Vili Tinakari

TESTAUS PALVELUNA – EROAVAISUUDET PERINTEISIIN TESTAUSMENETELMIIN

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2019

TIIVISTELMÄ

Tinakari, Vili
Testaus palveluna – Eroavaisuudet perinteisiin testausmenetelmiin
Jyväskylä: Jyväskylän yliopisto, 2019, 35 s.
Tietojärjestelmätiede, kandidaatin tutkielma
Ohjaaja(t): Halttunen, Veikko

Testaus on oleellinen ja suhteellisen vähän tutkittu osa informaatioteknologisten järjestelmien ohjelmistojen kehittämistä. Pilvilaskennan avulla testaus voidaan kuitenkin tehdä ulkoistettuna palveluna paljon tehokkaammin kuin perinteisillä testausmenetelmillä, mikä mahdollistaa testauksen kehittymisen pilvilaskennan tasolle. Tutkielmassa tutkittiin perinteisen testauksen yleispiirteitä ja tasoja, joita pyrittiin vertaamaan siihen, kuinka ne eroavat ulkoistetuista testauspilvipalveluista. Tutkimus koostuu kirjallisuuskatsauksesta tukeutuen aikaisempiin tutkimuksiin ohjelmistotestauksesta, järjestelmäkehityksestä ja testauksen ulkoistamisesta palveluksi. Aiemmat tutkimukset painottuivat paljolti itsessään perinteiseen testaukseen tai testauspalveluun erillisinä kokonaisuuksina, mutta vertailevia tutkimuksia oli harvassa. Tutkimuksen tarkoituksena oli vastata tutkimuskysymyksiin, jotka olivat: (1) Kuinka testaus palveluna eroaa perinteisistä testausmenetelmistä ja (2) kuinka yritykset hyötyvät ulkoistetusta testauspalvelusta? Tutkimuksessa käytiin läpi myös ilmenneitä testauspalvelun haasteita ja käytännön mahdollisuuksia ja pyrittiin löytämään hyödyllisiä ja haitallisia eroavaisuuksia testauspalvelusta perinteiseen testaamiseen verraten. Tutkimuksen tuloksina huomattiin, että testauspalvelu on varteenotettava ja suositeltava kilpailija perinteisen testauksen rinnalle, vaikka ulkoistetussa testauspalvelussa on vielä paljon kehitettävää turvallisuusriskien ja kannattavuuden parantamiseksi.

Asiasanat: järjestelmätestaus, järjestelmäkehitys, pilvilaskenta, TaaS, Testing as a service

ABSTRACT

Tinakari, Vili
Testing as a Service – differences to traditional testing
Jyväskylä: University of Jyväskylä, 2019, 35 pp.
Information systems, Bachelor's Thesis
Supervisor(s): Halttunen, Veikko

Testing is an essential and relatively little researched part of developing information technology software. However, with cloud computing, testing can be done as an outsourced service much more efficiently than traditional testing methods, allowing testing to develop to the level of cloud computing. The thesis examined the general characteristics and levels of traditional testing that were being compared to how they differed from outsourced testing as a service. The study consists of a literature review based on earlier studies about software testing, system development and testing as a service. Earlier studies focused largely on traditional testing or testing services as separate entities, but comparative studies were rare. The purpose of the study was to answer research questions that were: (1) How testing as a service differs from traditional testing methods and (2) how do companies benefit from outsourced testing? The study also examined the challenges and practical opportunities that have emerged from the testing service and sought to find useful and harmful differences between testing services and traditional testing. As a result of the research, it was found that the testing service is a viable and recommended competitor alongside traditional testing, although there is still much to be developed in the outsourced testing service to improve security risks and profitability.

Keywords: testing, TaaS, Testing as a Service, system development

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Testaus on tärkeä osa IT-järjestelmäkehitystä. Testauksella pyritään varmistamaan tuotettavien ohjelmistojen laatu ja toimivuus sekä varmistamaan, että tuote vastaa asiakkaiden tarpeita (Naik & Tripathy, 2008). Järjestelmän eri osia voidaan testata eri tavoin, mutta pääsääntöisesti IT-järjestelmistä testataan järjestelmän ohjelmoituja komponentteja eli ohjelmistoja (Sommerville, 2016).

Pilvilaskennan, eli verkkoyhteyksien kautta jaettavien informaatioteknologisten resurssien (Mell & Grance, 2011) yleistymisen myötä on pyritty löytämään pilvilaskennalla mahdollistettuja testausmenetelmiä. Pilvilaskennalla toteutetulle ulkoistetulle testauspalvelulle on yleistynyt termi ”Testing as a Service, TaaS. (Gao, Bai, Tsai & Uehara, 2013.) Testaus palveluna on IT-järjestelmiä kehittävien ja etenkin ohjelmistotuotantoa harjoittavien yritysten ja myös loppukäyttäjien keino testata IT-järjestelmien ohjelmistoja ulkoisella testauspalveluntarjoajalla (Yu, ym., 2010).

Englanninkieliselle termille ”Testing as a Service” ei ole vielä luotu vakioitua suomenkielistä määritelmää, sillä suomenkielistä tutkimusta aiheesta ei ole. Kyseisestä termistä käytetään tässä tutkielmassa suomenkielisiä käännöksiä testauspalvelu tai testaus palveluna, sillä ne kuvastavat TaaS:n ominaispiirteitä paremmin kuin esimerkiksi testipalvelu. Testipalvelun termin heikkoudeksi havaittiin se, että lukija voi mieltää testipalvelun palveluksi, jota vain testataan eikä palveluksi, joka testaa.

Motivaationa aiheen tutkimiseen toimii myös se, että useat tutkijat, kuten Myers, Sandler ja Badgett (2011) sekä Floss ja Tilley (2013), ovat todenneet testauksen olevan tärkeydestään huolimatta vähän tutkittu aihealue. Tutkielmassa oli huomattavissa myös se, että lähteitä testauksesta ja testauspalveluista oli kohtuullisen paljon, mutta lähteiden tieteellinen tutkimus oli suurimmaksi osaksi jäänyt vain esittelevälle ja esimerkiksi palvelumuotoja läpikäyvälle tasolle. Tästä syystä tulee tutkia myös enemmän perinteisiä testausmenetelmiä ja kuinka testaus toteutetaan ulkoistettuna palveluna, mitä jotkin yritykset jo tänä päivänä hyödyntävät. Monessa tutkimuksessa oli myös tuotu esille testauspalvelun hyötyjä ja haittoja eriteltyinä, mutta tutkimuksia, joissa olisi selkeästi ver-

rattu perinteistä ohjelmistokehittäjän talon sisäistä testausta ja ulkoista testauspalvelua, ei löytynyt.

Tutkielman tavoitteena on vastata seuraaviin kysymyksiin: (1) Kuinka testaus palveluna eroaa perinteisistä testausmenetelmistä ja (2) kuinka yritykset hyötyvät ulkoistetusta testauspalvelusta? Näiden kysymyksien vastausten saamiseksi tulee kuitenkin ensiksi tutkia testausta ja testauspalvelua.

Ensimmäisessä luvussa käydään ensiksi läpi perinteisen testauksen ja testausprosessin yleisiä piirteitä. Testauksesta esitellään myös nykyään yleisimmin käytetty testauksen V-malli, jota monet tutkijat, kuten Mili ja Tchier (2015) sekä Naik ja Tripathy (2008) käyttävät tutkielmissaan paljon. Ensimmäisen luvun käsiteltävät aiheet auttavat ymmärtämään kuinka testaus itsessään toimii ja kuinka ohjelmistoja voidaan testata IT-järjestelmissä. Tutkielman ensimmäisessä luvussa käydään myös läpi testauksen haasteita, sillä monet tutkijat, kuten esimerkiksi Naik ja Tripathy (2008) sekä Ammann ja Offutt (2016), ovat tuoneet useita testauksen haasteita esille teoksissaan. Haasteet on hyvä tietää, jotta niihin osataan varautua ja haasteista pystyttään pääsemään eroon.

Tutkielman kolmannessa luvussa kerrotaan pilvilaskennasta sekä pilvipalveluiden yleispiirteistä. Luvussa määritellään tarkemmin pilvilaskennan määritelmä ja esitellään, millainen testauspalvelun toimintakehys on. Kehysmallin esittämisen jälkeen kerrotaan hieman testauspalvelun yleisiä ominaisuuksia ja piirteitä sekä kuinka testauspalveluja voidaan luokitella.

Tutkielman neljännessä luvussa käsitellään, millaisia haasteita tutkimuksissa on havaittu testauspalvelulle ja hieman, kuinka näihin voi varautua. Tämän jälkeen arvioidaan ulkoistetun testauspalvelun hyviä ja huonoja puolia verrattuna perinteiseen organisaatioiden sisällä tapahtuvaan testaukseen. Viimeisimpänä tutkielmassa käydään läpi yhteenvedon muodossa sisältö, tutkimusrajoitteet ja jatkotutkimuskysymykset, joiden avulla testauksen ja testauspalvelun tutkimusta voisi vielä tutkia. Tutkielma painottuu siis kokonaisuudessaan paljolti perinteisiin testausmenetelmiin, testaukseen palveluna sekä testauksen palveluna tuomiin mahdollisuuksiin IT-järjestelmäkehityksessä.

Tutkimusta on rajattu käsittelemään tarpeellisimpia testauksen ja testauspalvelun ominaisuuksia. Esimerkiksi testauksesta esitellään vain yleisimmin esille tulleet testausmuodot, joiden alle voi luokitella useampia testausmenetelmiä. Myös testauspalvelusta oli paljon erilaisia variaatioita, kuten testauspalvelu ainoastaan pilvisovelluksille, mutta tutkielmassa pyritään avaamaan ainoastaan testauspalvelun käsitettä ymmärrettävälle tasolle. Tutkielma muodostuu suurimmaksi osaksi kirjallisuuskatsauksena viitaten pääsääntöisesti testausta palveluna, ulkoistettuja palveluita ja järjestelmäkehitystä tarkasteleviin tieteellisiin teoksiin. Kirjallisuuskatsauksessa tehdään synteettisen tutkimuksen eri tieteellisten teosten tuloksista ja tiedoista ja vertaillaan ja arvioidaan eri tutkimusten yhteneväisyyksiä ja eroavaisuuksia muun muassa testauksesta ja testauspalvelusta.

Suurin osa tutkielmassa käytetyistä lähteistä löydettiin Google Scholarista avainsanojen haun avulla. Testauspalvelun teoksien haussa käytettiin avainsanana pääsääntöisesti englanninkielistä termiä Testing as a Service, lyhennettynä

TaaS ja perinteisiä testausmenetelmiä tutkittaessa monet teoksista löytyi hakusanalla software testing. Muita käytettyjä hakusanoja oli muun muassa software engineering, unit testing, quality assurance, software testing as a service sekä IT-system development. Lähdeaineistoa etsittiin pääsääntöisesti Google Scholarista sen takia, että esimerkiksi yliopiston kirjaston hakukoneella ei löytynyt juurikaan tieteellisiä artikkeleita koskien testauspalvelua. Tämä on osittain selitettävissä sillä, että monet tutkielmassa käytetyt tieteelliset artikkelit olivat osa suurempaa tieteellistä julkaisua muun muassa IEEE:n konferensseista.

2 TESTAUS IT-JÄRJESTELMIEN KEHITYKSESSÄ

IT-järjestelmien yksi tärkeimmistä testattavista asioista on järjestelmien ohjelmistot. Ohjelmistot ovat nykyään oleellinen osa maailmaamme ja nykyajan yritykset pyrkivät löytämään keinoja parantaakseen digitaalisia järjestelmiään pysyäkseen mukana jatkuvasti vaativammassa ohjelmistokilpailussa. Tuotantovalmiit ohjelmistot vaativat paljon testausta ennen itse valmistusvaihetta. (Vocke, 2018.) Testaus on siis oleellinen osa ohjelmistoja sisältävien IT-järjestelmien kehittämistä ja luomista.

Jo vuonna 1979 oli huomattu, että yli puolet ohjelmistoprojektien kehittämisen ajasta ja kuluista oli koostunut testauksesta ja sama pätee tähän päivään (Myers, Sandler & Badgett, 2011). Aiemmissa tutkimuksissa, kuten Sommerville (2016) sekä Naik ja Tripathy (2008), on tullut ilmi, että testauksessa on oleellista varmistaa järjestelmän toimivuus ja haluttujen ominaisuuksien saavutettavuus. Testauksella pyritään vähentämään järjestelmän virhetiloja ja epähaluttuja ominaisuuksia jo ennen varsinaista käyttöönottoa (Sommerville, 2016).

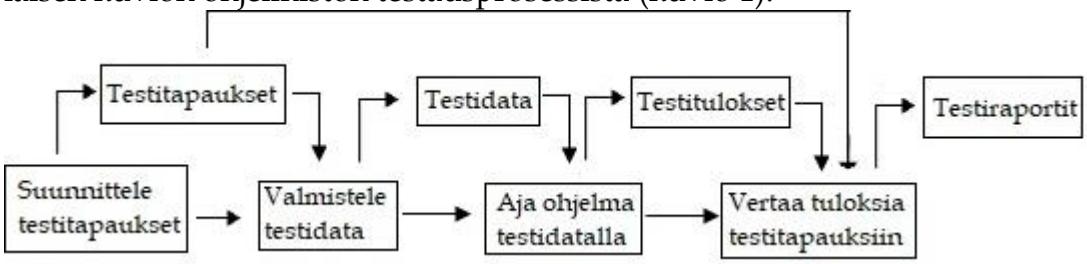
Järjestelmien testaus ei ole kuitenkaan ilmaista järjestelmiä ja ohjelmistoja valmistaville yrityksille. Jopa 40–70 prosenttia ohjelmiston kehityksen kuluista koostuu testauksesta ja osuus kasvaa, mikäli kyseessä on nopeasti toimintaan kehitetty sovellus (Yu, ym., 2010). Tähän suureen prosenttiosuuteen pyritään löytämään ratkaisuja muun muassa testauksen ulkoistamisella palveluksi. Asiaa käsitellään lisää myöhemmässä luvussa, jossa kerrotaan enemmän testauksesta pilvipalveluna.

Seuraavissa alaluvuissa käydään läpi tarkemmin ensiksi testausprosesseja ja testauksen yleispiirteitä. Tämän jälkeen esitellään tyypillisimpiä V-mallin mukaisia testaustasoja, kuten yksikkö-, integraatio- ja järjestelmätestaus sekä aiemmissa tutkimuksissa paljon esille tullut komponenttitestaus. Viimeisenä käsitellään testauksen haasteita, joita on esiintynyt lähdemateriaaleissa.

2.1 Testausprosessi ja testauksen yleispiirteet

Suurin osa järjestelmän testaamisesta itsessään tapahtuu järjestelmäkehityksen toteutuksen jälkeisessä ja käyttöönottoa edeltävässä vaiheessa. Järjestelmäkehitys, engl. system engineering, ovat prosesseja, joissa on tarkoituksena kehittää ja ylläpitää toimivia IT-järjestelmiä. Järjestelmäkehityksen vaiheisiin kuuluu: suunnittelu, analysointi, mallintaminen, implementaatio, testaus ja käyttöönotto sekä ylläpito. Järjestelmäkehityksen osana on myös ja ohjelmistokehitys, engl. software engineering, jossa ohjelmistoa kehitetään ja testataan. Järjestelmäkehityksen kaikissa vaiheissa tulee myös ottaa huomioon asiakkaan vaatimukset ja ohjelmiston haluttujen ominaisuuksien täyttäminen. (Sommerville, 2016.)

Testaus on osa järjestelmäkehitystä ja se on prosessi, joka noudattaa tiettyjä vaiheita. Testausprosessin ensimmäisessä vaiheessa suunnitellaan testaustilanne, joka sisältää muun muassa testiympäristön. Toisessa vaiheessa pitää valmistella testidata testitilanteita varten, eli mahdollisimman kattavat todellisuutta vastaavat tilanteet, joita ohjelmisto käsittelee. Kolmanneksi luodun testidatan avulla tulee suorittaa ohjelma, josta tulee testituloksia. Testituloksia tulee neljännessä vaiheessa verrata oletettuihin arvoihin ja tuloksiin, minkä avulla voidaan laatia testiraportti. Testiraportissa kerrotaan testauksessa ilmenneet eroavaisuudet ja yhtäläisyydet testitapausten odotusten ja varsinaisten testitulosten välillä. (Sommerville, 2016.) Sommerville (2016) on kuvannut seuraavanlaisen kuvion ohjelmiston testausprosessista (kuvio 1).



KUVIO 1 Ohjelmiston testausprosessin malli (Sommerville, 2016)

Testausprosessin kolmatta vaihetta eli ohjelmiston ajamista testidatalla, toistetaan useita kertoja eri testidatamahdollisuuksilla. Tämä mahdollistaa ohjelmiston varmemman toiminnan sekä luotettavamman testaustuloksen. (Mili & Tchier, 2015.) Usein myös testausprosessi itsessään on myös uusittava eri ohjelmiston osien kohdalla. Esimerkiksi eri tarkoitukseen luotuja aliohjelmia ei voida testata samanlaisella testidatalla, jos muun muassa toinen aliohjelma käsittelee tekstiä ja toinen käsittelee numeroita.

Nykyään suosituimmalle järjestelmäkehitystyylille, ketterälle järjestelmäkehitykselle (Urias, 2019), on yleistä, että järjestelmää kehitetään lyhyissä iteroivissa sykleissä. Ketterälle järjestelmäkehitykselle on yleistä, että ohjelmistokehitystä tehdään lyhyissä kehityssykleissä, joidenka lopuilla käsitellään edellisen syklin aikana aikaansaadut tuotokset ja selvitetään mitä uudessa aikasyklissä

tulisi tehdä (Deuff & Cosquer, 2013, s. 1-2). Testaustakin tapahtuu siis lähes jatkuvasti toistuvissa iteraatioissa.

Testaus voidaan jakaa myös valko- ja mustalaatikkotestaukseen (engl. white box testing ja black box testing). Jakaminen tapahtuu sen perusteella, kuinka paljon testattavan ohjelman koodista tiedetään. Valkolaatikkotestauksessa tiedetään tarkemmin ohjelmiston koodia ja sitä pystytään testaamaan enemmän rakenteellisesti. Puolestaan mustalaatikkotestauksessa pyritään testaamaan ohjelmiston toiminnallisuutta, eikä ohjelmiston koodia itsessään nähdä. (Hamill, 2004.)

Testausta kutsutaan usein myös ohjelmiston, tuotteen tai applikaation validoimisen eli selittämisen ja verifioimisen eli tarkistamisen prosessiksi (Singh & Kazi, 2016). Esimerkiksi Sommerville (2016) käsittelee testausta termein validation testing tai verification testing. Ohjelmistoa validoivalla testauksella testaajat pyrkivät selvittämään vastaako hyödyke haluttuja ominaisuuksia eli rakennetaanko oikeaa tuotetta (Sommerville, 2016). Tällä saadaan usein selville vastaako tuote asiakkaan vaatimuksia ja onko siinä kaikki halutut ominaisuudet. Esimerkiksi jos asiakas haluaa järjestelmän laskevan ohimenevän liikenteen määrää, niin asiakas ei silloin tee järjestelmällä mitään, mikäli se vain kuvaa ohimenevää liikennettä, mutta ei laske sen määrää.

Useat tutkijat, kuten Singh ja Kazi (2016) sekä Sommerville (2016), ovat osoittaneet tutkielmissaan validoivan testauksen olevan oleellinen osa järjestelmän kehittämistä, jotta asiakas saadaan otettua mahdollisimman hyvin huomioon. Singh ja Kazi (2016) kertovat muun muassa seitsemän eri syytä, miksi testaus on tärkeää ja niistä kolme keskittyy pelkästään asiakkaan tarpeiden, tyytyväisyyden ja vaivaamattomuuden täyttämiseen. Monet muutkin tutkijat, kuten Whittaker (2009), ovat todenneet, että mitä vähemmän järjestelmän käyttöönoton jälkeen järjestelmässä on havaittu virheitä, sitä enemmän asiakkaat ovat olleet tyytyväisiä järjestelmän valmistajaan. Mikäli järjestelmän selittävän testauksen jättää liian myöhäiseen vaiheeseen ohjelmiston kehitystä, järjestelmän kehityksen kulut saattavat kasvaa (Naik & Tripathy, 2008).

Verifiointitestaukselle tyypillisin kysymys on puolestaan, rakennetaanko tuotetta oikein (Sommerville, 2016). Verifiointitestausta voidaan suorittaa jo ohjelmistokehitysprosessin alkuvaiheissa toisin kuin validointitestausta, jota suoritetaan suurimmaksi osaksi kokonaisjärjestelmätestauksena ohjelmiston ollessa suurimmaksi osaksi valmis. Järjestelmän ohjelmistoa esimerkiksi voi testata jo kooditasolla ja varmistaa, että osa ohjelmiston koodista toimii kuten pitää. (Naik & Tripathy, 2008.) Verifiointitestauksessa ei siis ole niin tärkeää ottaa asiakkaan vaatimuksia huomioon, vaan pikemminkin saada ohjelmisto toimimaan niin kuin sen kuuluu toimia.

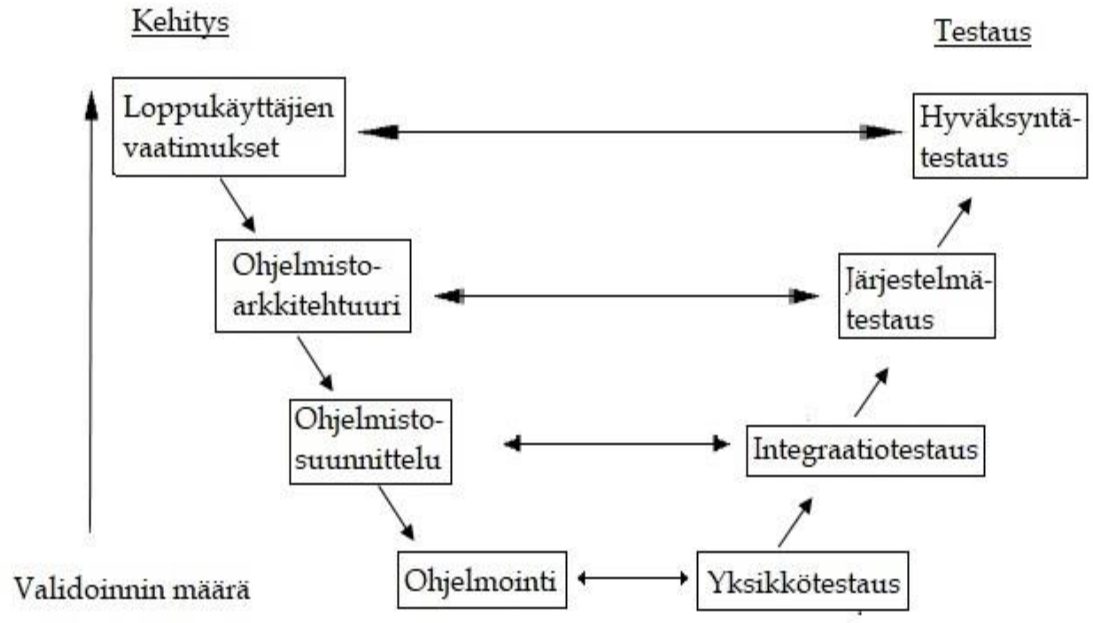
Ohjelmiston toimivuuden ja sen tuottamien ominaisuuksien lisäksi järjestelmän testaus pystytään jakamaan erilaisiin testaustasoihin, joista käydään muutamia läpi seuraavissa alaluvuissa. Tutkielmassa valittiin tarkasteltaviksi testaustasoiksi V-mallin mukaisia tasoja, joista on tehty aikaisemmin luotettavia tutkimuksia ja niitä on verrattavissa toteutettaviksi ulkoisina testauspalveluina. Esimerkiksi käyttöönoton vaiheen testauksessa järjestelmä yhdistetään asiak-

kaan organisaation järjestelmiin (Sommerville, 2016). Testauspalvelulla voisi olla haastavaa tehdä käyttöönottotestausta, mikäli käyttöönotettavat järjestelmät ovat fyysisiä koneita, eikä myöhemmin esitellyn pilvilaskennalla toteutettuja pilvessä toimivia infrastruktuureja.

2.2 Testauksen V-mallin tasot

Monet tutkijat, kuten Naik ja Tripathy (2008) sekä Mili ja Tchier (2015), käyttävät testauksen vaiheista V-mallia. V-mallissa tulee hyvin esille, kuinka ohjelmistokehityksen yhteydessä järjestelmää tulee jatkuvasti testata samalla kuin ohjelmistoprosessi etenee (Mili & Tchier, 2015). V-malli jakaa testauksen neljään eri tasoon, jotka ovat yksikkö-, integraatio-, järjestelmä- ja hyväksyntätestaaminen.

Tyypillisimmät testaustasot jakautuvat selkeästi V-mallissa. Naik ja Tripathy (2008) ja Mili ja Tchier (2015) esittelevät tutkielmissaan kuvion mukaisen V-mallin (kuvio 2). Kuviosta tulee selkeästi ilmi vesiputousmallia mukaileva järjestelmällinen eteneminen järjestelmän kehityksessä ja eri kehitysvaiheiden vastaavat testausvaiheet. Kuvion mukaisesti järjestelmä rakennetaan lähtökohtaisesti loppukäyttäjille, jotka asettavat vaatimukset. Vaatimukset voidaan testata hyväksyntätestauksella. Vaatimusten pohjalta voidaan rakentaa ohjelmistoarkkitehtuurimalli, joka voidaan puolestaan testausvaiheessa testata järjestelmätestauksessa. Arkkitehtuurin pohjalta voidaan suunnitella itse ohjelmisto, jota voidaan testata muun muassa integraatiotestauksella ja ohjelmoinnin aikana itsessään voidaan tehdä yksikkötestejä. Kuviosta myös nähdään, että validoivan testauksen määrä ja loppukäyttäjien huomioiminen yleisesti lisääntyy, mitä kokonaisempana kokonaisuutena järjestelmää kehitetään.



KUVIO 2 Testauksen V-malli (Naik & Tripathy, 2008, s. 16; Mili & Tchier, 2015 s. 33)

Esitellystä V-mallista on jätetty pois kuitenkin muun muassa Sommervillen (2016) esille tuoma komponenttitestaus, joka on sijoitettavissa integraatiotestauksen ja yksikkötestauksen väliin. Naik ja Tripathy (2008) tuovat teoksessaan ilmi myös näkökulman, että komponenttitestaus on osa yksikkötestausta heidän tutkimuksensa mukaisen V-mallin mukaan. Tässä tutkielmassa kuitenkin käsitellään komponenttitestauskin omana kokonaisuutenaan, sillä lähdemateriaalia tutkittaessa oli huomattavissa, että komponenttitestaus on oleellinen testauksen taso. Tutkielmassa ei myöskään käydä tarkemmin läpi hyväksyntätestausta, sillä hyväksyntätestauksen suorittavat järjestelmän loppukäyttäjät, kun taas ohjelmistokehittäjät tekevät yksikkö-, integraatio- ja systeemitestauksen (Naik & Tripathy, 2008).

Seuraavissa luvuissa käsitellään tarkemmin V-mallin mukaiset ohjelmistokehittäjien suorittamat testaustasot, eli yksikkö-, integraatio- ja järjestelmätestaus sekä yleisimmän V-mallin ulkopuolelle jätetty komponenttitestaus. Myös testaustasojen yhteneväisyyksiä käsitellään hieman ja arvioidaan myöhemmässä luvussa, jossa käsitellään palveluna tuotetun testauksen ja perinteisen testauksen eroavaisuuksia, kuinka kehitysvaiheen testaus on sovellettavissa ulkoistettuina pilvipalveluina.

2.2.1 Yksikkötestaus

Yksikkötestauksessa, engl. unit testing, on tarkoituksena testata ohjelmiston tietyn osan, kuten aliohjelman toimivuutta (Myers ym., 2011). Yksikkötestauksen tulee keskittyä etenkin objektien ja metodien, eli ohjelmistojen yksiköiden funktionaalisuuteen (Sommerville, 2016). Yksikkötestaus on siis käytännössä ensimmäinen mahdollinen testausmetodi, minkä ohjelmistolle itsessään voi suorittaa. Yksittäisten ohjelmiston osien testaus edesauttaa myös mahdollisten virheiden paikantamisen jo ohjelmiston varhaisessa kehitysvaiheessa (Myers ym., 2011).

Yksikkötestit muodostuvat ohjelmiston koodin luonnin yhteydessä, eli kun ohjelmiston osaan tulee uusi ominaisuus tai uusi osa koodia, niin kyseistä osaa testataan (Hamill, 2004). Yksikkötestausta tulee siis tehdä yhtäaikaisesti ohjelmistokoodin luonnin yhteydessä. Tällainen menettelytapa on yleistä perinteisessä ja ketterässä ohjelmistokehityksessä (Hamill, 2004), mutta on olemassa myös järjestelmäkehitysmalleja, kuten testauslähtöinen järjestelmäkehitys (TDD), jossa testit rakennetaan ennen ohjelmistokoodia (Sommerville, 2016).

Yksikkötestauksessa on oleellisinta kokeilla kaikkia mahdollisia tilanteita, mitä yksikön tulee käsitellä (Sommerville, 2016). Testausprosessin ensimmäinen vaihe, jossa testitapaukset suunnitellaan, ja toinen vaihe, jossa testidataa luodaan, ovat siis yksikkötestauksen ja koko testausprosessin kannalta oleellisia. Yksikkötestauksessa voidaan antaa erilaisia mahdollisia syötteitä irrallisena yksikkönä toimivalle järjestelmän osalle ja tarkistaa, että se palauttaa oikeanlaisia arvoja takaisin (Sommerville, 2016).

2.2.2 Komponenttitestaus

Komponenttitestauksella tarkoitetaan useamman yksikön yhdistelmän testausta. Komponenttitestauksen tarkoitus on testata järjestelmän yksiköistä muodostettuja osakokonaisuuksia tietyllä tasolla, mikä vastaa lähes järjestelmätestausta pienemmässä mittakaavassa (Naik & Tripathy, 2008). Komponenttitestaus on yleistä etenkin komponenttiperusteisissa järjestelmissä, jossa pyritään luomaan järjestelmiä, joissa on uudelleenkäytettäviä komponentteja (Gao, 2000).

Komponenttitestausta voidaan tehdä myös useammasta yksiköstä muodostettuun moduuliin. Komponenttitestauksessa on oleellista varmistaa, että useammat moduulin yksiköt pystyvät kommunikoimaan keskenään ja niiden käyttöliittymät ovat yhteensopivia (Sommerville, 2016). Komponenttitestaus on siis ikään kuin välivaihe yksittäisten yksiköiden ja kokonaisen järjestelmän testauksessa, jossa varmistetaan, että yksiköt toimivat keskenään komponentteina.

Yksiköiden muodostamilla komponenteilla on monenlaisia erilaisia jaettuja käyttöliittymiä, joiden välille voi muodostua virhetiloja. Tällaisia jaettuja käyttöliittymiä ovat esimerkiksi: parametrien, jaettujen muistien, proseduurien ja viestien välityksen käyttöliittymät. (Sommerville, 2016.) Kaikissa komponenttien välisissä käyttöliittymissä on siis tärkeää varmistaa käyttöliittymien yhteensopivuus, jotta data kulkee halutunlaisena komponentista toiseen (Gao, 2000).

Komponenttitestaus on haasteellista, sillä jotkin virhetilat käyttöliittymien välillä saattavat ilmetä vain poikkeuksellisissa tilanteissa. Komponenttitestauksessa on siis huomioitava paljon enemmän erilaisia testitilanteita kuin esimerkiksi yksikkötestauksessa. (Sommerville, 2016.) Tästä voi todeta, että on tärkeää tehdä huolellisesti ja monipuolisesti jo yksikkötestaukset, jotta komponenttitestauksen ja myöhemmässä vaiheessa tehtävät järjestelmätestaukset onnistuvat tarkemmin.

Sommerville (2016) tuo komponenttitestauksen selkeästi esille kehitysvaiheen testaustasona, mutta esittelee tutkimuksessaan myös V-mallin. Sommerville (2016) ei kuitenkaan yhdistä komponenttitestausta suoraan V-malliinsa, mutta tutkimuksen perusteella komponenttitestaus on toteutettavissa yksikkötestauksen ja integraatiotestauksen välissä. Myös esimerkiksi Ammann ja Offutt (2016) tuovat tutkimuksessaan esille moduulien testauksen, mikä vastaa hyvin paljolti Sommervillen (2016) komponenttitestausta.

2.2.3 Integraatiotestaus

Integraatiotestaus on puolestaan usean komponenttien tai yksikköjen käyttöliittymien yhteensopivuuden testausta. Integraatiotestauksessa on mahdollista testata useampien ohjelmistokehittäjien tekemiä moduuleiksi pakattujen ohjelmiston komponenttien käyttöliittymien ja kommunikaation yhteensopivuutta (Naik & Tripathy, 2008). Integraatiotestauksella pyritään siis löytämään käyttöliittymien ja yksiköiden tai komponenttien välisiä virhetiloja.

Integraatiotestauksen tarkoituksena on testata kokonaisen järjestelmän komponenttien yhteensopivuus sellaiselle tasolle, että järjestelmälle pystytään tekemään järjestelmätestausta myöhemmässä vaiheessa (Naik & Tripathy, 2008). Integraatiotestauksen avulla ohjelmistokehittäjät pystyvät myös varmistamaan, että komponenttien yhteensopivuus vastaa ohjelmistosuunnitelmaa (Mili & Tchier, 2015).

Naik ja Tripathy (2008) tuovat tutkielmassaan esille integraatiotestauksen olevan mahdollista jakaa aiemminkin esille tulleisiin valko- ja mustalaatikkotestaukseen. Mustalaatikkotestauksessa integroitavien komponenttien sisäisiä ominaisuuksia ei oteta huomioon. Komponenttien yhteensopivuutta testataan ikään kuin ne olisivat mustia laatikoita, joiden tarkempia sisältöjä ei tiedetä ja integraatiotestin tulokset määräytyvät lopputuloksesta. Valkolaatikkotestauksessa tuloksissa puolestaan otetaan integraatiotestauksessa huomioon komponenttien sisäiset rakenteet. (Naik & Tripathy, 2008.) Ohjelmistokehittäjien on myös mahdollista tehdä komponentteihin sisäisiä muutoksia nopeammin, kun integraatiotestaus tehdään valkolaatikkoisena (Vocke, 2018).

Integraatiotestaus tulee muistaa myös tehdä joka kerta, mikäli ohjelmistokoodiin tulee muutoksia (Myers, ym. 2011). Integraatiotestaus on myös helpompi tehdä lähes kokonaiselle järjestelmälle ja lisätä esimerkiksi vähittäin testauskokonaisuudessa olevien komponenttien määrää kuin testata suoraan kokonaista järjestelmää (Naik & Tripathy, 2008). Integraatiotestaus on siis erilaisten kokonaisuuksien yhteensopivuuden testausta, jossa testattavia komponentteja voidaan lisätä haluttuja määriä.

2.2.4 Järjestelmätestaus

Kokonaisen ohjelmiston testauksesta käytetään termiä järjestelmätestaus, engl. system testing. Järjestelmätestauksessa pyritään testaamaan IT-järjestelmän kokonaisuutta ja komponenttien yhteensopivuutta (Sommerville, 2016.) Kuinka järjestelmätestaus eroaa komponentti- tai integraatiotestauksesta, vaikka molemmissa testausmetodeissa kuitenkin testataan erilaisten komponenttien yhteensopivuutta ja käyttöliittymien välistä kommunikaatiota?

Sommerville (2016) tuo esille tutkielmassaan kaksi eroavaisuutta järjestelmätestauksen ja komponenttitestauksen välille. Ensimmäiseksi järjestelmätestauksessa otetaan enemmän huomioon uusiutuvien komponenttien tuomat ominaisuudet eli toimiiko järjestelmä oikein, jos siihen tuodaan uudempi komponentti, joka on aiemmin testattu toimivaksi. Toinen erottava tekijä on, että järjestelmätestauksessa voidaan testata eri komponentteja valmistavien ryhmien komponenttien yhteensopivuutta. Järjestelmätestaus on siis enemmän kokoava testausmuoto aiempiin testausmuotoihin verrattuna. (Sommerville, 2016.)

Järjestelmätestaus sisältää useita erilaisia testausmetodeja. Naik ja Tripathy (2008) kertovat tutkielmassaan järjestelmätestauksen koostuvan muun muassa järjestelmän toimivuuden, turvallisuuden, sitkeyden, kuormituksen, stabiilisuuden, stressin, suorituskyvyn ja luotettavuuden testauksesta. Järjestelmätes-

tauksen vaiheessa on siis useita erilaisia mahdollisuuksia testata, että järjestelmä vastaa haluttuja ominaisuuksia ja sisältää mahdollisimman vähän virheitä.

Järjestelmätestausta voidaan toteuttaa myös kytkemällä järjestelmä muihin ulkopuolisiin järjestelmiin ja näin kehittämällä suurempi järjestelmäkokonaisuus. Tällöin puhutaan suuremmasta integraatiotestaamisesta, jolloin tarkoituksena on varmistaa, että kehitetty järjestelmäkokonaisuus toimii yhteen muiden kehitettyjen järjestelmien kanssa (Mili & Tchier, 2015.) Järjestelmätestausta voi siis tehdä suurelle järjestelmäkokonaisuudelle.

Järjestelmätestauksen tarkoituksena on myös varmistaa, että lopputuote ei ole ristiriidassa alkuperäisiin tavoitteisiin nähden. Järjestelmätestausta ei siis voisi tehdä, mikäli lopputuotteena valmistetulla järjestelmällä ei olisi mitään tavoitetta tai tarkoitusta, mihin tämän tulisi vastata. (Myers, ym., 2011.) Järjestelmätestaus on siis hyvin kokonaisvaltaisempaa kuin yksikkö- ja komponentti- ja integraatiotestaus.

2.3 Testaamisen haasteet

Järjestelmiä testatessa on mahdotonta löytää kaikkia järjestelmän virhetiloja (Myers, ym., 2011). Testaaminen kuitenkin on tärkeässä roolissa mahdollisimman virheettömän ja laadukkaan ohjelmiston tuottamisessa (Naik & Tripathy, 2008). Testaamisessa pyritään siis etsimään mahdollisimman monia ohjelmiston virheitä, jotta virheet saadaan korjattua ennen ohjelmiston tuotantovalmistusta. Testauksella pystytään myös parantamaan tuotteen laatua, eikä pelkästään vain testaamaan järjestelmän toimivuutta (Naik & Tripathy, 2008).

Kuten jo aikaisemmin on todettu, testaaminen on yrityksille kallista. Seuraavaksi käydään läpi mistä testaukseen tulee kuluja. Ensimmäinen asia mikä testausprosessissa aiheuttaa kuluja on testitapausten suunnittelu, luominen, ylläpitäminen ja käyttäminen (Naik & Tripathy, 2008). Testitapauksien tulee olla mahdollisimman tehokkaita ja testata järjestelmän haluttuja ominaisuuksia oikealla tavalla. Testitapausten avulla tulee myös löytää erilaisten komponenttien ja järjestelmien virhetiloja jo testausvaiheessa. (Sommerville, 2016.) Testitapausten tulee siis olla yllättävän suuria, monipuolisia ja järjestelmän osalta kaiken kattavia.

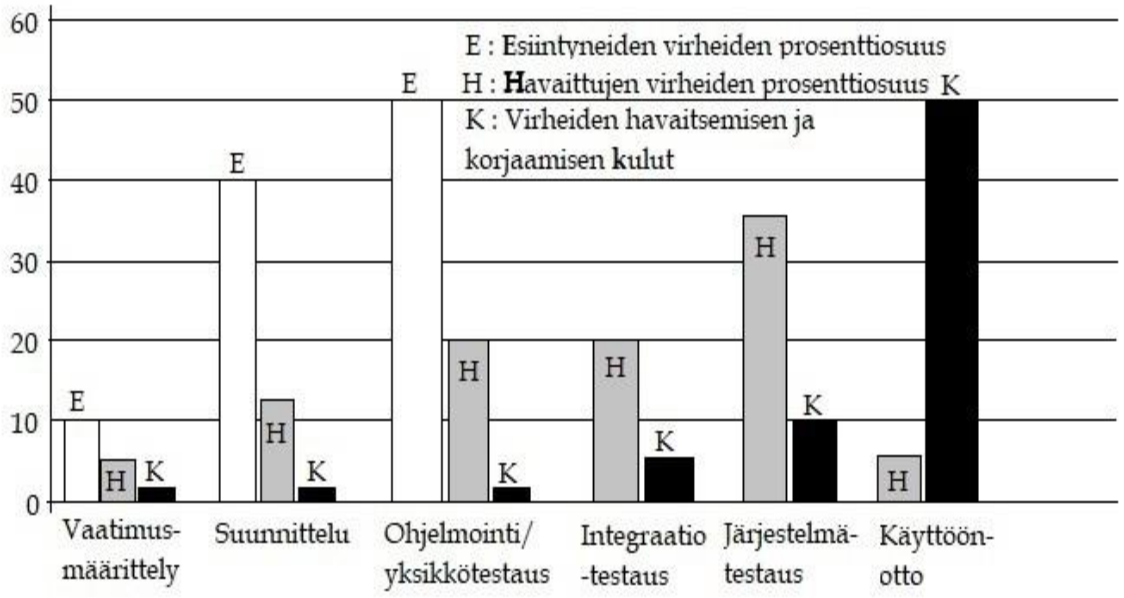
Testausta voisi tuottaa halvemmallakin vähentämällä testitapausten määrää, mutta samalla se vähentäisi järjestelmän testattavuutta (Naik & Tripathy, 2008). Järjestelmäkehittäjät pystyvät kuitenkin säästämään luomalla mahdollisimman tehokkaita testitapauksia. Myers, Sandler ja Badgett (2011) tuovat esille tutkielmassaan, että testauksen tehokkuuden ydinkysymys on: mikä joukko kaikista testitapauksista pystyisi löytämään kaikista eniten virheitä testattavasta ohjelmasta. Tähän ongelmaan on kehitetty omia tutkimuksia, muun muassa testitapausten suunnittelun menetelmäoppi, jonka mukaan esimerkiksi kaikista tehottomin menetelmä on satunnaisten syötteiden syöttäminen järjestelmään (Myers, ym., 2011).

Testaamista pystytään myös automatisoimaan ja testauksen automatisointi tekee testausprosessista paljon tehokkaampaa (Naik & Tripathy, 2008). Testaus voidaan automatisoida luomalla ja käyttämällä erilaisia testausoraakkeleita tai tehdä taustasovellus, joka testaa tuotettua ohjelmistokoodia. Testausoraakkeleita tutkittaessa on tullut esille muun muassa erilaisia tekniikoita, millä saadaan automatisoitua testauksen mallinnusta, määrittelyä, sopimusjohteista kehitystä ja muuntautumiskykyistä testausta. Testauksen automatisointia edesauttaa myös ohjelmistokoodin seikkaperäinen ja tarkka kuvailu, josta selviää tarkasti, mitä ohjelmistolla halutaan saavuttaa. (Barr, Harman, McMinn, Shahbaz & Yoo, 2015.)

Kuten aikaisemmin esiteltiin, mitä Naik ja Tripathy (2008) olivat teoksessaan todenneet, järjestelmän korjaaminen myöhemmissä vaiheessa voi osoittautua paljon kalliimmaksi, mitä myöhemmässä vaiheessa virhetila löydetään. Singh ja Kazi (2016) samaistuivat tähän faktaan tutkielmassaan. Myös Ammann ja Offutt (2016) ovat käsitelleet tutkielmassaan testauksen kulujen kasvua kehityksen edetessä lähemmäs käyttöönoton vaihetta ja selventäneet asian seuraavanlaiseen graafiseen kuvaajaan (kuvio 3).

Kuvaaja perustuu täsmälliseen analyysiin löydetyistä järjestelmävirheistä, joita ilmeni muutamissa suurissa valtiolle tehtävissä järjestelmissä. E-kirjaimella merkityt palkit kuvaavat prosenttiosuutta virheistä, jotka ilmenivät kyseisessä vaiheessa. H-kirjaimella merkityt palkit kuvaavat havaittuja virheitä ja K-kirjaimella merkityt palkit kuvaavat virheiden löytämisen ja korjaamisen kuluja kussakin vaiheessa. Ensimmäinen vaiheista on järjestelmän vaatimusmäärittely, toinen vaihe on suunnittelu, kolmas ohjelmointi sekä yksikkötestaus, neljäs integraatiotestaus, viides järjestelmätestaus ja kuudes vaihe on käyttöönotto. (Ammann & Offutt, 2016.)

Kuvaajasta huomaa selkeästi, miten paljon järjestelmän virhetilojen löytämisen ja korjaamisen kulut kasvavat, mitä myöhempään vaiheeseen siirrytään suhteessa kehitysprosessin aikaisempiin vaiheisiin. Myös löydettyjen virhetilojen prosenttisosuus kasvaa, kunnes käyttöönottovaiheessa ei löydetä enää virhetiloja, mutta virhetilojen korjaaminen silloin maksaa 50 kertaa enemmän kuin esimerkiksi vaatimusmäärittelyjen, suunnittelun tai ohjelmoinnin aikana. Taulukosta on myös havaittavissa, että järjestelmässä olemassa olevien virhetilojen määrä kasvaa myös alkuvaiheessa ohjelmointiin asti. Tämän jälkeen uusia virhetiloja ei tulisi syntyä, sillä kaikki järjestelmän uudet ominaisuudet ja mahdolliset virhetilat ovat jo järjestelmässä ohjelmoinnin jälkeen. Taulukon mukaan havaittujen virhetilojen määrä kasvaa myös järjestelmäkehityksen edetessä, mikä antaa käsityksen siitä, että suurin osa virhetiloista löydetään mahdollisesti ennen järjestelmän varsinaista käyttöönottoa.



KUVIO 3 Myöhäisen testauksen kulut (Ammann & Offutt, 2016)

3 TESTAUS PALVELUNA

Testaus pilvipalveluna tai myös testauspalveluna tunnettu testausmetodi on keino testata IT-järjestelmien ohjelmistojen suorituskykyä, skaalautuvuutta ja luotettavuutta oikeaa maailmaa vastaavilla pilvessä toimivilla simulaatiolla. Yksinkertaisemmin sanottuna testaus pilvessä mahdollistaa siis nykyaikaisten IT-järjestelmien täsmällisen testauksen suurien ja kattavien pilvitestausresurssien avulla. (Mittal, Nautiyal & Mittal, 2017.) Testauspalvelu on yksi nykyaikaisista pilvilaskennan työkaluista, joka mahdollistaa testauksen uudenlaisena pilvipalveluna (Poth, Werner & Lei, 2018, s. 401).

Pilvilaskenta on mahdollistanut uudenaikaisten järjestelmien tehokkaan ja ohjelmistokehittäjille vaivattomamman tavan kehittää IT-järjestelmiään. Pilvilaskenta mahdollistaa yritysten käyttää ja maksaa tarvittavista IT-järjestelmistä esimerkiksi virtuaalitietokoneista vain sen verran, mitä yritykset niitä käyttävät internetin välityksellä (Harikrishna & Amuthan, 2016).

Testausta on pyritty kehittämään tehokkaammaksi ja halvemmaksi muun muassa testiautomaation avulla. Testauspalvelussa on mahdollista toteuttaa suuri osa testauksesta automatisoidusti. Yhä enemmän järjestelmien kehittämisestä ja ylläpitämisestä, kuten myös testausta, tehdään ulkoistettuina pilvipalveluina. (Floss & Tilley, 2013.) Testauspalvelulla on myös omanlaisensa selkeä arkkitehtuuri (Yu, ym. 2010) eli kehysmalli, joten testausprosessi on myös hieman erilainen perinteiseen testaamiseen verrattuna.

Testaus ulkoistettuna palveluna on siis pilvilaskennalla toteutettava pilvipalvelu. Jotta testauksen pilvipalveluna ymmärtäisi paremmin seuraavissa luvuissa tullaan käsittelemään muun muassa seuraavat asiat : mitä tarkoittaa pilvilaskenta ja pilvipalvelu, mikä on testaus palveluna sekä millainen on testauspalvelun kehysmalli, eli millainen testauspalvelun prosessi käytännössä on. Testauspalvelun yleisimpiä piirteitä esitellään myös sekä millaisia mahdollisuuksia testauspalvelu tuo testaukseen.

3.1 Pilvilaskenta ja pilvipalvelujen yleispiirteet

Verkkoyhteyksien kautta toteutettavissa olevista, mukautuvista sekä fyysisistä tai virtuaalisesti jaettavista resursseista käytetään termiä pilvilaskenta (Mäkinen & Nurmela, 2015). Yhdysvaltalaisen kauppaministeriön alaisuudessa toimiva virasto NIST, National Institute of Standards and Technology, on määritellyt pilvilaskennan seuraavalla tavalla: Pilvilaskenta on malli, joka mahdollistaa kaikkialle jaettavia ja aina saavutettavia verkkoyhteyksien kautta jaettuja tietokoneresursseja. Resurssit ovat nopeasti järjestetty ja käytettävissä mahdollisimman pienellä vaivannäöllä tai palveluntarjoajan toimenpiteillä. (Mell & Grance, 2011.) Englanniksi pilvilaskennasta käytetään termiä cloud computing, mikä vaikuttaa kuvaavammalta termiltä kuin pilvilaskenta. Tutkielmassa käytetään kuitenkin termiä pilvilaskenta, sillä sitä esiintyy terminä suomenkielisissä tutkimuksissa useammin kuin esimerkiksi pilvitietojenkäsittelyä.

Pilvilaskenta on jaettavissa useaan pilvipalveluun, joista yleisimmät kolme ovat sovellus palveluna, engl. software as a service (SaaS), alusta palveluna, engl. platform as a service (PaaS) ja infrastruktuuri palveluna, engl. infrastructure as a service (IaaS) (Mell & Grance, 2011). Nykyään pilvilaskennan yleistyessä uusia pilvipalveluita on kehitetty ja ne ovat lisänneet myös suosiotaan, kuten testaus palveluna (Floss & Tilley, 2013). Testaus palveluna on yhä enemmän nouseva trendi ulkoistettavissa pilvipalveluissa (Ismail, Razali & Mansor, 2019, s. 1). Testauspalvelu tarjoaa ohjelmistokehittäjille mahdollisuuden käyttää laajaa ja joustavaa resurssikantaa, joka toimii pilvessä. Testauspalvelu itsessään voidaan automatisoida täysin, jolloin testausta suoritetaan ennalta-asetettujen parametrien mukaisesti. Testaus voidaan myös tehdä ihmisavusteisesti, jolloin testauksen suorittaa kolmas osapuoli, mutta kuitenkin pilven välityksellä. (Floss & Tilley, 2013.)

Testauspalvelu on houkuttelevaa yrityksille, sillä monet yritykset hyödyntävät entuudestaankin entistä enemmän pilvilaskentaa. Costello ja Hippold (2018) ovat julkaisseet Gartnerilla julkaisun, jossa arvioidaaan maailmanlaajuisten pilvipalveluiden markkinoiden kasvavan jopa 17,3 prosenttia vuonna 2019. Samassa julkaisussa he ovat myös julkaisseet taulukon maailmanlaajuisten julkisten pilvipalveluiden liikevaihdon tilastoista ja tulevaisuuden ennusteesta (Taulukko 1). Taulukossa on tilastoja vuodesta 2017 ja 2018 ja ennusteet vuodesta 2019 vuoteen 2021. Taulukossa on eritelty pilvipalveluista : bisnesprosessipalvelu, alustapalvelu, sovelluspalvelu, hallinta- ja turvallisuuspalvelu sekä infrastruktuuripalvelu. Taulukosta ilmenee selkeästi, että pilvipalveluiden liikevaihto on kasvanut ja sen on ennustettu kasvavan kokonaisuudessaan yli 10 miljardia dollaria vuodesta 2018 vuoteen 2021. Taulukosta voi myös huomata, että suurimpana pilvipalveluna rahallisesti on tällä hetkellä bisnesprosessipalvelu ja toiseksi suurimpana on aiemmin esittellyt sovellusalustapalvelu platform as a service (Costello & Hippold, 2018.)

TAULUKKO 1 Maailmanlaajuisten julkisten pilvipalveluiden liikevaihdon ennuste (miljardia Yhdysvaltain dollaria) (Costello & Hippold, 2018)

	2017	**2018**	**2019**	**2020**	**2021**
Bisnes-prosessi palveluna (engl. Business process as a service) (BPaaS)	42.2	46.6	50.3	54.1	58.1
Sovellusalusta palveluna (engl. Application Platform as a Service) (PaaS)	11.9	15.2	18.8	23.0	27.7
Sovellus palveluna (engl. Software as a Service) (SaaS)	58.8	72.2	85.1	98.9	113.1
Hallinnointi- ja turvallisuuspalvelut pilvessä	8.7	10.7	12.5	14.4	16.3
Infrastruktuuri palveluna (engl. Infrastructure as a Service) (IaaS)	23.6	31.0	39.5	49.9	63.0
Kokonaismarkkinat	**145.3**	**175.8**	**206.2**	**240.3**	**278.3**

Pilvipalvelut ovat siis pilvilaskennan lisäksi yleistymässä hyvin paljon. Kuten testausta käsittelevissä luvuissa mainittiin, testaus on kallista, joten pilvipalveluna tehdyllä testauksella pyritään tekemään testaus edullisemmin ja pienemmillä kuluilla. Suurin säästö pilvipalveluna tehdyssä testauksessa tulee testiympäristöjen ja testitapausten suunnittelun ja ylläpidon osalta. (Riungu-Kalliosaari, Taipale & Smolander, 2012.)

3.2 Testauspalvelun kehys

Kuten todettua, testaus palveluna on uudenaikainen testausmenetelmä IT-järjestelmien ohjelmistojen testaukseen. Tutkielmassa käydään läpi nimenomaan testauspalvelun arkkitehtuuri. Testauspalvelun arkkitehtuurinen kehys koostuu viidestä kerroksesta, jotka ovat : testin lähettäjän ja testauspalveluntarjoajan kerros, testitehtävien hallintakerros, testiresurssien hallintakerros, itse testauskerros ja testien tietokantakerros (Yu, ym., 2010, s. 182). Seuraavissa kappaleissa esitellään eri kerrokset tarkemmin.

Ensimmäinen kerros eli testin lähettäjän ja testipalvelun tarjoajan kerros mahdollistaa testin lähettäjän eli asiakkaan ja testipalvelun tarjoajan käyttää testauspalvelujärjestelmää (Bai, Li, Chen, Tsai & Gao, 2011). Kerros koostuu siis kahdesta osasta, joista asiakkaan osassa asiakas pääsee portaalin tai integroidun kehittämisympäristön kautta yhdistämään testauspalveluun internetin kautta. Testipalvelun tarjoaja puolestaan pystyy julkaisemaan uusia kehittämiään palveluita kerroksen kautta. (Yu, ym. 2010, s. 182.) Kyseistä kerrosta voi siis kutsua käyttäjien näkemäksi käyttöliittymäksi, jossa käyttäjinä toimii palvelun asiakas ja testipalvelun tarjoaja.

Toinen kerros eli testien tehtävänhallintakerros toimii väliohjelmistona, joka tukee testauspalveluja (Bai, ym., 2011). Kerros toimii myös testauksen palveluväylänä, joka toimii loppukäyttäjien ja testauspalveluiden välissä. Sillä on viisi erilaista tehtävää, jotka ovat : testauksen voimavarojen tarkistus, testaustehtävien klusterointi, järjestäminen sekä eteenpäin lähettäminen, testaustehtävien monitorointi, palvelujen rekisteröinti ja varastointi, ja palvelun julkaiseminen. (Yu, ym., 2010, s. 182.) Kyseinen kerros on siis ikäänkuin järjestelijä, jossa asiakkaalta tulleet testipalvelut nimetään ja järjestetään haluttuihin ryhmiin.

Kolmas kerros eli testiresurssien hallintakerros toimii pilvi-infrastruktuurina testauspalvelulle. Kerroksen vastuulla on valvoa ja hallita testauspalvelun pilvessä toimivia resursseja. (Bai, ym., 2011.) Fyysisten tietokoneiden resurssienhallinnan lisäksi kerroksessa on virtuaalitietokoneiden resurssienhallinta sekä virtuaalitietokoneiden käyttöliittymiä, joissa tapahtuu muun muassa käyttäjäryhmien ja virtuaalitietokoneiden hallinta, kirjanpito, valvonta ja järjestely (Yu, ym., 2010, s. 182). Kerros toimii siis valvovana ja ylläpitävänä osana testauspalvelun kokonaisuutta, josta voi tarvittaessa tarkistaa testaukseen saatavilla olevat resurssit.

Neljäs kerros on itse testauskerros, jossa testit suoritetaan. Kyseinen kerros on itse testauspalvelun ydinkerros, jossa tapahtuu myös palvelun kokoonpano, palveluiden jäsentely ja testitulosten keräys (Bai, ym., 2011). Palveluiden kokoonpanon alikerroksessa testauspalvelut järjestellään tiettyyn testausjärjestykseen ja samankaltaiset testauspalvelut ryhmitellään omiin käsiteltäviin ryhmiin. Palveluita jäsentelevässä kerroksessa tietynlaiset testityypit jaotellaan tietyille virtuaalitietokoneille suoritettavaksi. Testitulosten keräyksessä järjestelmä kokoaa testitulokset asiakkaalle eri virtuaalitietokoneilta. Testausten suorittamisessa käytetään virtuaalitietokoneita, sillä tietyt virtuaalikoneet voivat olla erikoistuneempia esimerkiksi yksikkötestauksiin tai testitapausten luomiseen tai järjestelmätestauksiin. (Yu, ym., 2010, s. 182.) Testauspalvelu on todella monipuolista ja virtuaalikoneita voi olla useita suorittamassa tietynlaisia testejä tai generoimassa testituloksia.

Viidennessä kerroksessa talletetaan testauksen tehtävät, testatut kohteet, palvelukuvaukset ja virhetilojen seurannan tulokset (Yu, ym., 2010, s. 182). Viides kerros toimii siis tietokantana testauspalvelulle. Tietokantaan talletettujen tietojen avulla pystyy mahdollisesti vertailemaan esimerkiksi

millaisia virhetiloja ja palveluita erilaiset testauspalvelut tuottavat ja millaisia eroavaisuuksia palveluntarjoajien välillä on.

3.3 Testauspalvelun yleisiä piirteitä

Yleisimmät pilvipalvelut ovat aikaisemminkin mainitut sovellus palveluna, sovellusalusta palveluna ja infrastruktuuri palveluna. Yleensä IT-järjestelmiä kehittävät yritykset testaavat itse kehittämiään järjestelmiä. Testaus palveluna mahdollistaa yritysten ulkoistaa ohjelmistojensa testausprosessin. Seuraavaksi käydään läpi vastauksia seuraaviin kysymyksiin: miksi yritykset haluaisivat ulkoistaa ohjelmistojensa testauksen pilvipalveluksi ja miten testausta voi käyttää palveluna.

Testauspalveluita on moneen eri tarkoitukseen. Testausvaihtoehtoja on lukuisia, riippuen siitä, kuinka palvelu on rakennettu ja minkälaisia testejä sen on tarkoitettu toteuttavan. (Floss & Tilley, 2013.) Jotkin testauspalveluntarjoajat ovat erikoistuneet esimerkiksi enemmän suorituskyvyn testaukseen ja jotkin järjestelmäkehityksen eri vaiheissa oleviin osatestauksiin, kuten yksikkötestaukseen.

Harikrishna ja Amuthan (2016, s. 2) tuovat esille teoksessaan seuraavanlaiset palvelumahdollisuudet ja niiden ominaisuudet. Prosessinhallintapalvelu, mikä tarjoaa testien projektin- ja prosessinhallinnan palveluna. Testiympäristöjen palvelu puolestaan mahdollistaa virtuaaliympäristön testauksen. Testauksen ratkaisupalvelu tarjoaa testien mallinnuksen ja testimetodit. Testaussimulaatio mahdollistaa testaussimulaatiot erilaisissa ympäristöissä. On myös olemassa testauspalvelu, jolla testipalvelun kehittäjät voivat testata, seurata ja monitoroida testauspalvelun toimintaa, jotta sen toimintaa voidaan kehittää. (Harikrishna & Amuthan, 2016, s. 2.)

Jokaiselle IT-järjestelmiä, -ohjelmistoja ja -sovelluksia kehittävälle yritykselle on siis varmasti keino hyödyntää testausta palveluna tarvitsemallaan tavalla. Riungu-Kalliosaari, Taipale ja Smolander (2012) esittävät hyvät ohjeet organisaatioille testauspalvelun käyttöönotosta, joita myös muut tutkijat, kuten Floss ja Tilley (2013) ovat käyttäneet tutkielmassaan. Seuraavaksi käydään kyseiset ohjeet läpi.

Ensimmäiseksi organisaatioiden tulee ymmärtää pilvilaskentaa ja pilvilaskennan tuomia mahdollisuuksia. Organisaatioiden ja yritysten tulee löytää itselleen oikeanlainen tapa hyödyntää pilvilaskennan ja pilvipalveluiden mahdollistamia ominaisuuksia. Organisaatiot voivat hakea tietoja pilvilaskennan tuomista mahdollisuuksista muun muassa suoraan pilvilaskennan palveluntarjoajalta. (Riungu-Kalliosaari, ym., 2012 s. 55.) Yritysten tulee siis ymmärtää mitä he itse voivat hyötyä pilvilaskennasta ja kuinka kannattavaa se heille on.

Toiseksi organisaatiot voivat joko tehdä pilottiprojekteja testauspalvelussa tai kehittää seikkaperäinen strategia palvelun hyödyntämisestä. Pilottiprojekteilla organisaatiot saavat käytännössä kokeilla testauspalvelun ominaisuuksia ja variaatioita, mikä sopii organisaatiolle parhaiten. Seikkaperäistä strategiaa

kehittäessä organisaatiot puolestaan voivat olla yhteydessä palveluntarjoajiin sekä pyytää konsultointiyrityksiltä neuvoja sopivan testauspalvelustrategian luomiseksi (Riungu-Kalliosaari, ym. 2012, s. 55.) Organisaatioiden tulee siis selvittää itselleen sopivin käytännön toteutusmalli, jota tulevat hyödyntämään käyttäessään testauspalvelua.

Kolmanneksi organisaatioiden tulee tehostaa heidän sisäisen ohjelmistokehitystiiminsä ja testauspalveluntarjoajan vuorovaikutusta. Tämä mahdollistaa paremmat testaustulokset sekä kehittää testauspalveluntarjoajan palveluja paremmiksi, mikä taas edesauttaa palvelun käyttäjiä. Myös erilaisiin testauspalvelun ja pilvilaskennan tuomiin käytännön muutoshaasteisiin tulee varautua. (Riungu-Kalliosaari, ym., 2012, s. 56.) Organisaatioiden tulee siis omaksua testauspalvelu omiin strategioihinsa ja käytänteisiin, vaikka kyseessä onkin ulkoinen palvelu.

Neljänneksi organisaatioiden tulee kehittää testauksen tutkimuksen ja käytännön toteutuksen yhteistyötä. Tällä tarkoitetaan sitä, että käytännön toteutuksen avulla organisaatiot ja yritykset voivat huomata, mitkä sovellukset ovat parhaiten testattavissa ulkoistettuina palveluina ja mitä erilaisia muuttujia testauspalvelussa tulee huomioida. (Riungu-Kalliosaari, ym., 2012, s. 56.) Tämä mahdollistaa sen, että testauksen tutkimus kehittyy samalla kun käytännön testausta tapahtuu.

Ulkoistetusta testauspalvelusta on jaoteltu myös erilaisia luokkia, kuinka testauspalvelu voi käytännössä toimia. Esimerkiksi Candea, Bucur ja Zamfir (2010) ovat tutkielmassaan jaotelleet automaattiset testauspalvelut kolmeen erilaiseen luokkaan. Ensimmäinen kolmesta testauspalvelun luokasta on testauspalvelu ohjelmoijan avustajana. Kyseinen testauspalvelun muoto mahdollistaa ohjelmistokoodin perusteellisen testauksen vähäisellä vaivannäöllä. Ohjelmoijan apurina toimiva testauspalvelu pystyy automaattisesti testaamaan ohjelmoijan tuottamaa ohjelmistokoodia ilman, että asiakkaan tarvitsee itse tehdä juuri mitään. Tällainen testauspalvelumalli pystyy myös arvioimaan ja ennaltaehkäisemään osan epähalutuista ohjelmisto-ominaisuuksista. Tämä lyhentää testausaikaa ja vähentää ohjelmoijan työtaakkaa. (Candea, Bucur & Zamfir, 2010.)

Toisena testauspalvelun mallina on esitetty testauspalvelu loppukäyttäjille, eli niin sanottu testauspalvelun kotiversio. Kyseisessä mallissa myös sovellusten ja ohjelmistojen loppukäyttäjät pystyvät käyttämään testauspalvelua varmistaakseen sovelluksen oikean ja halutunlaisen toiminnan. Palvelun käyttöliittymä on yksinkertaisempi ja selkeämpi kuin aiemmin esitellyssä ohjelmoijan apuri -mallissa. (Candea, Bucur & Zamfir, 2010.) Testauspalvelua voi siis yleisestikin käyttää esimerkiksi kotiversiona myös loppukäyttäjät, eikä pelkästään yritykset ja organisaatiot.

Kolmantena esiteltynä testauspalvelun mallina on sertifikaattipalvelut. Palveluiden tarkoituksena on testata sovelluksia, kuten yleisesti testauspalvelussa, mutta sovellus luokitellaan tietylle asteikolle löydettyjen virhetilojen määrään perustuen. Esimerkiksi Microsoftin laitteiston laadun tutkimuskeskus ja Apple Apple-kaupassaan sertifioi sovelluksiaan hieman esitellyn sertifikaattitestauspalvelun kaltaisesti. (Candea, Bucur & Zamfir, 2010.) Sertifikaattien

avulla käyttäjät pystyvät tarvittaessa vertailemaan paremmin eri ohjelmistoja toisiinsa juuri testauksessa löydettyjen virhetilojen perusteella.

Edellä mainittujen kolmen testauspalvelun luokittelumallin lisäksi, Gao, Bai, Tsai ja Uehara (2013) jakoivat testauspalvelut kahdeksaan erityyppiseen palvelumuotoon. Kyseiset palvelumuodot ovat osittain luokiteltavissa aiemmin esitettyihin palvelumuotoihin, mutta esimerkiksi laajan testisimulaation sekä sopimusten ja laskutuksen palveluja ei esitelty aiemmin. Laaja testisimulaatio mahdollistaa esimerkiksi käyttöliittymien laajamittaisen skenaariotestauksen, eli kyseessä on hieman laajemman testausympäristön ja testityökalujen valinta testauspalvelua varten. Sopimus- ja laskutuspalvelu puolestaan helpottaa palveluntarjoajien laskutusta ja sopimusten tekoprosesseja sekä mahdollistaa loppukäyttäjien tarkastella palveluntarjoajien sopimusvaihtoehtoja. (Gao, ym., 2013, s. 214.)

4 TESTAUSPALVELUN HAASTEET JA MAHDOLLISUUDET

Kuten aikaisemmissa luvuissa on käyty läpi: testaus on oleellinen prosessi ohjelmistokehityksen aikana laadun varmistamiseksi ja järjestelmiä pystytään testaamaan myös ulkoistettuina palveluina. Seuraavissa luvuissa pyritään vastaamaan molempiin tutkimuskysymyksiin, eli kuinka testaus palveluna eroaa perinteisistä testausmenetelmistä ja kuinka yritykset hyötyvät ulkoistetusta testauspalvelusta.

Testauspalvelussa on paljon hyviä ominaisuuksia, joista monet ovat pilvilaskennan avulla nykyajan vaatimuksia vastaavia. Hyvien ominaisuuksien lisäksi testauspalveluissa on kuitenkin vielä paljon haasteita, muun muassa turvallisuudessa ja skaalautuvuuden hallinnassa (Floss & Tilley, 2013). Haasteet eivät kuitenkaan ole täysin ylitsepääsemättömiä ja lisätutkimuksen avulla yritykset pystyvät vastaamaan haasteisiin.

Seuraavissa luvuissa käsitellään, millaisia haasteita testauspalveluissa on havaittu tieteellisissä tutkimuksissa. Tämän jälkeen vertaillaan perinteistä testausmenetelmiä ja testauspalvelua. Testauspalvelun vertailussa pyritään tuomaan esille niin hyviä kuin kehitettäviäkin puolia testauspalvelusta viitaten aiemmin ilmenneisiin haasteisiin ja ominaisuuksiin niin testauksessa kuin testauspalvelussakin.

4.1 Testauspalvelun haasteet

Aiemmissa luvuissa käsiteltiin ulkoistetun testauspalvelun ominaisuuksia. Kuten aiemmin ilmeni, useat tutkijat, kuten Floss ja Tilley (2013) sekä Riungu-Kalliosaari, Taipale ja Smolander (2010), ovat löytäneet paljon uudenaikaisia ominaisuuksia testauspalveluista. Nämä esitellyt hyvät ominaisuudet ei kuitenkaan tule ilman haasteita, joita tullaan seuraavaksi esittelemään tarkemmin.

Yleisimmin esitetyksi haasteeksi ulkoistetuissa testauspalveluissa nousi tutkimusten kesken tarpeeksi kattavan, aina saatavilla olevan, kehittyvän ja

monipuolisen testauspalvelun taloudellinen kannattamattomuus ja haastavuus. Yritykset voivat kokea haastavaksi olla jatkuvasti ja globaalisti saavutettavissa palveluntarjoajana (Floss & Tilley, 2013). Ongelmakysymykseksi on esitetty, mistä yritykset saavat kannattavat tulot, joilla pystyy ylläpitämään jatkuvasti käytössä olevaa testauspalvelua (Gao, ym., 2013). Testauspalvelun yleisin ongelma on siis palveluntarjonnan kannattavuuden ja uusien teknologioiden kehittämisen tuomat haasteet.

Toinen tutkimuksissa yleisimmin ilmennyt testauspalveluiden haaste oli turvallisuus. Turvallisuuden haasteet ovat tulleet palveluiden toteutuksessa esille etenkin datan hallinnan ja käsittelyn yhteydessä. Asiakkaat voivat kuitenkin käyttää testauspalvelua lähes täysin ilman turvallisuusongelmia testaamalla tietoturvaluokitukseltaan avoimempia sovelluksia ja ohjelmia. (Riungu-Kalliosaari, ym., 2013.) Myös tekijänoikeusrikkeiden ja asiakastietojen välittymisen riskit on havaittu testauspalvelussa. Yksi esitetty keino välttyä kyseisiltä ongelmilta on tekemällä testidatan mallit binäärimuodossa. (Floss & Tilley, 2013.) On kuitenkin tärkeää, että ohjelmistoja ja sovelluksia pystytään testaamaan sellaisella datalla, jollaista järjestelmä yleisimmin vastaanottaa.

Kolmas tutkimuksissa esiintynyt haaste testauspalveluun on testien skaalautuvuuden ja kattavuuden hallinta. Palveluntarjoajien on otettava huomioon monia asioita, kuten eri pilvilaskennallisten palveluiden yhteensopivuudet ja käyttöliittymät. Testauspalvelussa pitää siis tehdä paljon integraatioita eri pilvipalveluiden teknologioiden välillä, jotta palvelusta saadaan tarpeeksi kattava myös pilvilaskennalla toteutettuihin järjestelmiin. (Riungu-Kalliosaari, ym., 2013.) Testauspalvelun skaalautuvuus saattaa asettaa haasteita palveluntarjoajille, sillä esimerkiksi sovelluspalveluna tuotettuja sovelluksia pitää pystyä testaamaan useampia kerrallaan laajamittaisessa testauspalvelussa (Gao, ym., 2013).

Testauspalvelun on nähty myös lisäävän ohjelmistotuottajien tarvetta ja painetta tuottaa virheettömämpiä sovelluksia ja ohjelmistoja. Testauspalvelun avulla ohjelmistokehittäjät pystyvät testaamaan tuotteitaan vaivattomammin, mutta myös loppukäyttäjät ja vakoiluohjelmien käyttäjät pystyvät testaamaan tuotteen heikkouksia ja löytämään heikkouksia sovelluksista, joita ei ole testattu tarpeeksi hyvin. (Candea, ym., 2010, s. 160.) Osa loppukäyttäjistä voi käyttää testauspalvelun ominaisuuksia löytääkseen sovelluksista haavoittuvaisuuksia ja käyttää niitä vahingollisiin tarkoituksiin (Floss & Tilley, 2013). Toisaalta haaste ei ole täysin negatiivinen loppukäyttäjille tai markkinoille, sillä tuottajien tulee valmistaa entistä varmempia ja laadukkaampia tuotteita.

4.2 Testauspalvelun vertailu perinteiseen testaukseen

Testausta käsittelevässä kappaleessa kävi ilmi testauksen hyötyjä ja haasteita. Aiemmin käsiteltiin myös testauspalvelun yleisiä piirteitä ja haasteita. Kuinka testauspalvelu käytännössä siis eroaa perinteisestä testauksesta? Tässä luvussa

kerrotaan testauspalvelun tutkituista hyödyistä ja selkeistä eroavaisuuksista perinteiseen testaukseen.

Suurin ja selkein testauspalvelun hyöty verrattuna perinteiseen testaukseen on testaukseen kuluvat kustannukset. Kuten aiemmin pilvilaskentaa ja testauspalveluja käsittelevässä luvussa mainittiin, testauspalvelulla pystytään luomaan testitapaukset, testausympäristöt ja testaukseen vaadittavat sovellukset pilvilaskennan avulla. Pilvilaskennalla luodut testiympäristöt ovat paljon halvempia tuottaa virtuaalisesti kuin fyysiset testiympäristöt. Testiympäristöt on myös helpompi purkaa virtuaalisessa ympäristössä kuin perinteisesti fyysisessä ympäristössä (Riungu-Kalliosaari, ym., 2013.) Perinteisen testauksen luvuissa esitellyistä tutkimuksista puolestaan käy ilmi, että perinteinen testaus on kallista ja etenkin testausprosessin alkuvaiheen testitapausten luonti luo yrityksille paljon kustannuksia (Naik & Tripathy, 2008). Aikaisemmin kävi myös ilmi, että testausta voi automatisoida, mutta automatisointikin vaatii resursseja yrityksen sisältä.

Pilvilaskennan myötä tulevat mahdollisuudet käyttää pilvipalveluiden mahdollistamia pilvessä toimivia resursseja tekee testauksesta tehokkaampaa kuin perinteisestä testauksesta. Riungu-Kalliosaari, Taipale ja Smolander (2013) haastattelivat tutkimuksessaan useita testauspalveluja käyttäviä yrityksiä. Yritysten haastatteluista ilmenee, että testauspalvelu on koettu paljon tehokkaammaksi kuin perinteinen testaus. Testauspalvelun tehokkuus on ilmennyt muun muassa useiden virtuaalikoneiden käyttöönoton helppoudessa, sillä perinteisesti manuaalisten testauskäyttöön tarkoitettujen koneiden ostaminen ja käyttöönotto on kallista. Myös virtuaalikoneiden määrä ja siten testattavan ohjelman kuormituksen testaaminen oli paljon helpompaa kuin perinteisillä testausmenetelmillä. (Riungu-Kalliosaari, ym., 2013.) Testauspalvelulla yritykset voivat siis käyttää palveluntarjoajan virtuaalikoneita ja maksaa vain sen verran, mitä ajallisesti tai sopimuksen mukaan palvelua käyttävät (Floss & Tilley, 2013). Testauspalvelun on myös sanottu vähentävän löydettyjen virheiden määrää jopa 15% enemmän perinteiseen testaukseen verrattuna (Gao, ym., 2013).

Virtuaalikoneiden avulla käytettävästä testausympäristöstä saa myös paljon dynaamisemman ja skaalautuvamman. Testauspalvelujen virtuaalikoneiden avulla yritykset pystyvät luomaan täsmällisempiä ympäristöjä, mitkä vastaavat paremmin ohjelmistojen todellisten käyttäjien työskentely-ympäristöjä. Virtuaalikoneita pystyy myös muokkaamaan helpommin vastaamaan testattavaa skenaariota kuin perinteisiä koneita perinteisessä testauksessa. (Mittal, ym., 2017.) Asiakkaiden toimintaympäristöjä vastaaviksi testiympäristöiksi muokkautuvat virtuaalitietokoneiden tilat mahdollistavat myös realistisempia testituloksia, kuin pelkästään tietyssä testiympäristössä tehdyt testit, jotka eivät välttämättä vastaa asiakkaiden toimintaympäristöjä (Riungu-Kalliosaari, ym., 2013). Virtuaalikoneiden testiympäristöjä on siis helpompi muokata halutunlaiseksi.

Perinteinen testaus vaatii myös enemmän aikaa. Sovelluksen kilpailukykyyn vaikuttaa se, kuinka nopeasti sovellus saadaan markkinoille, mihin puolestaan vaikuttaa muun muassa kuinka nopeasti sovellus saadaan testattua (Naik & Tripathy, 2008, s. 24). Testauspalvelulla testien suorittaminen uudelle

ohjelmiston osalle on suoritettavissa paljon nopeammin useammassa ympäristössä kuin perinteisessä testauksessa. Organisaatiot pystyvät myös varaamaan enemmän resurssejaan muihin liiketoimintansa osa-alueisiin, kun testauspalvelu hoitaa testauksen. (Riungu-Kalliosaari, ym., 2013.)

Testauspalvelu ei ole myöskään niin rajoittunutta kuin perinteinen testaus. Aiemmin ilmenneiden pilvilaskennan tuomien ominaisuuksien lisäksi testauspalvelussa on tiedossa kaikki olemassa olevat testaustekniikat. Perinteisessä testauksessa puolestaan testitekniikat rajoittuvat testaajan tekniikoiden tietämykseen. (Gao, ym., 2013.) Testauspalveluihin on myös lisätty ominaisuudet valko- ja mustalaatikkotestauksesta mukaan lukien yksikkötestaus (Floss & Tilley, 2013, s. 422).

Pilvilaskennan tuomista ominaisuuksista huolimatta testausta ei voida kuitenkaan täysin ulkoistaa ja automatisoida. Sommerville (2016) kertoo teoksessaan täysin automatisoidun testauksen olevan mahdotonta. Olisi mahdotonta muun muassa automatisoida testausta, joka testaa järjestelmiä, joiden toiminnan oleellinen osa on, miltä järjestelmä näyttää. On myös mahdotonta testata, onko ohjelmistoilla epähaluttuja sivuvaikutuksia, joita ei näe sovelluksen toimivuudesta. (Sommerville, 2016.) Kaikkea ei pysty siis testauspalvelulla vielä tänä päivänä testaamaan.

Aiemmin esitelty testauspalvelun malli, testauspalvelu ohjelmoijan avustajana, mahdollistaisi myös yksikkötestauksen nopeamman ja automaattisemman suorittamisen. Kuten aiemmin kävi ilmi, kyseinen palvelumalli hakee automaattisesti uusimman luodun ohjelmistokoodin ja testaa sen (Candea, Bucur & Zamfir, 2010). Vastaavanlaisesti testauspalvelussa on siis mahdollista tehdä kaiken tasoisia testejä eri kehityksen vaiheissa oleville ohjelmistoille (Gao, ym., 2013).

Perinteinen testaus on myös tietoturvallisempaa kuin testauspalvelu. Kuten testauspalvelun haasteita käsittelevässä luvussa ilmeni, testauspalvelussa on haasteena muun muassa turvallisuus. Perinteinen testaus on siis turvallisempaa, sillä siinä testattavaa testidataa tai ohjelmistoa ei lähetetä kolmannelle osapuolelle testattavaksi ja täten kolmas osapuoli ei saa yhtä helposti käsiinsä ohjelmiston tietoja. On myös pääteltävissä, että erityisesti tietoturva-alttiit ja pienet yritykset, joiden tekijänoikeudet eivät välttämättä ole yhtä huippuluokkaa kuin suurten yritysten, pitäytyvät todennäköisemmin perinteisessä testauksessa välttääkseen testauspalvelun turvallisuusriskit.

Testauspalvelussa testaus on kuitenkin itsessään halvempaa myös pienemmille yrityksille kuin perinteinen testaus juuri pilvilaskennan avulla. Pilvilaskennan avulla pienemmätkin yritykset ja jopa loppukäyttäjät saavat enemmän erilaisia testaustyökaluja käyttöönsä ja maksavat vain palvelun käyttöajan verran. Pilvilaskenta testauspalvelussa mahdollistaa myös aiemmin mainittujen ominaisuuksien lisäksi muiden pilvipalveluiden, kuten sovelluspalveluiden avulla luotujen ohjelmistojen testauksen, mikä ei ole mahdollista perinteisin testausmenetelmin. (Gao, ym., 2013.)

5 YHTEENVETO JA POHDINTA

Tässä tutkielmassa vertailtiin testauspalvelun ja perinteisen testauksen eroavaisuuksia sekä näiden ominaisuuksien vertailua kirjallisuuskatsauksena. Tutkielmassa käytettiin lähteinä aikaisempia tutkimuksia muun muassa testauksesta, ohjelmistokehityksestä, pilvilaskennasta ja testauspalvelusta. Lähdemateriaaleista löytyi hyviä yhteneväisyyksiä, mutta myös eroavaisuuksia esimerkiksi testauksen tasojen määrittelyssä ja esittelyssä, mikä mahdollisti myös tutkimusten välistä vertailua.

Perinteistä testausta käsittelevissä luvuissa käytiin läpi testauksen merkitystä järjestelmäkehityksessä sekä testauksen V-mallin mukaiset testaustasot. Testauksesta käytiin läpi myös testausprosessi pintapuolisesti sekä testaamisen haasteita yleisellä tasolla. Testaamisen perusteiden käsittely pohjusti testauspalvelun käsittelyä. Testauspalvelua läpikäyvissä luvuissa puolestaan käsiteltiin pilvilaskennan yleispiirteitä sekä testauspalvelun kehys. Tämän jälkeen tuotiin esille testauspalvelun yleisiä piirteitä ja lopuksi testauspalvelun haasteita ja sitä millaisia erilaisia ominaisuuksia testauspalvelussa on verrattuna perinteiseen testaukseen.

Testausta käsittelevissä luvuissa ilmeni, että testaus itsessään on todella tärkeää IT-järjestelmäkehityksessä. Järjestelmistä tärkeimpänä testattavana osana on nimenomaan ohjelmistot, joiden testaaminen on prosessi. Prosessia on pyritty automatisoimaan, jotta testaaminen ei olisi niin kallista ja testaaminen olisi tehokkaampaa. Testausprosessilla pyritään verifioimaan ja validoimaan ohjelmistoja, jotta asiakkaat olisivat mahdollisimman tyytyväisiä kehitettyyn hyödykkeeseen ja sen ohjelmisto toimii kuten pitääkin. Testaamisella pyritään myös parantamaan tuotteitten laatua. Testaamisessa on myös haasteita. Testaaminen on kallista eikä testaamisella löydetä kaikkia järjestelmän virhetiloja, mutta monista tutkimuksista ilmeni, että mitä aikaisemmin virhetilat havaitaan, sitä vähemmän virheiden korjaaminen maksaa.

Tutkielmassa käytiin läpi testausprosessin pintapuolisesti, sillä pääsääntöisesti pyrittiin keskittymään testaustasojen ja perinteisten testaamisen haasteiden käsittelyyn. Testauksesta käsiteltiin juurikin tyypillisimmät testaustasot,

joita on tarkasteltu tutkimuksissa myös testauspalveluna. Testaustasot toimivat myös hyvin taustalla olevina kokonaisuuksina testaamisessa, jotta testauspalvelun vaiheitakin pystyi jaottelemaan eri tasoille.

Pilvilaskenta ja pilvipalvelut puolestaan ovat tuoneet uudenaikaisia mahdollisuuksia IT-järjestelmäkehitykseen, mikä näkyy myös testauksessa. Internetissä luodut virtuaalitietokoneet ja niillä luodut ohjelmistot vaativat myös uudenaikaisia testausmetodeja, jotka vastaavat niiden suorituskykyjä. Testauspalvelu on siis yksinkertaisesti pilvilaskennalla suoritettavaa testausta ulkoistettuna palveluna.

Testauspalvelu on tutkimusten mukaan uusi palvelumuoto ja monet tutkijat toivat sen hyvin esille. Uudenlainen palvelukehys mahdollistaa pilvilaskennan mukaiset palvelut testauksen muodossa vaivattomammin kuin perinteisessä testauksessa ja jokaiselle yritykselle pienestä suureen on tarjolla monenlaisia testauspalveluntarjoajia erilaisine palveluineen. Useat tutkijat toivat paljon hyviä puolia testauspalvelusta esille, kuten edullisuuden järjestelmäkehittäjille ja helppokäyttöisyyden, sillä yritysten ei tarvitse fyysisesti luoda kalliita testiympäristöjä, sillä palveluntarjoajan virtuaaliympäristöt ovat käytettävissä kellon ympäri.

Testauspalvelussa on kuitenkin ilmennyt myös omia haasteitaan, kuten turvallisuus, kannattavuus ja skaalautuvuuden ja teknologian hallinta. Turvallisuutta on pyritty ratkaisemaan muun muassa muokkaamalla testattavat ohjelmistot sellaiseen muotoon, ettei niitä pysty ulkopuolinen kopioimaan. Kannattavuuden haasteeseen yritykset eivät puolestaan ole vielä löytäneet ratkaisua, sillä monet kannattavat testauspalveluntarjoajat ovat osa suurempaa yritystä, jolla on muitakin pilvilaskennan työkaluja ja palveluita tarjottavana.

Tutkimus vastasi tutkimuskysymyksiin juurikin tutkielman loppuosassa ja testauksen sekä testauspalvelun esittely ja läpikäynti pohjusti vertailun mahdollistamista. Testausta ja testauspalvelun yleispiirteitä käsittelevät kappaleet tuovat lukijalle selkeät käsitykset siitä, kuinka testausta ja testauspalvelua voidaan toteuttaa. Viimeiset kappaleet käsittelevät paremmin eroavaisuuksia ja hyötyjä, mitä yritykset voivat saada ulkoistetuista testauspalveluista.

Kaiken kaikkiaan on havaittavissa, että testauspalvelu on hyödyllinen ja varteenotettava mahdollisuus testata IT-järjestelmiä. Testauspalvelu vaikuttaa hyvältä keinolta ulkoistaa ja tehostaa yrityksen järjestelmien testausta. Turvallisuus paranee tulevaisuudessa mahdollisesti paremmaksi esimerkiksi erilaisten salaustekniikoiden avulla, joita pystytään hyödyntämään etenkin pilvilaskennassa. Kannattavuutta todennäköisesti tulee lisäämään ainakin se, että järjestelmiä kehittävät yritykset käyttäisivät testauspalvelua entistä enemmän.

Tutkimuksessa käsiteltiin pääsääntöisesti vuonna 2008 ja sen jälkeen julkaistuja tutkimuksia. On otettava myös huomioon, että testauspalvelusta ei ole vielä tehty yhtä paljoa tutkimusta kuin vanhemmista aihealueista kuten perinteisestä testaamisesta, joten uusimpien artikkeleiden ja tutkimusten arviointi oli haasteellista, sillä useaan niistä oli viitattu vähän.

Haasteellista tutkielmassa oli testauspalvelun moninaisuus. Testauspalvelusta on olemassa nykyään myös pilvipalveluiden testausta, pilvessä tapahtu-

vaa testausta sekä pilvestä riippumatonta testauspalvelua. Tutkielmassa koetettiin kuitenkin selittää pääsääntöisesti yleisimmän pilvessä tapahtuvan testauspalvelun periaatteita kyseisiin tutkimuksiin vedoten.

Tutkimusta rajoitti myös testauspalvelua käsittelevien tutkimusten tieteellisyyden kritisointi. Osa löydetyistä lähteistä oli sellaisia, joiden tieteellisyyttä piti arvioida hieman enemmän kuin toisten lähteiden. Monet tutkijat ja informaatioteknologian asiantuntijat olivat laatineet julkaisuja esimerkiksi erilaisille verkkosivuille, joiden tieteellinen arvo oli vähäistä.

Vähäisen tutkimuksen vuoksi monet tutkijat, kuten Floss ja Tilley (2013), tuovat tutkimuksessaan juurikin esille, että ulkoistetussa pilvitestauspalvelussa on vielä paljon tutkittavaa. Heidän esille tuomia jatkotutkimusaiheita oli muun muassa testauspalvelun vaikutuksia asiakkaan liiketoimintaan sekä testauspalvelun palveluntasovaatimusten määrittäminen (Floss & Tilley, 2013, s. 424). Joten aiemmatkin tutkijat ovat nähneet monia jatkotutkimusaiheita asiasta.

Aiemmista tutkimuksista eroten tutkimukseni käsitteli perinteistä ja pilvipalveluna toteutettua testausta suhteellisen tasapuolisesti. Aiemmat tutkimukset ovat painottuneet pääsääntöisesti vain joko testauspalvelun tai perinteisen testauksen tutkimiseen. Tämä tutkimus helpottaa siis myös perinteisen ja pilvilaskentana toteutetun testauksen vertailua ja yhteneväisyyksien ja eroavaisuuksien ymmärtämistä. Tutkimuksesta on todennäköisesti hyötyä testaamisen tutkimisesta kiinnostuneille ja tukea aiemmin tehdylle tutkimukselle testaamisesta ja ulkoistetuista pilvitestauspalveluista.

Jatkotutkimusaiheena on nähtävissä myös hieman samankaltainen tutkimus, mitä Riungu-Kalliosaari, Taipale ja Smolander (2012) tekivät tutkimuksessaan, eli haastattelujen analysointia eri testauspalveluita käyttäviltä yrityksiltä. Yrityksiä haastatellessa olisi kuitenkin mahdollista pyrkiä muun muassa tarkastelemaan tarkemmin, miten yritykset testasivat aiemmin verrattuna ennen testauspalvelun käyttöönottoa, miten yritykset päätyivät valitsemaan oman palveluntarjoajansa ja miten testauspalvelun käyttöönotto on hyväksytty yrityksen sisällä. Oletettavissa on myös, että testaamisesta ja testauspalveluista tehdään tutkimuksia tulevaisuudessa testauksen tärkeyden vuoksi.

LÄHTEET

Ammann, P., & Offutt, J. (2016*). Introduction to software testing*. Cambridge University Press.

Bai, X., Li, M., Chen, B., Tsai, W. & Gao, J. (2011) Cloud testing tools, *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE*), Irvine, CA, 2011, s. 1-12.

Barr, E.T., Harman, M., McMinn, P., Shahbaz, M., & Yoo, S. (2015) The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering*, vol. 41, s. 507-525.

Candea, G., Bucur, S., & Zamfir, C. (2010). Automated software testing as a service. *Proceedings of the 1st ACM symposium on Cloud computing* (s. 155-160). ACM.

Costello, K. & Hippold, S. (2018) Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17.3 Percent in 2019, *Gartner Newsroom,* haettu osoitteesta https://www.gartner.com/en/newsroom/press-releases/2018-09-12-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2019

Deuff, D. & Cosquer, M. (2013) *User-centered agile method*, John Wiley & Sons.

Floss, B. & Tilley, S. (2013) Software Testing as a Service: An Academic Research Perspective, *IEEE Seventh International Symposium on Service-Oriented System Engineering* (s. 421-424). IEEE.

Gao, J., Bai, X., Tsai, W. & Uehara, T. (2013) Testing as a Service (TaaS) on Clouds, IEEE *Seventh International Symposium on Service-Oriented System Engineering* (s. 212-223). IEEE.

Gao, J. (2000), Component testability and component testing challenges. *Proceedings of International Workshop on Component-based Software Engineering (CBSE2000, the 22nd International Conference on Software Engineering (ICSE2000).*

Hamill, P. (2004) *Unit Test Frameworks: Tools for High-Quality Software Development*, O'Reilly Media, Oy.

Harikrishna, P., & Amuthan, A. (2016). A Survey of Testing as a Service in Cloud computing, *International Conference on Computer Communication and Informatics (ICCCI)* (s. 1-5). IEEE.

Ismail, F. F., Razali, R., & Mansor, Z. (2019). Considerations for Cost Estimation of Software Testing Outsourcing Projects. *International Journal on Advanced Science, Engineering and Information Technology, 9(1), s. 142-152*.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.

Mili, A., & Tchier, F. (2015) *Software testing : Concepts and operations*, John Wiley & Sons.

Mittal, V., Nautiyal, L. & Mittal, M. (2017) Cloud Testing- The Future of Contemporary Software Testing, *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, Jammu, 2017, s. 131-136.

Myers, G.J., Sandler, C. & Badgett, T. (2011) *The Art of Software Testing Third edition*, John Wiley & Sons.

Mäkinen, P. & Nurmela, T. (2015) *Pilvilaskennan perusteet ja sanasto*, Suomen Standardisoimisliitto SFS ry, SFS SR 310 Verkkosovellukset - seurantaryhmä, haettu 19.3.2019: https://www.slideshare.net/SFSedu/pilvilaskennan-perusteet-ja-sanasto

Naik, K. & Tripathy, P. (2008) *Software Testing and Quality Assurance: Theory and Practice*, John Wiley & Sons.

Poth, A., Werner, M. & Lei, X. (2018) How to Deliver Faster with CI/CD Integrated Testing Services?. *Larrucea X., Santamaria I., O'Connor R., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2018. Communications in Computer and Information Science*, vol 896. Springer, Cham.

Riungu-Kalliosaari, L., Taipale, O., & Smolander, K. (2012). Testing in the cloud: Exploring the practice. *IEEE software*, 29(2), s. 46-51.

Singh, A. & Kazi, N. (2016) *Software Testing,* Himalaya Publishing House.

Sommerville, I. (2016). Software *engineering* (Tenth edition, global edition.). Boston: Pearson.

Urias, E. (2019) Best Software Development Methodologies In 2019, *INVID*. Haettu osoitteesta https://invidgroup.com/best-software-development-methodologies-in-2019/

Vocke, H. (2018) The Practical Test Pyramid, *martinfowler.com, ThoughtWorks*. Haettu osoitteesta https://martinfowler.com/articles/practical-test-pyramid.html

Whittaker, J.A. (2009) *Exploratory Software Testing : Tips, Tricks, Tours, and Techniques to Guide Test Design*, Pearson Eduation

Yu, L., Tsai, W.-T., Chen, X., Liu, L., Zhao, Y., Tang, L. & Zhao, W. (2010) Testing as a Service over Cloud, *Fifth IEEE International Symposium on Service Oriented System Engineering* (s. 181-188). IEEE.