemän testaajaa, joista kolme oli ulkopuolisia konsultteja. Testaajien määrä ei ole suuri, vaikka organisaatioyksikkö kuului isoon organisaatioon.

> ”Yeah, I’m senior test designer in the organisation. Um, there’s one other senior test designer, we’ve got a very small test group.” (Organisaatioyksikkö C, testaaja)

7.1.2 Testausosaaminen

Testaajien osaamisessa toimialatuntemus nähtiin selkeästi tärkeimpänä osaamisalueena. Toimialatuntemuksen tärkeyttä korostettiin lähes jokaisessa organisaatioyksikössä. Poikkeuksina olivat vain organisaatioyksiköt A ja D, joista organisaatioyksikkö A:ssa luotettiin kehittäjien testausosaamiseen ja organisaatioyksikkö D:ssä luotettiin kenttätestaukseen osallistuneiden käyttäjien tietämykseen. Teknistä osaamista painotettiin myös, mutta selkeästi vähemmän kuin toimialatuntemusta. Esimerkiksi organisaatioyksikkö I:n tapauksessa toimialatuntemuksen tarve oli huomattava:

> ”Se on se toimialatuntemus, eli ilman sitä ei. Meil on kokeiltu sitäkin että testais vähän semmoset jotka ei sitä toimialaa tiiä ja, se on ihan (kaihdettu), jos ei oo sitte jotain tiettyjä arvoja, tarkastelee tommosia, semmosta nyt vois panna mutta, yleensä tässä tulee se toimiala tullee niin korostetusti esille että, kyl se pittää tuntee se toimiala.” (Organisaatioyksikkö I, testaaja)

Testaajan teknisen osaamisen tärkeyttä korostettiin organisaatioyksiköissä C, E, G ja J. Teknistä osaamista pidettiin tärkeänä, koska toimialatuntemuksen lisäksi testaajan oli ymmärrettävä tekniset perusasiat onnistuakseen tuotteen testauksessa. Esimerkiksi J:n tapauksessa testaajan oletettiin osaavan testata tuote vähäisenkin ohjeistuksen perusteella.

> ”Sit täs se tekninen puoli taas meil on kuitenkin kohtuutarkal tasol niit testikeissei määritelty, et se on tärkeetä, et osaaminen riittää siihen, että on sen järjestelmän peruskäyttö hallussa ja tota niin, testikeissin pohjalta yleensä pystyy sitten järkeilemään loput.” (Organisaatioyksikkö J, suunnittelija)

Organisaatioyksikkö C:n testaajan mukaan heidän käyttämä Scrum-menetelmä vaatii testaajalta enemmän testaustietämystä kuin mitä vesiputousmallin mukainen testaus vaatisi. Testaaja piti teknistä osaamista tärkeämpänä, koska toimialatuntemusta he saivat käyttöönsä tarvittaessa toisaalta.

> “I think especially for testing, it’s introducing new issues that haven’t yet been solved. So that means actually, the work from the tester’s point of view is much more difficult, and I think, at a much more senior level than for instance, a tester working in a waterfall project.” (Organisaatioyksikkö C, testaaja)

Organisaatioyksikkö L:n ja I:n tapauksessa testausosaamista ja kokemusta testaustoimenpiteistä vaadittiin myös testausta johtavalta testausvastaavalta tai projektipäälliköltä. Organisaatioyksikkö L:n testaus oli todella riippuvainen siitä, kuinka hyvin testausvastaava ymmärsi testattavaa kohdetta. Testausvastaavalla oli kuitenkin apunaan testausmalli testauksen suunnittelussa.

> ”Periaattees mahdollisuudet siihen, että se olis kattava on, on hyvät, koska se testausmenettely on ihan erikseen ohjeistettu ja siit on semmonen malli olemassa, testausmalli. Mutta tietysti tapauskohtasesti se riippuu aika pitkälti siitä testausvastaavasta, et kuinka hyvin se pääsee sisään siihen testattavaan asiaan ja paneutuu siihen ja, sitte laatii ne testit ja testitapaukset.” (Organisaatioyksikkö L, suunnittelija)

Toimialatuntemuksen tarvetta kuvastaa hyvin organisaatioyksikkö H:n suunnittelijan näkemys testausosaamisesta. Organisaatioyksikkö H:n suunnittelija oli vahvasti sitä mieltä, että testaustoimenpiteissä testaajalta vaadittiin toimialatuntemusta.

> ”Me ei voida testausta ulkoistaa, et tää vaatii tämmöstä vahvaa tuotteen tuntemusta ja markkinatuntemusta.” (Organisaatioyksikkö H, suunnittelija)

7.1.3 Standardoinnin taso

Organisaatioyksiköille oli tyypillistä, että niissä noudatettiin organisaation itsensä määrittelemiä käytäntöjä. Viralliset standardit toimivat pääasiassa taustavaikuttajina omien käytäntöjen tukena. Organisaatioyksikkö J oli ainoa, joka noudatti ISO 9001 -standardin asettamia vaatimuksia tarkasti. ISO-standardeille tyypillisesti, standardia oli kuitenkin muokattu omiin tarpeisiin sopivaksi.

> ”Siis laatusertifikaatio ISO jotain, ja mä luulen että sieltä tulee aika vahva ohjaus nyt, mutta meidän liiketoimintayksikön tarpeisiin sitä on kyllä säädetty. Kyl se aika tiukasti ohjaa ja aika tiukasti on käytössäkin, että ei oo siitä hirveemmin lipsuttu.” (Organisaatioyksikkö J, testaaja)

Muiksi virallisia standardeja noudattaviksi organisaatioyksiköiksi tunnustautuivat organisaatioyksikkö B ja L. Organisaatioyksikkö B:ssä käytettiin ISO 9001 -standardia, mutta sitä ei noudatettu kovin tarkasti. Organisaatioyksikkö L noudatti CMMI-mallia tuotekehityksessä, mutta testauksessa organisaatiossa käytettiin itse kehitettyä prosessia.

> ”CMMI on meillä käytössä, se on pari vuotta, tainnu kolmekin vuotta olla kohta.” (Organisaatioyksikkö L, testaaja)

Huomattavan yleistä oli, että organisaatioyksiköiden omat käytännöt perustuivat viralliseen standardiin, kuten edellä mainittuihin ISO 9000 -standardisarjaan tai CMMI-malliin. Organisaatioyksiköiden E ja G omat käytännöt perustuivat ISO 9001 -standardiin ja käytäntöjä myös noudatettiin. Organisaatioyksikkö G:ssä testaukselle oli määritelty testauspolitiikka standardia soveltaen. Organisaatioyksiköt H ja K käyttivät CMMI-mallin mukaisia toimenpiteitä tukemaan omaa toimintaansa.

> ”Ei oo, orjallisesti lähdetty hakemaan mitään sertifikaattia, että tehdään jotain prosessia standardin mukasesti, mutta standardit on siinä mielessä käytössä, että niinku veikkaan, että monessa muussaki yrityksessä toimitaan just näin, että ne toimii semmosina hyvinä muistilap-

> puina, että mitä pitäs kuitenkin käydä läpi ja mitä pitäs miettiä, elikä tommosta niinku CMMI:tä ainakin on tässä käytetty semmosena tukena, että miten toimitaan.” (Organisaatioyksikkö H, suunnittelija)

Organisaatioyksikössä A:ssa yhtenäisten käytäntöjen noudattaminen oli kehittäjien vastuulla. A:lla ei ollut käytössä yleisiä ohjelinjoja. Organisaatioyksiköissä D ja I noudatettiin omia käytäntöjä ja sisäisiä ohjeita tuotekehityksessä, mutta testaukselle ei ollut yhtenäistä käytäntöä jokaiselle projektille. Organisaatioyksiköissä C ja F noudatettiin myös omia käytäntöjä. Organisaatioyksikkö C:n oli pitänyt kehittää omia käytäntöjä varsinkin testaustoimenpiteille, koska perinteiset mallit eivät sopineet käytettäviksi ketterissä menetelmissä.

> ”Että ehk ei prosessia, mutta meillä on niitä omia tiettyjä käytäntöi, nimenomaan tähän ketterään, ketterissä menetelmissä, testaamiseen, siihen kun ei noi perinteiset oikein käy, että niin ku oma.” (Organisaatioyksikkö C, projektipäällikkö)

Jälleen oli havaittavissa eroja erikokoisiin organisaatioihin kuuluvien organisaatioyksiköiden välillä. Isoihin organisaatioihin kuuluvissa organisaatioyksiköissä noudatettiin standardia tai tiettyä organisaation määrittelemää käytäntöä kehityksessä ja testauksessa, joka perustui standardiin. Pienissä organisaatioissa standardit tiedostettiin, mutta niitä noudatettiin löyhästi tai luotettiin omiin menettelytapoihin enemmän. Keskikokoisiin organisaatioihin kuuluvissa organisaatioyksiköissä vain yhdessä ei huomioitu standardeja lainkaan.

7.1.4 Asiakas

Pääsääntöisesti organisaatioyksiköissä asiakkaalle annettiin mahdollisuus vaikuttaa tuotteen kehitykseen ja testaukseen. Useissa organisaatioyksiköissä asiakkaan tehtävänä oli osallistua määrittelyjen tekoon ja testaustoimenpiteissä tehdä hyväksymistestaus tuotteelle. Yleisesti ottaen asiakkaan kanssa tehty yhteistyö oli tiiviimpää sisäisen asiakkaan kanssa kuin ulkoisen asiakkaan kanssa.

Organisaatioyksikkö B:ssä hyvää asiakassuhdetta pidettiin tärkeänä, koska testaus oli joissain tapauksissa jopa ulkoistettu asiakkaan tehtäväksi. Lisäksi tuotteen lopullinen testaus oli aina projektipäällikön tai asiakkaan vastuulla. Organisaatioyksikkö B:n suunnittelija koki, että organisaation kasvun myötä asiakkaan testaaminen ei välttämättä enää riitä ja testauksen kattavuutta tulisi parantaa, koska virheiden kustannukset nousevat, kun tuotteen käyttäjämäärät kasvavat.

> ”…koska aikaisemmin kun on oltu pienempi niin on voitu mennä ihan hyvin sillä että testaaminen on aika vähäistä ja se jää ihan sinne vaan asiakkaan kontolle. Asiakkaita ollu muutamia, mut nyt kun alkaa olla et asiakkaita samalla tuotteella on kymmeniä, no ei ehkä vielä nyt satoja, mut jos aatellaan sataa esimerkiks lähenee tietyissä tuotteissa käyttäjä-/asiakasmäärä asennusten määrä niin siinä tietysti yksittäisten virheiden, jotka pääsee meidän omasta testausprosessista läpi niin kustannus nousee niin suureks että sen pitäs olla selvästi normaalimpia tarkempi.” (Organisaatioyksikkö B, suunnittelija)

Organisaatioyksiköissä C, F, G, K ja L asiakaspuolta edusti sisäinen asiakas. Organisaatioyksiköissä K ja L asiakas osallistui testauksen katselmointiin ja L:n tapauksessa asiakas teki myös hyväksymistestausta. Kehitykseen K:n asiakas osallistui katselmointien avulla ja L:n asiakas oli tekemässä määrittelyjä. Lisäksi organisaatioyksikkö K:n ulkoinen asiakas hyväksyi testaussuunnitelman.

Organisaatioyksikkö C:ssä asiakas oli Scrum-menetelmän mukaisesti tiiviisti mukana kehitystyössä ja määrittelyjen teossa. C:n tapauksessa asiakas ei tehnyt testausta itse, mutta asiakas saattoi keskustella testaajien kanssa siitä, milloin tuote oli testaajien mielestä tarpeeksi kattavasti testattu.

> ”Eli sen, että millon se softa on valmis, kaikkien kriteerien mukaan, ja näin, joskus ne [asiakas] kysyy testaajilta esimerkiks tai testimanagerilta, että onks tää niin ku valmis tää softa” (Organisaatioyksikkö C, projektipäällikkö)

Organisaatioyksikössä F sisäinen asiakas oli vastuussa testauksen järjestämisestä. Räätälöityä tuotetta valmistaessaan, organisaatioyksikkö F:n ulkoisella asiakkaalla oli mahdollisuus vaikuttaa testaukseen toimittamalla itse tekemänsä testaussuunni-

telma. Organisaatioyksikkö G:ssä kaikenlainen yhteistyö asiakkaan kanssa oli suotavaa kehitysvaiheesta riippumatta. Testaukseen asiakas osallistui katselmoimalla testaustuloksia.

> ”Ja siihen kohtaan ICT voi jo osallistua, ja kaikenlainen yhteistyö liiketoiminnan kanssa ICT:n välillä olis suotavaa koko tän kehitysprosessin ajan. Ja sen jälkeen kun on tää tavoitemäärittely saatu tehtyä ja ehkä mahollisesti katselmoitua, niin se siirtyy vaatimusmäärittelyyn jolloin se niin kuin, siirtyy ikään kuin ICT:n tehtäväks työks, jossa kuitenkin liiketoiminta on edelleen mukana asiantuntijana.” (Organisaatioyksikkö G, suunnittelija)

Organisaatioyksiköissä A, E, H ja J ulkoisella asiakkaalla oli mahdollisuus vaikuttaa testaus- ja tuotantoprosesseihin. Organisaatioyksikkö J:ssä asiakkaan osallistuminen oli riippuvainen asiakkaan kanssa tehdystä sopimuksesta. Asiakas osallistui vähintään määrittelyjen tekoon, mutta sopimuksesta riippuen, asiakkaalla oli mahdollisuus myös katselmoida projektin etenemistä ja tehdä oma hyväksymistestaus. Organisaatioyksikkö E:n asiakas jopa vaati hyväksymistestauksen suorittamista, sen lisäksi asiakkaan toiveet otettiin huomioon muissakin testaustoimenpiteissä.

> ”Asiakashan sinällään on niinku pääarkkitehti, että jos se haluaa vaikuttaa siihen niin silloin se saa vaikuttaa ja totta kai kuunnellaan sitten sen mukaan. Nytten juuri olikin että se asiakas osti sen testaussuunnitelman validoinnin ja testauksen validoinnin, eli täällä oli henkilöt kahtomassa kun testattiin, [toiselta yritykseltä] niin totta kai saa tehä semmosta ja otetaan mielellään vastaankin, jos joku sitä haluaa.” (Organisaatioyksikkö E, testaaja)

Asiakasosallistumisen suhteen organisaatioyksiköt D ja I poikkesivat muista organisaatioyksiköistä. Kummassakaan organisaatioyksikössä asiakas ei osallistunut kehitystyöhön. Organisaatioyksikkö I:ssä tuote kuitenkin annettiin valituille asiakkaille testattavaksi ennen julkaisua. Molemmissa organisaatioyksiköissä asiakkaan antamaa palautetta pidettiin kuitenkin tärkeänä.

> ”Sanotaan et tää enemmänki herättää sit kysymyksii meissä että no miksi ne [asiakkaat] ei ole meillä mukana siellä näissä tekemisissä, mut se varmaan johtuu siitä et meil ei ole tällästä

niinkun konseptia olemassa, johonka me ois voitu kutsua niitä.” (Organisaatioyksikkö D, projektipäällikkö)

7.1.5 Testausstrategia

Monissa organisaatioyksiköissä testausstrategiassa keskityttiin siihen, että sovelluksen ydintoiminnot tai toiminnallisuus pyrittiin varmistamaan. Siten myös monissa tapauksissa testausresurssit keskitettiin kriittisien kohteiden testaamiseen. Esimerkiksi organisaatioyksiköissä B, D ja F, resursseja keskitettiin kriittisten kohteiden testaamiseen. Lisäksi organisaatioyksiköissä B ja F testaustoimenpiteissä keskityttiin uusien toimintojen testaamiseen.

”Kyl se vaikuttaa, että niihin korkee kriittisiin käytetään enemmän paukkuja.” (Organisaatioyksikkö B, testaaja)

”Kyllä se [kriittisyys] näkyy siinä ja oikeestaan se näkyy sillä tavalla, että melkein pelkästään näillä kriittisillä osilla on regressiotestit.” (Organisaatioyksikkö F, suunnittelija)

”Siel on se meijän ensimmäinen runko että ne uudet asiat, niissä tulee testitapauksia jotka me testataan läpi et ne toimii. Ja muuten meil on semmonen tietty, se on nyt ihan sellasella mututuntumalla sellanen standardi et me aina, testataan kevyesti koko ohjelmisto samalla läpi. Mutta aina keskitytään niihin, uusiin ominaisuuksiin jotka tulee, tuotepäälliköltä projektipäälliköltä jostain muualta meille.” (Organisaatioyksikkö B, testaaja)

Organisaatioyksikkö C:n lähtökohta erosi tästä linjasta täysin. Organisaatioyksikkö C:n testaajan mukaan organisaatioyksikön lähtökohtana testaukseen oli, että kaikki testataan. Testeille etsittiin nopeampi tapa suorittaa ne, ennemmin kuin jätettiin testi suorittamatta. Jos testausta jouduttiin kuitenkin optimoimaan, niin testauskohteet valittiin riskien tai kehittäjien ja arkkitehtien neuvojen perusteella.

Organisaatioyksikkö C:n testaajan mielestä heidän testaus kattoi suuren osan sovelluksen toiminnoista ja testaus suoritettiin nopeasti. Kattavuus ja nopeus perustuivat

siihen, että mahdollisimman paljon testejä automatisoitiin ja testausresurssit keskitettiin siten, että kukin testaaja toimi vain yhdessä projektissa kerrallaan. Testaaja esitti myös väitteen, että heillä suoritetaan testejä enemmän kuin monissa muissa organisaatioissa.

> "Our coverage is very high. The approach that we take to testing, is that, we're very good at rapid testing, we always look for faster ways to execute test cases instead of, eliminate tests from being executed. So we try, do a lot of scripting, for instance, to automate test articles, should we say. So, we execute a lot more tests than a lot of other organisations." (Organisaatioyksikkö C, testaaja)

Organisaatioyksikössä A testausstrategiaa ei varsinaisesti ollut määriteltynä, mutta se oli kehittynyt kuitenkin siihen suuntaan, että he testasivat tuotteen muutoksien jälkeen sekä tuotteen uudet toiminnot. Testausmenetelmänä he käyttivät tutkivaa testausta.

> "Ei meillä varsinaista strategiaa ole, että. Jokainen testaa niin kun parhaaksi näkee. Muutos testataan, ja uus kehitys, että se näyttäs toimivan." (Organisaatioyksikkö A, suunnittelija)

Organisaatioyksikössä E testausta pyrittiin valvomaan siten, että testauksesta vaadittiin dokumentit. Testaus perustui etukäteen määriteltyyn suunnitelmaan, johon siirryttiin pikaisesti suoritetun tutkivan testauksen jälkeen. Testauksen tarkoituksena oli tarkastaa tuotteen toiminnallisuus. Kattavuuteen kiinnitettiin huomiota, jos oli aihetta epäillä, että tuotteessa olisi vielä löytymättömiä virheitä.

> "Koska meillä on tää laatujärjestelmä käytössä, niin siitä tulee tavallaan, meiän täytyy pystyä todistamaan tämä, miten tämmönen testaus on tehty, toisin sanoen, meille semmonen testaus, mistä ei löydy mitään dokumenttia, niin on hukkaan heitettyä." (Organisaatioyksikkö E, suunnittelija)

> "Siinä on, se etukäteen suunniteltu on tietysti siinä vaiheessa kun on saatu, eli ihan ensimmäisenä sen kanssa leikitään vähän aikaa ja katotaan, miten se toimii ja sitten kun on sieltä

> saatu semmosia yksinkertasimpia ja helpompia virheitä pois, niin sen jälkeen siirrytään siihen, mikä on se oikee testaussuunnitelma..” (Organisaatioyksikkö E, testaaja)

Organisaatioyksikössä G:n testaus perustui testauspolitiikkaan. Projekteilla oli oma testausvastaava, jonka tehtävänä oli tehdä testaussuunnitelma vaatimusmäärittelyjen ja käyttötapauksien perusteella. Ydintoiminnot ja sovelluksen toiminnallisuus pyrittiin varmistamaan automatisoidun regressiotestauksen avulla. Uusien toimintojen testaaminen tehtiin kuitenkin manuaalisesti.

> ”Testausvastaava on yksittäisen projektin testausvastaava, eli hän kirjoittaa testaussuunnitelman siihen projektiin ja huolehtii siitä, että resursoi sen, tai ottaa ne vastausresurssit käyttöön ja sitten huolehtii siitä, että se testaus etenee, ja raportoi testauksen etenemisestä johtoryhmälle tai ohjausryhmälle.” (Organisaatioyksikkö G, testaaja)

> ”Regressiotestaus, ja nimenomaan sitä että se on automatisoitu. Et sitä ei joka kerta taota erikseen vaan siellä ajetaan ne testisetit, ja sit jos tulee poikkeavia tuloksia, sitte ruvetaan pureutumaan et miksi.” (Organisaatioyksikkö G, projektipäällikkö)

Organisaatioyksikkö H:n testaaja kertoi, että heillä testausresurssit keskitettiin tärkeimmän tuotteen testaamiseen. Tästä johtuen, testausresursseja ei välttämättä riittänyt kaikille projekteille. Testauksen automatisointia hyödynnettiin öisin suoritettavien testien muodossa ja uudet testit pyrittiin automatisoimaan myös.

> ”Kyllä tietysti etukäteen määritellään, että mitkä alueet ois niitä tärkeimpiä, mihin pitää saada testausta, ja sen mukaan sitten aina paras mahdollinen arvaus yritetään tehdä. Kyllä se on, että projektista tulee, että katsotaan, mitkä on tärkeimmät projektit, mihin tarvitaan testausta, ja sen mukaan sitte tehdään.” (Organisaatioyksikkö H, testaaja)

Organisaatioyksikkö I keskittyi varmentamaan sovelluksen toiminnan yksikkö- ja integraatiotestauksen avulla. Testauksen suunnittelu oli projektipäällikön vastuulla, joka kirjasi testaustoimenpiteet muiden projektiin liittyvien tehtävien joukkoon.

> ”Se projekti etenee vaiheittain ni yksikkötestaus, se on nyt ihan, oikeestaan sanois sitä, pitkin matkaa koko ajan tullee sitä ja sitte, sitten aina kun tulleepi toimitus niin sitte otetaan tämmönen integraatiotestaus tyyppisenä vedetään se läpi että kaikki hommat pyörii myös ne mitkä aikasemmin toimi ni toimiiko ne vielä.” (Organisaatioyksikkö I, testaaja)

Laatujärjestelmä ohjasi organisaatioyksikkö J:n testaustoimenpiteitä. Testaussuunnitelma noudatti laatujärjestelmän vaatimaa pohjaa. Testauksen pääpaino oli uusien ominaisuuksien testaamisessa sekä toiminnallisuuden varmistamisessa regressiotestauksen avulla. Testauksen kattavuus oli varmennettu siten, että kehityksen ja testauksen projektipäälliköt määrittelivät testauksen pääpiirteet.

> ”K: Joo, kuitenkin niinkun pääpaino ilmeisesti selkeesti näiden uusien ominaisuuksien toiminta?
> V: Joo, ja sitten on se regressiotestaus mistä oli just puhetta.” (Organisaatioyksikkö J, testaaja)

> ”Mut sitten testitapausten osalta kattavuus on oikeastaan pyritty hoitaa sillä että testauksen projektipäällikkö ja sen toteutuksen projektipäällikkö on yhdessä miettineet sen niinkun tavallaan otsikkotasolla läpi että tuleeko kattavasti testattua näillä.” (Organisaatioyksikkö J, testaaja)

Myös organisaatioyksikkö K:ssa testaustoimenpiteet ja testaussuunnitelman pääpiirteet oli selkeästi määritelty testauspolitiikkaa vastaavassa ohjeistuksessa. Järjestelmällisen suunnitelmaan perustuvan testauksen lisäksi organisaatioyksikössä tehtiin vähän myös tutkivaa testausta.

> ”Meijän yksikössä on ainakin tosi tarkka et missä järjestyksessä pitää tehdä. Muuten sä et pysty jatkamaan seuraavaan testausvaiheeseen ennen kuin sul on yks vaihe tehty. Et se on tosi järjestelmällistä. Et ei voi poiketa siitä mallista.” (Organisaatioyksikkö K, testaaja)

Organisaatioyksiköissä L testauksen lähtökohdat oli määritelty erikseen ohjeistetussa testausmenettelyssä, jota testausvastaava noudatti testaussuunnitelmaa tehdessään. Käyttötapausten testauksella pyrittiin varmistamaan tuotteen toiminnallisuus. Testa-

usta valvottiin siten, että organisaatioyksikössä katselmoitiin testausvastaavan suunnitelma sekä testauksen tulokset. Testausautomaatiota he hyödynsivät kuormitustestauksessa.

> ”Eli tavallaan meillä rakentuu käyttötapauksiin meiän tämä testauksen, ni käyttötapauksii testataan sitten ihan, kokonaan sitten” (Organisaatioyksikkö L, testaaja)

7.1.6 Testauksen ongelmat

Pieniin organisaatioihin kuuluvissa organisaatioyksiköissä oli nähtävissä yhtäläisyyksiä testauksen ongelmien suhteen. Organisaatioyksiköissä A, B, D ja I jokaisella oli ongelmia resurssien ja testausstrategian tai testaussuunnitelman kanssa. Organisaatioyksikössä A testauksen ongelmat johtuivat osaksi testaustyön arvostuksen puutteesta. Myös organisaatioyksiköissä B, D ja I testauksen ongelmia aiheutti se, että kehitystyötä pidettiin tärkeämpänä kuin testauksen suorittamista.

> ”Sitä ei valitettavasti katota niin tärkeeks hommaks et siihen delegoitas joku ihminen. Et se on aina vähän OTOna [oman työn ohella]. Et jokainen vähän tekee sitä silloin kun ehtii tyyppisesti, vaikka se oikeesti säästäis sit pidemmäl tähtäimel aika paljon aikaa ja vaivaa, jos sen joku kunnolla kävis läpi.” (Organisaatioyksikkö A, testaaja)

> ”Mut se et se organisointi resurssointi ja se, tietynlaisen sen prosessin vakiinnuttaminen. Niin mä luulen et siinä on se, ainaki meillä se suurin sudenkuoppa tällä hetkellä.” (Organisaatioykikkö B, testaaja)

> ”Jos sitä keritään tekemään sitä suunnittelua etukätteen nii, ei siellä niin kun paneuduta ollenkaan semmoseen sannaan ku testaukseen.” (Organisaatioyksikkö I, testaaja)

Testaustyön arvostuksen puute johti ongelmiin myös organisaatioyksikössä F. Sekä organisaatioyksikkö F:n suunnittelija, että testaaja pitivät sitä ongelmana, että heille ei palkattu erillisiä testaajia. Suunnittelija ja testaaja olivat yhtä mieltä siitä, että tes-

taustoimenpiteet jätetään helposti suorittamatta, kun testaus ei ole ensisijaisena työtehtävänä.

> ”Ei oo, henkilöitä joil ois niinku pääasiallisena tehtävänä softan testaaminen, niitä on tasan yks tai nolla, riippuen miten lasketaan. Ja jostain syystä meiän firmas on sellanen asenne joillain henkilöillä että, testaajia ei palkata vaan että pitää olla joku muu ihminen joil on muu tehtävä yritykses joka suorittaa testausta, eli esimerkiks myyjillä saattaa olla testausvastuuta, tai niin et ne joutuu suorittaa testausta ja se on niitten toissijanen tehtävä ja se ei oikeen toimi.” (Organisaatioyksikkö F, testaaja)

Organisaatioyksiköissä E ja H testauksen ongelmat olivat tyypillisiä ongelmia vesiputousmallin mukaisessa kehityksessä. Testaukselle ei riittänyt resursseja ja aikaa. Organisaatioyksikkö H:n projektipäällikkö määritteli, että organisaatiolla oli vain kolmasosa tarvittavien testaajien määrästä. Testausresurssien riittämättömyydestä johtuen, testaustoimenpiteiden suorittamiseen ei ollut aikaa. E:n tapauksessa erillisien testaajien puutteesta seurasi samankaltaisia ongelmia kuin organisaatioyksikkö F:llä, kiireessä testaustoimenpiteet jäivät suorittamatta, koska ensisijaisiin tehtäviin kiinnitettiin enemmän huomiota.

> ”Jos aattelee taas, kuinka kattavaa se on, niin välillä se sitten jää siitä kiinni, että se ei oo riittävän kattavaa kuitenkaan Ja myöskään, kaikkeen uusimpaan kehitykseen, niin läheskään kaikkeen ei synny niitä testitapauksia samassa tahdissa, kun uudiskehitys tapahtuu” (Organisaatioyksikkö H, testaaja)

> ”Mun mielestä niitä ei ole tarpeeks, et nyt se on vähä semmonen, et meil ei päätoimisia testaajia oikeestaan ole. Ja nyt sitte, kun ajatellaan, et tekijä ei oikeestaan oo niinkun oikea ihminen tuotoksiaan testaamaan, tietysti sitäki täytyy jokaisen tekijän tietenki tehä, mutta loppupelissä, niin joskus on aika vaikee löytää, niinku projektiaikataulut on, meil on hirveesti töitä, niin mahotonta löytää enää aikaa sille testaukselle, koska ei oo semmosta, ulkopuolista tahoa, joka sitä edes meiltä siinä viime kädessä vaatis välttämättä.” (Organisaatioyksikkö E, suunnittelija)

Isoihin organisaatioihin kuuluvissa organisaatioyksiköissä testauksen ongelmat olivat monimuotoisia. Organisaatioyksikkö C:n ongelmana oli, että Scrum-menetelmä aset-

ti vaatimukset testaajien osaamiselle ja testauksen dokumentointi ei ollut riittävällä tasolla. Resurssien puute johtui testaajan mukaan taloudellisesta tilanteesta, jota pyrittiin ratkaisemaan ulkoisten konsulttien avulla.

> ”Resurssit, resurssit on suurin puute tällä hetkellä. Sanotaan että testausresursseihin, sen puuttumiseen se o ihan selvä yhteys taloudelliseen tilanteeseen.” (Organisaatioyksikkö C, testaaja)

Organisaatioyksikössä G testauksen dokumentointi oli hyvässä mallissa, mutta tiedon siirtyminen oli ongelma. Organisaatioyksikkö G:n projektipäällikön oli vahvasti sitä mieltä, että riittämätön kommunikointi testaajien ja tuotekehittäjien välillä hidasti virheiden korjaamista. Lisäksi suunnittelijan mukaan liiketoimintapuolen henkilöstö ohitti suunnittelijat ja keskustelivat suoraan tuotekehittäjien tai sovelluspäälliköiden kanssa. Erillisestä testausyksiköstä huolimatta, resurssit jakautuivat epätasaisesti testaushenkilöstön vaihtelevan liiketoiminnallisen osaamisen vuoksi.

> ”Meillähän on siis oma yksikkö testaukselle ja siellä pyritään resursoimaan, mutta siinä on se ongelma niinkun löytää jokaisen projektin testaukseen sellaiset henkilöt, joilla on riittävää liiketoiminnallista ymmärtämistä. Eli sellaisten resurssien saaminen on joskus tiukalla.” (Organisaatioyksikkö G, testaaja)

Organisaatioyksikkö J:llä oli resurssien riittämättömyyden lisäksi ongelmia myös testausstrategian kanssa. Testausstrategian puutteet olivat nousseet esiin testaajille kokonaan uusien asioiden testaamisessa. Tämä aiheutti testaajien työmäärän kasvua, josta seurasi edelleen testien heikentyneet suunnitelmat, jotka vaikuttivat testauksen kattavuuteen.

> ”Sitten kun me on menty vähä uusille uusille urille niin sit se ei oo ollu aivan niin helppoa, et siin olis todennäkösesti tarvittu erilaista vaihejakoa tai muuta erilaista.” (Organisaatioyksikkö J, testaaja)

Testausympäristöjen vähyys aiheutti organisaatioyksikkö K:n aikatauluongelmat. Testaajia oli enemmän kuin testausympäristöjä, joten testausympäristöjä ei riittänyt

jokaiselle ja sen vuoksi huonoimmassa tapauksessa projektit joutuivat odottamaan testausympäristöjen vapautumista. Toinen aikatauluihin vaikuttanut tekijä oli asiakkaan vaatimat muutokset projektiin.

> ”Kyllä se mun mielestä on ongelma. Et kyl me tarvittais lisää niitä testikantoja. Eli jotkut projektit joutuu sit odottaan” (Organisaatioyksikkö K, suunnittelija)

Organisaatioyksikkö L:n projektipäällikön mukaan heidän suurin ongelmansa testauksessa oli, että heillä ei ollut varsinaisesti testaukseen erikoistunutta osaajaa. Suunnittelijan mielestä nykyisen käytännön heikkous oli siinä, että testauksen kattavuus oli riippuvainen projektille määrätystä testausvastaavasta.

> ”Meil ei oo ammattilaisprofiilia siin testauksessa. Siel se ongelma on mun mielestä.” (Organisaatioyksikkö L, projektipäällikkö)

> ”Mutta tietysti tapauskohtasesti se sitte riippuu aika pitkälti siitä testausvastaavasta, et kuinka hyvin se pääsee sisään siihen testattavaan asiaan ja paneutuu siihen.” (Organisaatioyksikkö L, suunnittelija)

7.2 Hypoteesien muodostaminen

Seuraavat hypoteesit muodostettiin yllä esiteltyjen kategorioiden ja havaintojen perusteella. Hypoteeseja muodostaessa huomiota kiinnitettiin muun muassa siihen, miten organisaatioyksiköt kokivat ketterien menetelmien käytön kehitys- sekä testaustyössä.

7.2.1 Ketterillä menetelmillä järjestetään lisää aikaa testaustoimenpiteisiin, vaikka samalla projektin kokonaisaikaa pyritään lyhentämään

Tapa, jolla ketterissä menetelmissä voidaan nopeuttaa tuotantoa sekä järjestää lisää aikaa testaukselle, on tuotteen jakaminen osiin. Pienissä osissa toimitettava kokonaisuus voidaan myös testata osissa, jolloin testaajien ei tarvitse odottaa koko tuotteen valmistumista. Lisäksi tuotteen kriittisimmät osat voidaan tehdä ensimmäisinä, jolloin niiden testaamiseen on eniten aikaa.

> ”Uskoisin, että sitä kautta saatais paljon nopeammin tuotantoon. Eli voitais viedä pienemmissä erissä sitä kokonaisuutta tuotantoon ja saatais niinkun tavallaan ne ensimmäiset kriittisimmät osat paljon nopeammin” (Organisaatioyksikkö G, testaaja)

Suunnitelmalähtöisissä menetelmissä testausajasta usein tingitään, jos toteutus tai määrittelyjen teko myöhästyy. Kiireissä aikataulussa testaus jätetään helposti suorittamatta, varsinkin jos testausvastuu ei ole erillisillä testaajilla. Lisäksi kun testaus on viimeinen suunnitelmalähtöisen menetelmän vaihe, projektin lopussa suoritetussa testauksessa löydettyjä virheitä ei välttämättä ehditä korjata toimitusajankohtaan mennessä. Toimivissa projekteissa testaus ei muodosta ongelmia projektin loppuvaiheessa. Ketterissä menetelmissä testaus on järkevämmin mukana projektin alusta alkaen.

> ”Being involved in the project planning stages already, requirement definition and so on, is really good. I have much deeper knowledge of what I’m testing when I get around to actually testing it.” (Organisaatioyksikkö C, testaaja)

> ”Se ehkä mitä se [ketterä menetelmä] tekee, se luontevasti ohjaa siihen että se testaus tapahtuu, et se ei ole vaan projektin päätteessä tapahtuva asia.” (Organisaatioyksikkö B, suunnittelija)

> ”Ketterässä menetelmässähän se jakautuu se testauksen osuus niin, mun mielestä se kuulostaa huomattavasti paremmalta ja järkevämmältä.” (Organisaatioyksikkö G, testaaja)

Testauksen aikainen osallistuminen antaa testaukselle enemmän aikaa ja tuotteesta saadaan palautetta jo projektin alusta lähtien. Testaajilla on myös parempi käsitys testattavasta tuotteesta, koska he osallistuvat projektiin jo määrittelyn ja suunnittelun aikana. Siten virheitä löydetään aiemmin projektin aikana ja virheiden korjaukseen on enemmän aikaa. Varhaisen testauksen avulla säästetään kuluissa, koska kaikkia virheitä ei löydetä vasta projektin lopussa.

> ”Kyllä se [ketterä menetelmä] mun mielestä jollain tavalla varmaan parantais [laatua]. Siinä kun aikasemmin jonkun ohjelman tekee ja lähtee väärään suuntaan tekemään sitä niin päästään nopeemmin siihen ongelmaan käsiks. Nopeemmin pystyttäs sit korjaamaan ja säästämään sitä myötä sit kuluissa.” (Organisaatioyksikkö K, suunnittelija)

> ”Se vaikuttaa siihen että joutuu testaamaan pienempiä osia, joutuu testaamaan aiemmin, ku nyt on jäähdytyskausi releasee varten on alkanu ni sillon testaus on lähteny hirveel vauhdil käyntiin ja sit, bugit on löytyny liian myöhää, et se on pääasiallinen ero. Ja tää on hyvä asia, koska se pitäis tuoda esiin bugit aiemmi” (Organisaatioyksikkö F, testaaja)

Organisaatioyksiköissä C, D ja G oltiin sitä mieltä, että ketterien menetelmien käytöllä voidaan saavuttaa nopeampia tuloksia. Lisäksi organisaatioyksikkö F:n testaajan mukaan ketterien menetelmien avulla tiedetään paremmin miten projekti etenee. Organisaatioyksikkö C:n testaaja oli havainnut, että aikataulut pyrittiin säilyttämään työskentelemällä ahkerammin.

> ”I think the schedules are probably better. What we have noticed is, all of the testers want to keep to their schedules, because they’re so much shorter. And that’s a really good thing, from the SCRUM point of view, you’ve got two weeks to do it, instead of three months to do it. So, we have ended up working much harder than we used to work before, to meet the schedules.” (Organisaatioyksikkö C, testaaja)

7.2.2 Ketterien menetelmien käyttö tasaa testausresurssien kuormitusta mutta ei vähennä itse resurssitarvetta

Ketterät menetelmät tasaavat testausresurssien kuormitusta tuotteen pilkkomisella pienempiin osiin, joita voidaan testata aikaisessa vaiheessa. Tällainen lähtökohta voi olla jopa tehokkaampi ratkaisu testauksen suorittamiseksi, koska iteraatioiden aikana testauksen kohteena on rajattu kokonaisuus. Suunnitelmalähtöisiin menetelmiin verrattuna testaajien kuormitus tasaantuu, koska tuote tehdään osakokonaisuus kerrallaan, joiden testaus jakautuu koko projektin ajalle. Testausta tehdään siten jatkuvasti projektin kuluessa, eikä vain projektin viimeisten viikkojen aikana. Ketterät menetelmät eivät kuitenkaan vähennä testausresurssien tarvetta, koska testausresursseja tarvitaan jatkuvasti kehitysprosessin aikana.

> ”Se muuttaa sitä et tarvittais, tarvitaan jatkuvasti resursseja, ei tarvita vaan kerran tai kaks vuodes vaan pitäis koko ajan löytyy.” (Organisaatioyksikkö F, testaaja)

> ”Jos se [lopputuote] on osissa niin sillon se aiheuttaa siinä mieles vähä vähemmän työtä et siinä on aina se tietty rajattu kokonaisuus mitä testataan. Ja toisaalta sieltä saadaan pala kerrallaan paremmin ne, mahdolliset virheet kiinni” (Organisaatioyksikkö B, testaaja)

> ”Se tavallaan tasais sitä, koska testaus on just semmoista ryöpsäyttelyä, että kun se tuote on valmis niin sit on yhtäkkiä niinkun kaks, kolme viikkoa aikaa testata ja sitä tehdään yötäpäivää, että tavallaan toi tasais sitä testauksen kuormitusta.” (Organisaatioyksikkö G, testaaja)

Lyhyellä aikavälillä tarkasteltaessa testausresurssien tarve ketterissä menetelmissä on kuitenkin vaihtelevaa. Organisaatioyksikkö C:n testaaja esitti, että jos resurssitarvetta tarkastellaan päivittäin, niin resurssitarve vaihtelee.

> ”Vaikuttaa sillä lailla että, ketterällä puolella testaukse resurssitarve saattaa olla semmosella lyhyellä aikajänteellä elikkä se tulee vähä sillai, sanotaanko nykäyksittäin. Eli saattaa olla että yhtenä päivänä tarvitaan jopa useita testaajia ja sitten taas, seuraavana päivänä, on sanotaanko virheiden korjaus tai jotain tämmöstä näin ja sinä päivänä ei tarvittaisi ensimmäistäkään testaajaa. Eli se on hyvin tämmöstä, vaihtelevaa se tarve.” (Organisaatioyksikkö C, testaaja)

Ketterissä menetelmissä testaajat toimivat tehokkaammin, jos testaajat osallistuvat yhteen projektiin kerrallaan. Organisaatioyksikössä C oli saavutettu tehokkaampi testaus keskittämällä testausresurssit toimimaan yhdessä projektissa kerrallaan. Tehokkuutta organisaatioyksikkö C:n testaaja perusteli sillä, että testaajien ei tarvinnut vaihdella tehtäviään eri projektien välillä.

> ”What we have noticed is that, before, when we’re using traditional projects, a tester might be spread across two or three different projects, and it was quite difficult because, you spend so much time changing what you were working on, that you’re a lot less efficient. So now, testers are usually dedicated just to one project” (Organisaatioyksikkö C, testaaja)

7.2.3 Hallinnon ja asiakkaan on ymmärrettävä ja noudatettava ketterissä menetelmissä käytettäviä työtapoja

Toimittajaorganisaation on lisättävä yhteistyön ja kommunikoinnin määrää asiakkaan kanssa ketteriä menetelmiä varten. Kaikkien asianosaisten on ymmärrettävä työtavat, joita ketterissä menetelmissä käytetään. Asiakkaalta vaaditaan erilaista ajatusmallia kuin suunnitelmalähtöisessä kehitysprosessissa, sillä alussa tehtävien tarkkojen määrittelyjen ja sopimuksien sijaan, määrittelyjen annetaan muotoutua kehitysprosessin aikana. Asiakkaan on osallistuttava projektiin tiiviimmin, koska määrittelyjä pitää tarkentaa kehityksen aikana. Asiakkaan on muutettava suhtautumistaan ohjelmistokehitykseen, koska sovellusta ei toimiteta pakettiratkaisuna.

> ”On semmosia asiakasprojekteja, joissa on tarkkoja sopimuksia ja määräyksiä ja joissa on tämmösii niinkun, sopimusteknisistä syistä joudutaan menemään siihen, että asiakas tekee speksin, jota me yritetään toteuttaa ja jos se ei mee hyvin, niin sitten lakimiehet pui asiaa. Eli tämmösis tapauksissa ei valitettavasti voida käyttää ketterii menetelmiä.” (Organisaatioyksikkö F, suunnittelija)

> ”Varsinaisesti, ihan niin älyttömän ketteriä menetelmiä me ei käytetä ja se on hyvin pitkälti meidän asiakasrajapinnan sanelema elikä on vaikea löytää sellasia asiakkaita, jotka haluis

> lähtee tällaseen sykliseen malliin ja tarkennettavaan malliin mukaan, vaan hyvin pitkälti tilaukset on paketti ja sille joku pakettihinta.” (Organisaatioyksikkö H, suunnittelija)

> ”Se ehkä vaatis mun mielestä aika paljon semmosta uutta ajattelumallia enemmän asiakkailta ja meiltä ja silleen että me ymmärrettäis se” (Organisaatioyksikkö L, testaaja)

Lisääntyneellä yhteistyöllä ja kommunikoinnilla asiakkaan kanssa, asiakkaan palaute ja mielipide tuotteesta saadaan nopeammin esiin. Hyväksymistestaus on tehokkaampaa, koska osatoimitus on pienempi kokonaisuus, jonka testaus on nopeampaa kuin koko järjestelmän testaus kerralla. Monet tutkimukseen osallistuneet organisaatioyksiköt toivoivat asiakkaalta enemmän osallistumista projekteissa. Asiakkaan mukanaolo parantaa kehittäjien näkemystä siitä, mitä asiakas oikeasti tuotteelta odottaa ja haluaa.

> ”Kyllä mun mielest se toimii sillon paremmin. Varsinki ku tehään asiakkaalle uutta niin kun hän saa sen palanen kerrallaan ja jos hän ei oo tyytyväinen siihen palaseen niin hän voi ilmasta sen heti.” (Organisaatioyksikkö B, testaaja)

Lisäksi asiakkaan luonteva mukanaolo tehostaa eri asianosaisten kommunikointia. Organisaatioyksikkö C:n tapauksessa asiakkaan oli helpompi tehdä projektiin liittyviä päätöksiä, koska he pystyivät keskustelemaan eri osapuolien kanssa suoraan. Esimerkiksi testauksen osalta asiakas voi ottaa yhteyden suoraan testaajiin saadakseen pätevää tietoa tuotteen testauksen edistymisestä.

> ”Eli siis, sen, että millon se softa on valmis, kaikkien kriteerien mukaan, ja näin, että joskus ne [asiakkaat] kysyy testaajilta esimerkiks tai testimanagerilta, että onks tää niin ku valmis tää softa” (Organisaatioyksikkö C, projektipäällikkö)

Hallinnon on myös ymmärrettävä roolinsa ketterää menetelmää käytettäessä. Hallinnon on noudatettava ketterissä menetelmissä käytettäviä työtapoja sekä ymmärrettävä niitä, jotta menetelmien käytöstä saadaan todellista hyötyä. Esimerkiksi pariohjelmointi, hallinnon on ymmärrettävä miksi kaksi kehittäjää työskentelee yhden koneen ääressä. Ketterissä menetelmissä kehitystyötä lähestytään kehittäjien näkökul-

masta, joten usein kehittäjät ovat niitä, jotka ketteriä menetelmiä haluavat käyttää. Hallinnon on sopeuduttava tilanteeseen ja annettava kehitystyöryhmälle sen tarvitsema tuki sekä työrauha. Kuten muissakin tuotantomenetelmissä, ohjelmistotestauksen toteutus on riippuvainen hallinnon suhtautumisesta testaukseen. Ketterät menetelmät eivät poista testausresurssien tarvetta, mutta ne kohentavat testauksen asemaa kehitystyön rinnalla.

> ”Mut et jos hallinto ymmärtää sen, että tässä saadaan todellista arvoa eikä estä sitä, niin se voi vapauttaa tän kehittäjäryhmän käyttämään näitä asioita ja mun kokemus on se, että usein kehittäjät on niitä, jotka haluaa käyttää näitä kun ne kokee, että ne helpottaa sitä omaa elämää.” (Organisaatioyksikkö F, suunnittelija)

Ketterien menetelmien työtapojen noudattaminen on tärkeää. Organisaatioyksikkö K:n testaajalla oli ollut huono kokemus Scrum-menetelmän käytöstä, koska he eivät noudattaneet menetelmän työtapoja ja ohjeita. Scrum-menetelmässä tärkeimmät osat tehdään ensimmäisinä, mutta organisaatioyksikkö K:n kokeilussa he olivat ensimmäisen sprintin tehneet loppujen lopuksi viimeiseksi. Organisaatioyksikössä C Scrum-menetelmän käytäntöjä noudatettiin, organisaatioyksikkö C:n testaajan mukaan Scrum-menetelmän ensimmäiset iteraatiot ovat usein vaikeita, mutta jatkossa ne helpottuvat.

> ”Eka asia mitä asiakas halus, että pitää toteuttaa ensimmäisenä, se toteutettiin viimesenä. Koska se oli kaikista hankalin ja kuitenkin sprintissä pitäs mun käsityksen mukaan tehä ensin kaikista vaikeimmat tapaukset ja sit vasta lopuks ne helpoimmat.” (Organisaatioyksikkö K, testaaja)

> ”Sanotaan että aiheuttaa aina alkuvaiheessa sitä että kun, projekti lähtee käyntiin niin Scrumi useimmiten lähtee hyvin jähmeästi liikkeelle elikkä ensimmäiset sprintit ovat vaikeampia kun jatkossa myöhemmin.” (Organisaatioyksikkö C, testaaja)

7.2.4 Sisäinen asiakas tukee ketterien menetelmien käyttöönottoa

Ketterien menetelmien perusarvoissa on korostettu yhteistyön merkitystä asiakkaan kanssa. Tutkimukseen osallistuneissa organisaatioyksiköissä yhteistyö asiakkaan kanssa oli huomattavasti tiiviimpää, jos asiakaspuolta edusti sisäinen asiakas. Sisäinen asiakas on luontevampaa ottaa mukaan kehitystyöhön kokoaikaisesti. Ulkoista asiakasta on vaikeampaa saada sitoutumaan projektiin, koska ketterässä kehityksessä asiakkaalta toivotaan jatkuvasti tiivistä yhteistyötä ja osallistumista. Lisäksi ulkoisen asiakkaan ja toimittajan välillä vaaditaan luottamusta, jotta ketteriä menetelmiä voidaan käyttää ilman pelkoa riitatilanteista.

> ”We have a lot of internal customers. And most of them participate, more than you would find in most other organisations, actually.” (Organisaatioyksikkö C, testaaja)

> “On sit tiettyjä asiakkaita, joiden kans on nyt päästy hyvään suhteeseen et voidaan mennä myös ketterillä” (Organisaatioyksikkö B, suunnittelija)

Sisäisen asiakkaan kanssa luottamussuhde on valmiiksi vakaalla pohjalla, joten kehitystyö voi olla joustavampaa. Hyvä luottamussuhde lisää asiakkaan ymmärrystä valmistettavasta tuotteesta ja mahdollisista ongelmatilanteista kehitysprosessissa. Myös kommunikointi helpottuu, kun toimittaja ja asiakas tuntevat toisensa jo valmiiksi.

> ”Positiivista on se, että kun, täähän on hyvin vakiintunutta toimintaa, eli yleensä tekijät ja teettäjät tuntee toisensa valmiiks. Ja siinä mieles, ei tarvii hakee semmosta yhteistä kieltä.” (Organisaatioyksikkö L, suunnittelija)

7.2.5 Ketterät menetelmät soveltuvat hyvin muutos- ja päivitysprojekteihin

Organisaatioyksikössä I käytössä olevaa kehitysmenetelmää ei oltu tunnistettu, mutta osa kehittäjistä halusi tehdä tuotteen muutokset lyhyissä aikaväleissä. Syynä tähän

oli, että siten tuote oli jatkuvasti toimituskelpoinen. Ketterissä menetelmissä iteraation pituus voidaan sovittaa organisaatiolle ja muutokselle sopivaksi, joten toteutettavat osat tai muutokset voidaan myös pilkkoa sopiviksi kokonaisuuksiksi.

> ”Osa koittaa tehä sitte lyhyempiä kerralla tehtäviä [muutosprosesseja] että se, projekti pysys pitempään kasassa. Vaihtelee projektissaki aika paljo että, sillonku on stabiili tilanne projektissa niin se on, hyvinki lyhyellä jänteellä tehhään sitä että, projekti on koko ajan toimituskeleposessa kunnossa.” (Organisaatioyksikkö I, suunnittelija)

Organisaatioyksikkö A:ssa käytettiin ketterille menetelmille tyypillisiä piirteitä tuotteen päivitysprojekteissa. Projekteissa päivitettiin alkuperäistä tuotetta asiakkaiden toiveiden mukaisesti. Organisaatioyksikkö keräsi asiakkaiden toiveita, jotka analysoitiin ja siirrettiin toteutettaviksi sopivana ajankohtana. Päivitysprojektin laajuudesta ja muutosalttiudesta riippuen, ketterät menetelmät soveltuvat niihin hyvin. Pienten projektien on aikaisemmissa tutkimuksissa todettu soveltuvan hyvin ketterille menetelmille.

Ketterien menetelmien avulla voidaan helpottaa kokonaan uusien asioiden kehittämistä, koska prosessia voidaan tarkastella ja muokata tietyin väliajoin. Siten nähdään nopeammin, jos projektissa tehdään vääriä päätöksiä. Tällainen lähtökohta voidaan tietenkin omaksua myös testaustoimenpiteisiin siten, että projektin edetessä kiinnitetään huomiota, miten testaustoimenpiteet ovat onnistuneet ja voidaanko testauksen toteutusta kehittää.

> ”Sitten kun me on menty vähä uusille uusille urille niin sit se ei oo ollu aivan niin helppoa, et siin olis todennäkösesti tarvittu erilaista vaihejakoa tai muuta erilaista.” (Organisaatioyksikkö J, testaaja)

> ”Siinä kun jonkun ohjelman tekee ja niinkun lähtee väärään suuntaan tekemään sitä niin päästään nopeemmin siihen ongelmaan käsiks.” (Organisaatioyksikkö K, suunnittelija)

Dokumentoinnin näkökulmasta organisaation on määriteltävä itselleen sopiva dokumentoinnin määrä. Scrum-menetelmää käyttävässä organisaatioyksikössä C dokumentointi oli vähäistä ja testaajat kokivat sen ongelmaksi, jos projektin testaustehtävät siirrettiin toiselle henkilölle. Toisaalta esimerkiksi organisaatioyksikössä L dokumentoitu tieto vääristyi, koska eri ihmiset tulkitsivat kirjoitetut asiat eri tavalla. Tieto vääristyi myös siksi, että heiltä puuttui suora kommunikaatio asianosaisten välillä. Organisaatioyksikkö L:n edustaja pitikin kommunikoinnin puutetta ongelmana.

7.2.6 Aikaisemmin demonstroitavissa oleva järjestelmä ja lisääntynyt kommunikointi parantavat asiakkaan tyytyväisyyttä lopputulokseen

Yhteisymmärrys asiakkaan ja toimittajan välille löydetään paremmin iteraatioiden avulla, koska täyttä yhteisymmärrystä ei tarvitse löytää heti projektin alusta lähtien. Lisääntyneen kommunikoinnin avulla toimittajaorganisaatiolla on paremmat mahdollisuudet toimittaa sellainen tuote, joka täyttää asiakkaan tarpeet. Demonstraation avulla asiakkaan mielipide saadaan paremmin esiin, koska ilman demoa asiakkaan on vaikea tietää, mitä hän tuotteelta odottaa.

> ”Kun nyt mennään tolla vesiputousmallilla jossa se muutosten tekeminen sit myöhemmin aiheuttaa aiheuttaa varmasti harmia ja työtä lisää meille. Toinen vaihtoehto olis mennä enemmän semmoseen iteratiiviseen jollon ei tarttis siinä alussa löytää sitä täyttä yhteisymmärrystä.” (Organisaatioyksikkö J, testaaja)

> ”Tässä mielessä vastaus on, että ketterät menetelmät, nimenomaan tää iteratiivinen ja nopee kommunikointi parantaa tuotteen laatua siinä mielessä, että tehdään sitä asiaa mitä asiakas oikeesti haluaa, että ohjelmistossa on ne piirteet mitä se käyttäjä tarvii.” (Organisaatioyksikkö F, suunnittelija)

Järjestelmän aikainen demonstrointi on osoittautunut toimivaksi lähestymistavaksi tilanteissa, joissa demonstrointi on ollut mahdollista. Etenkin käyttöliittymää tehdes-

sä aikaisesta demosta ja asiakkaan läsnäolosta on hyötyä, koska asiakas voi kertoa mielipiteensä käyttöliittymästä välittömästi. Asiakas pääsee kommentoimaan käyttöliittymää aikaisessa vaiheessa ja asiakkaan on helppo todeta, mitä hän käyttöliittymältä odottaa tai ei odota. Asiakkaan palaute on siten konkreettisempaa ja tulokset näkyvät valmiissa käyttöliittymässä.

> ”Lähdetään sillä tavalla liikkeelle, että softa on kommunikaation väline. Eli tehdään vähän, annetaan asiakkaille, asiakas sanoo ”okei, tää näyttää hyvältä, mut sen ehkä pitäs toimia näin” sellasis tapauksissa, jolloin tätä on pystytty käyttämään, niin asiat on menny hirvittävän paljon jouhevammin. Monet erityisesti tämmöset, käyttöliittymään ja käyttökokemukseen liittyvät asiat on hirvittävän vaikeita speksata kynällä ja paperilla tai sanoilla. On paljon helpompaa, pystytään toimimaan sillä tavalla, että on jo jonkinmoinen idea asiasta, tehdään toteutus ja sitten tehään kevyt toteutus.” (Organisaatioyksikkö F, suunnittelija)

Taulukko 6. Hypoteesit

Hypoteesi 1	Ketterillä menetelmillä järjestetään lisää aikaa testaustoimenpiteisiin, vaikka samalla projektin kokonaisaikaa pyritään lyhentämään
Hypoteesi 2	Ketterien menetelmien käyttö tasaa testausresurssien kuormitusta mutta ei vähennä itse resurssitarvetta
Hypoteesi 3	Hallinnon ja asiakkaan on ymmärrettävä ja noudatettava ketterissä menetelmissä käytettäviä työtapoja
Hypoteesi 4	Sisäinen asiakas tukee ketterien menetelmien käyttöönottoa
Hypoteesi 5	Ketterät menetelmät soveltuvat hyvin muutos- ja päivitysprojekteihin
Hypoteesi 6	Aikaisemmin demonstroitavissa oleva järjestelmä ja lisääntynyt kommunikointi parantavat asiakkaan tyytyväisyyttä lopputulokseen

Taulukossa 6 on tiivistettynä vielä esitelty työssä muodostetut hypoteesit. Kaiken kaikkiaan näyttäisi siltä, että noudattamalla ketterien menetelmien työtapoja ohjelmistotestauksen asemaa voidaan parantaa ohjelmistokehitysprojekteissa. Lisäksi ketterien menetelmien periaatteissa korostetun tiiviin asiakasyhteistyön avulla voidaan parantaa asiakkaan tyytyväisyyttä lopputulokseen.

8 TULOSTEN POHDINTAA

Työn tulokset koostuvat kuudesta aineiston perusteella muodostetusta hypoteesista. Tutkimuskysymyksen kannalta tärkeimmät havainnot liittyivät ohjelmistotestauksen toteutukseen ketterissä menetelmissä.

Tässä työssä havaittiin, että ketterissä menetelmissä ohjelmistotestaus on paremmin huomioitu projekteissa vaikka ketterien menetelmien avulla pyritään samaan aikaan lyhentää projektien kokonaisaikaa. Testaukselle järjestetään enemmän aikaa osatoimituksien avulla. Rajattujen kokonaisuuksien valmistuminen iteraatiovälein mahdollistaa testaustoimenpiteiden aloittamisen ajoissa, sekä testauksen suorittamisen kehitystyön rinnalla. Siitä huolimatta, että monessa organisaatioyksikössä ketterät menetelmät tunnustettiin nopeammiksi kehitysmenetelmiksi, suunnitelmalähtöisiä menetelmiä pidettiin turvallisempana vaihtoehtona. Esimerkiksi organisaatioyksikkö D:n suunnittelija myönsi ketterien menetelmien olevan nopeampia, mutta suunnitelmalähtöistä menetelmää käytettiin, koska sen avulla saatiin varmempia tuloksia. Aineistossa esiintyi myös väite, että ketterät menetelmät olisivat loppujen lopuksi hitaampia menetelmiä ja ne vain venyttäisivät aikataulua. Lisäksi organisaatioyksikkö K:n testaaja oli sillä kannalla, että ohjelmistotestauksen toteutus ei ollut riippuvainen kehitysmenetelmästä: samat asiat testataan, käytetään sitten ketterää tai suunnitelmalähtöistä tuotantomenetelmää.

Työn toinen tärkeä havainto oli, että ketterien menetelmien käyttö tasaa testausresurssien kuormitusta mutta ei kuitenkaan vähennä itse testausresurssien tarvetta. Ketterissä menetelmissä testaus kuuluu projektiin jo alusta lähtien, joten testaajat ovat kokoajan mukana projekteissa. Testausta tehdään koko projektin ajan, jolloin testaukseen on enemmän aikaa ja testaus ei välttämättä muodosta suurta työkuormaa projektin loppuun. Aineistossa esiintyi myös eriäviä mielipiteitä ketterissä menetelmissä tarvittavista testausresursseista. Organisaatioyksikkö I:n testaaja oli sitä mieltä, että ketterien menetelmien käyttö lisäisi tarvittavien testausresurssien määrää vähäi-

sen suunnittelun vuoksi. Ketterää tuotantomenetelmää käyttävän organisaatioyksikkö C:n testaus perustui kuitenkin testaussuunnitelmiin enemmän kuin tutkivaan testaukseen. Lisäksi organisaatioyksikkö F:n suunnittelijan mukaan ketterien menetelmien käyttö ei estä suunnitelmien ja arkkitehtuurin tekoa.

Itkosen et al. (2005) tutkimus tukee tässä työssä tehtyä havaintoa siitä, että ketterissä menetelmissä testaukselle pyritään järjestämään lisää aikaa siten, että testausta tehdään jatkuvasti projektin aikana. Itkonen et al. (2005) käyttävät tutkimuksessaan CoC-aikakehystä kuvaamaan testaustoimenpiteiden sijoittumista ketterässä prosessimallissa. Testauksen liittäminen projekteihin aikaisemmin tuo mukanaan kuitenkin uusia ongelmia kehitysprosessiin. Esimerkiksi Puleio (2006) ja Stolberg (2009) kohtasivat ongelmia testauksen toteuttamisessa ketterissä menetelmissä, koska testaus piti sisällyttää iteraatioihin. Vaikka testaus voidaan aloittaa aikaisemmin projekteissa, testaustoimenpiteiden sovittaminen nopeisiin iteraatioihin on haastavaa. Sekä Puleio että Stolberg, löysivät ratkaisun tähän ongelmaan ja lopulta pitivät testauksen ja kehitystyön rinnakkaista suorittamista järkevämpänä ratkaisuna, kuin testauksen suorittamista vasta projektin lopussa.

Testausresurssien suhteen on ollut keskustelua siitä, tarvitaanko ketterissä menetelmissä erillisiä testaajia lainkaan. Esimerkiksi Talby et al. (2006) kyseenalaistavat erillisten testaajien tarpeen, jos testaustoimenpiteiden suorittamisesta tehdään luonnollinen osa kehittäjien työtehtäviä. Tämän työn tulokset antavat kuitenkin syyn olettaa, että erilliset testaajat ovat tarpeen myös ketteriä menetelmiä käytettäessä. Organisaatioyksikkö C tarvittaessa osti ulkopuolisia testauskonsultteja, koska omia testaajia ei ollut riittävästi käytettävissä. Lisäksi organisaatioyksikössä F korostettiin erillisten testaajien tarvetta, koska heillä ei erillisiä testaajia ollut käytössä. Oman työn ohella testausta ei välttämättä ehditty tekemään, koska ensisijaiset työtehtävät olivat tärkeämpiä. Myös Shayen (2008) tutkimuksessa testaajat kuuluivat oleellisena osana kehitysprosessiin. Testaajat toimivat yhteistyössä kehittäjien kanssa, jolloin kehittäjät saivat palautetta uusista toiminnoista päivittäin.

Muut työntulokset liittyivät läheisesti asiakkaaseen sekä yhteistyöhön asiakkaan ja toimittajan välillä. Ketterissä menetelmissä nojataan voimakkaasti asiakkaan läsnäoloon ja yhteistyöhön, joten asiakkaalta vaaditaan erilaista ajatusmallia kuin suunnitelmalähtöisiä menetelmiä käytettäessä. Lisääntynyt asiakkaan ja kehittäjäorganisaation välinen kommunikointi edesauttaa molemminpuolista ymmärtämistä kehitysprojektista. Asiakas pääsee tiiviimmin osalliseksi kehitystyöhön ja siten voi lisätä ymmärrystään ohjelmistokehityksestä. Asiakkaan läsnäolo helpottaa myös kehittäjien työtä, koska he voivat tarkentaa epäselviä vaatimuksia tai toiminnallisuuksia suoraan asiakkaalta. Suoran kommunikoinnin tehokkuus korostuu, mutta toisaalta tehokas kommunikointi voi johtaa puutteelliseen dokumentointiin.

Hypoteesien ulkopuolelle jäi muutamia havaintoja ohjelmistotestaukseen liittyen. Tutkimukseen osallistuneissa organisaatioyksiköissä ohjelmistotestauksen organisoinnin ja testaustyön arvostuksen välillä oli nähtävissä yhteys. Erillisiä testaajia ei rekrytoitu esimerkiksi organisaatioyksiköihin A ja F, koska johto ei kokenut testaustyötä tärkeäksi. Lisäksi joissain tapauksissa testaukselle ei varattu riittävästi aikaa, vaikka organisaatioyksiköllä olisi ollut erilliset testaajat käytössään. Toinen hypoteesien ulkopuolelle jäänyt havainto liittyi testaajien osaamiseen ketterissä menetelmissä. Aineistossa esiintyi havainto, että ketterissä menetelmissä testaajalta vaaditaan enemmän osaamista. Organisaatioyksikkö C:n testaajan mukaan testaustyö oli vaikeampaa käytettäessä ketterää tuotantomenetelmää. Hypoteeseissa ei myöskään keskitytty testausautomaatioon, jota käytettiin kuitenkin lähes joka toisessa organisaatioyksikössä. Esimerkiksi Organisaatioyksikkö C:n testauksen nopeus pyrittiin säilyttämään testien automatisoinnilla.

Tutkimuksen tulokset voivat olla hyödyllisiä ketteriä menetelmiä harkitseville organisaatioille. Tuloksissa on arvokasta tietoa siitä, mitä etuja ketterien menetelmien käytöllä voidaan saavuttaa suunnitelmalähtöisiin menetelmiin verrattuna, ja mitä ketterien menetelmien käyttö vaatii etenkin ohjelmistotestauksen osalta. Lisäksi havainnoissa esiintyy tietoa siitä, millaiset asiat tukevat menetelmien käyttöönottoa ja millaisissa tilanteissa menetelmien käytöstä on todettu olevan hyötyä.

Tämän työn tuloksien pätevyyttä rajoittaa hieman se, että vain yhdessä organisaatioyksikössä tietoisesti käytettiin ketterää menetelmää, Scrumia. Ketteriä menetelmiä haluttiin käyttää myös organisaatioyksikössä F, mutta käytännön toteutus oli jäänyt hieman vajaaksi. Ketterille menetelmille tyypillisiä piirteitä käytettiin kuitenkin useammassakin organisaatioyksikössä ja lähes kaikki tutkimukseen osallistuneet organisaatioyksiköt olivat kokeilleet ketteriä menetelmiä. Toisaalta ketterien menetelmien käytön harvinaisuudesta voidaan päätellä, että menetelmiä ei ollut otettu käyttöön kovin laajasti tutkimuksen tekohetkellä.

9 YHTEENVETO

Tämän tutkimuksen tarkoituksena oli löytää vastauksia siihen, kuinka ketterät menetelmät vaikuttavat ohjelmistotestauksen toteuttamiseen. Vastauksia pyrittiin löytämään aineistopohjaisen menetelmän avulla. Aineisto koostui haastattelujen nauhoituksista, jotka oli kerätty haastattelemalla 12 organisaatioyksikön suunnittelijaa, projektipäällikköä sekä testaajaa kolmen haastattelukierroksen aikana. Toisella haastattelukierroksella kerättiin myös tilastollista tietoa yhteensä 31 organisaatioyksiköstä. Tälle tutkimukselle tärkeimmät tilastotiedot esiteltiin luvussa 5.

Aineistopohjaista menetelmää noudattaen, ensin aineistossa esiintyneet ilmiöt ja havainnot hajautettiin osiin. Seuraavassa vaiheessa havainnot luokiteltiin kategorioihin ja sen jälkeen muodostettiin yhteydet kategorioiden välille. Analyysin aikana aineistosta löytyneet havainnot luokiteltiin tutkimuskysymyksen kannalta tärkeisiin kategorioihin. Näiden kategorioiden ja havaintojen perusteella muodostettiin kuusi hypoteesia ohjelmistotestaukseen ja ketteriin menetelmiin liittyen.

Tutkimuksen aikana muodostettujen hypoteesien perusteella vaikuttaisi siltä, että ketterissä menetelmissä ohjelmistotestauksen asema kohenee ja sen lisäksi testausresurssien kuormitus on tasaista. Ketteriä menetelmiä käytettäessä testaus pitää huomioida projektin alusta lähtien, jolloin testaus kuuluu kehitysprosessiin jatkuvasti. Kun kehitys ja testaus suoritetaan rinnakkaisesti, testaus ei muodosta suurta työkuormaa projektin loppuun, tällöin myös virheiden korjaamiseen on enemmän aikaa. Vaikka ketterissä menetelmissä kannustetaan kehittäjiä testaamaan tuotetta, erillisten testaajien tarve ei poistu.

Ketterien menetelmien periaatteiden mukaisesti, asiakkaan on oltava aktiivisemmin mukana kehitysprosessissa. Ketterissä menetelmissä asiakkaan ja toimittajan välistä yhteistyötä ja kommunikointia on lisättävä, jotta kehitys voi olla joustavaa ja nopeaa. Tämä lähestymistapa vaatii ajatusmallin muutosta sekä toimitus- että asiakasorgani-

saatioissa, sillä molempien osapuolien on ymmärrettävä iteratiivisen kehityksen piirteet. Lisäksi asiakkaan ja toimittajan välillä on oltava hyvä luottamussuhde. Tässä tutkimuksessa havaittiin, että sisäinen asiakas tukee ketterien menetelmien käyttöä, koska sisäisen asiakkaan kanssa luottamussuhde on valmiiksi kunnossa.

Tämä tutkimus tehtiin osana suurempaa MASTO-hanketta. Jatkossa tutkimuksen tuloksia hyödynnetään MASTO-hankkeessa ohjelmistotestaukselle tehtävän referenssimallin rakentamisessa. Yksittäisistä tuloksista on hyötyä organisaatioille, jotka haluavat käyttää ketteriä menetelmiä kehitystyössään. Ohjelmistotestaukseen liittyvät tulokset tarjoavat arvokasta tietoa organisaatioille testauksen organisoinnin ja testausstrategian määrittelemisessä, kun halutaan siirtyä ketterämpään kehitysmalliin.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta J. (2002), Agile Software Development Methods: Review and Analysis. *VTT Publications 478*.

Ambler, S. (2008), Agile Software Development at Scale. *Balancing Agility and Formalism in Software Engineering*, 5082, 1-12, Springer, Berlin.

ATLAS.ti - The Knowledge Workbench (2005). Scientific Software Development. Saatavissa: http://www.atlasti.de/ [Viitattu 13.8.2009]

Beck, K. (1999), Embracing change with extreme programming. *Computer*, 32(10), 70-77.

Bertolino, A. (2007), Software Testing Research: Achievements, Challenges, Dreams. *Future of Software Engineering, 2007 FOSE '07*, 85-103.

Boehm, B. (2002), Get Ready for Agile Methods, with Care. *Computer*, 35 (1), 64-69.

Boehm, B. & Turner, R. (2004), *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison Wesley, Boston.

Boehm, B. (2006), A View of 20th and 21st Century Software Engineering. *Proceedings of the 28th international conference on Software engineering*.

Coghlan, D. & Brannick, T. (2005), *Doing Action Research in Your Own Organization*, 2nd edition, SAGE.

Cohen, C. F., Birkin, S. J., Garfield, M. J. & Webb, H. W. (2004), Managing conflict in software testing. *Communications of the ACM*, 47(1), 76-81.

Cusumano, M., MacCormack, A., Kemerer, C. F. & Crandall, B. (2003), Software Development Worldwide: the State of the Practice. *IEEE Software*, 20(6), 28-34.

Fitzgerald, B., Russo, N. L. & Stolterman, E. (2002), *Information Systems Development - Methods in Action*, McGraw-Hill, London.

Fowler, M. & Highsmith, J. (2001), The Agile Manifesto.

Glaser, B. & Strauss, A., L. (1967), *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine, Chicago.

Glaser, B. (1992), *Basics of Grounded Theory Analysis: Emergence vs. Forcing,* Sociology Press, Mill Valley.

Haikala, I. & Märijärvi, J. (2004), *Ohjelmistotuotanto,* Talentum, Helsinki.

Highsmith, J. & Cockburn, A. (2001), Agile Software Development: the Business of Innovation. *Computer*, 34(9), 120-127.

Hirsjärvi, S., Remes, P. & Sajavaara, P. (2009), *Tutki ja kirjoita*, 15. painos, Kariston Kirjapaino Oy, Hämeenlinna.

ISO, Information Technology – *Process Assessment – Part 1: Concepts and Vocabulary*, ISO/IEC 15504-1.

ISO, International Software Testing Standard, ISO/IEC 25010.

ISO, International Software Testing Standard, ISO/IEC 29119.

Itkonen, J., Rautiainen, K. & Lassenius, C. (2005), Towards Understanding Quality Assurance in Agile Software Development. *Proceedings of the International Conference on Agility (ICAM 2005)*, 201-207.

Itkonen, J. & Rautiainen, K. (2005), Exploratory Testing: A Multiple Case Study, *Proceedings of the International Symposium on Empirical Software Engineering* (*ISESE 2005)*, 84-93.

Judy, K. H, & Krumins-Beens, I. (2007), Using Agile Practices to Spark Innovation in a Small to Medium Sized Business. *40th Annual Hawaii International Conference on System Sciences (HICSS 2007)*, 275b.

Kit, E. (1995), Software Testing in the Real World: Improving the Process, Addison-Wesley, Reading, MA.

Luomansuu, R. (2009), Tilastollinen tutkimus ohjelmiston laatuun vaikuttavista tekijöistä, Lappeenranta University of Technology.

Myers, G. J. (2004), *The Art of Software Testing*, 2nd edition, John Wiley & Sons, Inc., Hoboken, New Jersey.

Nandhakumar, J. & Avison, J. (1999), The fiction of methodological development: a field study of information systems development. *Information Technology & People*, 12(2), 176-191.

Nerur, S., Mahapatra, R. & Mangalaraj, G. (2005), Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5), 72-78.

Puleio, M. (2006), How Not to Do Agile Testing. *Proceedings of AGILE 2006 Conference (AGILE'06)*, pp. 7 pp.-314.

Roosmalen, R. (2007), Testing and Scrum, Fall 2007 Scrum Gathering: London. Saatavissa: http://www.scrumalliance.org/resources/270 [Viitattu 14.7.2009]

Salo, O. & Abrahamsson, P. (2008), Agile Methods in European Embedded Software Development Organisations: a Survey on the Actual Use and Usefulness of Extreme Programming and Scrum. *IET Software*, 2(1), 58-64.

Schwaber, K. & Beedle, M. (2002), *Agile Software Development with Scrum*, Prentice-Hall, Upper Saddle River, NJ.

Seaman, C. (1999), Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, 25(4), 557-572.

Shaye, S. (2008), Transitioning a Team to Agile Test Methods. *Agile 2008 Conference (AGILE '08)*, 470-477.

Sommerville, I. (1995), *Software Engineering*, 5th edition, Addison Wesley.

Stolberg, S. (2009), Enabling Agile Testing Through Continuous Integration. *Agile Conference, 2009 (AGILE '09)*, 369-374.

Strauss, A. & Corbin, J. (1990), *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*, SAGE Publications, Newbury Park, CA.

Sumrell, M. (2007), From Waterfall to Agile - How does a QA Team Transition? *AGILE 2007*, 291-295.

Sutherland, J. (2001), Agile can scale: inventing and reinventing SCRUM in five companies. *Cutter IT Journal*, 14 (12), 5-11.

Taipale, O. (2007), *Observations on Software Testing Practice*, Lappeenranta, Acta Universitatis Lappeenrantaensis.

Vriens, C. (2003), Certifying for CMM Level 2 and IS09001 with XP@Scrum. *Proceedings of the Agile Development Conference (ADC 2003)*, 120-124.

LIITE 1. Ensimmäisen haastattelukierroksen kysymykset

Laadullisen tutkimuksen kysymykset

Ensimmäinen haastattelukierros (suunnittelijat)

- *Jatkokysymyksiä käytetään tarvittaessa, ei pakollisia*

Tunnistetiedot

Päivämäärä, haastateltavan nimi, organisaatioyksikkö, tehtävä

Aihe 1: Suunnittelu- ja toteutusmenetelmät

1.1 Millaisia ohjelmiston toteutusmenetelmiä teillä käytetään? Hyödynnetäänkö teillä ketteriä menetelmiä? Millä tavoin?/Miksi ei?

1.2 Millainen tuotteidenne kriittisyystaso on? Vaihteleeko se? Jos kyllä, niin vaikuttaako tämä suunnittelu- ja toteutusmenetelmien valintaan?

Aihe 2: Testausstrategiat ja -resurssit

2.1 Miten tehtäväsi liittyvät testaukseen?

2.2 Kuinka päätätte, mitä testataan? Omat kokemuksenne tämän testausstrategian kattavuudesta/toimivuudesta?

2.3 Mitä osaa testauksesta haluaisit kehittää? Miksi?

2.4 Vaikuttaako lopputuotteen kriittisyys testausstrategiaan? Kuinka?[1]

2.5 Onko testaus riittävästi resursoitu? Jos ei, niin miksi arvioisit näin olevan? Mitä mielestäsi asialle pitäisi tehdä?

Aihe 3: Ketterät menetelmät

Kysytään jos yritys sanoo käyttävänsä ketteriä menetelmiä

3.1 Millaisia kokemuksia teillä henkilökohtaisesti on ketterien menetelmien toiminnasta?

3.2 Vaikuttaako ketterien menetelmien käyttö komponenttien uudelleenkäyttöön tai laatuun? Millä tavoin?

3.3 Vaikuttaako ketterät menetelmät testauksen resurssitarpeeseen? Entä aikatauluihin? Millä tavoin?

[1] Kysytään jos tuotteiden kriittisyys vaihtelee

Aihe 4: Standardit

4.1 Käytättekö standardeja ohjelmistotuotannon tukena? Jos kyllä, niin mitä? Omat kokemuksenne/mielipiteenne standardin vaikutuksesta ohjelmistoprosessiin ja lopputuotteeseen.

- ISO 29119

4.2 Seuraatteko testausprosessin laatua/tehokkuutta? Käytättekö tähän jotain standardeja? Jos kyllä, niin mitä? Jos ette seuraa, niin miksi arvioisit näin olevan?

- Mitä mittareita
- Miten ulkoistetun testausprosessin seuranta

Aihe 5A: Ulkoistaminen

5.1 Millaista tietämystä tuotteidenne testaus vaatii? Mistä tämä tietämys on hankittavissa?

5.2 Hankitteko testauspalvelua tai ohjelmistoja ulkoisilta tuottajilta? Mitä palveluja/komponentteja?/Miksi ette?

Aihe 5B: Ulkoistaminen jatkokysymykset

Kysytään jos yritys on ulkoistanut testausta/ostanut komponentteja

5.3 Tukeeko tuotantomenetelmä mielestänne ulkoa hankittavan testauspalvelun käyttämistä? Miksi? (Entä jos kyseessä on kriittinen tuote?[1])

5.4 Tukeeko tuotantomenetelmä mielestänne ulkoa hankitun komponentin/ohjelmiston käyttämistä osana tuottetta? Miksi?

5.5 Vaikuttaako ulkoistamisen tuotteen testausstrategiaan? Miksi?

Aihe 6: Testausautomaatio, testauspalvelut sekä –työkalut

6.1 Käytättekö testausautomaatiota? Jos kyllä, mitä asioita olette automatisoineet? Jos ette, niin miksi?

6.2 Millainen mielikuva/Millaisia kokemuksia teillä on testausautomaatiosta ja sen käytöstä?

6.3 Oletteko löytäneet tai käyttäneet verkosta testauspalveluja ja tuotteita? Jos kyllä, niin mitä? Millaisia verkosta saatavia testauspalveluja haluaisitte? Miksi?

6.4 Mitä testauspalveluja/tarjontaa haluaisitte jo käytössä olevien lisäksi? Miksi?

Aihe 7: Laatu sekä Asiakkaan ja toimittajan väliset suhteet

7.1 Miten määrittelette laadun, eli mikä osa laadusta on teille tärkeää. Miten laatutavoite näkyy tuotekehityksessä ja testauksessa? ISO 25010:

- Toiminnallisuus
- Luotettavuus
- Tehokkuus
- Käytettävyys
- Tietoturvallisuus
- Yhteensopivuus
- Ylläpidettävyys
- Siirrettävyys

7.2 Vaikuttaako lopputuotteen kriittisyys laatumääritelmään? Miten? Miksi?[2]

7.3 Vaikuttaako ulkoa hankittujen palveluiden/komponenttien määrä laatuun? Miten? Miksi?

7.4 Miten asiakkaiden osallistuminen vaikuttaa laatuun?

- Suuri kokoero toimittaja-asiakassuhteessa
- Toimittajan ja asiakkaan välinen luottamus
- Asiakastyytyväisyys

Aihe 8: Muita tärkeitä asioita?

8.1 Mihin tutkimusalueeseen mielestäsi pitäisi panostaa?
-Testauspolitiikka
-Testausstrategia
-Testauksen hallinta
-Testauksen toteutus

8.2 Jäikö mielestäsi jotain keskeistä kysymättä tai sanomatta?

[2] Kysytään jos tuotteiden kriittisyys vaihtelee

LIITE 2. Toisen haastattelukierroksen kysymykset

Industry survey: Software testing and quality

The scope of the survey is organizational units and their main software products.

General information of the organizational unit

1. Interview

- Date
- Place
- Interviewer
- Interview started

2. Respondents

Name	Occupation	Responsible for development /testing/both

3. Company

- Name
- Organizational unit (OU)
- Industry sector

4. Personnel, methods, and automation

- Number of employees in the whole company?
- Number of SW developers and testers in the OU?
- Percentage of automation in testing?
- Percentage of agile (reactive, iterative) vs plan driven methods in projects?
- Percentage of existing testers vs resource need?

5. Please, estimate the distribution of the turnover in your OU.

Percentage of the turnover (0 - 20%, 21 - 41%, 41 - 60%, 61 - 80%, 81 - 100%)

- Product: Customized product (based on a product kernel)
- Product: Uniform product kernel in all deliveries
- Product: Product family composed of distinct components
- Product: Standardized online service product (e.g. product/service prov.)
- Service: Training and consulting

- Service: Subcontracting
- Service: System integration
- Service: Installation service
- Service: Self service (e.g. service/service provider)
- Other, specify:

Processes and tools

6. Please, estimate how the following claims describe your software development.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- We like to transfer knowledge more by face-to-face conversation than by documents as the primary method of knowledge transfer.
- Progress of the software is more important than thorough documentation.
- Business people and developers work daily together in the projects.
- Our process is able to cope with late changes in requirements, design, and technical platform.
- We prefer more individuals, collaboration, and interaction than processes and tools.

7. Faults in your products can cause (please, select all suitable points)

Irritation and dissatisfaction

Disturbance in the normal operation of the organization or a person

Remarkable economical losses

Interruption in the normal operation of the organization or a person

Loss of human life/lives

Other, specify:

8. Please, estimate following claims concerning your software testing. When the claim is not applicable leave the scale empty.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Our software correctly implements a specific function. We are building the product right (human examination).
- Our software is built traceable to customer requirements. We are building the right product.
- Our formal inspections are ok (document to be inspected).
- We go through checklists (req., func., tech., code).
- We keep code reviews.
- Our unit testing (modules or procedures) is excellent.
- Our integration testing (multiple components together) is excellent.
- Our usability testing (adapt software to users' work styles) is excellent.
- Our function testing (detect discrepancies between a program's functional specification and its actual behaviour) is excellent.
- Our system testing (system does not meet requirements specification) is excellent.
- Our acceptance testing (users run the system in production) is excellent.
- We keep our testing schedules.
- Last testing phases are kept regardless of the project deadline.
- We allocate enough testing time.

9. Please, estimate following claims. New testing standard ISO/IEC 29119

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Quality is built in development.
- Quality is built in testing.
- Our test policy is excellent (principles, approach, and high-level objectives).
- Our test strategy is excellent (a reusable set of guidelines for all projects).
- Our test management is excellent (strategizing, planning, monitoring, control, and reporting of testing).
- Our test execution is excellent (testing within a particular level of testing (e.g. unit, integration, system or acceptance) and/or type of testing (e.g. performance testing, security testing, functional testing).

10. Do you follow a systematic method or process in the software testing (e.g. TPI, or standard, or your own specified process)?

- No
- To a certain extent; which one:
- Yes; which one:

11. The available testing tools, if any

Tool	Description/Experiences/Recommendation	In-house	Vendor

Customer participation

12. Please, estimate your most important customer's participation during specification phase of the development.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Our most important customer is a member of the project team and responsible for the definition of the system.
- Our most important customer takes part in the project management schedules and progress reports for the development of the system.
- Our most important customer develops and evaluates the budget for the system.

13. Please, estimate your most important customer's participation during design phase of the development.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Our most important customer is a member of the project team for user interface design.
- We develop a prototype for our most important customer.
- Our most important customer defines system controls and security procedures.

- Our most important customer defines and reviews technical designs.

14. Please, estimate your most important customer's participation during testing phase.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Our most important customer develops test specifications.
- Our most important customer evaluates test data specifications developed by us.
- Our most important customer reviews results of system test done by us.
- Our most important customer conducts the system tests.

15. Please, estimate your most important customer's participation in general control.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Needed new features are paid by our most important customer.
- Our most important customer reviews project management schedules and progress reports made available by us.
- Our most important customer provides domain training to us.
- Our most important customer's employees are evaluated by their own management in our collaboration projects.

Quality vs customer – supplier relationship

16. Please, estimate the following claims. Relationship to the customer.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Our most important customer has experience on the area of business.
- Our most important customer has power on the area of business.
- Our most important customer has strict contractual agreements.
- Our most important customer has requests and suggestions.
- Our most important customer co-operates and communicates excellently with us.

17. Please, estimate the following claims. Trust.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Our most important customer is concerned about our welfare and best interests.
- Our most important customer considers how their decisions and actions affect us.
- We trust our most important customer.
- Our most important customer trusts us.

Software quality

18. Do you have a quality system certificate or a capability-maturity classification (e.g. CMM, SPICE, ISO-9001)?

- No
- Yes; which one:

19. Please, estimate following claims concerning quality attributes of your software. When the quality attribute is not applicable leave the scale empty.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- The functional stability is excellent. Our software is suitable for functions it is developed for (appropriateness).
- The reliability is excellent. The availability, fault tolerance, and recoverability of our software are excellent.
- The performance efficiency is excellent. Our software consumes a reasonable amount of resources and time.
- The operability is excellent. Our software is useful and usable according to the users (ease of use).
- The security is excellent. The security issues (malicious access, use, modification, destruction, or disclosure) have been taken into account.

- The compatibility is excellent. Our software is compatible with relevant software or components.
- The maintainability is excellent (e.g. modifications after delivery).
- The transferability is excellent. Our software can be transferred to another platforms.
- The installability is excellent. Our software can be installed (first time) with relative ease to the operating environment.
- The updateability is excellent. Our software can be updated (after first installation) with relative ease to the operating environment.

20. Please, estimate the following claims related to your software.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- We have identified the most important quality attributes.
- We have prioritized the most important quality attributes.
- We have documented the most important quality attributes.
- We have communicated the most important quality attributes within our OU using some other way than documentation.
- We follow regularly through measurement the achievement of the most important quality attributes.

21. How many percent of the development effort is spent on testing?

22. Please, estimate following claims concerning problems.

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Complicated testing tools cause test configuration errors.
- Commercial testing tools do not offer enough hardware support.
- It is difficult to automate testing because of low reuse and high price.
- Insufficient communication slows the bug-fixing and causes misunderstanding between testers and developers.

- Feature development in the late phases of the product development shortens testing schedule.
- Testing personnel do not have expertise in certain testing applications.
- Existing testing environments restrict testing.

23. Please, estimate following claims concerning enhancement proposals?

Scale: 1=fully disagree, 3=neutral, 5=fully agree

- Fault database hepls in observing testing process.
- Test report automation decreases testers' work load.
- New features are not allowed after a set deadline to help test planning.
- Test results should be open for all process groups.
- Products should be designed to promote testability.
- Testing strategy helps in avoiding unnecessary testing.
- We should have dedicated testers.
- Development of testing environment makes testing more efficient.

24. Name and explain the three most significant factors how customer affects software quality in projects (in descending order).

25. Name and explain the three most efficient tools or methods of test automation (in descending order).

26. Name and explain most important advantages and disadvantages of outsourcing in development and testing.

Comments

Interview ended:

LIITE 3. Kolmannen haastattelukierroksen kysymykset

Laadullisen tutkimuksen kysymykset

Tunnistetiedot

Päivämäärä, haastateltavan nimi, organisaatioyksikkö, tehtävä

Aihe 1: Testausmenetelmistä

1.1 Millaisia testausmenetelmiä ja –vaiheita (yksikkö, integrointi, käytettävyys, alpha jne) teillä käytetään?

1.2 Vaikuttaako tuotteen käyttökohde tai kriittisyys testaukseen? Vaihteleeko testaustavat projektien välillä? Jos kyllä, niin kuinka? Jos ei, niin pitäisikö? (Selitä kriittisyys käsitteenä)

Aihe 2: Testausstrategiat ja -resurssit

2.1 Miten tehtäväsi liittyvät testaukseen?

2.2 Kuinka päätätte, mitä testataan? Omat kokemuksenne tämän testausstrategian kattavuudesta/toimivuudesta?

2.3: Testauksen dokumentointi
- Kuinka tarkalla tasolla, testitapauksenne ja -suunnitelmanne on dokumentoitu (kuvaile kuinka yksityiskohtaisia testitapaukset ovat)
- Minkälainen testidokumentaatio on sinulle testaajana tärkeintä käytännössä?
- Teettekö testausta ilman etukäteen suunniteltuja testitapauksia (tutkiva- / ekploroiva testaus)

2.4 Voiko testauksen tarve vaikuttaa tuotteen julkaisuun? Kuinka? Mistä arvioisit tämän johtuvan?

2.5 Miten testaus on resursoitu? Jos heikosti, niin miksi arvioisit näin olevan?

2.6 Haluaisitko kehittää jotain osaa testauksesta? Mitä/Miksi?

Aihe 3: Testaus ja ketterät menetelmät

3.1 Käytetäänkö yrityksessänne ketteriä menetelmiä? Vaikuttaako se testausstrategiaan/tapoihin? Entä aikatauluihin?

3.2 Vaikuttaako ketterien menetelmien käyttö lopputuotteen tai komponenttien laatuun? Entä resurssitarpeeseen?

Aihe 4: Standardit

4.1 Käytättekö standardeja testaustoiminnan tukena? Jos kyllä, niin mitä, miten ne vaikuttavat? Omat kokemuksenne/mielipiteenne standardin vaikutuksesta

4.2 Seuraatteko testausprosessin tehokkuutta tai kattavuutta? Käytättekö tähän jotain mittareita? Jos kyllä, niin mitä? Jos ette seuraa, niin miksi arvioisit näin olevan?

- Miten ulkoa hankitun moduulin seuranta?

Aihe 5: Ulkoistaminen

5.1 Millaista tietämystä tuotteidenne testaus vaatii? Mistä tämä tietämys on hankittavissa?

5.2 Hankitteko testauspalveluja ulkoisilta tuottajilta? Mitä?/Miksi ette?

Ulkoistaminen jatkokysymykset

Ohitetaan jos ei ulkoistettua testausta

5.3 Tukeeko testausorganisaationne mielestänne ulkoa hankittavan testauspalvelun käyttämistä? Miksi? (Entä jos kyseessä on kriittinen tuote?[1])

5.4 Vaikuttaako ulkoa hankittujen komponenttien käyttö osana tuotetta sen testausstrategiaan? Miksi? Entä laatuun?

Aihe 6: Testausautomaatio

6.1 Käytättekö testausautomaatiota? Jos kyllä, mitä asioita olette automatisoineet? Jos ette, niin miksi?

6.2 Millainen mielikuva/Millaisia kokemuksia teillä on testausautomaatiosta ja sen käytöstä?

6.3 Kuinka suuri teillä on manuaalisen testauksen osuus testaustyöstä? Mitä se merkitsee lopputuotteen laadulle?

Aihe 7: Testaustyökalut

7.1 Käytättekö erikseen testaukseen tarkoitettuja työkaluja tai ohjelmia? Jos kyllä, niin mitä?

- Oma mielipiteenne näiden työkalujen toimivuudesta

7.2 Onko työkalunne vendor- vai inhouse-tuotantoa? Mistä arvioit tämän johtuvan?

- Oma mielipiteenne vender-testaustyökalujen laadusta ja tehokkuudesta
- Oma mielipiteenne inhouse-työkalujen laadusta ja tehokkuudesta

7.3 Oletteko löytäneet tai käyttäneet verkosta testauspalveluja? Jos kyllä, niin mitä? Millaisia verkosta saatavia testauspalveluja haluaisitte? Miksi?

7.4 Tarvitsisitteko mielestänne jotain testaustyökaluja/palvelua jo käytössä olevien lisäksi? Millaisia/Miksi?

Aihe 8: Laatu

8.1. Tiedätkö mitkä ovat teidän laatutavoitteet järjestelmälle/tuotteelle jota testaat? Mitkä ne ovat? Miten laatutavoite näkyy testauksessa? Jos et, miten määrittelisit ne? ISO 25010:

- Toiminnallisuus
- Luotettavuus
- Tehokkuus
- Käytettävyys
- Tietoturvallisuus
- Yhteensopivuus
- Ylläpidettävyys
- Siirrettävyys

8.2 Vaikuttaako lopputuotteen kriittisyys laatumääritelmään? Miten? Miksi?

Aihe 9: Asiakas projektissa

9.1 Miten asiakkaiden osallistuminen prosessiin vaikuttaa testaukseen? Voiko asiakas vaikuttaa testaussuunnitelmaan tai testicasejen päättämiseen?

- Suuri kokoero toimittaja-asiakassuhteessa
- Toimittajan ja asiakkaan välinen luottamus

Aihe 10: Muita tärkeitä asioita?

10.1 Mihin tutkimusalueeseen mielestäsi pitäisi panostaa? Minkä alan tutkimustiedosta teille olisi eniten hyötyä testauksen suunnittelussa, toteutuksessa tai organisoinnissa?
-Testauspolitiikka
-Testausstrategia

-Testauksen hallinta
-Testauksen toteutus

10.2 Jäikö mielestäsi jotain keskeistä kysymättä tai sanomatta?

LIITE 4. Kategoriataulukko

Kategoria / OU	Testauksen organisointi	Testausosaaminen	Standardoinnin taso	Asiakas	Testausstrategia	Testauksen ongelmat
A	Testausvastuu kehittäjillä	Kehittäjät kokeneita	Ei ole	Jokseenkin mukana kehitysvaiheessa, tekee hyväksymistestausta	Uudet toiminnot ja muutokset, eksp. (tutkiva testaus)	Ei strategiaa suunnitelmaa, resurssit (aika, henkilöstö)
B	Testausvastuu projektipäällikköllä tai asiakkaalla, asiakastuki testaa	Riippuu henkilöstä, toimialatuntemus tärkeää	Virallinen, ISO 9001, ei noudateta tarkasti	Osallistuu testaukseen, asiakassuhde tärkeä, tiivistä yhteistyötä	Uudet ja ydintoiminnot, eksp., keskitetään resurssit	Suunnitelma, strategia, resurssit (aika, henkilöstö)
C	Testausvastuu erillisillä testaajilla, jotka toimivat Scrum - työryhmän jäseninä	Senior-tason testaajat tai testauskonsultteja, tekninen ja toimialatuntemus tärkeää	Omia käytäntöjä, noudatetaan vaihtelevasti	Sisäinen asiakas, tiivis yhteistyö kehityksessä, testauksessa vähemmän	Testataan kaikki tai riskien perusteella, automaatio, nopeus, keskitetään resurssit, testaussuunnitelma	Dokumentointi, testaus vaatii senior-tason testaustietämystä, resurssit (henkilöstö)
D	Kenttätestaus käyttäjillä	Käyttäjän tietotaito	Oma, noudatetaan, testaukselle ei ole prosessia	Asiakas ei ole mukana kehityksessä eikä testauksessa	Ydintoiminnot ja kriittiset kohteet, testausprojektit	Suunnitelma, prosessi, kustannukset, resurssit
E	Testausvastuu kehittäjillä ja suunnittelijoilla	Suunnittelijan (testaajan) tekninen ja toimialatuntemus tärkeää	Oma, perustuu ISO 9001, noudatetaan	Testauksessa mukana, määrittelyssä mukana	Ydintoiminnot, toiminnallisuus ja kattavuus, testaussuunnitelma ja valvonta, autom.	Suunnitelma, valvonta heikkoa, resurssit (aika, henkilöstö)
F	Testausvastuu sisäisellä asiakkaalla	Toimialatuntemus tärkeää	Oma, noudatetaan	Sisäinen asiakas järjestää testauksen, tiivis yhteistyö	Uudet toiminnot ja riskikohteet, eksp., keskitetään resurssit, autom.	Automaatio, ei palkata testaajia, resurssit (aika, henkilöstö)
G	Testausvastuu erillisellä testausyksiköllä	Testaajan toimialatuntemus ja kokemus sekä tekninen osaaminen tärkeää	Oma, perustuu ISO 9000 sarjaan, noudatetaan tarkasti (politiikka)	Sisäinen asiakas määrittelyssä ja kehityksessä mukana, testausta katselmoivat	Uudet sekä ydintoiminnot ja toiminnallisuus, testauspolitiikka, suunnitelma ja valvonta, autom.	Testausautomaation ylläpito, kommunikointi, resurssit jakautuvat epätasaisesti
H	Testausvastuu erillisellä testaustyöryhmällä	Testaajan toimialatuntemus tärkeää	Oma, perustuu CMMI, noudatetaan	Tekee hyväksymistestausta, voi määritellä testejä	Toiminnallisuus, keskitetään resurssit, autom.	Kattavuus, suunnitelma, valvonta, resurssit (aika, henkilöstö)
I	Testausvastuu asiakasneuvojilla, jotka toimivat testaajina	Asiakasneuvojan (testaajan) toimialatuntemus tärkeää, projektipäällikön kokemus tärkeää	Oma, sisäisiä ohjeita, omia menetelmiä, noudatetaan, testauksessa ei	Palaute tärkeää, muuten ei juuri osallistu kehitystyöhön	Uudet toiminnot, toiminnallisuus ja kattavuus, projektipäällikkö kertoo miten, eksp	Strategia, automaation rakennus ja kehitys, resurssit (aika, henkilöstö)
J	Testausvastuu erillisellä testausyksiköllä	Testaajan toimialatuntemus ja tekn. osaaminen tärkeää, tiedettävä myös organisaation toimintatavat	Virallinen, ISO 9001, noudatetaan tarkasti	Hyväksyy määrittelyt ja osallistuu hyväksymistestaukseen, katselmoi edistymistä	Uudet toiminnot, toiminnallisuus ja kattavuus, standardi, testaussuunnitelma, valvonta	Strategia uuden asian testaamisessa, resurssit (aika, henkilöstö)
K	Testausvastuu erillisellä testausyksiköllä	Testaajan toimialatuntemus tärkeää	Oma, (CMMI) noudatetaan tarkasti	Ulkoinen asiakas hyväksyy testaussuunnitelman, sisäinen asiakas katselmoi kehitystä ja testausta	Toiminnallisuus, testauspolitiikka, suunnitelma, vähän eksp.	Resurssit (testausympäristöt, aiheuttaa aikatauluongelmia), muutoksien testaus
L	Testausvastaava ohjaa kehittäjät testaamaan tuotetta	Testausvastaavan kokemus tärkeää, henkilöstön toimialatuntemus tärkeää	CMMI ja oma prosessi testaukselle, noudatetaan	Sisäinen asiakas, tiivis yhteistyö, katselmoi testausta ja on mukana määrittelyjen teossa, tekee hyväksymistestausta	Toiminnallisuus, kattavuus, kuormitus, testausvastaavan suunnitelma, valvonta, autom.	Suunnitelma ja strategia, ei testausammattilaista, hiljaisen tiedon tarve, dokumentointi käytäntöä ei noudateta