Lauri Humalamäki

DEVOPSIN TOTEUTTAMINEN PILVIPALVELUNA: KEINO LAADUKKAASEEN SOVELLUSKEHITYKSEEN?

JYVÄSKYLÄN YLIOPISTO
INFORMAATIOTEKNOLOGIAN TIEDEKUNTA
2023

TIIVISTELMÄ

Humalamäki, Lauri
DevOpsin toteuttaminen pilvipalveluna: Keino laadukkaaseen sovelluskehitykseen?
Jyväskylä: Jyväskylän yliopisto, 2023, 69 s.
Tietojärjestelmätiede
Ohjaaja(t): Marttiin Pentti, Halttunen Veikko

Ketterät sovelluskehitysmallit ovat yleistyneet merkittävästi ja suurelta osin korvanneet perinteisen vesiputousmallin. Ketterän sovelluskehityksen rinnalle syntynyt DevOps on toimintamalli, joka tukee ketterää sovelluskehitystä ja mahdollistaa nopeat ja jatkuvat julkaisut sekä automaation lukuisissa eri sovelluskehityksen vaiheissa. Lisäksi pilvipalvelut ovat yleistyneet selvästi viime vuosina, ja monet organisaatiot päätyvät uusiin teknologioihin sekä käytänteisiin pärjätäkseen markkinoilla. Pro gradu -tutkielmassa tarkastellaan, miten toimintamalli DevOpsia voidaan toteuttaa PaaS-pilvipalveluna (Platform as a Service). Tutkielma antaa lukijalle kattavan kuvan DevOpsista ja sen tärkeimmistä työkaluista sekä käytänteistä. Lisäksi tutkielmassa esitellään yleisimmät pilvipalvelumallit ja niiden ominaisuudet. Tämä luo tarvittavaa pohjaa sille, miten DevOps voidaan yhdistää PaaS-pilvipalveluun. Tutkielmassa myös selvitetään, miten pilvipalvelupohjaiset DevOps-palvelut voivat edistää ohjelmistotuoteprojektin laatua. Laatu terminä on monimerkityksinen käsite, joten se määritellään tarkkaan ja tutkimuksen tueksi luodaan uusi viitekehys. Esimerkkinä PaaS-pohjaisesta pilvipalvelusta on Microsoft Azure DevOps. Tutkielmassa käydään Azure DevOps käydään läpi, jotta havaitaan, miten se toteuttaa DevOps-toimintamallin käytänteitä. Lisäksi Azure DevOpsia käytetään apuna tarkastelemaan, miten DevOps-pilvipalvelulla voidaan parantaa ohjelmistotuotteen laatua. On huomioitava, että Azure DevOps on tuore tuote, joten aiempi tutkimusaineisto siitä on vielä pirstaloitunutta. Tutkielma toteutettiin tapaustutkimuksena ja haastatteluosuus on suoritettu teemahaastattelun, jossa haastateltiin Azure DevOpsia käyttävää tiimiä.

Asiasanat: Platform as a Service (PaaS), Microsoft Azure DevOps, pilvipalvelu, sovelluskehitysprojekti, jatkuva integraatio ja julkaisu

ABSTRACT

Humalamäki, Lauri
Deploying DevOps to the cloud: A means to high-quality software development?
Jyväskylä: University of Jyväskylä, 2023, 69 pp.
Information System Sciences
Supervisor(s): Marttiin Pentti, Halttunen Veikko

Agile software development methodologies have gained significant popularity and have largely replaced the traditional waterfall model. Alongside agile software development, the concept of DevOps has emerged as an operational model that supports agile practices and enables rapid and continuous releases, as well as automation across various stages of software develpopment. Additionally, cloud services have become more prevalent in recent years, and many organizations are adopting new technologies and practices to stay competitive in the market. This Master's thesis examines how the DevOps operational model can be implemented as a Platform as a Service (PaaS) cloud service. The thesis provides the reader a comprehensive overview of DevOps and its key tools and practices. Moreover, the thesis introduces common cloud service models and their characteristics. This lays the necessary groundwork for integrating DevOps into a PaaS cloud service. The thesis also explores how cloud-based DevOps services can enhance the quality of a software product project. Since quality is a multifaceted term, it is precisely defined, and a new framework is established to support the research. An example of a PaaS-based cloud service is Microsoft Azure DevOps. The thesis examines Azure DevOps to understand how it implements DevOps practices. Furthermore, Azure DevOps is utilized to explore how a DevOps cloud service can improve the quality of a software product. It is important to note that Azure DevOps is a relatively new product so previous research material on it is still fragmented. The thesis was conducted as a case study, and the interview segment was carried out through thematic interviews with a team utilizing Azure DevOps.

Keywords: Platform as a Service (PaaS), Microsoft Azure DevOps, cloud service, software development project, continuous integration and delivery

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Sovelluskehitys on muuttanut muotoaan viime vuosina merkittävästi. Monet uudet teknologiat ovat muuttaneet ympäristöä, ja organisaatioiden on sopeuduttava muutokseen pärjätäkseen markkinoilla. Ihmisen osuus ja kokemus on yhä keskeisemmässä osassa, ja kehityksen tueksi on noussut useita eri toimintamalleja ja työkaluja. Terminä työkalulla ohjelmistokehityksessä tarkoitetaan apuvälinettä, joita voidaan hyödyntää esimerkiksi ongelmanratkaisuun, kehityksen nopeuttamiseen ja helpottamiseen sekä ideoiden esittämiseen ja kommunikointiin (Riddle & Fairley, 2012). Ketterä sovelluskehitys (agile development), josta käytetään myös nimitystä agile, on suurilta osin korvannut perinteisen vesiputousmallin, jossa sovelluskehitysprojekti etenee lineaarisesti vaiheesta toiseen. Vesiputousmallin ongelmina ovat tyypillisesti jäykkyys ja kyvyttömyys sopeutua muutoksiin. Eri tiimit työskentelevät erillään toisistaan, mikä vaikeuttaa kommunikaatiota ja ongelmanratkaisua. Agilessa taas projektia iteroidaan ja kehitetään jatkuvasti ja se sopeutuu nopeasti eri olosuhteisiin. Eri tiimit voivat olla läsnä jo projektin alkuvaiheista alkaen. Myös asiakkaat voivat osallistua kehitysvaiheisiin mukaan. Scrum ja Kanban ovat yleisesti käytettyjä menetelmiä agilessa. Scrum sisältää useita sprinttejä, jotka sisältävät omat prosessinsa sovelluskehitysprojektin vaiheen mukaan. Kanban on taas visuaalinen esitystapa työn roolinjaon ja aikataulutuksen hallintaan. (Hoda ym., 2018.)

DevOps, eli Development and Operations tukee agilea ja tuo organisaatiolle työkaluja sekä käytänteitä projektien hallintaan. Nimensä mukaisesti keskeisessä osassa ovat kehityksen ja operoinnin avoin yhteistyö erillään työskentelevien tiimien sijaan. DevOps pyrkii integroimaan kehityksen, toimituksen ja operatiivisen toiminnan ja ylläpitää jatkuvaa ominaisuuksien toimitusta. Jatkuvien toimituksien ansiosta voidaan julkaista tiheästi pieniä päivityksiä ja jatkuva integrointi varmistaa koodin luotettavuuden ja kääntyvyyden. Automaatio ja testaaminen ovat keskeisessä osassa jatkuvan integroinnin ja toimituksen onnistumiseksi. DevOps-toimintamallin käyttöönotto organisaatiossa voi kuitenkin vaatia merkittäviä muutoksia työkulttuuriin. Operoinnin, kehityksen ja laadunhallinnan yhteistyön on oltava saumatonta. DevOps pyrkiikin irrottamaan organisaatiot perinteisestä siiloutuneesta rakenteesta. (Ebert ym., 2016.)

Yksi huomattava muutos organisaatioiden toimintaan ovat myös pilvipalvelut, jotka ovat yleistyneet viime vuosina merkittävästi. Käsitteenä pilvipalvelulla tarkoitetaan palvelua, joka tarjoaa erilaisia tietokoneresursseja verkon kautta. Pilvipalveluilla voi olla lukuisia samanaikaisia käyttäjiä ja resurssien määrää voidaan mukauttaa tarpeen mukaan. Pilvipalvelun keskeisiä piirteitä ovat muun muassa saavutettavuus, skaalautuvuus ja välitön käyttöönotto. Pilvipalvelut voidaan siis saavuttaa verkon kautta missä ja milloin vain, ne skaalautuvat resurssien tarpeen mukaan ja käyttöönotto on usein nopeaa ja automatisoitua verkon kautta. (Mell & Grance, 2010.) Yhä useampi yritys turvautuu pilvipalveluiden käyttöön ja myös pilvipalveluntarjoajien määrä kasvaa jatkuvasti. Käytössä pilvipalvelut tuovat käyttäjilleen lukuisia hyötyjä. Tarvittavia ohjelmistoja ei tarvitse kehittää itse, eikä yrityksen tiloihin tarvitse esimerkiksi koota palvelinkoneita laskentaa ja tiedon tallentamista varten. Yleistymisestä kertoo, että kevään 2022 aikana tehdyssä tutkimuksessa jo 81% suomalaisista yrityksistä käytti pilvipalveluita ja määrä on kasvanut tasaisesti vuodesta toiseen (Tilastokeskus, 2022). Suomi on myös EU-alueen maista prosentuaalisesti suurin pilvipalveluiden käyttäjä (Laine, 2021).

Tutkielmassa tutkitaan, miten DevOps ja pilvipalvelut voidaan yhdistää toisiinsa. Käytännössä tämä tarkoittaa sen määrittelyä, mitä ominaisuuksia pilvipalvelun on sisällytettävä, jotta se voi toteuttaa DevOpsia. Jotta aiheesta voidaan nostaa johtopäätöksiä, on DevOps-toimintamalli itsessään käytävä tarkkaan läpi. Erityistä huomiota saavat DevOpsiin keskeisesti liittyvät automaatio ja testaus sekä jatkuvat integrointi ja toimitukset. On hyvä muistaa, ettei DevOps itsessään ole teknologia, vaan käytänteitä ja työkaluja sisältävä toimintamalli (Khan, M.S. ym., 2022). Kun organisaatio ottaa DevOpsin käyttöön, on myös hankittava siihen soveltuvat ohjelmistot. Ohjelmistojen ostaminen erikseen voi olla työlästä ja kallista. Samoin ohjelmien asentaminen käytettäville laitteille pilven sijaan voi aiheuttaa rajoituksia käytölle. Lisäksi usean palveluntarjoajan ohjelmistot eivät välttämättä toimi keskenään. Pilvipalvelu taas voi sisältää kaiken samassa paketissa.

Käytännön toteutuksen esimerkkikohteeksi valitaan Microsoftin PaaS-pilvipalvelu Azure DevOps. PaaS (Platform as a Service) tarkoittaa pilvessä toimivaa palvelualustaa, joka on helppo ja nopea ottaa käyttöön ja sisältää kattavia konfigurointimahdollisuuksia ja työkaluja sovelluskehittäjille. (Soni, 2017.) PaaS-palveluille tyypillisesti Azure DevOps tarjoaa käyttäjilleen ympäristön sovelluskehitykseen. Azure DevOps on ilmainen alle viiden hengen tiimeille ja tarjoaa erilaisia hinnoittelupaketteja suuremmille käyttäjämäärille. Tärkeimpiä ominaisuuksia ovat muun muassa Azure Pipelines, joka tarjoaa versionhallinnan ja yhteyden GitHubiin, Azure Boards työn etenemisen ja vaiheiden monitorointiin ja Azure Test Plans manuaaliseen ja automatisoituun testaamiseen. Azure DevOpsia voidaan käyttää myös sekä yrityksen yksityisen pilven että julkisen pilven kanssa. Azure DevOpsin työkalujen avulla sovelluskehitysprojekteista pyritään saamaan onnistuneempia ja laadukkaita. (Krill, 2018.) Microsoft ei ole ainoa palveluntarjoaja, joka tuottaa PaaS-palveluita DevOps-toimintamallin hyödyntämiseen. Esimerkiksi Amazon AWS sisältää DevOps ominaisuuksia, kuten AWS

CodePipeline jatkuvaan toimitukseen ja integraatioon, AWS CodeBuild koodin kääntämiseen ja testaamiseen ja AWS CodeDeploy julkaisun automaatioon (Amazon, 2023). Myös Google Cloud tukee DevOpsia ja noudattaa ominaisuuksillaan samoja periaatteita kuin Azure ja AWS (DevOps, 2023).

DevOps-toimintamallista ja sen hyödyntämisestä yrityksen toiminnassa on paljon aiempaa tutkimusta. Esimerkiksi Noorani, Zamani, Alenzei ja Shameem (2022) tutkivat DevOpsin hyödyntämistä ohjelmistokehitysyrityksissä. Organisaatiot hyödyntävät DevOpsia yhdistämään kehityksen ja toiminnan saman katon alle ja tuottamaan tehokkaasti asiakkaiden toiveita vastaavia tuotteita. (Noorani ym., 2022.) Angara, Prasad ja Sridevi (2020) taas käsittelivät DevOpsia projektinhallinnan ja projektin eri vaiheiden kannalta. He totesivat, että projektipäällikön on kyettävä jatkuvasti johtamaan jatkuvaa suunnittelua, integrointia, julkaisua ja monitorointia projektin eri vaiheissa. Roolien muutokset ja automatisointi DevOpsin myötä voivat tehdä projekteista monimutkaisempia ja jopa heikentää projektin lopputulosta. (Angara ym., 2020.) Samoin pilvipalveluista ja niiden luokittelusta ja määrittelystä löytyy paljon lähdemateriaalia. Esimerkiksi Yhdysvaltain standardisointi- ja teknologiainstituutti NIST:n raportti ”The NIST Definition of Cloud Computing” on yleisesti tunnettu ja usein viitattu asiantuntijoiden kohdalla. Se tarjoaa kattavan kuvan eri pilvipalvelumallien toteutuksesta ja ominaisuuksista. (Mell & Grance, 2010.)

Tutkimuksen määrä taas Microsoftin Azure DevOpsista on puutteellista. Tutkimuksia siitä, miten Azure DevOps toteuttaa DevOps-toimintamallin periaatteita, on hyvin vähän. Myös Azure DevOpsin hyödyntäminen sovelluskehitysprojektien laadun parantamisessa on nykyään ajankohtainen asia, josta on vaikea löytää aiempia tutkimuksia. Kuten aiemmassa kappaleessa mainittiin, ei DevOps ole toteutettavana helppo malli ja siksi Azure DevOpsin käyttöönottoa ja toteutusta pitäisi pohtia tarkkaan. Azure DevOpsin joistakin toiminnoista ja ominaisuuksista löytyi yksittäisiä raportteja, kuten Web 3 -teknologioiden integroinnista, Microsoft .NET -ympäristön käyttämisestä ja Azure DevOpsin hyödyntämisestä maatalousteknologian kohdalla. Ylipäätään tutkimukset Azure DevOpsista ovat kuitenkin pirstaleisia ja aiemmilla tutkimuksilla on vaikea saada kokonaista ja selkeää kuvaa palvelun käytöstä. Tämä on selkeä aukko, jonka tutkimus pyrkii täyttämään.

Tutkielma myös selvittää, miten DevOpsiin pohjautuvat pilvipalvelut auttavat ohjelmistoyrityksiä projektin laadussa. Ohjelmistotuotannon projektit ovat paikoin isoja ja haastavia kokonaisuuksia, joissa aikataulut voivat venyä ja budjetti rikkoutua. Lisäksi valmis tuote ei välttämättä vastaa asiakkaan tai kehittäjän odotuksia. Laatu ei kuitenkaan ole terminä yksiselitteinen, vaan sitä voidaan mitata eri näkökulmista. DevOps ja pilvipalvelut tukevat luonnollisesti myös ohjelmistokehityksen sisäisiä prosesseja. Ne nopeuttavat sovelluskehitystä, vähentävät manuaalisia ja toistuvia työvaiheita ja löytävät automaattisen testauksen avulla virheitä (M. S. Khan ym., 2022). Laatua mitatessa tutkielma kuitenkin keskittyy enemmän itse lopputuotteen laatuun. Aiemman tutkimuksen pohjalta luodaankin laadun mittaamisessa hyödynnettävä viitekehys. Viitekehys käsittelee projektin laatua eri näkökulmista ja sisältää keskeisimmät vaatimukset.

Tutkimuskysymykset ovat siis seuraavat:

- Miten DevOps-toimintamallin käytänteitä voidaan toteuttaa PaaS-palveluna?
    - Miten Azure DevOps toteuttaa niitä?
- Miten DevOpsiin pohjautuvat pilvipalvelut auttavat ohjelmistoyrityksiä projektien laadussa?

Tutkielma sisältää yhteensä yhdeksän lukua, joista kirjallisuuskatsauksen osuus loppuu lukuun viisi. Ensimmäinen luku on johdanto, jossa käydään läpi tutkielman ja teorioiden taustoja ja tuodaan ilmi, miksi aihetta pitää tutkia. Tutkielman toisessa luvussa esitellään ensin DevOps-toimintamalli. Alaluvuissa käydään läpi DevOpsin keskeisimmät käytänteet ja käsitteet, eli organisaation kulttuuri, DevOps eri sovelluskehitysmallien kanssa ja automaatio ja jatkuva toimitus. Lisäksi luvussa tuodaan ilmi, mitä laatu käsitteenä tarkoittaa ja miten laatua mitataan tutkielman kohdalla. Tämä on tärkeää erityisesti toisen tutkimuskysymyksen kannalta. Kolmas luku taas keskittyy pilvipalveluihin. Pilvipalvelun luokittelu käydään läpi ja niiden keskeisiä ominaisuuksia tarkastellaan. PaaS-pilvipalvelu Azure DevOps esitellään tarkkaan. Kun pilvipalveluiden ja DevOpsin keskeiset käsitteet on tuotu ilmi, tutkii neljäs luku DevOpsin toteuttamista pilvessä. Luvussa myös käsitellään, miten Azure DevOps toteuttaa DevOpsin käytänteitä pilvessä. Luku viisi on yhteenveto teoriaosuudesta.

Kirjallisuuskatsauksessa kiinnitettiin huomiota lähdemateriaalin korkeaan laatuun. Hakualustana käytettiin Jyväskylän yliopiston JykDokia ja Google Scholaria. Hakukohteena olivat erityisesti kansainväliset tieteelliset ja vertaisarvioidut tutkimusraportit. Lähdekirjallisuus käytiin tarkkaan läpi oikeudellisuuden varmistamiseksi ja siitä johdettiin sekä kirjoittajan omia päätelmiä että esimerkkejä. Hakusanoja olivat esimerkiksi ”Azure DevOps, Azure DevOps and project management, DevOps and software development”. Lähteissä oli tärkeää niiden julkaisuvuosi. Azure DevOps on saanut alkunsa vuodesta 2005, mutta on muuttunut vuosien varrella merkittävästi ja saavutti nykyisen muotonsa vasta vuonna 2018. Sitä vanhempi Azure DevOpsiin liittyvä lähdemateriaali ei välttämättä ole enää oikeellista. Toisaalta DevOps ja pilvipalvelut juontavat juurensa kauemmaksikin, joten niiden kohdalla ei julkaisuvuoden kohdalla ole yhtä tiukkoja rajoituksia. Tämänhetkisistä Azure DevOpsin ominaisuuksista paras tieto löytyy Microsoftin sivuilta tai itse palvelun portaalista. Toisaalta on otettava myös puolueellisuus huomioon. Jos valmistaja ylistää oman tuotteensa sivulla, kuinka Azure DevOps parantaa merkittävästi projektin laatua, ei siitä voida vetää suoria johtopäätöksiä.

Empiirinen tutkimus alkaa luvusta kuusi. Tutkimus toteutettiin laadullisena teemahaastatteluna, jonka valinta perusteltiin tutkimuskohtaisesti. Lisäksi luku tuo ilmi, miten varmistetaan kerätyn aineiston luotettavuus ja huolellinen käsittely. Haastateltavien joukko esitellään ja tutkittavasta organisaatiosta annettiin lyhyt kuvaus. Seitsemäs luku esittelee haastatteluiden tuloksia. Se kokoaa ensin haastateltavien taustat. Sen jälkeen tuodaan aihealueittain ilmi syntyneet

tulokset. Pienetkin huomiot on pyritty tuomaan mukaan. Lopulta luvussa kahdeksan muodostetaan johtopäätöksiä haastatteluista kerätyn aineiston pohjalta. Tutkimuskysymyksiin pyritään vastaamaan kattavasti ja tuloksia verrataan myös kirjallisuuskatsauksen tuloksiin. Lopulta yhdeksäs ja viimeinen luku kokoaa yhteenvedon koko tutkielmasta.

2 DEVOPS

Organisaatiot ovat jatkuvan muutoksen kourissa ja myös sovelluskehitys on muuttanut muotoaan ajan kuluessa. Jatkuva käyttöönotto ja integrointi ovat yhä useammin läsnä sovelluskehittäjien arjessa ja testaus ja julkaisu ovat automatisoituja. Päivityksiä monitoroidaan jatkuvasti ja mahdolliset korjaukset voidaan julkaista nopeasti. (Zhu ym., 2016.) DevOps (Development and Operations) voidaan nähdä ketterän ohjelmistokehityksen evoluution tuotoksena. Ennen kaikkea, DevOps edistää kehityksen ja operoinnin saumatonta yhteistyötä. Aiemmassa siiloutuneessa sovelluskehityksen rakenteessa kehittäjät (developers) ja ylläpitäjät (operators) työskentelivät erillisissä yksiköissä ja omasivat omat prosessit, työkalut ja toimintaperiaatteet. Kehittäjät loivat ja julkaisivat uusia ominaisuuksia ja toimintoja järjestelmille, kun taas ylläpitäjät huolehtivat siitä, että järjestelmä pysyy päivityksistä huolimatta toiminnassa. Tyypillinen kommunikointiväline kehittäjien ja ylläpitäjien välillä oli tikettijärjestelmä, jossa tieto siirtyi tiketeillä. Päivityksien julkaisussa saattoi olla useiden päivien viiveitä ja kehittäjät ja ylläpitäjät saattoivat syyttää toisiaan syntyneistä ongelmista. Ongelmanratkaisu oli myös hidasta johtuen kehittäjien ja ylläpitäjien välisestä etäisyydestä ja hitaasta kommunikoinnista (Leite, 2020.). Tähän ongelmaan Humble ja Molesky (2011) lisäsivät, että monet kehittäjät eivät välttämättä ollenkaan ajaneet kehittämiään järjestelmiä, eivätkä välttämättä ymmärtäneet käytännön vaatimuksia niiden toiminnalta ollenkaan (Humble & Molesky, 2011).

DevOps-toimintamalli toi muutoksen sovelluskehitysorganisaatioiden kulttuuriin. Se integroi tehokkaasti aiemmin erilliset kehityksen ja ylläpidon tiimit nopeuttaen toimituksia ja päivityksiä ja vähentäen puutteellisesta kommunikaatiosta aiheutuvia ongelmia. DevOps vaikuttaa ohjelmiston elämänkaaren eri vaiheissa. Ohjelmisto käy alussa hyväksyttävänä eri tahoilla, toiminta eri ympäristöissä varmistetaan, päivityksiä julkaistaan jatkuvasti ilman taukoja ja järjestelmiä monitoroidaan julkaisun jälkeen mahdollisten ongelmien havaitsemiseksi. Myös testaus on automatisoitua, jolloin mahdolliset ongelmat havaitaan nopeasti. (Zhu ym., 2016.) DevOpsin työkalut sekä helpottavat sovelluskehittäjien työtä

että tuottavat asiakkaille ja loppukäyttäjille arvoa laadukkaamman ohjelmiston muodossa.

DevOps tuo siis sekä teknologian että kulttuurin muutoksen organisaatioille. DevOpsin tärkeimmät pilarit ovat ihmisten välinen yhteistyö eri yksiköiden välillä ja automaatio. Jotta DevOps voidaan ottaa organisaatiossa käyttöön, tulee varmistua siitä, että tiimien välinen yhteistyö saadaan nopeaksi ja läpinäkyväksi. Tämä voi olla monissa siiloutuneissa organisaatioissa haasteellista ja vaatii suuria muutoksia. (Leite, 2020.) Uudet teknologiat ja työkalut vaativat myös opettelua ja muutoksia toiminnalle. Koko ohjelmiston arkkitehtuuri voi mennä uusiksi. Jos ohjelmisto on DevOpsin käyttöönoton aikana käytössä, muutokset pitää tehdä inkrementaalisesti. (Zhu ym., 2016). Luonnollisesti uudet teknologian työkalut ja muutokset vaativat muutoksia myös ihmisten toimintatavoilta.

Luvussa esitellään DevOpsin tärkeimmät käytänteet aihealueittain. Ensimmäisessä alaluvussa käydään läpi DevOpsin vaikutuksia organisaation kulttuuriin. Toisessa alaluvussa taas tarkastellaan, miten DevOps toimii eri sovelluskehitysmallien kanssa. Kolmannessa alaluvussa käydään läpi automaation ja jatkuvan toimituksen käsitteitä ja toteuttamista. Lopuksi neljännessä alaluvussa tarkastellaan, mitä termillä ”laatu” tarkoitetaan ja miten sitä voidaan mitata.

2.1 Organisaation kulttuuri

DevOps tuo muutoksen organisaation kulttuurille. Kommunikaatio on tärkeää, jotta tieto kulkee eri tiimien ja yksiköiden välillä. Organisaation on tarjottava työkalut, käytänteet ja verkostot kommunikointiin. Tämä voi kuitenkin olla haastavaa etenkin toisistaan kaukana olevien tiimien vuoksi. (Leite, 2020.) Eri maantieteelliset sijainnit ja aikavyöhykkeet monimutkaistavat viestintää. Humble ja Molesky (2011) painottavat avoimuutta ja asioista puhumista seuraavasti: ”If your IT teams don’t talk and collaborate with each others throughout the service/system delivery lifecycle, bad things will happen” (Humble & Molesky, 2011). Tällä he tarkoittavat sitä, että onnistunut DevOps toimintamallin toteutus vaatii ihmisten ja tiimien avointa keskustelua. Tiedon pitää virrata eri tiimien välillä ja avoin keskustelu tuo ilmi myös ideoita järjestelmän tai palvelun parantamiselle. Ihmiset jakavat osaamistaan ja onnistumisiaan avoimesti organisaation sisällä. Puhumattomuus taas lisää riskiä toimituksien viiveelle, bugeille ja osapuolien turhautumiselle. (Humble & Molesky, 2011.) Siksi kommunikaatioon panostettava, vaikka se olisi kallista ja haastavaa esimerkiksi etäisyyksien takia (Leite, 2020).

Eri tiimien ja jäsenten on luotettava toisiinsa, mutta toisaalta organisaation kulttuurin on annettava tilaa myös virheille. Jotta uutta voidaan keksiä ja kehitys ja julkaisut voivat olla jatkuvia, tapahtuu väistämättä virheitä, joista lopulta opitaan. Kommunikaatio sisältää myös tiedon jakamisen eri tiimien ja yksiköiden välillä. Tämä sisältää esimerkiksi organisaation sisäisen Wiki-sivuston luomisen, josta eri osapuolet voivat hakea tietoa. Uuden toimintamallin käyttöönotto vaatii myös työkalujen, esimerkiksi automaation liittyvien, opettelun. Organisaation

on hyvä tarjota tehokkaat koulutukset sekä vanhoille työntekijöille, että uusille työntekijöille niin sanotun on-boarding-prosessin kautta. Ylipäätään sekä ylemmällä että alemmalla johdolla on tärkeä tehtävä sopivan kulttuurin luomiselle DevOpsin käyttöönottamiseksi. (Leite, 2020.)

2.2 DevOps ja eri sovelluskehitysmallit

Vesiputousmallia pidetään perinteisenä sovelluskehitysmallina, joka on jäämässä hiljalleen pois. Vesiputousmallissa sovelluskehitysprojekteilla on selkeitä vaiheita, joihin edetään tyypillisesti aina yksi kerrallaan (katso kuva 2). Vaiheita voivat olla esimerkiksi vaatimukset, suunnittelu, toteutus, testaus ja lopulta ylläpito. Kun agilessa palataan sovelluskehityksen vaiheisiin aina uudelleen, ei perinteisessä vesiputousmallissa palata enää taaksepäin. Yleisesti vesiputousmallin on katsottu olevan kankea ja huonosti nykyaikaisiin ohjelmistokehitysprojekteihin mukautuva jo pitkään. (Hoffman, 2020). McConnel (1996) nostaa kuitenkin esille termin muokattu vesiputousmalli, jossa vaiheissa voidaan palata takaisin. Esimerkiksi tietty suunnitelmavaihe voi sisältää useita aliprojekteja, joiden pariin palataan useita kertoja. Vesiputousmallia voidaan myös toteuttaa asteittain, jolloin edellisiin kokonaisvaiheisiin palataan uudelleen. (McConnell, 1996.) Tämä on selvä ero Hoffmanin (2020) kuvailuun perinteisestä vesiputousmallista, jossa vaiheissa ei palata enää ylöspäin (Hoffman, 2020).

Vesiputousmalli soveltuu parhaiten sovelluskehitysprojekteihin, jotka ovat selkeästi määriteltyjä, ennustettavia, eivätkä muutu merkittävästi kehityksen aikana. Esimerkkinä vesiputousmallia käyttäville projekteille voidaan nostaa aiemmin mainitut tiukkojen lainsäädäntöjen ja byrokratian alla toimivat projektit. Vesiputousmalli ja DevOps ovat erilaisia lähestymistapoja sovelluskehitykseen, eikä niiden katsota toimivan kovin hyvin yhdessä. Esimerkiksi jatkuva toimitus ja integrointi ovat hankalia toteuttaa, jos sovelluskehityksessä ei voida palata eri vaiheisiin. Agilen sprinteissä näihin taas palataan luonnostaan. Vesiputousmalli ei yksinkertaisesti jousta tarpeeksi DevOpsin jatkuvaan kehitykseen ja iterointiin ja viivästys yhdessä vaiheessa viivästyttää koko projektia. Siiloutuneissa organisaatioissa myös eri tiimit toimivat erillään ja kommunikaatio ja yhteistyö eivät ole riittävän avointa DevOpsin käytänteiden täyteen hyödyntämiseen. Toki joitakin DevOpsin työkaluja voidaan käyttää myös vesiputousmallin kanssa. Eri vaiheiden lopussa testaaminen voidaan hoitaa DevOpsin automaation avulla, mikä nopeuttaa vaiheita ja parantaa tuotteen lopullista laatua. Samoin työkalut, kuten säilytyspaikat, versionhallinta ja kontit, voidaan katsoa eduiksi kehityksessä. (Hoffman, 2020.)

Mohammad (2017) mainitsee tutkimuksessaan, että DevOps sekoitetaan usein agile-toimintatapaan (Mohammad, 2017). Agile tarkoittaa inkrementaalista lähestymistapaa sovelluskehitykseen, jossa kehitystä käydään pienissä vaiheissa ja näihin vaiheisiin palataan aina uudestaan (katso kuva 1). Ohjelmistoa parannetaan jatkuvasti esimerkiksi asiakkaalta saadun palautteen pohjalta. Agilen

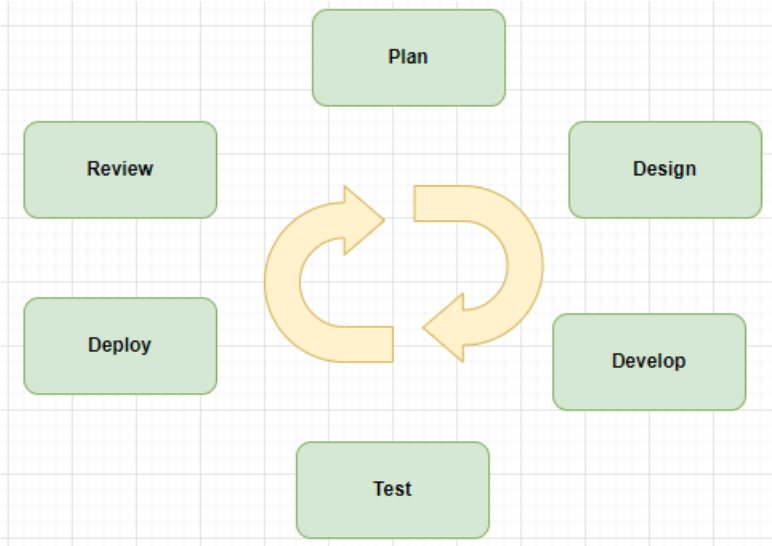
tärkeimmiksi peruspilareiksi voidaan katsoa asiakkaan osallistuminen kehitykseen jo aikaisessa vaiheessa, iteratiivinen kehitys, itseorganisoituvat tiimit ja sitoutuminen muutoksiin (Leau ym., 2012.). Sekä agilea että DevOpsia markkinoidaan laajoilla termeillä ja veteen piirretyillä rajoilla ja molemmat keskittyvät enemmän toimintatapoihin ja käytänteisiin aktiviteettien sijaan. Lisäksi sekä agile että DevOps auttavat tiimejä keskittymään työhön nopeammin ja tehokkaammin. Mohammed kuitenkin painottaa, että Agile keskittyy enemmän projektinhallinnan johtamiseen ja joustavuuteen, kun taas DevOps keskittyy julkaisuputken optimointiin ja jatkuvuuteen. Käytännössä Agile siis painottaa toimintoja yleisesti ja DevOps toimintojen tehokkuutta ja automaatiota. Agile on korvaamassa kovaa vauhtia vanhemman vesiputousmallin. DevOps tukee agilen ketteryyttä, reagointikykyä ja nopeutta. (Mohammad, 2017.)

DevOps toimii tukena kaikissa kuudessa tyypillisessä agilen vaiheessa (katso kuva 1). DevOps auttaa projektin tavoitteiden ja työnjaon määrittelyä ja tukee tiimin järjestämistä suunnittelussa. Kehityksen ja testauksen vaiheessa jatkuva integraatio ja automaattinen testaus varmistavat ohjelmistotuotteen toimivuuden ja laadun ja toimituksessa DevOpsin jatkuva toimitus nousee osaan. Lopuksi DevOps tukee julkaistun tuotteen monitorointia ja tuottaa tilastoja, joka tukee Agilen arviointivaihetta. (Mohammad, 2018.) Almeida ym. (2022) lisäävät, että Ylipäätään automatisaatio nopeuttaa koko agilen ”pyörimistä”. Kun ylimääräiset, manuaalista työtä vaativat vaiheet on korjattu automatiikalla, kestävät agilen vaiheet vähemmän aikaa (Almeida ym., 2022.)

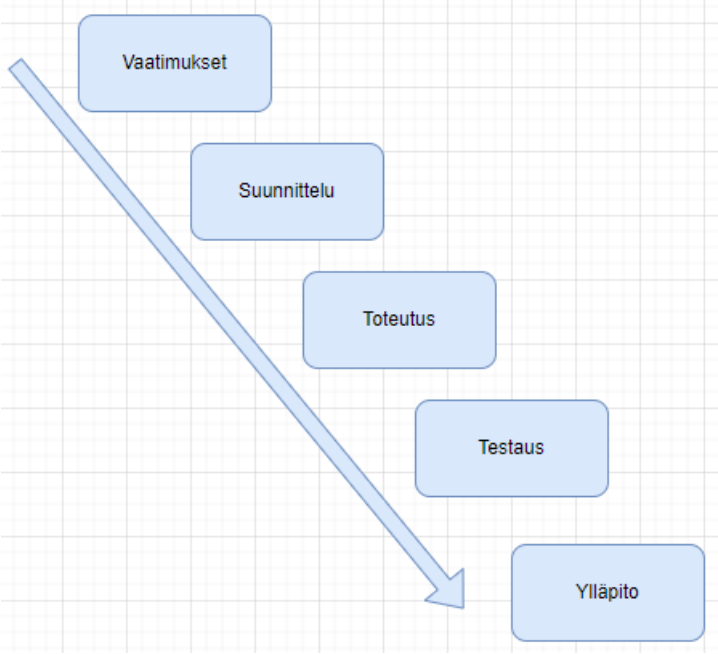
Yhtenä trendinä sovelluskehityksessä on asiakkaiden vaatimuksien jatkuva kasvu mikä kasvattaa halukkuutta tehostaa läpinäkyvyyttä ja avointa kommunikaatiota. Agilessa asiakas osallistuu projektin suunnitteluun ja luomiseen jo alkuvaiheessa. Näin tuotteesta saadaan jo alussa palautetta ja sitä on helppo muokata oikeaan suuntaan. Myös DevOpsissa sovelluskehitykseen voidaan ottaa mukaan sovelluskehitysprojektin ulkopuolisia tiimejä, kuten markkinointi, tai liiketoimintayksikkö, tai myös asiakkaita. Nämä osapuolet saavat läpinäkyvän kuvan, missä vaiheessa projekti on ja missä vaiheessa esimerkiksi toivottujen ominaisuuksien toteutukset ovat. Heidän osallistamisensa projektiin mahdollistaa kaiken potentiaalin käyttämisen projektiin ja korkeamman arvon tuottamisen lopputuotteessa. Jos johonkin sovelluskehityksen vaiheeseen pitää palata asiakkaan toiveen vuoksi, DevOps mahdollistaa muutoksen toteuttamisen käytännössä nopealla aikataululla. (Almeida ym., 2022.)

On myös olemassa lukuisia malleja, joiden avulla agilea voidaan hyödyntää käytännössä. Näitä menetelmiä ovat esimerkiksi Scrum, Kanban ja SAFe. Scrumin keskeissä osassa on tyypillisesti yhdestä neljään viikkoon kestävät sprintit, joiden aikana kehitetään tietyt toiminnallisuudet. Tiimityö on keskeisessä osassa ja Scrum-tiimin roolit onkin ennalta määrätyt. (Jacobson, 2022.) Kanban on visuaalinen tiedonesitysmuoto, jossa projektin kehitystä esitellään taulun muodossa. Taulu jakaa työn eri vaiheisiin, esittelee roolit ja näyttää, mitä milläkin hetkellä on työn alla. (Braglia ym., 2020.) SAFe, eli Scaled Agile Framework on viitekehys, joka sisältää malleja ja käytäntöjä ja tukee organisaatioita hyödyntämään agilea laajasti. SAFe koostuu kolmesta eri tasosta, eli Portfolio-, Program-, ja Team-

tasoista, joilla on tehtävänsä projektien eri vaiheissa. Lisäksi SAFe sisältää kattauksen työkaluja ja käytäntöjä agilen soveltamiseen. (Turetken ym., 2017.) Kuten agilen eri vaiheissa, tukee DevOpsin jatkuva toimitus ja integraatio ja automaattinen testaus Scrumia tehokkaasti. Työkalujen avulla huolehditaan, että Scrumin sprintit pysyvät tavoitteissaan ja aikatauluissaan. Kanban käytöstä DevOpsissa taas voidaan nostaa esimerkiksi Azure DevOpsin Boards-välilehti, joka on Kanban-muodossa. Toisaalta Mohammad (2017) nostaa esille sen, että DevOps ei välttämättä suoraan tue kaikkia agilen menetelmiä ja joissakin prosesseissa on tehtävä kompromisseja (Mohammad, 2017). Myös SAFe nousee ongelmalliseksi Turetken ym. (2017) tutkimuksessa. Vaikka SAFe:n katsotaan pääasiassa toimivan DevOpsin kanssa, voi tiukka tiimien määrittely rajoittaa tiimien autonomiaa ja joustavaa yhteistyötä muiden tiimien välillä (Turetken ym., 2017).



KUVIO 1 Agilen vaiheet kuvattuna: Vaatimuksien suunnittelu, tuotteen suunnittelu, kehittäminen, testaaminen, julkaisu ja arviointi.



KUVIO 2 Perinteinen vesiputousmalli kuvattuna

2.3 Automaatio ja jatkuva toimitus

DevOpsin tärkeimmiksi periaatteiksi voidaan nostaa automaatio, jatkuva toimitus ja jatkuva integrointi. Automatisoinnilla tarkoitetaan DevOpsissa eri ohjelmistokehityksen vaiheiden ja prosessien automatisointia manuaalisen työn sijaan. Aktiviteettien, kuten integraation, testauksen, koodin kääntämisen ja ohjelmiston toimituksen, automatisointi vähentää inhimillisiä virheitä ja nopeuttaa julkaisua. Automatisointiin kuuluvat vaiheet aina lähdekoodin kirjoittamisesta loppukäyttäjälle näkyvään palveluun tai järjestelmään asti. Automatisointiin on myös tarjolla lukuisia työkaluja. Lähdekoodin hallinta ja versionhallinta ovat joukko käytänteitä, jotka mahdollistavat lähdekoodin tallennuksen, muutoksien tarkastelun ja useiden tiimien työskentelyn samaan aikaan. Koontityökalut kääntävät koodia ja valmistavat sen ajettavaksi ohjelmaksi. Tähän liittyy myös aiemmin mainittu jatkuva integrointi. Konfiguroinnin hallinta sisältää ominaisuuksien ja tuotteiden jäljittämisen koko niiden elinkaaren aikana. Esimerkiksi asiakkaan toivoma ominaisuus tulee ensin toiveena ja ohjeistuksena ja lopussa toteutettua ominaisuutta monitoroidaan ja varmistetaan, että se toimii oikein. (Mohammad, 2018.)

Automaattinen testaus on DevOpsin yksi tärkeimpiä periaatteita. Testaamisella varmistetaan ohjelmiston toimivuus ja tarjolla on lukuisia eri työkaluja sen toteuttamiseksi. Lisäksi työkaluja ovat vielä tietokantojen hallinta, monitorointi

ja yhteistyö. Tietokannat sisältävät ja käsittelevät ohjelmistokehityksessä syntyvää ja käytettävää tietoa. Monitorointi tarkkailee palveluiden ja järjestelmien toimintaa aina muistin käyttöasteesta tiettyjen ominaisuuksien soveltuvuuteen. Lisäksi kuten aiemmin mainittiin, on yhteistyö ja viestintä avainasemassa DevOpsin onnistumisessa. Yhteistyön käytänne sisältää avoimuuden käsitteen, jossa omia ideoita, ajatuksia ja onnistumisia jaetaan tiimin kesken. (Mohammad, 2018)

Jatkuva toimitus mahdollistaa päivityksien jakamisen ilman järjestelmän tai palvelun saavutettavuuden heikentymistä. Käytännössä tämä tarkoittaa sitä, että pieniä päivityksiä voidaan jakaa hyvinkin nopealla ja tiheällä aikataululla ilman käyttökatkoksia palvelussa. Tiimit vastaanottavat nopeasti palautetta päivityksistä, saavat kattavamman kuvan toimitusprosesseista ja myös riskit ja kustannukset jakelusta pienentyvät. Päivityksistä tulee organisaatiolle, kehittäjille ja käyttäjille rutiinia. (Humble & Molesky, 2011.) Jokainen säilytyspaikkaan (eng. repository) tallennettu versio ohjelmistosta on oltava tuotantoon tarkoitettu. Automaattisten testien ja ohjelman koonnin jälkeen päivitys voidaan julkaista nopeasti. (Leite, 2020.) Rossel (2017) määrittelee kirjassaan jatkuvan käyttöönoton (Continuous Deployment) sijoittuvan jatkuvan toimituksen prosessin sisään. Jatkuva käyttöönotto sisältää aina jonkun manuaalisen toimenpiteen, joka on tehtävä, jotta jatkuva toimitus toteutuu. Koska DevOpsissa käyttöönotettava koodi on aina tuotantoon tarkoitettu, on käyttöönotto kuitenkin usein nopeaa ja jopa vain yhden napin painallus. Sekä jatkuva käyttöönotto ja toimitus loppuvat tietyn päivityksen osalta, kun ohjelmisto on julkaistu ja otettu käyttöön. (Rossel, 2017.) Jatkuva käyttöönotto ja toimitukset kuitenkin jatkuvat aina uusien päivityksien kohdalla.

Jatkuvalla integroinnilla (Continuous Integration) puolestaan tarkoitetaan sitä, että ohjelmisto on aina julkaistavassa kunnossa. Koodin pitää kääntyä ja olla laadultaan hyvää. Koodit ajetaan säilytyspaikkaan useasti päivässä ja jos testit menevät läpi, koodi voidaan julkaista ja integroida päätuotteeseen. Koodia voidaan testata inkrementaalisesti pienissä osissa tai kerralla, kun kaikki komponentit ovat valmiita. (Rossel, 2017.) Jatkuvalla integroinnilla voidaan siis varmistaa se, että koodi toimii kuten pitääkin ja on yhteensopivaa kaikkien tiimien ja työntekijöiden työn kanssa.

DevOpsin kohdalla voidaan myös nostaa esiin termi ALM, eli Application Lifecycle Management. ALM on viitekehys sovelluskehityksen elinkaaren hallintaan, joka sisältää koko ohjelmiston elämänkaaren aina ideasta käytöstä poistamiseen asti. DevOps ei suoraan sisällä ALM:ää, mutta ALM tukee sekä Agilen että DevOpsin lähestymistapoja sovelluskehitykseen liittyen. Samoin jatkuvat toimitukset, automatisoidut prosessit ja kattava testaaminen kuuluvat viitekehykseen. (Redhat, 2020.) Myös Microsoft Azure DevOpsin ohjesivut ohjaavat organisaatioita automatisoidun ALM:n käyttöön (Microsoft, 2023). Tämän myötä ALM on hyvä esitellä terminä DevOpsin rinnalla.

ALM kattaa aktiviteetit hallintoon, kehitykseen ja ylläpitoon liittyen. Kolme ALM:n keskeistä käsitettä ovat jäljitettävyys, näkyvyys ja prosessien automaatio. Jäljitettävyys tarkoittaa kykyä jäljittää ohjelmistokehityprojektiin

liittyviä artefakteja ja linkittämään niitä toisiinsa. Mitä isompia projektit ovat, sitä vaikeampaa jäljitys on ja kokonaisuuden hallinta vaikeutuu. Läpinäkyvyys sisältää avoimen kuvan siitä, mitä osia sovelluskehitykseen kuuluu ja missä vaiheessa projekti on. Tämä on tärkeää tiimien koordinoinnin ja valvomisen suhteen. Lopuksi ALM painottaa ohjelmistokehityksen prosessien automatisointia koko projektin läpi. Tällä pyritään vähentämään virheitä ja nopeuttamaan kehitystä. (Tüzün, 2019.)

Kun DevOpsin työkaluja käy läpi, on tärkeä muistaa, että DevOps itsessään ei ole teknologia. Se on toimintamalli, joka sisältää käytänteitä ja työkaluja ohjaamaan yrityksen toimintaa. DevOps ohjaa yrityksiä käyttämään tiettyjä teknologioita esimerkiksi prosessien automatisointiin. Yhdessä DevOps-kulttuurin kanssa, yritykset voivat saavuttaa tavoitteitaan tehokkaan ja nopean sovelluskehityksen saralla. DevOps-pilvipalvelut, kuten Azure DevOps ja AWS DevOps, ovat puolestaan teknologioita, jotka toimivat DevOpsin käytänteiden mukaan ja ohjaavat organisaatioita jatkuvaan julkaisuun ja automaatioon.

DevOps ei ole automaattinen valinta ja ongelmien ratkaisija kaikille organisaatioille. Monilla organisaatioilla on yhä käytössään vanhoja ja monimutkaisia Legacy-järjestelmiä. Legacy-tietojärjestelmät voivat edelleen täyttää organisaation ja asiakkaiden vaatimukset ja niiden uudistaminen olisi valtava ja kallis projekti. Usein Legacy-tietojärjestelmät koostuvat monista pienistä järjestelmistä ja jatkuvat toimitukset ja päivityksien jakelu olisivat hyvin vaikeita totetuttaa. Schaller (2016) toteaakin, että Agilen ja DevOpsin sijaan Legacy-tietojärjestelmien kohdalla perinteinen vesiputousmalli toimii edelleen hyvin. Projekteilla on selkeät ja muuttumattomat vaatimukset ja muutokset dokumentoidaan kattavasti. (Schaller, 2016.)

Tiukkojen lainsäädäntöjen alla toimivat yritykset voivat kohdata haasteita DevOpsin toteuttamisessa, eivätkä esimerkiksi jatkuvat toimitukset ja automaatio välttämättä sovi niille (Ebert ym., 2016). Tähän Laukkarinen ym. (2018) lisäävät, että laki määrittää vaatimukset luotettavuudelle, näkyvyydelle ja jäljitettävyydelle ohjelmistokehitysprojektien kohdalla (Laukkarinen ym., 2018). 2014 julkaistu DevOpsin koulutusta ja tietoa sisältävä sivusto listaa, miksi DevOps soveltuu hyvin myös tiukasti säännellyille organisaatoille. Sivusto perustelee eduiksi muun muassa avoimen kommunikaation kaikkien sidosryhmien kanssa, selkeät prosessit ja jäljitettävyyden ja automaation, joka vähentää virheitä ja lisää ennustettavuutta. (Davies, 2018.) Tämä on kuitenkin osin ristiriidassa tutkimuksien kanssa. Esimerkiksi Laukkarinen ym. (2018) totesivat, että ohjelmistokehityksen lainsäädännössä on uudistamisen varaa, jotta ne sopisivat paremmin DevOps-käytänteiden kanssa. Samoin DevOps-työkaluissa pitäisi keskittyä vielä enemmän jäljitettävyyteen, hierarkiaan ja standardeihin malleihin, jotka sopivat lainsäädännön kanssa yhteen. Tutkimuksessa myös todettiin, että aihetta pitää tutkia lisää eri näkökulmista. (Laukkarinen ym., 2018.) Ebert (2014) taas painotti, kuinka jatkuvat pienet päivitykset voivat vaarantaa kriittisien järjestelmien toiminnan, joissa virheitä ja toimintakatkoksia ei sallita (Ebert ym., 2016). Tämän myötä DevOpsiin keskittyvän sivuston väitteitä ei voida pitää puolueettomina ja täysin luotettavina.

2.4 Ohjelmistokehityksen laadun määrittely ja mittaaminen

Yhtenä tutkimuskysymyksenä on, miten DevOpsiin pohjautuvat auttavat ohjelmistoyrityksiä projektin laadun toteutumisessa. Tässä tilanteessa kysymyksellä viitataan lopputuotteen laatuun ja onnistumiseen. Tämä tarkoittaa valmista ja käytettävää tuotetta. Toinen näkökulma olisi tarkastella ohjelmistokehityksen eri vaiheita ja prosesseja ja niiden laatua ja onnistumista. Tämä näkökulma kuitenkin jätetään pois tutkimuskysymyksestä. Jotta tutkimuskysymykseen voidaan vastata, pitää projektin laatu ja onnistuminen määritellä.

DeLone-McLeanin malli on yleisesti tunnettu ja hyväksytty viitekehys informaatioteknologian tutkimuksissa. Mallilla voidaan mitata ohjelmiston laatua ja alun perin se koostui järjestelmän laadusta (SysQ, System Quality) ja informaation laadusta (InfQ, Information Quality). Myöhemmin muuttujaksi tuotiin myös palvelun laatu (SerQ, Service Quality), joka yhdistää yksilön vaikutukset ja organisaation vaikutukset. Malli katsoi keskeiseksi tekijäksi projektin onnistumiselle erityisesti käyttäjätyytyväisyyden. Käyttäjätyytyväisyys itsessään on haastava mitattava. Malli kuitenkin katsoo järjestelmän laadun, informaation laadun ja palvelun laadun käyttäjän näkökulmasta tarkasteltaviksi piirteiksi. Tämän myötä käyttäjätyytyväisyyttä voidaan mitata niillä. (Montesdioca & Maçada, 2015.) Myös Akbar ym. (2018) määrittelevät projektin onnistumisen merkittävimmäksi mittariksi loppukäyttäjän tyytyväisyyden. Projekti on ainakin loppukäyttäjän tai asiakkaan kannalta onnistunut, jos se tuottaa arvoa ja täyttää vaatimukset. He myös huomauttavat, että ohjelmistotuotteen laatu ei voi kuitenkaan muodostua pelkästään loppukäyttäjän tyytyväisyydestä. Jos esimerkiksi budjetti on ylittynyt moninkertaisesti, ei projekti ole silloin onnistunut. Laadun määrittelyyn on siis syytä ottaa loppukäyttäjän tyytyväisyyden lisäksi huomioon myös aika ja budjetti. (Akbar ym., 2018.) Projekti voidaan katsoa onnistuneeksi ja laadultaan hyväksi, kun se on valmistunut aikataulun ja budjetin puitteissa ja täyttää loppukäyttäjän tai asiakkaan vaatimukset. Näiden kaikkien kolmen kriteerin täyttäminen ei kuitenkaan ole nykypäivän ohjelmistokehitysyrityksille helppoa.

Thomas (2008) tutki 36 organisaation määrittelyä projektin onnistumiselle ja esittelivät tarkempia kriteereitä IT-projektien onnistumiselle ja niiden mittaamiselle. Tärkeänä huomiona nostettiin, että organisaatiot, jotka määrittelevät onnistumisen, mittaavat onnistumista ja toimivat mittaustulosten mukaan, onnistuvat projekteissaan keskimääräistä paremmin. Toisaalta yhtä selkeää metodia onnistumisen mittaamiselle ei löytynyt. Vain kolmen määritteen sijaan tutkimus löysi yhteensä 14 kriteeriä onnistumiselle. Pelkästään asiakkaan tai loppukäyttäjän tyytyväisyys ei riittänyt, vaan myös sijoittajien, ohjausryhmien, projektitiimin ja muiden sidosryhmien tyytyväisyyttä tarkasteltiin. Teknisestä näkökulmasta otettiin huomioon järjestelmän käyttöönotto, täytetyt vaatimukset, järjestelmän laatu ja käytettävyys. Liiketoiminnan kannalta otettiin huomioon toiminnan jatkuvuus, liiketoiminnan tavoitteet ja etujen toimitus. (Thomas, 2008.)

Ghanbari ym. (2018) totesivat laadun mittaamisen haasteeksi ihmisten erilaiset näkökulmat ja asenteet teknologiaa kohtaan. Tavalliset käyttäjät eivät

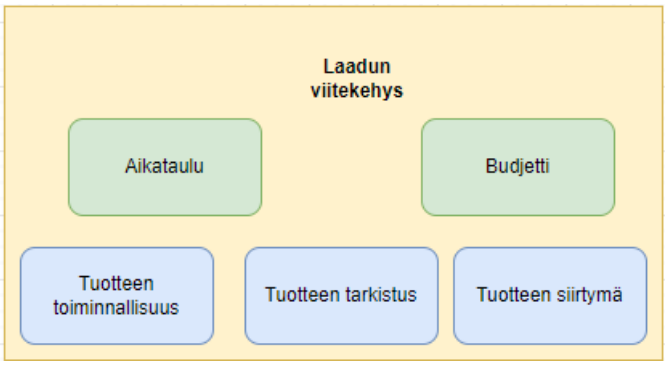
välttämättä välitä pienistä ongelmista ohjelmiston käytössä ja hyväksyvät ne. (Ghanbari, 2018.) Tämä on yksi syy asiakastyytyväisyyden mittaamisen haastavuudelle, josta myös Montesdioca ja Maçada mainistivat (Montesdioca & Maçada, 2015). Jos pienet ongelmat tuotteessa katsotaan jatkuvasti sormien läpi, voidaan ohjelmistokehityksessä oikaista vaatimuksien kohdalla yhä enemmän. Tästä voi seurata kriittisiä bugeja ja haavoittuvuuksia. Yhden laadun kriteerin täyttämiseksi voidaan myös tehdä uhrauksia toisen kriteerin kohdalla. Esimerkiksi aikataulussa pysymiseksi ominaisuuksia voidaan jättää pois, jolloin kuitenkin loppukäyttäjän tyytyväisyys laskee. Toisaalta se voi laskea yhtä paljon, jos ominaisuudet säilytetään, mutta lopputuote julkaistaan selvästi aiottua myöhemmin.

Lee (2014) käsittelee artikkelissaan tarkemmin itse ohjelmiston laatua, eli projektin lopputuotteen laatua. Vertailussa on neljä eri laatumallia ja iästään huolimatta keskeiseen osaan nousee McCallin laatumalli. (Lee, 2014.) Alkuperäisessä, vuoden 1977 tutkielmassa käydään läpi lukuisia eri ohjelmiston toiminnan piirteitä. Näistä tärkeimmät jaetaan kolmeen eri luokkaan: tuotteen toiminnallisuus, tuotteen tarkistus ja tuotteen siirtymä. Ohjelmiston käytöltä, eli toiminnallisuudelta vaaditaan oikeellisuutta, luotettavuutta, tehokkuutta, integroitavuutta ja käytettävyyttä. Tuotteen tarkistaminen edellyttää ylläpidettävyyttä, joustavuutta ja testattavuutta. Lopulta siirtymän vaatimuksia ovat liikuteltavuus, uudelleenkäytettävyys ja yhteentoimivuus. (McCall ym., 1977.) Käytännössä McCallin laatumalli ottaa huomioon käyttäjien käyttökokemuksen, tuotteen ylläpidettävyyden jatkossakin ja helpon käyttöönoton. Laatumalli auttaa kehittäjiä ymmärtämään vaatimuksia ohjelmiston laadun varmistamiseksi. Se ei kuitenkaan ota huomioon esimerkiksi organisaation strategisia tai liiketoiminnallisia tavoitteita tai itse projektikehitysprosessia ja sen vaiheita.

Ohjelmistokehityksen laatua voidaan siis tarkastella ja mitata useista eri näkökulmista. Esimerkiksi loppukäyttäjällä ja organisaation johdolla voi olla ihan eri ajatuksia ohjelmistokehityksen laadun kannalta. Jonkun osapuolen mielestä projekti voi olla onnistunut, kun toisen mielestä katastrofi. Aiempi tutkimus myös keskittyi laadun tutkimisessa erilaisiin näkökulmiin. Esimerkiksi DeLone-McLeanin malli pohjautui pitkälti käyttäjätyytyväisyyden mittaamiseen (Montesdioca & Maçada, 2015). Ghanbari (2008) taas painotti käyttäjätyytyväisyyden olevan vaikeasti mitattava mittari, johon asenteet ja suhtautuminen teknologiaa kohtaan vaikuttavat merkittävästi (Ghanbari, 2018).

Voidaan ajatella, että teknisen toteutuksen mittaamisella saadaan arvioitua myös asiakastyytyväisyyttä. Esimerkiksi McCallin laatumallin toiminnallisuus sisältää tehokkuuden ja käytettävyyden (McCall ym., 1977.). Molemmat ovat varmasti piirteitä, jotka lisäävät asiakastyytyväisyyttä. Käyttöliittymän testauksen ja arvioinnin ammattilaisen puolesta voidaan arvioida antavan kattavamman kuvan sen toimivuudesta kuin satunnaisen loppukäyttäjän. Ylipäätään McCallin laatumallin muutkin mallit voivat antaa tarkempaa tietoa projektin onnistumisesta. Asiakas voi olla tyytyväinen käyttöön, vaikka tuote on hankalasti integroitava ja päivitettävä ja sen elinkaari tulee olemaan lyhyt. McCallin laatumalli huomioi myös uudellenkäytettävyyden ja ylläpidon (McCall ym., 1977.).

Laadun mittaamiseksi tutkielmassa luodaan uusi viitekehys. Haastavan mitattavuuden vuoksi asiakastyytyväisyys jätetään arvioinnista pois. Sen sijaan tuotteen laatua katsotaan McCallin laatumallin kolmella mitattavalla kohdalla, eli tuotteen toiminnallisuus, tarkistus ja siirtymä. Lisäksi Azure DevOpsin mainostetaan vähentävän organisaation kustannuksia ja nopeuttamaan projektin toteutumista (Microsoft, 2023). Tämän myötä budjetti ja aikataulu ovat myös tärkeät mittarit laadun mittaamisessa. Tutkielmassa tutkitaan DevOps-pohjaisten pilvipalveluiden hyödyntämistä erityisesti sovelluskehittäjien näkökulmasta. Kehittäjät voivat arvioida projektin onnistumista tuotteen toiminnallisuuden, tarkistuksen ja siirtymän kautta. Vaikka aikataulun ja budjetin ylittyminen voi ohjelmistokehitystiimin sijaan vaikuttaa enemmän organisaation muihin yksiköihin, on niiden noudattaminen tärkeää myös kehittäjille. Kehittäjät voivat saada nuhteita ja painostusta johdolta, jos annetuissa kriteereissä ei pysytä.



KUVIO 3 Viitekehys ohjelmistokehityksen laadun mittaamiselle

3 PILVIPALVELUT

NIST:n (National Institute of Standards and Technology) artikkelissa Mell ja Grance (2011) määrittelevät pilvipalveluiden keskeisiä ominaisuuksia ja piirteitä tarkasti. Pilvipalvelut ovat tietokoneresursseja, kuten laskentateho, serverit, tallennustila ja ohjelmat, jotka voidaan saavuttaa mistä vain milloin vain internet-yhteyden kautta. Pilvipalvelut ovat jaettuja resursseja, eli niillä voi olla useita käyttäjiä samaan aikaan. (Mell & Grance, 2010.)

Pilvipalveluille määritellään viisi tärkeintä ominaispiirrettä: pyynnöstä itsenäinen palvelu (on-demand self-service), laaja saavutettavuus verkosta (broad network access), resurssien jako (resource pooling), nopea joustavuus (rapid elasticity) ja mitattavat palvelut (measured service). Pilvipalvelut voidaan siis ottaa käyttöön nopeasti ja itsenäisesti. Pilvipalveluntarjoajan sivujen kautta pystyy tyypillisesti valitsemaan tarpeisiin sopivan resurssit ja ottamaan ne nopeasti itse käyttöön ilman vastapuolen ihmisresurssien käyttöä. Pilvipalvelut voidaan myös saavuttaa eri standardeilla laitteilla ja verkoilla, kuten matkapuhelimella tai tietokoneella. Pilvipalveluntarjoajat jakavat resursseja asiakkaille tarpeen mukaan. Tietty asiakas saattaa esimerkiksi tarvita selvästi enemmän laskentatehoa kuin toinen. Loppukäyttäjä ei myöskään välttämättä tiedä, missä paikassa fyysiset tietokoneresurssit sijaitsevat. Nopea skaalautuvuus taas tarkoittaa sitä, että saatavilla olevien resurssien määrää voidaan muuttaa nopeasti esimerkiksi suuren laskentatehon tarpeen iskiessä. Lopuksi sekä asiakas että palveluntarjoaja monitoroivat ja tarkkailevat palvelun käyttöä optimoidakseen resurssien jakoa. (Mell & Grance, 2010.)

Pilvipalveluiden käyttö on kasvanut viime vuosina räjähdysmäisesti. Vuonna 2022 Suomen kaikista yrityksistä 81 prosenttiyksikköä käytti pilvipalveluita ja suurista yrityksistä taas 97 prosenttiyksikköä. Määrä on lähes tuplaantunut vuoteen 2014 verrattuna. Tyypillisimpiä pilvipalveluita olivat toimisto-ohjelmat, sähköposti, kirjanpito ja tiedostojen tallennus. (Tilastokeskus, 2022.) Euroopan alueella Suomi on ollut pitkään edelläkävijä pilvipalveluiden käytössä, mutta sai Ruotsin rinnalleen vuonna 2022 (Eurostat, 2023). Voidaan siis päätellä, että pilveen tallennettavan datan määrä tulee vain kasvamaan jatkossa ja myös pilvipalveluntarjoajia tulee markkinoille lisää.

Luvussa käydään läpi pilvipalveluiden ominaisuuksia. Ensimmäinen alaluku tarkastelee, miten pilvipalvelut voidaan luokitella niiden ominaisuuksien mukaan. Luokittelun voi määrittää esimerkiksi se, kuinka paljon käyttäjä pääsee hallitsemaan infrastruktuuria. Samoin luokitteluun vaikuttaa se, missä pilven infrastruktuuri fyysisesti sijaitsee. Toinen alaluku käsittelee Microsoftin Azure DevOpsia ja antaa siitä kattavan yleiskuvauksen.

3.1 Pilvipalveluiden luokittelu

Pilvipalvelut voidaan jakaa kolmeen palvelumalliin. Näitä ovat Ohjelmisto palveluna (Software as Service, SaaS), palvelualusta palveluna (Platform as a Service, PaaS) ja Infrastruktuuri palveluna (IaaS) (Mell & Grance, 2010). SaaS on pilvipalvelumalleista yleisin (Eurostat, 2023). SaaS tarkoittaa pilvessä sijaitsevia ohjelmistoja, joihin käyttäjä pääsee käsiksi verkon kautta. Pääsy mahdollistuu esimerkiksi internet-sivulla tai erillisessä ohjelmassa olevan käyttöliittymän kautta. SaaS on käyttäjille helpoin mutta myös rajoittunein pilvipalvelu. Palveluntarjoaja hallitsee palvelusta lähes kaikkea ja käyttäjälle jää tyypillisesti vain sovellus- tai ohjelmistokohtaiset asetukset tai käyttäjien hallinta. (Mell & Grance, 2010.) Esimerkkeinä tyypillisistä SaaS-kohteista voidaan nostaa toimisto-ohjelmat, kuten Microsoft 365 tai Salesforce.

Palvelualusta palveluna on palveluntarjoaman pilvessä ylläpitämä alusta ohjelmistokehitykseen. Palvelu tarjoaa valmiita työkaluja ohjelmistokehitykseen. Näitä ovat esimerkiksi versionhallinta, tuki eri ohjelmointikielille ja testaus. Koko pilvi-infrastruktuuri, kuten verkko, tallennustila ja käyttöjärjestelmät, on palveluntarjoajan vastuulla. Käyttäjä voi tyypillisesti hallita palvelualustan ohjelmia ja työkaluja ja konfiguroida ympäristöä. (Mell & Grance, 2010.) Tutkielmassa käsiteltävä Azure DevOps on Microsoftin tarjoama PaaS-pilvipalvelu. PaaS-mallille tyypillisesti se tarjoaa kattavan ympäristön ja suuren määrän työkaluja sovelluskehitykseen DevOps-mallin mukaisesti.

Infrastruktuuri palveluna on pilvipalvelumalleista laajin. Palveluntarjoaja tarjoaa asiakkaille infrastruktuurin, joka sisältää laskentatehon, tallennustilan, verkkolaitteet ja muut tarvittavat resurssit. Asiakas pystyy luomaan oman tietojärjestelmänsä pilvi-infrastruktuurin päälle ja hallitsemaan käyttöjärjestelmiä, tallennustilaa, käytettäviä ohjelmia ja joissain määrin konfiguroimaan verkon laitteita (Mell & Grance, 2010). IaaS vaatiikin asiakkailtaan pilvipalvelumalleista kaikista eniten tietoteknistä tuntemusta. IaaS-palveluita löytyy muun muassa Googlelta, IBM:ltä ja Amazonilta.

Pilvipalvelut voidaan luokitella myös käyttöönottomallien mukaan. Yksityinen pilvi on vain tietyn organisaation käyttöön tarkoitettu pilvi. Sitä voi ylläpitää organisaatio itse omissa tiloissaan tai se voi olla kolmannen osapuolen toimijan vastuulla. Yhteisöpilvi palvelee useampia organisaatioita, joilla on samanlaisia vaatimuksia esimerkiksi liiketoiminnan tai lainsäädännön suhteen. Samoin kuin yksityisen pilven kohdalla, sitä voi ylläpitää käyttäjäorganisaatio itse tai kolmas osapuoli. Julkinen pilvi on kaikille avoin pilvipalvelu. Palvelu voidaan

ottaa verkon kautta käyttöön ja sijaitsee fyysisesti palveluntarjoajan ylläpitämässä pilvessä. Lopuksi on vielä olemassa hybridipilvi. Hybridipilven kohdalla organisaatiolla on käytössään kaksi eri pilvi-infrastruktuuria, kuten yksityinen pilvi ja julkinen pilvi. Pääasiassa toimintoja voidaan pyörittää yksityisessä pilvessä, mutta julkista pilveä hyödynnetään esimerkiksi varmuuskopiointiin tai äkillisen laskentatehon tarpeeseen. (Mell & Grance, 2010.) Taulukko 1 antaa yhteenvedon eri pilvipalvelumallien hyödyistä ja haitoista.

TAULUKKO 1: Yhteenveto pilvipalvelumallien hyödyistä ja haitoista

Pilvipalvelu	Hyödyt	Haitat
Yksityinen On-Premise	- Muokattavuus - Data fyysisesti lähellä	- Ylläpito ja luominen työlästä - Fyysisen tilan tarve
IAAS (Infrastructure as Service)	- Laaja muokattavuus ja skaalautuvuus - Ei tarvitse omia palvelinkoneita	- Vaatii IT-asiantuntemusta - Voi olla kallis ratkaisu
PAAS (Platform as Service)	- Valmis sovellusalusta - Paljon hyötyjä ohjelmistokehittäjille	- Muokattavuus rajallisempi kuin IAAS:ssa - Mahdolliset rajoitukset resursseissa
SAAS (Software as Service)	- Valmiit ohjelmistot - Vaatii vain vähän teknistä osaamista käyttäjiltä	- Erittäin rajallinen muokattavuus
Hybridipilvi (Hybrid Cloud Infrastructure)	- Skaalautuvuus - Voi auttaa laskemaan infrastruktuurin kustannuksia	- Julkisen pilven toistuva käyttö nostaa kustannuksia

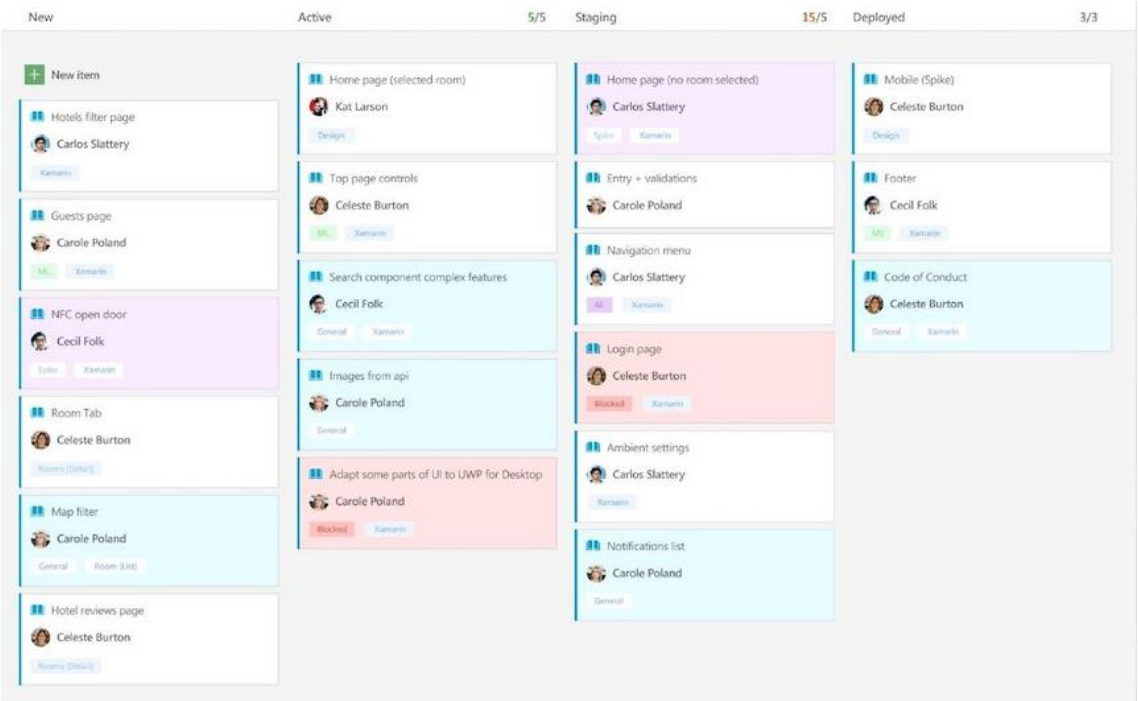
3.2 Microsoft Azure DevOps

Kuten tutkielmassa aiemmin todettiin, on Microsoftin Azure DevOps PaaS-pilvipalvelu. Azure DevOps nykyisessä muodossaan esiteltiin vuonna 2018 ja se mahdollistaa käyttäjille pääsyn pilvessä sijaitseviin ohjelmistokehitystyökaluihin joko selaimen tai integroidun kehitysympäristön kautta. Työkaluja ovat muun muassa Git-hakemistot versionhallintaan, työkalut ohjelmistojen jatkuvaan julkaisuun, Agile-työkalut työn suunnitteluun ja kartoittamiseen, testauksen työkalut, projektin mukaan mukautettava käyttöliittymä ja sisäänrakennettu Wiki tiedon jakamiseen tiimin keskuudessa. Azure DevOps tukee kattavasti eri lisäosia, mikä mahdollistaa palvelun kustomoinnin organisaation tarpeiden mukaan. (Rossberg, 2019.) Azure DevOpsia käyttävät niin pienet kuin suuretkin yritykset. PeerSpotin sivuston mukaan DevOpsia käytetään aina infrastruktuurin

suunnittelusta Agile-toimintatapojen toteuttamiseen sovelluskehityksessä ja jatkuvien toimituksien jakelun varmistamiseksi. Osa käyttäjistä ylistää Azure DevOpsin hyötyjä koko projektin elinkaaren hallinnassa, kun taas osa käyttää sitä vain suunnitteluvaiheessa, eikä itse ohjelmiston rakentamiseen. Monet konsultointiyritykset myyvät Azure DevOps-pilvipalvelua ja sen käyttöönoton opastusta muille yrityksille. (PeerSpot, 2023.)

Azure DevOpsin historia ulottuu kuitenkin kauemmaksi kuin vuoteen 2018. Vuonna 2004 Microsoft julkaisi Application Lifecycle Management (AML) -palvelun nimeltään Microsoft Visual Studio Team Systems ohjelmistojen elämänkaaren hallintaan (Arora, 2016). Sitä seurasi vuonna 2005 julkaistu Microsoft TFS, eli Team Foundation Server. TFS:n avulla sovelluskehitystiimit pystyivät kommunikoimaan paremmin ja hallitsemaan projektien kulkua yhdessä. Käyttöliittymä ei ollut käyttäjäystävällinen ja esimerkiksi nyt tärkeät testaamisen työkalut käytännössä puuttuivat kokonaan. Kovasti ohjelmistokehityksessä yleistyvä agile tuotiin TFS:n rinnalle vuonna 2012. Projektinhallintaan esitellyt uudet työkalut helpottivat Agile-prosessien noudattamista. Team Foundation Serveristä tuli lopulta Azure DevOps vuonna 2018. Muutos oli myös palvelun arkkitehtuurin kannalta merkittävä. Team Foundation Server keskittyi aiemmin toimintaan organisaatioiden omilla alustoilla ja tietokoneilla, mutta Azure DevOpsin myötä palvelu siirtyi pääasiassa kokonaan pilveen. (Gehman, 2018.)

Azure DevOpsissa on lukuisia keskeisiä ominaisuuksia käyttäjille. Azure Boards koostuu Kanban-tauluista, jotka voidaan luokitella projektin eri vaiheisiin (katso KUVIO 4). Eri työt (eng. work item), kuten bugit, uusi ominaisuus tai yksittäinen tehtävä, voidaan sijoittaa vaiheisiin, kuten kehitys tai julkaisu. Boards-välilehdeltä pystyy siis seuraamaan projektin etenemistä ja tiimin eri jäsenten tehtäviä. Projektien kaikki vaiheet ideasta ylläpitoon ovat näkyvillä ja lisäksi Boards tukee tiimin tapaamisien järjestämistä. Azure Pipelines on jatkuvan integroinnin ja toimituksen työkalu. Azure Pipelines on suunniteltu toimimaan kaikkien ohjelmointikielien, pilvien ja alustojen kanssa ja synkronoi Git-versionhallinnan kanssa. Kontit (eng. container) ja testaamisen työkalut tukevat sovelluskehittäjiä jatkuvassa toimituksessa. Azure Artifacts tarjoaa keskitetyn varastoinnin pakkauksille, kuten NuGet, npm ja Maven. Pakkauksien avulla voidaan hallita ja tallentaa ohjelmistoprojektien käyttämien riippuvuuksien ja komponenttien koodipaketteja. Artifacts myös mahdollistaa sekä julkisten että yksityisten pakkauksien sisällyttämisen projektiin. Azure Repos sisältää rajattoman määrän Git-hakemistoja pilvessä. Käyttäjät voivat hallita muutoksia koodiin vetopyynnöillä (eng. pull request) ja tiedostojen ja lähdekoodin hallinta on tehty helpoksi. Useampi henkilö voi työskennellä saman projektin kanssa samaan aikaan tai toinen henkilö voi jatkaa toisen työn loppuun. Lopuksi Azure Test Plans sisältää työkalut sekä automaattiseen että manuaaliseen testaamiseen. Testaamisella saadaan dataa ja tilastoja ohjelmiston toiminnasta ja vaatimuksien täyttämisestä. Näillä ohjelmistokehittäjät kykenevät parantamaan tuotettavan ohjelmiston laatua. (Rossberg, 2019.)



KUVIO 4 Azure Devopsin Boards-näkymä

Microsoftin Azure DevOps sisältää luonnollisesti useita DevOpsin käytänteitä. Jatkuvilla toimituksilla voidaan toimittaa useita pieniä päivityksiä ja muutoksia ohjelmistoon. Eri ympäristöt, kuten staging ja production, mahdollistavat muutoksien ja uusien ominaisuuksien testaamisen käytännössä ennen lopullista julkaisua. Jatkuvan integraation myötä myös kehitysympäristön koodi on aina oikeellista ja testeillä varmistettua. Testaamisen työkalut parantavat ohjelmiston laatua ja Azure DevOps tarjoaa kattavat versionhallinnan ominaisuudet ja säilytyspaikat koodille. Monitorointi ja ohjelmoinnin työkalut, kuten kontit, ovat nekin DevOpsin käytänteitä. Teknologian lisäksi Azure DevOps tukee DevOpsin kulttuuria. Projektit ovat erityisesti Kanban-taulujen vuoksi läpinäkyviä. Jokainen tiimin jäsen voi seurata, missä vaiheessa projekti on ja minkä parissa kukin työskentelee. Kommunikointi on tehty helpoksi ja tiimin jäsenet voivat kommentoida projektin eri töitä ja tuoda omia ideoitaan esille. Työkalut kannustavat ihmisiä opettelemaan uutta ja vetopyynnöt tarkastetaan aina jonkun toisen osapuolen toimesta, jolloin virheet voidaan käydä yhdessä läpi. Azure DevOps tukee myös ketteriä kehitysmenetelmiä, kuten Agile, jotka ovat keskeisessä osassa nykyajan ohjelmistokehitysorganisaatioita. (Microsoft, 2023.)

Tutkielmassa käytiin aiemmin läpi pilvipalvelun määritteitä ja ominaisuuksia. Microsoft markkinoi Azure DevOpsia PaaS-pilvipalveluna. PaaS-palvelulle tyypillisesti Azure DevOps tarjoaakin sovelluskehitysalustan kehittäjille ja laajan mukautettavuuden eri ohjelmistoille ja ohjelmointikielille. Se on laajennettavissa lähes mille tahansa ohjelmistokehitykseen liittyvälle käyttötarkoitukselle sisältää laajan kirjaston teknisiä ominaisuuksia ja työkaluja. Azure DevOps on helposti

ja itsepalveluna saavutettavissa. Kuka tahansa voi ottaa ilmaisen version välittömästi käyttöönsä Microsoftin internetsivuilta. Lisäksi sivustolta voi valita organisaatiolleen sopivan maksullisen paketin. Tilauksessa voi valita tarvittavien ominaisuuksien ja samanaikaisten käyttäjien määrän. Azure DevOps on saatavilla kellon ympäri selaimen kautta millä tahansa laitteella missä tahansa, jolloin korkean saatavuuden kriteerit täyttyvät. Organisaatio voi valita, kuinka paljon resursseja palvelulta tarvitsee ja palveluntarjoaja huolehtii tarvittavien resurssien jaon. Tilausta voi muokata ja palvelua laajentaa esimerkiksi lisäosilla, mikä lisää skaalautuvuutta. Lopuksi myös mitattavuuden piirre täyttyy, koska Azure DevOps antaa tilastoja käyttäjilleen ja palveluntarjoajalle. (Microsoft, 2023.)

4 DEVOPS-TOIMINTAMALLIN TOTEUTTAMINEN PILVESSÄ JA AZURE DEVOPSISSA

Agile on noussut keskeiseen osaan sovelluskehityksessä. Ketterät sovelluskehitysmenetelmät ovat korvaamassa tehokkaasti perinteisen vesiputousmallin, vaikka silläkin on vielä paikkansa. DevOps on myös noussut selvään suosioon sovelluskehittäjien keskuudessa. Se lupaa parantaa tiimin yhteistyötä ja läpinäkyvyyttä ja tehostaa projektin etenemistä aikataulun ja budjetin puitteissa. PaaS-pilvipalvelut kykenevät tarjoamaan DevOpsin työkalut ja toiminnot organisaatioille joustavasti ja kustomoitavasti. Tässä luvussa tarkastellaan, mitä PaaS-pilvipalvelulta vaaditaan DevOpsin toteuttamiseksi. Keskeisenä tutkimuksen kohteena on Azure DevOps, jota käytetään esimerkkinä DevOps-pilvipalvelusta. Azure DevOpsin ominaisuudet ja yhtäläisyydet DevOpsin kanssa käydään tarkkaan läpi. Lopuksi tarkastellaan, miten DevOps-pilvipalvelut auttavat organisaatioita projektin laadussa. Tässä käytetään myös esimerkkinä Azure DevOpsia.

4.1 DevOps PaaS-pilvipalveluna

DevOpsin keskeisiä työkaluja ja käytänteitä ovat automaatio, testaus, jatkuva integrointi ja toimitukset ja avoimuuden kulttuuri (Rossel, 2017). Automaatiolla tarkoitetaan ohjelmistokehityksen eri vaiheiden ja prosessien automatisointia manuaalisten prosessien vähentämiseksi (Mohammad, 2018). Pilvipalvelun on siis kyettävä toteuttamaan prosesseja automaattisesti. Esimerkiksi testauksen automatisointi on tärkeää. Testaamisella varmistetaan koodin kääntyminen ja oikeellisuus, eli toimiiko se tarvittavalla tavalla. Samoin kattava testaaminen lisää myös tietoturvaa. Manuaalisten testien tekeminen ja ajaminen taas vie paljon resursseja ja aikaa sovelluskehittäjiltä. (Mohammad, 2017.) PaaS-pilvipalvelut tukevat tyypillisesti lukuisia eri ohjelmointikieliä (Mell & Grance, 2010). Microsoft mainostaa myös Azure DevOps -palvelustaan, että se tukee mitä tahansa

ohjelmointikieliä (Microsoft, 2022). Tällöin testien on oltava hyvin monipuolisia ja konfiguroitavissa kaikkiin tarpeisiin. Ylipäätään mitä monimutkaisemmiksi ohjelmistot kasvavat, sitä haastavampaa kattava manuaalinen testaaminen on. Tekoäly (AI) ja koneoppiminen (ML) ovat nousseet esille ohjelmistokehitykseen liittyvän automaation kohdalla. Niiden avulla erilaisia testiskenaarioita voidaan luoda automaattisesti. Tekoäly huolehtii, että kaikki ohjelmiston osat käydään testeissä läpi. (Trudova ym., 2020.)

Automaatio sisältyy vahvasti myös jatkuvaan toimitukseen ja integraatioon, mikä korostaa sen merkitystä DevOps-pilvipalveluissa. Jotta julkaisuja voidaan toteuttaa nopealla aikataululla, on projektien oltava vahvasti automatisoituja. Automaattiset testit tukevat sitä, että ongelmat havaitaan nopeasti ja tiimiläisiltä ei mene aikaa manuaalisten testien suorittamiseen. Versionhallinta pitää automaattisesti yllä ohjelmiston historiaa ja päivityksiä ja Docker-kontit voidaan luoda, hakea ja avata automaattisesti prosessin nopeuttamiseksi. Lopulta koodin käyttöönotto pyritään tekemään automaation avulla mahdollisimman nopeaksi. Kun koodi on testattu automaation avulla oikeelliseksi ja kääntyväksi, voidaan se julkaista eri ympäristöihin helposti ja nopeasti. Projektista ja organisaatiosta riippuen valmiin koodin julkaisu tuotantoon voi vaatia vain yhden napin painalluksen ja kaikki muu on automatisoitua. (Rossel, 2017.)

Shahin ym. (2013). tutkivat artikkelissaan jatkuvan integroinnin ja jatkuvan toimituksen vaatimuksia ja haasteita. Jatkuvan integraation mahdollistamiseksi lähdekoodin historian ja muutoksien on oltava helposti saatavilla. Toiseksi ajettujen testien tuloksien pitää olla kaikkien saatavilla. Kaikkien koodiin liittyvien tapahtumien lisäksi on myös tärkeää esittää selkeästi tieto siitä, kuka työskentelee minkäkin työn parissa. Samalla on varmistettava, että eri komponentit ja jopa arkkitehtuurit toimivat yhdessä saumattomasti. Jatkuvan integraation myötä on myös tärkeää, että käytössä on useampi erillinen virtuaalinen ympäristö. Kriittinen bugi tai tietoturvahaavoittuvuus voi olla ohjelmiston toiminnan kannalta vaarallinen, mutta useamman eri ympäristön käyttö vähentää riskejä. Toisaalta kattavilla ja luotettavilla testeillä, jotka antavat palautetta ohjelmistokehittäjille, ehkäistään riskejä pidemmälle. Toistuvista julkaisuista tarvitaan dataa ja tilastoja niiden onnistumisen arvioimiseksi. Mitä tiheämpään tahtiin julkaisuja jaetaan, sitä kriittisempää tilastojen kerryttäminen on. Lopuksi työntekijöiden puutteellinen osaaminen voi olla haitallista jatkuvan integroinnin ja toimituksen kannalta. Esimerkiksi vesiputousmallin kaltaisten projektien parissa työskennelleille DevOpsin periaatteet voivat olla uusia ja vaativat merkittävästi opettelua. Monilla organisaatioilla jatkuvassa integraatiossa ja toimituksessa ongelmaksi on noussut kustannukset ja ympäristön ylläpitäminen. Ylipäätään tietokoneresursseille on korkeita vaatimuksia, koska koodin kääntäminen ja testaaminen vaatii laskentatehoa ja kaistaa merkittävästi. Palvelun on toimittava myös useiden eri ympäristöjen ja koodikirjastojen kanssa. Tämä nostaa ympäristön vaatimuksia entisestään. (Shahin ym., 2017).

DevOpsin onnistumisen edellytyksenä oli avoin kulttuuri ja tiedon jakaminen tiimiläisten keskuudessa (Humble & Molesky, 2011). Pilvipalvelun kohdalla tämän toteuttaminen on mahdollista useilla eri tavoilla. Projektia ja sen

etenemistä on kyettävä seuraamaan selkeästi. Jokaisen tiimin jäsenen on tiedettävä, missä vaiheessa projekti on. Samoin roolien on oltava helposti jaettavissa ja näkyvillä. Erilaiset keskustelupalstat ja kommenttikentät mahdollistavat huomioiden ilmaisun ja toisten auttamisen. Organisaation käytössä oleva sisäinen Wiki on myös yksi DevOpsin käytänteitä (Leite, 2020). Käytössä oleva pilvipalvelu on oiva paikka Wikin perustamiselle ja oman tietämyksen jakamiselle muun tiimin keskuudessa.

Yhteenvetona DevOps-pilvipalvelun ominaisuuksista voidaan sanoa, että se on tuettava automaatiota, testausta, jatkuvaa integrointia ja toimitusta ja avoimuuden kulttuuria. Samoin sen on kyettävä mukautumaan lukuisille eri käyttötarkoituksille ja ohjelmointikielille. Ohjelmistokehitysprojekteja on valtavasti erilaisia ja myös projektitiimien ja organisaatioiden koot voivat vaihdella. Jos mietitään DevOpsia pilvipalvelun ominaisuuksien kannalta, ovat ne poikkeuksestta PaaS-palveluita. PaaS-palvelut ovat juuri ohjelmistokehittäjille tarkoitettuja ja sisältävät valmiin, konfiguroitavan ympäristön ohjelmistokehitystä varten. Samoin kuin muiden pilvipalveluiden, on PaaS-pilvipalveluiden oltava saavutettavia, skaalautuvia ja itsepalveluina hankittavia. (Mell & Grance, 2010.) Organisaatioiden ja ohjelmistokehitystiimien erot tarpeissa ja koossa edellyttävät myös, että pilvipalvelun tilaus on mukautettavissa. Ei ole esimerkiksi järkevää maksaa 50 hengen paketista, jos palvelua käyttää vain 15 hengen tiimi.

4.2 Azure DevOps ja DevOpsin toteuttaminen pilvessä

Azure DevOps on Microsoftin mukaan PaaS-pilvipalvelu. Palvelulle tyypillisesti se tarjoaa sovelluskehitysympäristön sovelluskehittäjille. Microsoft ylläpitää itse infrastruktuuria tarjoten ympäristön ja työkalut. Käyttäjät voivat kuitenkin konfiguroida Azure DevOpsin tarpeidensa mukaan esimerkiksi tiettyjä projekteja ja ohjelmointikieliä varten. (Microsoft, 2022.) Myös kaikille pilvipalveluille tyypillisten piirteiden voidaan ajatella täyttyvän (Mell & Grance, 2010.) Azure DevOpsin voi tilata Microsoftin sivuilta itsenäisesti käyttöön. Sen voi konfiguroida tarpeiden mukaan, jolloin hinta lasketaan erikseen. Skaalautuvuus riittää suurillekin projekteille ja tiimeille ja palvelu on aina saavutettavissa internet-yhteyden kautta. Lopuksi Azure DevOps jakaa käyttäjilleen tilastoja ja dataa palvelun käytöstä ja tuloksista. (Microsoft, 2022.)

4.2.1 Automaatio

Azure DevOpsin portaalissa monet toiminnot on automatisoitu. Työt (work item) voidaan merkitä automaattisesti valmiiksi, kun tietyt, ennalta määrätyt kriteerit, ovat täyttyneet. Hakemistot (Repos) voidaan määritellä synkronoitumaan automaattisesti. Näin hakemisto on aina ajan tasalla ja tiimiläiset näkevät tehdyt muutokset ja koodin nykytilan. Pipelines-välilehden toimintojen automatisointi

on tärkeää sovelluskehityksen sujumisen kannalta. Julkaisuputket voidaan luoda ja määrittää automaattisesti YAML-tiedostojen avulla ja sama pätee myös ympäristöjen määrittelyyn. Julkaisujen automaatiota varten vaatimukset automaattiselle julkaisulle voidaan määritellä julkaisuputkessa ja kirjastot voidaan synkronoida automaattisesti hakemiston kanssa. Testaamisen automatisoinnille on kattava määrä toimintoja. Testit voidaan määrittää vaihe vaiheelta, jotta niitä voidaan ajaa jatkossa automaattisesti. Azure Test Plans varmistaa, että koodi toimii oikein ja täyttää vaatimukset. Azure DevOps kokoaa muun muassa dataa ja tilastoja siitä, miten testit ovat onnistuneet ja miten kattavia testit ylipäätään ovat toimivuuden varmistamiseksi. Lopuksi Azure Artifactsiin voidaan automaationa avulla tallentaa paketteja jokaisen onnistuneen koodin koonnin jälkeen. (Microsoft, 2023.)

Azure DevOpsin kohdalla voidaan puhua termistä ”Azure Automation”. Se sisältää julkaisun ja hallinnan, reagoinnin, orkestroinnin ja integroinnin automatisoinnin Azure DevOps-palvelussa. Azuren automaatio toimii lukuisten eri käyttöjärjestelmien, ohjelmointikielien tai lisäosien kanssa. Prosessien automatisoinnilla Microsoft korostaa tehokkuuden parantumista, virheiden vähentymistä ja aikataulujen nopeutumista. (Microsoft, 2023.) Eri automaation työkaluja käydään seuraavaksi tarkemmin läpi.

Pipelines as a Code mahdollistaa julkaisuputken määrittämisen YAML-muotoisessa koodissa. YAML-tiedostot ovat luettavissa olevia ja tekstipohjaisia tiedostoja ja ne mahdollistavat helposti hallittavan versionhallinnan ja julkaisuputken kehityksen, testauksen ja julkaisun sujuvan hallinnan. Azure DevOps sisältää automaatikan, joka osaa luoda automaattisesti YAML-määrittelyn uudelle julkaisuputkelle. (Azure DevOps Labs, 2023.) Azure Rest API on ohjelmallinen rajapinta, joka mahdollistaa säännöllisesti ajettavien toimenpiteiden automatisoinnin. Toimenpiteiden määrä on merkittävä ja esimerkkinä voidaan nostaa sertifikaattien, työjonojen, tilastojen ja yhteyksien automatisointi. Rest API:ssa tieto siirtyy http-protokollan avulla. (Microsoft, 2023.) Azure Command Line Interface (CLI) mahdollistaa DevOps-palveluiden hallinnan komentorivin kautta. CLI:tä voidaan hyödyntää automaatiossa esimerkiksi ajastettujen Bash-skriptien tai toistettavien komentojen avulla. Azure Runbooks tarjoaa myös skriptien automatisointia. Pelkkien Azure DevOpsin-toimintojen hallinnan sijaan Runbookeilla voidaan suorittaa monimutkaisempia komentoja palveluiden ylläpitämisen ja kehittämisen tueksi. Automatisointi, kuten ajastus, on myös Runbookeissa mukana. Service Hooks välittää tietoa tapahtumista. Palvelussa voidaan määrittää, mistä tapahtumista tietoa välitetään ja kenelle. Esimerkiksi koodin kääntymisen epäonnistumisesta voidaan lähettää tieto sovelluskehittäjille. Azure Functions tarjoaa monien eri palveluiden kanssa integroitavan pilven, jossa voidaan suorittaa eri toimintoja. Esimerkiksi koodi voidaan määrittää ajettavaksi tietyn tapahtuman sattuessa tai Azure Functions voidaan määritellä käsittelemään ja muokkaamaan dataa automaattisesti. Azure Logic App on myös pilvialusta, joka tukee tapahtumien suorittamista automaattisesti. Kun Service Hooks oli enemmän sovelluskehittäjille tarkoitettu, voi Logic App automatisoida esimerkiksi sähköpostiviestien lähettämisen tai tiedostojen siirron. Azure Resource Manager

(ARM) on työkalu Azuren hallintaan. Käytännössä se on ylläpidon työkalu, jolla koko infrastruktuuria voidaan hallita. ARM:n toiminnot ovat myös automatisoitavissa, eli esimerkiksi komponentteja voidaan luoda automaattisesti useita kerrallaan manuaalisen työn sijaan (Microsoft, 2023).

Vuppalapati ym. (2020) ylistivät artikkelissaan Azure DevOpsin koneoppimista ja tekoälyn hyödyntämistä ja mainitsivatkin Azure DevOpsin olevan sen ansiosta yksi nykypäivän ohjelmistokehitysorganisaatioiden tärkeimpiä viitekehyksiä (Vuppalapati ym., 2020). Nimitys MLops tulee termistä ”machine learning DevOps” eli koneoppimisen DevOps. Koneoppimisen käyttöönotto on pyritty tekemään helpoksi ja automatisoiduksi. Tekoälylle syötetään kouluttamista varten dataa ja määritellään algoritmit ja tekoäly kokoaa datasta tulokset, joita voidaan verrata tavoitteisiin. Tekoälymalli ja sen ympäristö kootaan Docker-kontiksi, joka sisältää metadatan, kuten koodikirjastot mallin toiminnan mahdollistamiseksi. Mallin toiminta varmistetaan vielä kokonaisuudessaan, kunnes se voidaan ottaa käyttöön. Tekoälyn toimintaa monitoroidaan jatkuvasti ja sitä voidaan kouluttaa uudelleen. Kun koneoppiminen ja tekoäly on toiminnassa, voidaan sitä hyödyntää projektin monissa eri vaiheissa. Näitä ovat esimerkiksi julkaisun nopeuttaminen, läpinäkyvyyden ja jäljitettävyyden parantaminen, testaaminen ja ennustaminen, datan analysointi ja versionhallinnan ylläpitäminen. (Guthrie, 2021.)

4.2.2 Jatkuva toimitus ja integrointi

Azure DevOps tukee jatkuvaa toimitusta ja integrointia useilla eri tavoilla. Automaatio ja testaaminen on keskeisessä roolissa nopeatahtisessa ohjelmistokehityksessä. Azure DevOpsin testit varmistamat, että julkaistava ja integroitava koodi on luotettavaa ja toimii tarvittavalla tavalla. Versionhallinta kattaa koko lähdekoodin historian ja tarvittaessa useampi henkilö pystyy työskentelemään sen kanssa samaan aikaan. Azure DevOps mainostaa tukevansa kaikkia eri ohjelmointikieliä ja myös lukuisia komponentteja ja lisäosia, jolla pyritään varmistamaan yhteensopivuusongelmien vähäisyys. Eri ympäristöt, kuten Staging, Development ja Production, mahdollistavat päivityksien testaamisen ilman muutoksia tuotannossa ja asiakkaan päässä. Lisäksi Azure DevOps kerää kaikista päivityksistä dataa, josta voidaan koota tilastoja tukemaan sovelluskehitystä. Kattavilla ohjeistuksilla, Wikillä ja avoimella tiimin välisellä kommunikaatiolla pyritään parantamaan tiimin osaamista ja saamaan aikaan positiivista suhtautumista jatkuvaan integraatioon ja toimitukseen. Lopuksi PaaS-palvelu Azure DevOps tarjoaa myös ratkaisun infrastruktuurin luomiseen ja kustannuksiin. Asiakas voi valita haluamansa resurssit ja palvelut ja maksaa ennalta määrätyn summan. (Microsoft, 2023.) Jatkuva integraatio ja toimitus eivät siis kasvata kustannuksia, eikä resurssien loppuminen nouse herkästi ongelmaksi.

4.2.3 Kulttuuri

DevOpsin kulttuurin keskeisimpiä teemoja ovat avoimuus ja tiimityöskentely (Humble & Molesky, 2011). Tiimin jäsenet jakavat avoimesti toisilleen onnistumisia ja ideoita ja osallistuvat koko projektin elinkaaren aikana eri vaiheisiin. Tähän Leite (2020) korosti, että organisaation on tarjottava kommunikaatiokanavat ja huomioitava erilaiset haasteet, kuten maantieteelliset sijainnit ja eri aikavyöhykkeet (2020). DevOpsissa ei ole siiloutunutta rakennetta, jossa kehityksen ja ylläpidon tiimit ovat toisistaan erillään. Azure DevOps pyrkii ylläpitämään avointa keskustelua palvelussaan.

Azure Boards -välilehden Kanban-taulut kuvaavat koko tiimille avoimesti ja visuaalisesti, mikä on kenenkin tehtävä, missä vaiheessa projekti on ja mitä seuraavana on tulossa. Tämä edistää avointa kommunikaatiota ja tehtävien jakamista. Taulun statuksen voi päivittää ja siihen voidaan lisätä esimerkiksi tehtävään liittyvää lisätietoa tai se voidaan määrittää uusien henkilöiden vastuulle. Work Itemit kuvaavat, kenellä tehtävä on työn alla, niitä voidaan kommentoida ja asettaa uusia henkilöitä vastuulle. Pull Requestien sisältö kattaa kommentoinnin ja arvioinnin. Esimerkiksi tietylle päivitykselle voidaan määrittää arvioija, joka käy läpi muutokset koodissa. Mahdolliset virheet tai ongelmat merkitään ja arvioija jättää palautteen työstä. Kun työn omistaja tai muu tiimiläinen on korjannut puutteet, merkitsee tämä ongelmat korjatuksi ja kuittaa kommentilla. DevOps-kulttuurissa katsotaan, että virheitä tapahtuu väkisin ja tärkeämpää on kokeilla omia ideoita rohkeasti. Azure DevOpsin kattava virheentarkastus mahdollistaa sen, etteivät virheet haittaa tuotannossa olevaa versiota. Manuaalisen, ihmisen kautta tapahtuvan tarkastamisen lisäksi myös automaattinen testaus tukee tätä periaatetta. Wiki taas tarjoaa tehokkaan kommunikointivälineen jakaa omaa tietoa ja osaamistaan koko tiimin kesken. Wikiin voidaan koota projekteihin tai palveluihin tarvittavaa tietoa tai ohjeita. Esimerkiksi uusien työntekijöiden on helpompi päästä alkuun, kun Wiki on valmiiksi selkeä ja kattava. Lopuksi Azure Devopsin retrospektiivit kokoavat sprinttien lopussa informaatiota siitä, miten sprintti meni, mikä onnistui ja mikä ei. Tavoitteena on jälleen avoin keskustelu tiimin kanssa ja periaate siitä, että virheitä sattuu ja niistä opitaan. (Microsoft, 2023.)

4.3 Azure DevOpsin hyödyt ja haasteet

Azure DevOpsin hyötyjä ja haasteita tutkiessa voidaan myös tukeutua tutkimuksiin DevOpsin ongelmista. Edellisissä kappaleissa Azure DevOpsin todettiin jo sisältävän kattavasti DevOpsin toimintatapoja. Shahin (2017) artikkeli tutki erityisesti jatkuvan integroinnin ja toimituksen haasteita käytännön olosuhteissa (Shahin ym., 2017). Monet haasteet ovat yleisesti koko organisaation tasolla, eivätkä ole Azure DevOpsissa sellaisenaan. Esimerkiksi projektin läpinäkyvyys koettiin joissain tilanteissa ongelmaksi, mutta Azure DevOps esittää selkeästi

koko tiimille, missä vaiheessa itse projekti on ja missä vaiheessa ja kenen vastuulla yksittäiset työt ovat. Samoin Shahin totesi ongelmaksi työkalujen ja teknologioiden puutteen jatkuvan integroinnin ja toimituksen toteuttamiseksi. Azure DevOps sisältää nämä työkalut, joten sen kohdalla työkalut nousevat haasteista hyödyksi. Puuttuva testausstrategia ja heikot testit vaikeuttavat selvästi jatkuvan integroinnin ja toteuttamisen luomista. Azure DevOps kuitenkin sisältää kattavat työkalut sekä automaattisten että manuaalisten testien toteuttamiselle. Eri ohjelmointikielet, komponentit ja jopa arkkitehtuurit voivat auttaa ohjelman koonnissa ongelmia. Azure DevOps kuitenkin tukee lukuisia määriä eri ohjelmointikieliä, arkkitehtuureja ja työkaluja. Sama pätee tietokantojen rakenteeseen, joka voi ilman Azure DevOpsia muodostua ongelmaksi. (Microsoft, 2023.) Voidaan siis todeta, että Azure DevOpsin kohdalla monet tutkitun artikkelin haasteet kääntyivätkin Azure DevOpsin hyödyiksi.

Avinash (2022) kokoaa artikkelissaan Azure DevOpsin hyötyjä tiivistetysti. Merkittävämmiksi hyödyiksi se nostaa automaation työkalut, saumattoman viitekehyksien integroinnin, koneoppimisen ja infrastruktuurin konfiguroinnin. Azure DevOps poistaa automaation avulla usein toistuvia ja yksinkertaisia prosesseja ja auttaa työntekijöitä keskittymään täysin arvon tuottamiseen asiakkaalle. Tehokkaat testit tukevat koodin integraatiota ja auttavat löytämään
ongelmia ja näistä tuotetut raportit kertovat kehittäjille ohjelmiston laadusta. Ylipäätään Azure DevOpsin ominaisuudet ja työkalut auttavat organisaatioita sekä julkaisemaan, ottamaan käyttöön ja monitoroimaan entistä korkealaatuisimpia ohjelmistoja. Myös Avinash mainitsee Azure DevOpsin olevan yksi nykypäivän ohjelmistokehittäjien tärkein viitekehys, joka vähentää projektien kustannuksia ja parantaa asiakastyytyväisyyttä. (Avinash, 2022.) Seremet ja Rakic (2021) taas nostivat esiin, kuinka DevOps ja Azure DevOps yhdessä vähentävät kitkaa ohjelmistojen julkaisun ja ongelmanratkaisuprosessien välillä. Kehitys- ja ylläpitotiimi työskentelevät saumattomasti yhdessä samassa Azure DevOps-pilviympäristössä. Azure DevOps voidaan siis nähdä myös keinona lisätä yhteistyötä ja avoimuutta organisaation sisällä.(Seremet & Rakic, 2021.)

Toki DevOpsin käyttöönotto ei ole täysin ongelmatonta. Azure DevOps on suuri kokonaisuus, jossa on valtava määrä työkaluja, ominaisuuksia ja konfigurointimahdollisuuksia. Sen sujuva käyttö vaatii siis kattavaa perehdyttämistä ja opettelua. Puutteellinen osaaminen voi aiheuttaa viivästyksiä projektissa, kustannuksien nousua ja jopa tietoturvariskejä. Samoin DevOps-toimintamallin ja myös Azure DevOpsin käyttöönotto voivat aiheuttaa muutosvastaisuutta organisaation sisällä. Organisaatioiden on huolehdittava avoimuudesta ja kommunikaatiosta työntekijöiden kohdalla ja varmistettava riittävä perehdytys turvalliseen Azure DevOpsin käyttöön. Azure DevOps aiheuttaa myös kustannuksia organisaatiolle. Mitä suurempi määrä käyttäjiä on, sitä kalliimpi palvelu on. Myös erilaiset ostettavat lisäpalvelut ja lisenssit niille maksavat lisää. Lisäksi vaikka Azure DevOps tukee kattavasti eri viitekehyksiä ja muita sovelluksia, ei pidä luottaa sokeasti, että se toimii saumattomasti kaikkien näiden kanssa. (Soni, 2017.) Aiemmin mainittiin myös, että tiukan lainsäädännön määrittelemät projektit voivat olla haastavia toteuttaa DevOpsin avulla, mihin pätee myös Azure DevOps.

Toimintakatkoksia tai tuotantoon päässeitä bugeja ei yksinkertaisesti voida sallia, kun taas Azure DevOpsilla julkaistaan useita pieniä päivityksiä jatkuvasti ja DevOpsin kulttuuriin kuuluu, että virheistä opitaan. (Ebert ym., 2016.) Nämä ovat ristiriidassa keskenään. Seremet ja Rakic olivat myös huolissaan tietoturvan toteuttamisesta Azure DevOpsissa. Laaja valikoima työkaluja ja ominaisuuksia ja pilviympäristön konfigurointi vaativat osaamista, jotta palvelusta saadaan luotua turvallinen. Lisäksi riskit ovat läsnä myös sovelluskehittäjien työssä. Perinteiset haavoittuvuusskannerit eivät ole kehittyneet samaa tahtia Agilen ja DevOpsin nopeuden kanssa. Jos joka viikko julkaistaan uusi päivitys, ei sen testaamiselle tietoturvan kannalta ole välttämättä aikaa. Kehittäjät voivat jopa pilkkoa tai piilottaa tietoisesti koodia, jossa on turvallisuuden kannalta ongelmia, mutta aikataulujen vuoksi päivitys on pakko julkaista. (Seremet & Rakic, 2021.) Seuraava taulukko (TAULUKKO 2) kokoaa Azure DevOpsin hyötyjä ja haasteita.

TAULUKKO 2: Azure DevOpsin hyödyt ja haitat

Azure DevOpsin hyödyt	**Azure DevOpsin haasteet**
Kattava valikoima työkaluja ja ominaisuuksia.	Turvallinen ja tehokas käyttö vaatii perehtymistä.
Tukee eri ohjelmointikieliä, arkkitehtuureja ja lisäosia.	Voi aiheuttaa muutosvastaisuutta.
Auttaa toteuttamaan DevOpsia käytännössä.	Palvelun käyttö on maksullista.
Koneoppiminen ja tekoäly automaation tukena.	Ei sovi kaikkiin projekteihin ja sovelluskehitysmenetelmiin.
Testauksen työkalut parantavat ohjelmiston laatua.	Nopea julkaisutahti saa helposti oikaisemaan tietoturvan kohdalla.
Tukee yhteistyötä, avoimuutta ja kommunikaatiota organisaatiossa.	

4.4 DevOps-pilvipalveluiden vaikutus projektin laatuun

Projektin laadun mittareiksi valittiin aikataulu, budjetti, tuotteen toiminnallisuus, tuotteen tarkistus ja tuotteen siirtymä. Aikataulu määrittää projektin valmistumista sille annettujen aikarajojen puitteissa ja budjetti taas lopullisten kulujen määrän verrattuna ennalta arvioituun ja määrättyyn projektiin käytettävään rahasummaan. Toiminnallisuus sisältää ohjelmiston käytön, luotettavuuden, tehokkuuden ja oikeellisuuden, tarkistaminen taas ylläpidettävyyden, joustavuuden ja testattavuuden ja siirtymä kattaa liikuteltavuuden, uudelleenkäytettävyyden ja yhteentoimivuuden.

Oikein toteutettu DevOps-pilvipalvelu vähentää toistuvia ja yksinkertaisia prosesseja automaation avulla. Esimerkiksi testaus voidaan automatisoida. Lisäksi avoimuus, ennustaminen ja palvelussa tehtävät suunnitelmat varmistavat, että projekteissa kuluu vähemmän aikaa yllättäviin työtehtäviin ja korjauksiin. Eri ympäristöt ja kattava versionhallinta taas varmistavat, että jos virheitä tapahtuu, toivutaan niistä nopeasti. Ylipäätään jatkuvan integroinnin ja jatkuvan

toimituksen optimointi ja automaatio nopeuttavat projektin toteuttamista. (Khan, M. ym., 2020.) Projektin ennustaminen auttaa myös budjetissa. Kun tehtäviä jaetaan tiimiläisille, nähdään, kuinka paljon ihmisresursseja niiden suorittaminen vaatii. Esimerkiksi Azure DevOpsin tukeman ketterän kehityksen myötä organisaatio voi reagoida helposti muutoksiin ilman töiden seisastumista. Lisäksi valmis valikoima kattavia työkaluja ja ominaisuuksia vähentää tarvetta hankkia uusia työkaluja projektin aikana. Azure DevOps voidaan myös integroida esimerkiksi kirjanpito- tai taloushallintajärjestelmien kanssa, mikä tukee kustannuksien seurantaa. (Microsoft, 2023). Ylipäätään DevOps pilvipalvelun kautta voidaan nähdä sekä aikataulun että budjetin kannalta positiivisena tekijänä. Kun projekti valmistuu aikataulun puitteissa, pienenee myös riski budjetin ylittymiselle. Toki aikataulun kohdalla on huomioitava se, että tiimin pitää osata käyttää palvelua. Tämä ei ole ongelma, jos pilvipalvelu on ollut jo aiemmin organisaation käytössä. Jos taas pilvipalvelu otetaan nimenomaan uutta projektia varten käyttöön, voi palvelun käyttöönotto ja opettelu viedä suuren osan aikataulusta. Samoin budjetissa pitää huomioida käytettävä pilvipalvelu ja sen mahdolliset lisäosat. Esimerkiksi Azure DevOps ei ole täysin ilmainen palvelu ja monet sen ominaisuudet ovat lisämaksullisia.

Tehokas testaaminen varmistaa, että palvelu toimii odotetulla tavalla. Testaaminen kattaa esimerkiksi haavoittuvuuksien havaitsemisen tai toiminnan kuormituksen alla. Azure DevOps tarkastaa ohjelmointikielen ja kannustaa kehittäjiä helposti ymmärrettävän ja muokattavan ohjelmointikielen luomiseen. Hyvin kirjoitettu koodi helpottaa myös testaamista. Lisäksi eri ympäristöt auttavat testaamaan palvelua käytännössä ilman muutoksia tuotantoversioon. Näin myös käytettävyyden ongelmat ihmisen näkökulmasta paljastuvat helposti. Samalla muutoksia voidaan ottaa vaiheittain esimerkiksi ensin development-ympäristössä, sitten staging-ympäristössä ja lopulta tuotannossa. Tällä saadaan joustavampi käyttöönotto aikaan ja voidaan varmistaa muutoksien yhteentoimivuus. Asiakas hyötyy myös jatkuvasta integroinnista ja julkaisusta. Pieniä päivityksiä saadaan useasti ja myös mahdollisia muutoksia voidaan tehdä aikaisessakin vaiheessa. Monitoroinnin avulla taas ylläpitäjät voivat tarkastella palvelun toimintaa ja huomata ja korjata virheitä ennen kuin loppukäyttäjä kohtaa niitä. Monitorointi parantaa ylläpidettävyyttä. Khan ym. (2020) tutkivat DevOpsin ja Azure DevOpsin hyötyjä ohjelmistokehitysprojekteissa ja selvittivät, mitkä tekijät erityisesti auttavat tiimejä suoriutumaan projekteista onnistuneesti. Kehittäjien näkökulmasta automaattinen testaus nähtiin Azure DevOpsin merkittävämmäksi ominaisuudeksi. Samoin tiimin valinta ja roolinjako, tehokas versionhallinta ja selkeä infrastruktuuri sovelluskehitykseen nähtiin tärkeiksi. Ylipäätään sekä DevOps-toimintamalliin ja Azure DevOpsiin suhtauduttiin positiivisesti ja niiden koettiin auttamaan ohjelmiston nopeassa jakelussa, ongelmanratkaisussa ja virheiden löytämisessä. (Khan ym., 2020.)

Pilvipalvelun edut projektin aikataulun ja budjetin toteutumisessa vaikuttavat selviltä. Myös McCallin laatumallin kolme mallia täyttyvät hyvin toteutetun DevOps-palvelun myötä. Ohjelmiston oikeellisuus täyttyy helpommin, kun työkalut ja toiminnot tukevat ohjelmistokehitystä ja automaattinen testaus löytää

virheitä. Hyvin toteutettu ohjelmisto on myös helppo ylläpitää ja testata. Samoin sitä voidaan käyttää uudelleen tarpeiden muuttuessa ja se toimii yhdessä muiden järjestelmien kanssa.

5 TEORIAOSUUDEN YHTEENVETO

Sovelluskehityksen maailma on muuttunut vuosien myötä vahvasti ketterämpään suuntaan. Esimerkiksi vesiputousmalli nähdään nykyään kankeana ja raskaana työkaluna sovelluskehitykseen. Agile ja DevOps taas edustavat ketterämpää ja kevyempää tapaa sovelluskehitykseen. On yleistä, että asiakas osallistuu kehitykseen jo projektin alkuvaiheessa ja muutoksien tekeminen on helppoa. DevOps taas kokoaa sovelluskehittäjille kattavasti työkaluja ja käytänteitä, jotta ohjelmistojen kehittäminen olisi tehokasta ja päivityksiä voidaan jaella nopeasti. DevOpsin tärkeimpiä käytänteitä ovat automaatio ja jatkuva integrointi ja toimitus. Automaatiolla pyritään vähentämään turhia manuaalisia työn vaiheita ja parantamaan ohjelmiston laatua. Jatkuva integraatio taas varmistaa koodin oikeellisuuden ja kääntyvyyden ja jatkuva toimitus ja julkaisu ylläpitävät nopeaa julkaisu- ja päivitystahtia. (Rossel, 2017.) Asiakkaan näkökulmasta tämä näkyy tiheässä päivitystahdissa, laadukkaassa ohjelmistossa ja osallistamisessa. Kehittäjien näkökulmasta taas DevOps luo avoimen kommunikaation kulttuuria, selkeämpää suunnittelua ja läpinäkyvää työnjakoa ja seurantaa. Monia prosesseja voidaan korvata automaatiolla ja versionhallinta on aina ajan tasalla näyttäen kaikki tehdyt muutokset. Organisaation johto taas hyötyy budjetin ja aikataulun puitteissa pysymisestä ja paremmasta asiakastyytyväisyydestä. Toki DevOps on myös iso muutos koko organisaation kulttuurissa. Sen käyttöönotto voi vaatia isoja muutoksia organisaation toimintatavoissa ja työkaluissa ja riski muutosvastaisuudelle on olemassa. (Khan, M. ym., 2020.)

Myös pilvipalveluiden yleistyminen on muuttanut sovelluskehittäjien työtapoja. Nykyään organisaatioiden ei välttämättä enää kannata perustaa omaa yksityistä pilveä (Mell & Grance, 2010). Pilvipalveluntarjoajien valikoima on valtava ja pilvipalvelut palvelevat useita eri käyttötarkoituksia. Sovelluskehitystä voidaan hoitaa kokonaan pilvessä pyörivän palvelun turvin. Tästä on esimerkkinä juuri Azure DevOps, joka on pilvessä toimiva PaaS-palvelu. Se tarjoaa sovelluskehittäjille kattavan määrän työkaluja ja sitä voidaan hyödyntää aina projektin suunnittelusta julkaistun ohjelmiston monitorointiin ja ylläpitämiseen (PeerSpot, 2023).

Kuten markkinoilla saatavilla olevat palvelut osoittavat, DevOpsia voidaan toteuttaa tehokkaasti PaaS-pilvipalveluiden avulla. Teoriassa organisaatiot voisivat toteuttaa DevOpsia myös niin sanotuilla on-premise-palveluilla, joissa tarvittavat ohjelmistot asennetaan fyysisesti tietokoneille. Käytännössä tämä voi olla kankea ja hidas prosessi. Lisäksi mikään ei takaa, että eri ohjelmat toimivat saumattomasti yhdessä. PaaS-pohjaiset DevOps-pilvipalvelut taas sisältävät kaiken valmiina pakettina.

Ensimmäinen tutkimuskysymys oli seuraava: ”Miten DevOps-toimintamallin käytänteitä voidaan toteuttaa PaaS-palveluna?” ja apukysymyksenä ”Miten Azure DevOps toteuttaa niitä?”. Tutkielmassa ensinnäkin todettiin, että DevOps ei itsessään ole teknologia, vaan toimintamalli, joka sisältää lukuisia käytänteitä ja työkaluja organisaatioille. DevOps-pilvipalvelu voidaan ajatella teknologiana, joka toteuttaa näitä käytänteitä ja työkaluja käytännössä. Tärkeimmiksi käytänteiksi nousivat automaatio, jatkuva integraatio ja jatkuvat toimitukset (Rossberg, 2019). Jos pilvipalvelu ei toteuta näitä käytänteitä, ei myöskään DevOpsin toteuttaminen onnistu palvelulta. Automaatio on vahva pilari jatkuvan integraation ja toimituksen toteuttamiselle ja koneoppiminen ja tekoäly ovat nousseet sen rinnalle. Jatkuva integraatio ei voi toteutua, jos koodi on virheellistä, eikä käänny. Automaattiset testit auttavat tässä. Samoin versionhallinnan ylläpitäminen käsin on työlästä. Jatkuva toimitus taas on nopeaa, kun päivitysten oikeellisuudesta on varmistuttu ja julkaisuprosessi on pitkälle automatisoitua. (Rossberg, 2019.)

Tutkielmassa huomattiin myös, että Azure DevOps paikkaa joitakin DevOpsin haasteita. Esimerkiksi heikko läpinäkyvyys ja puutteellinen testausstrategia koettiin organisaatiossa usein DevOpsin toteuttamisen haasteeksi. (Shahin ym., 2017). Azure DevOpsissa samaa ongelmaa ei kuitenkaan ole. Projektin eteneminen ja työnjako on kaikkien nähtävillä ja testauksen työkalujen valikoima on kattava. Myös avoimuuden kulttuuri näkyy vahvasti esimerkiksi edellä mainitussa läpinäkyvässä työnjaossa ja projektin etenemisen seurannassa. Ylipäätään Azure DevOpsin havaittiin sisältävän kaikki DevOpsille oleelliset käytänteet ja työkalut. Esimerkiksi automaation teknologioita on palvelussa valtavia määriä ja monet prosessit voidaan automatisoida. Tekoäly tuo uusia mahdollisuuksia ja Microsoft myös parantaa ja kehittää palvelua tasaiseen tahtiin (Microsoft, 2022).

Toinen tutkimuskysymys taas oli seuraava: ”Miten DevOpsiin pohjautuvat pilvipalvelut auttavat ohjelmistoyrityksiä projektien laadussa?”. Laadun mittaamiselle luotiin oma viitekehys viidellä eri tekijällä käyttäen pohjana muun muassa McCallin laatumallia. Kysymyksen kohdalla haasteena oli vähäinen aiempi tutkimus DevOps-pilvipalveluista, kuten esimerkiksi Azure DevOpsista. Suuntaa saatiin kuitenkin itse DevOpsiin liittyvää tutkimusta tarkastelemalla ja sen hyötyjen ja haittojen vertailulla. Samoin Azure DevOps ja sen vaikutukset olivat tarkastelussa. Azure DevOps voitiin katsoa hyödylliseksi työkaluksi kaikistä McCallin laatumallin näkökulmista. Tehokkuuden parantuminen näkyy positiivisesti sekä aikataulussa että budjetissa ja kun lopputuote on laadukas, ovat sekä sen käyttöönotto, käyttö ja ylläpito onnistuneita. Automaatio ja kattava testaus

paljastavat ongelmia ohjelmistossa ja hyvin toteutettu palvelu on helppo ottaa käyttöön, integroida ja ylläpitää. Laatua mitatessa kuitenkin ongelmaksi nousi aiemman tutkimuksen vähäisyys DevOps-pilvipalveluihin, kuten Azure DevOpsiin, liittyen. Pelkkään DevOps-kirjallisuuteen ei voida nojautua loputtomiin. Azure DevOpsia käyttävien organisaatioiden haastattelulla saadaan lisää näkökulmia Azure DevOpsin vaikutuksista projektin laadulle. Sen sijaan ensimmäiseen tutkimuskysymykseen ja apukysymykseen saatiin jo kirjallisuuden ja Microsoftin sivuston tuella kattavasti vastauksia (Katso TAULUKKO 3). DevOps-toimintamalli on tarkkaan esitelty ja kuvattu aiemmassa tutkimuksessa. Toimintatapoja voitiin verrata itse Azure DevOps-palveluun ja Microsoftin sivuston ohjeisiin toimintojen käytöstä. Toki empiirisellä tutkimuksella saadaan myös tähän tutkimuskysymykseen lisää vastauksia. Esimerkiksi organisaation työntekijöiden työpäivää seuraamalla nähdään, miten avoimen kommunikaation kulttuuria toteutetaan Azure DevOpsin avulla.

TAULUKKO 3: DevOps ja Azure DevOps

DevOps käytänne	**Miten Azure DevOps toteuttaa?**
Avoimen kommunikaation kulttuuri	Läpinäkyvä roolinjako ja projektin seuranta, avoin kommunikaatio, kommentointi- ja keskustelumahdollisuus
Automaatio	Lukuisat automaation työkalut, automaattinen testaus, tekoäly ja koneoppiminen
Jatkuva integraatio	Automaattinen versionhallinta, koodin oikeellisuuden ja kääntyvyyden tarkistus, eri tuotantoympäristöt
Jatkuvat toimitukset	Jatkuvan integraation varmistaminen, automaatio julkaisun tukena, monitorointi ja seuranta

6 HAASTATTELUTUTKIMUKSEN TOTEUTUS

Oikean tutkimusmenetelmän valinta on tutkimuksen kannalta tärkeää. Tässä tutkielmassa käytetään teemahaastattelua laadullisen tutkimuksen toteuttamiseksi. Vaikka sopiva tutkimusmenetelmä on jo valittu, on tärkeä kiinnittää huomiota muihinkin tekijöihin, kuten haastateltavien valintaan, itse haastatteluun ja litterointiin. Seuraava luku perustelee, miksi kyseinen tutkimusmenetelmä on valittu ja miten kysymykset toteutetaan. Lisäksi tuodaan ilmi, miten varmistetaan mahdollisimman korkealaatuinen ja puolueeton tutkimus.

6.1 Aineistonkeruun menetelmät ja haasteet

Sovelluskehitys muuttaa muotoaan jatkuvasti. Organisaatiot joutuvat mukautumaan jatkuvaan muutokseen ja myös ohjelmistokehittäjät kohtaavat uusia ohjelmistoja ja toimintamalleja. Perinteinen ja siiloutunut vesiputousmalli hitaine julkaisuineen on jäämässä taka-alalle. Ketterä kehitys ja jatkuvat toimitukset ja julkaisu ovat nousseet merkittävässä osassa korvaajaksi. Vaikutukset organisaatioihin ja sovelluskehittäjiin voivat kuitenkin olla erilaisia. Esimerkiksi muutosvastaisuuden määrä riippuu merkittävästi yksilöstä. (Khan, M. ym., 2020.) Jotta aiheesta saadaan lisää tietoa, on tärkeä haastatella erilaisia organisaatioita ja yksilöitä.

Laadullinen tutkimus on tehokas työkalu keräämään uutta tietoa tutkittavasta aiheesta ja joko luomaan uutta teoriaa tai määrittelemään aiempaa teoriaa uudelleen. Laadullisella haastattelulla päästään lähelle haastateltavia ja heidän jokapäiväistä elämäänsä. Haastattelu pyrkii keräämään haastateltavien ajatuksia, tunteita ja näkemyksiä. Kysymykset voivat olla erilaisista teemoista ja haastateltavat eri taustoista. Näillä toimenpiteillä voidaan luoda erilaisia näkökulmia tutkimukseen ja tuomaan ilmi ilmiöiden suhteita ja riippuvuuksia keskenään. (Shah & Corley, 2006.)

DevOpsin kaltaisen käsitteen tarkastelu edellyttää ymmärrystä siitä, miten ihmiset sen kokevat ja miten se vaikuttaa heidän työskentelyynsä. DevOpsiin liittyy usein omakohtaisia käsityksiä. Laadullinen tutkimus tarjoaa mahdollisuuden saada syvempää tietoa osallistujien näkemyksistä ja ajatuksista, koska se perustuu avoimiin ja joustaviin haastatteluihin. Laadullisen haastattelututkimuksen avulla voidaan kysyä avoimia kysymyksiä, jolloin vastaajat voivat vapaasti ilmaista mielipiteensä, ajatuksensa ja kokemuksensa DevOpsista. Samoin laadullinen tutkimus ottaa huomioon haastateltavien ihmisten erilaisen taustan. Tämän myötä laadullinen tutkimus on tämän tutkielman kohdalla tehokas tutkimusväline. Se mahdollistaa avoimet huomiot haastattelun eri aiheista. Haastateltavat ovat myös taustoiltaan erilaisia ja joillakin saattaa olla selvästi enemmän DevOps-kokemusta kuin muilla. Avoin haastattelu tuo tilaa lisäkokemusten ja huomioiden kertomiseen.

Haastatteluilla toteutettava laadullinen tutkimus ei ole täysin ongelmatonta. Haastattelu ei pääse yhtä lähelle tutkittavia kuin kenttätutkimus. Kenttätutkimuksessa voidaan tarkkailla tutkittavien käyttäytymistä ja havaita ilmiöitä reaaliajassa. Goldkuhl (2019) mainitsi myös, että laadullisessa tutkimuksessa ja haastatteluissa tutkimuksen tekijän omat tulkinnat voivat muuttaa tuloksien suuntaa. On mahdotonta vetää tarkkaa rajaa sille, missä määrin oma tulkinta on sallittua. Kaikki omat tulkinnat on kuitenkin tärkeää kirjata ylös ja pyrkiä havainnoimaan puolueettomasta näkökulmasta (Goldkuhl, 2019.)

Shah ja Corley (2006) käsittelivät tutkielmassaan laadullisen tutkimuksen haasteita ja vaatimuksia. Laadullisessa tutkimuksessa syntyvät ideat voivat olla epäjärjestelmällisiä. Tämä voi johtaa haasteisiin raporttia kirjoittaessa. Ilmiö on myös aiemmin aiheuttanut negatiivista suhtautumista laadullista tutkimusta kohtaan, mutta nykyään hyväksyttävämpää. Tutkielmassa nostettiin esille erilaisia piirteitä, joita laadullisen tutkimuksen on sisällytettävä oikeellisuuden varmistettavaksi. Tutkimuksen on oltava sisäisesti oikeellinen, eli haastateltavat on valittava oikein. Ulkoisella oikeellisuudella taas tarkoitetaan yhdistettävyyttä muihin ilmiöihin ja teorioihin. Luotettavuudella tarkoitetaan huolellisuutta ja luottamuksellisuutta tutkimuksessa ja objektiivisuudella puolueettomuutta ja tarkkojen huomioiden tekemistä. (Shah & Corley, 2006.)

Hirsjärvi ja Hurme (2022) esittelivät termejä reliaabelius ja validius laadullisen tutkimuksen ohella. Käytännössä näillä tarkoitetaan sitä, että tutkija pääsee kiinni objektiiviseen totuuteen ja todellisuuteen. Tutkimus on reliaabelia, jos samaa henkilöä kahdesti tutkittaessa saadaan kahdesti sama tulos tai jos kaksi arvioitsijaa päätyy samanlaiseen tulokseen. Samoin tutkimus on reliaabelia, jos kahdella rinnakkaisella tutkimusmenetelmällä saadaan sama tulos. Käytännössä reliaabeliuksen toteuttaminen täydellisesti ei ole pakollista. Olosuhteet tai haastateltavan suhtautuminen tutkittavaan asiaan voivat muuttua. Tällöin esimerkiksi eri tulos kahdelta eri haastattelukerralta ei ole tutkimusmenetelmän vika. Validiuksella taas tarkoitetaan sitä, että tutkimus koskee juuri sitä asiaa, mitä sen ajatellaan koskevan. Käytännössä tutkimusmenetelmän kuuluu mitata juuri sitä, mitä sen pitääkin mitata. (Hirsjärvi & Hurme, 2022.)

6.2 Teemahaastattelu

Teemahaastattelu puolistrukturoidusti toteutettuna on yleinen haastattelumuoto laadullisessa tutkimuksessa. Nimensä mukaisesti teemahaastattelu koostuu useista aiheeseen liittyvistä teemoista. Teemat voivat syntyä esimerkiksi haastattelua edeltäneen kirjallisuuskatsauksen pohjalta. Teemahaastattelu sisältää valmiita teemakohtaisia kysymyksiä. Haastattelussa annetaan kuitenkin tilaa haastateltavan omalle pohdinnalle ja kysymykset ovat tyypillisesti väljiä. Haastattelija voi ohjailla haastattelua eri suuntaan esimerkiksi uusien näkökulmien ja havaintojen ilmetessä. Haastatteluun valitaan henkilöitä, joilla on jo valmiiksi taustaa tutkittavan aiheen ilmiöistä. Tällä varmistetaan, että haastattelut ovat tehokkaita ja tuottavat mahdollisimman paljon uutta tietoa ilmiöstä. (Palonen & Kylmä, 2022)

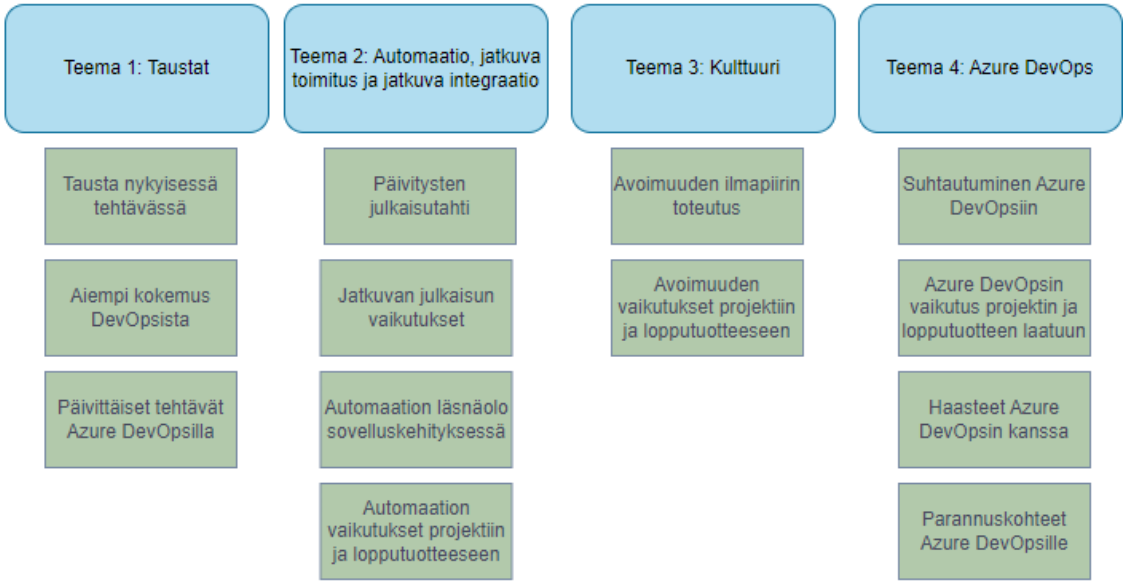
Teemahaastattelu sekoitetaan toisinaan avoimeen haastatteluun. Avoin haastattelu on väljä ja on lähempänä ohjattua keskustelua kuin haastattelua. Selvää kysymysrakennetta ei välttämättä ole. Palonen ja Kylmä (2022) toteavat, että avoin haastattelu sopii parhaiten ilmiöihin ja aiheisiin, joista tiedetään vasta hyvin vähän. Sen avulla pyritään keräämään mahdollisimman syvällistä tietoa ja ymmärrystä osallistujien kokemuksista ja asenteista. Avoimessa haastattelussa pyritään ennemmin ohjaamaan tai rohkaisemaan haastateltavaa kertomaan ilmiöistä, kuin kysymään suoraan. Strukturoitu haastattelu on taas rakenteeltaan selvästi avointa haastattelua ja teemahaastattelua tiukempi. Se sisältää tietyt kysymykset, jotka kysytään kaikilta osallistujilta samalla tavalla ja samassa järjestyksessä. Tilaa omalle pohdinnalle tai haastattelun myötä syntyville kysymyksille ei ole. (Palonen & Kylmä, 2022.)

Teemahaastattelu katsottiin tutkielmaan parhaiten sopivaksi. Tutkielmassa tutkittiin aiheita, joista kaikista ei ole vielä merkittävästi tietoa. Esimerkiksi Azure DevOpsiin liittyvä tutkimus on vähäistä. Lisäksi teemahaastattelu toimii hyvin avoimen haastattelun kanssa. Haastateltava voi tuoda ilmi omia huomioita ja havaintoja, jotka voidaan yhdistää juuri sillä hetkellä käsiteltävään teemaan. Esimerkiksi strukturoidulla haastattelulla havainnot voisivat helposti jäädä tulematta. Tutkielman kirjallisuuskatsauksessa syntyi myös useita teemoja, jotka loivat pohjan teemahaastattelulle. Aiheet oli helppo jakaa neljään eri osa-alueeseen. Näillä haastattelut saadaan käynnistettyä tehokkaasti ja edettyä loogisesti vaiheesta toiseen. Haastatteluiden pohjarakenne pysyy kutakuinkin samana, mikä helpottaa tuloksien läpikäyntiä.

6.3 Haastatteluiden teemat

Kirjallisuuskatsauksen pohjalta luotiin neljä eri teemaa haastattelurunkoa varten (katso KUVIO 5). Teemat on luotu tutkittavien ilmiöiden pohjalta ja tarjoavat vastauksia tutkimuskysymyksiin. Keskeisessä osassa ovat DevOps-

toimintamalli, jatkuvat toimitukset ja integraatio ja Azure DevOps. Myös haastateltavien taustat pyrittiin selvittämään ensimmäiseksi.



Kuvio 5 Haastattelurungon teemat

Ensimmäinen haastatteluteema pyrkii selvittämään haastattelijoiden taustat. Taustoituksilla kartoitetaan, kuinka pitkään haastateltava on toiminut nykyisessä tehtävässään Azure DevOpsin parissa. Samoin tuodaan ilmi, mikäli haastateltavalla on aiempaa kokemusta Azure DevOpsista tai DevOps-toimintamallin alaisuudessa toimimisesta. Koska haastateltavilla on erilaisia taustoja ja työtehtäviä, saadaan myös Azure DevOpsissa toteutettavien päivittäisten tehtävien kyselyllä läpileikkaus heidän työtehtävistään. Ylipäätään taustoitus varmistaa, että haasteltavat ovat kykeneviä vastaamaan DevOpsiin ja Azure DevOpsiin liittyviin kysymyksiin ja tuovat lisäarvoa tutkielmalle.

Automaatio, jatkuva toimitus ja jatkuva integraatio keskittyvät päivitysten ja ominaisuuksien julkaisuun ja automaation toteuttamiseen organisaatiossa. Teema tuo ilmi, kuinka usein päivityksiä ja uusia ominaisuuksia julkaistaan. Koska haastateltavana on DevOpsia käyttävä tiimi, oletuksena on nopea julkaisutahti. Lisäksi kysytään, aiheutuuko nopeasta julkaisutahdista ongelmia tai haasteita. Automaation todettiin jo kirjallisuuskatsauksessa olevan merkittävässä osassa nykypäivän sovelluskehitystä. Haastateltavilta kysytään, miten automaatio on läsnä sovelluskehityksessä ja lisäksi tiedusteltiin sen roolia erityisesti eri vaiheissa, kuten testaamisessa ja julkaisussa. Lisäksi pyritään selvittämään, miten automaatio vaikuttaa projektin lopputuotteen laatuun ja miten automaatioon yleisesti suhtauduttiin ohjelmistokehityksessä.

Kulttuurin teemassa huomion kohteena on DevOpsin läpinäkyvyys ja avoimuus. Teeman tavoitteena on selvittää, miten tiimissä toteutetaan

läpinäkyvyyden ja avoimuuden periaatteita. Haastateltavia voidaan esimerkiksi johdatella kertomaan tapaamisista, palavereista tai Azure DevOpsin projektin seurannasta, mikäli esimerkkejä ei tule heti mieleen. Teemassa myös kysytään, miten avoimuuden ja läpinäkyvyyden koetaan parantavan projektin ja lopputuotteen laatua.

Lopuksi Azure DevOpsiin liittyvä viimeinen teema tuo ilmi haastateltavien asenteita Azure DevOpsia kohtaan. Ensin selvitetään, mikä on ylipäätään suhtautuminen Azure DevOpsiin, eli onko se toimiva palvelu. Seuraavaksi pyritään tuomaan ilmi Azure DevOpsin vaikutukset projektiin laatuun. Lisäksi teemassa selvitetään, mitä haasteita Azure DevOpsin kanssa on ja mitä uusia ominaisuuksia tai parannuksia se kaipaisi.

Neljä eri teemaa luovat kattavan kuvan nykyajan DevOps-sovelluskehitystiimin toiminnasta ja PaaS-pilvipalvelu Azure DevOpsin käytöstä. Niillä saadaan kattavasti mielipiteitä eri taustoilla olevilta henkilöiltä. Kysymykset tuovat ilmi yksilön mielipiteitä ja suhtautumista DevOpsiin ja Azure DevOpsiin. Toimintamalli ja PaaS-pilvipalvelu ovatkin merkittäviä kokonaisuuksia, joihin yksilöt voivat suhtautua täysin eri tavalla. Teemat tuovat myös painoarvoa itse tutkimuskysymyksille. Ensinnäkin saadaan selville, mitkä ominaisuudet ja työkalut ovat tärkeimpiä DevOpsin toteuttamiselle pilvessä. Eri tehtävissä työskentelevillä ohjelmistokehittäjillä on erilaisia vaatimuksia. Samoin saadaan läpileikkaus siitä, mikä Azure DevOpsissa toimii ja mikä ei. DevOps-kulttuuri ei suoraan ollut tutkimuskysymyksissä, mutta avoimuuden toteuttaminen on tärkeää myös PaaS-pilvipalvelussa, jos sen tarkoituksena on toteuttaa DevOpsia. Lopulta selvitetään DevOpsin ja Azure DevOpsin vaikutuksia projektin laatuun. McCallin laatumallin pohjalta luotiin viitekehys, joka sisältää keskeisimmät periaatteet projektin laadun mittaamiselle (McCall ym., 1977.). Viitekehys keskittyi budjetin ja aikataulun toteutumiseen ja lopputuotteen siirtymän, toiminnallisuuden ja ylläpidettävyyden mittaamiseen.

6.4 Haastateltavien valinta

Shash ja Gorley (2006) painottivat, että laadullisessa tutkimuksessa haastateltavien tarkka valinta on ensiarvoisen tärkeää luotettavien tuloksien saamiseksi (Shash & Gorley, 2006). Tämän tutkielman kannalta perusedellytyksenä oli se, että haastateltavat käyttävät sekä DevOps-toimintamallia että Azure DevOpsia. Haastatteluun toteutettiin suuressa ja monipuolisia palveluita tuottavassa IT-alan yrityksessä. Tarjottavia palveluita on aina kuluttajatuotteista isoille organisaatioille tarjottaviin luottamuspalveluihin ja tietoturvaratkaisuihin. Haastatteluun valittiin yksikkö, jossa tuotetaan sähköisen allekirjoituksen palveluita yritysasiakkaille. Palvelua tuotetaan yksikön sisällä omaan käyttöön itse. Käytännössä tämä tarkoittaa sitä, että kehitys ja tuki on tiimin sisällä. Sen sijaan palvelun myynti on ulkoistettu eri yksikölle. Azure DevOpsiin on päädytty pitkälti siitä syystä, että myytävä ja kehitettävä palvelu on rakennettu Microsoftin Azure-

ympäristöön. Suurella yrityksellä on paljon kokemusta teknologiatuotteiden kehittämisestä. Johdon luomat periaatteet ja käytänteet ohjaavat suurilta osin myös tämän yksikön toimintaa. Lisäksi ilmeni, että kyseisessä yksikössä on tiukat kriteerit työntekijöiden valintaan. Työhaastattelut ovat tarkkoja ja haastateltavien osaamista arvioidaan huolellisesti. Haastatteluhetkellä tiimissä oli useita eri rooleja. Mukana oli lukuisia sovelluskehittäjiä, ohjelmistoinsinööri, tuotteen omistaja ja ohjelmistoarkkitehti. Suurella yrityksellä on paljon kokemusta ohjelmistotuotteiden kehityksessä ja toiminta on parhaiksi todettujen toimintamallien ja viitekehyksien mukaista.

Haastateltavien määrälle ei ole laadullisissa tutkimuksissa tarkkaa ohjeistusta. Voidaan kuitenkin todeta, että liian pienellä otannalla aineistosta ei voida tehdä yleistyksiä. Liian suuri määrä taas voi kadottaa yksittäisiä huomioita ja yksityiskohtia. (Hirsjärvi & Hurme, 2022.) Tämän tutkielman kohdalla haastateltavien määrää rajoitti ensisijaisesti tiimin koko. Tiimissä oli yhdeksän Azure DevOpsia käyttävää henkilöä. Toisaalta moniin haastattelun teemoihin ja kysymyksiin ei löytynyt aiemmasta kirjallisuudesta vastauksia. Lisäksi haastateltavissa oli sekä kokeneita kehittäjiä että eri tehtävissä työskenteleviä henkilöitä. Monipuolisilla teemoilla ja eri taustoista tulevilla haastateltavilla pyrittiin varmistamaan riittävä tiedonkeruu.

Haastateltavia lähestyttiin henkilökohtaisesti Slack-viestintäsovelluksen kautta sen jälkeen, kun tiimin tuotteen omistajalta oli saatu siihen lupa. Haastateltavat ohjelmistokehittäjät pyydettiin haastatteluun satunnaisessa järjestyksessä ja haastattelut toteutettiin haastateltavien omien aikataulujen mukaan. Kaikki haastateltavat suostuivat haastatteluun mielellään ja kaikki yhdeksän henkilöä saatiin haastateltua kahdessa viikossa. Ensimmäinen haastattelu toimi mahdollisena testihaastatteluna ja siitä tiedotettiin haastateltavaa. Haastattelurunko ja kysymykset todettiin kuitenkin heti toimivaksi. Lisäksi teemahaastattelun tyyli ja joustava rakenne mahdollistivat lisäkysymyksien kysymisen tarvittaessa ja omaan pohdintaan johdattelun.

6.5 Aineiston käsittely

Haastattelut toteutettiin Teams-sovelluksen kautta ja nauhoitettiin varmuuden vuoksi kahdella eri laitteella. Haastatteluita purkaessa nauhoitus toistettiin ja samalla kirjoitettiin olennaisia kohtia Wordiin. Myös litteroinnissa hyödynnettiin selkeyden vuoksi teemoja. Nauhoituksissa ja litteroinnissa edettiin teema kerrallaan. Ensimmäinen haastateltava litteroitiin ensimmäisenä ja sen jälkeen edettiin haastattelujärjestyksessä.

Hirsjärvi ja Hurme (2022) painottivat, ettei litteroinnin tarkkuudelle ole määriteltynä yleispätevää tarkkuutta. Käytännössä tämä tarkoittaa sitä, ettei läheskään aina tarvitse litteroida sanantarkasti. Sanasta sanaan litterointi voi lisätä työmäärää ja käytettävää aikaa merkittävästi. Vapaamman litteroinnin edellytyksenä tälle on kuitenkin se, että litteroija tuntee aineiston hyvin ja tunnistaa eri

teema-alueet. Litteroinnissa on myös tärkeää pyrkiä säilyttämään konteksti, mikäli se on tulkinnan kannalta olennaista. Lisäksi jos litterointiin lainataan haastateltavan maininta sanasta sanaan, on se tyypillisesti hieman epätavallinen ja muusta massasta erottuva heitto. (Hirsjärvi & Hurme, 2022). Tämän tutkielman kohdalla haastattelija suorittaa myös litteroinnin itse. Kun aihe ja teemat ovat tuttuja, voidaan materiaalista poimia olennaisimmat osat. Tarvetta sanasta sanaan kirjoittamiselle ei havaittu.

Kun haastattelut on litteroitu, on olemassa riski, että kokonaisuus pirstaloituu (Hirsjärvi & Hurme, 2022). Ongelma pyritään kuitenkin korjaamaan itse tutkielmassa. Haastatteluista muodostetaan uusi kokonaisuus, jota hyödynnetään tutkimuskysymyksiin vastaamisessa ja teorian luomisessa. Koska sekä haastattelut että litteroinnit ovat teemoittain, on kokonaisuuden rakentaminen uudestaan sen myötä helpompaa.

7 TULOKSET

Tässä luvussa käydään läpi haastatteluiden tuloksia ilman vertailua kirjallisuuskatsauksen tuloksiin. Luku on jaettu neljään alalukuun teemahaastattelun neljän eri teeman mukaan. Ensimmäisessä alaluvussa selvitetään haastateltavien taustat. Taustoitus sisälsi työnimikkeen, työkokemuksen nykyisessä tehtävässä ja aiemman DevOps-kokemuksen. Toinen luku keskittyi automaatioon, jatkuvaan toimitukseen ja integraatioon. Kolmas luku tarkasteli DevOpsin avoimen ja läpinäkyvän kulttuurin toteuttamista. Lopuksi neljäs luku tarkasteli haastateltavien suhtautumista Azure DevOpsiin.

7.1 Haastateltavien taustat

Haastateltavien taustoja selvitettiin haastattelun ensimmäisessä teemaosiossa. Taustoissa oli havaittavissa selviä eroja. Osa haastateltavista oli ollut talossa jo pitkään, kun taas osa oli aloittanut vasta viime vuosina. Monella oli DevOps-kokemusta jo pitkältä aikaväliltä. Uran aikana oli nähty DevOpsin kehittyminen uudesta asiasta monien huulilla olevaksi trendiksi. Samoin Azure DevOpsista oli monella aiempaa kokemusta. Tämä päti myös Azure DevOpsin aiempiin versioihin, kuten Microsoft Team Foundation -palveluun. Tämän myötä saatiin näkökulmaa siitä, miten palvelu on kehittynyt vuosien varrella ja miten sen ongelmia on parannettu. Samoin ilmeni myös ei-toivottavia muutoksia esimerkiksi GitHubin suhteen. Monipuolisilla taustoilla ja pitkällä kokemuksella saatiin myös paikattua haastatteluiden lyhyitä kestoja. Haastatteluiden pituus ei myöskään kerro kaikkea. Esimerkiksi H9 oli selvästi kokenut kehittäjä. Haastattelu ei ollut kovin pitkä, mutta hyvin tiivis ja täynnä tutkielman kannalta konkreettista tietoa. Haastateltava tiesi, mitä sanoi, eikä taukoja juuri muodostunut. Myös muissa haastatteluissa ilmeni samanlaista tehokasta tahtia. Selvästi suurin osa haastateltavista oli ohjelmistokehittäjän työnimikkeen alla, joista yksi oli senioritason ohjelmistokehittäjä. Yksi oli ohjelmistoinsinööri ja lisäksi tiimissä oli tuotteen omistaja (Product owner) ja arkkitehti. Haastatteluiden kestossa oli selvää hajontaa. Lyhyin

haastattelu oli 12 minuuttia ja pisin 37 minuuttia. Yhdeksän haastattelun pituuden keskiarvo oli 21 minuuttia. (Katso TAULUKKO 4).

Taustoituksessa selvitettiin myös, millaisia työtehtäviä haastateltavat päivittäin tekevät. Erilaiset version- ja koodinhallintaan liittyvät työkalut nousivat monen kohdalla esiin, kuten Git-versionhallinta, koodin tarkastelu ja koodin koonti. Pipelines oli käytössä julkaisussa eri ympäristöihin ja artifactsia käytettiin pakettien ja kirjastojen tallentamiseen ja jakamiseen. Pull requestit ja work itemit katsottiin keskeiseksi ominaisuudeksi ja samoin projektin etenemisen ja tarjolla olevien tehtävien tarkastelu Boardsilta. Myös Nugget-pakettien tarkistaminen ja wikin kirjoittaminen ja lukeminen nousivat esille. Ylipäätään käytettävien ominaisuuksien käyttäminen riippui vahvasti henkilön omista työtehtävistä ja taidoistaan. Yksi haastateltava myös näki Azure DevOpsin yhtenä suurena työkaluna, eikä palveluna, joka sisältää useita eri työkaluja.

TAULUKKO 4 Haastateltavien taustat

Koodi	Titteli	Kokemus nykyisessä tehtävässä	Aiempi DevOps-kokemus	Haastattelun kesto
H1	Software Developer	5 vuotta	Ei ole	17 min
H2	Software Developer	2 vuotta	Ei ole	12 min
H3	Software Engineer	11 vuotta	Kyllä	25 min
H4	Software Developer	2 vuotta	Kyllä	24 min
H5	Software Developer	4.5 vuotta	Ei	18 min
H6	Senior Software Developer	1.5 vuotta	Kyllä	18 min
H7	Software Developer	4 vuotta	Ei	37 min
H8	Product Owner	1 vuosi	Kyllä	15min
H9	Lead Software Architect	6 vuotta	Kyllä	21 min

7.2 Automaatio, jatkuva toimitus ja integraatio

Automaatio, jatkuva toimitus ja integraatio kuvattiin luvussa 2 merkittäviksi piirteiksi DevOps-mallin toteuttamiseksi. Automaatiolla voidaan korvata toistuvia ja rutiininomaisia manuaalisia työvaiheita automatiikalla. Jatkuva toimitus mahdollistaa päivitysten ja ominaisuuksien nopean jakelun. Jatkuva integraatio taas tarkoittaa sitä, että koodi on aina kääntyvää ja luotettavaa. (Rossel, 2017.) Tässä haastattelun teemassa selvitettiin, miten usein päivityksiä julkaistiin ja miten nopeaan julkaisutahtiin suhtauduttiin. Samoin tarkasteltiin automaation roolia testaamisessa, julkaisussa ja sovelluskehityksessä yleisesti. Lisäksi pyrittiin selvittämään haastateltavien mielipiteet automaatiota, jatkuvaa toimitusta ja jatkuvaa integraatiota kohtaan ohjelmistokehityksessä yleisesti.

Jatkuvan toimituksen kohdalla haasteena oli erilaiset roolit ja eri ympäristöt. Tiimillä oli käytössään kolme eri ympäristöä: kehitys (development), testaus (staging) ja tuotanto (production). Eri henkilöt julkaisivat eri ympäristöihin ja

esimerkiksi kaikki eivät osanneet sanoa, kuinka usein tuotantoon, eli loppuasiakkaille, näkyviä päivityksiä julkaistiin.

H1, H2, H3, H9 ja H4 kertoivat, että tuotantoon julkaistaan päivityksiä noin kahden viikon välein. H1 mainitsee, että mikropalveluiden vuoksi joskus päivityksiä on useamminkin ja H4 kertoo, että pienet bugikorjaukset eivät yleensä odota kahta viikkoa. H3 ilmaisee, että regressiotestit ovat pullonkaulana, mikä estää päivityksien julkaisun vieläkin nopeammin. Ylipäätään päivityksiä voi olla päällekkäin samanaikaisia, eli todellisuudessa päivityksiä tulee tiheästi. Product Owner H8 mainitsee kokonaismääräksi kuukaudelle noin 3-4 päivitystä ja pääarkkitehti H9 lisää vielä, että säännöllistä julkaisuväliä ei ole, vaan mennään päivitysten koon ehdoilla. Kun päivitys on valmis ja testattu, voidaan se julkaista nopeasti. Päivitysten koko vaihtelee suuresti. H5, H6 ja H7 eivät vastaa tuotantoympäristön julkaisusta. Sen sijaan he julkaisevat kehitysympäristöön. Nopean julkaisun etuna on se, että kehitysympäristöön voi julkaista välittömästi. Käytännössä tämä tarkoittaa yhden napin painamista. H7 mainitsee automaation olevan tällöin isossa roolissa.

Jatkuvaan julkaisuun suhtauduttiin yksimielisen positiivisesti, mutta myös tärkeitä huomioita ja rajoituksia nostettiin esille. H1, H9 ja H8 kertovat eduiksi sen, että mikäli ongelmia tulee, tiedetään heti syy. Pienten päivitysten seuranta on helpompaa, kun tiedetään, mikä päivitys tai muutos ongelman aiheutti. Suurissa päivityksissä taas voi olla useita muutoksia, jotka on käytävä läpi ongelmien juurisyyn selvittämiseksi. Tämä viivästyttää sekä korjausta että projektin etenemistä kokonaisuudessaan. H9 lisää, että rollback on pienillä päivityksillä kevyempää ja asiakkaalle voidaan ladata päivitykset valmiiksi ja kytkeä näkyviin vasta myöhemmin. H3:n mukaan asiakas hyötyy, kun saa päivitykset nopeammin käyttöönsä ja mahdolliset ongelmat selvitetään nopeasti. Samoin kuin H3, myös H4, H5 ja H6 korostavat automaation ja testauksen roolia jatkuvien toimituksien edellytyksenä. Jos julkaisutahti on nopea, on testauksen pysyttävä perässä. Jos näin ei ole, voi projektin laatu heikentyä tai julkaisuun syntyy pullonkauloja. H3 ja H4 kertovat, että nopeisiin julkaisuihin tähtääminen voi saada kehittäjät hätiköimään, mikä heikentää lopputuotteen laatua. H5 mainitsee useat eri branchit, säilytyspaikat ja riippuvuudet myös haasteiksi useita pieniä päivityksiä samaan aikaan kehittäessä. Kuitenkin jos testaus on kunnossa, ei jatkuva toimitus lisää olennaisesti virheiden riskiä. H1, H2, H6 kertovat, että samat bugit voivat lipsahtaa julkaisuun, oli julkaisutahti sitten hidas tai nopea. Nopeissa ja pienissä julkaisuissa virheiden paikantaminen ja korjaus on vain helpompaa. Aiemman DevOps-kokemuksen perusteella H6 lisää vielä: ”Pienet päivitykset ovat hyviä, ei välttämättä nopeat”. Tähän haastateltava täsmentää, että testauksen lisäksi muut prosessit eivät välttämättä pysy perässä. Tällöin nopea päivitystahti voi olla huono asia.

Automaatio on tiimin ympäristössä mukana sekä testaamisessa että julkaisussa. H1 mainitsee, että jokaisen commitin ja mergen jälkeen on automaatiota ja H3 lisää, että CI Pipelines huolehtii koodin kääntyvyydestä ja luotettavuudesta. Jos koodissa on merkittäviä ongelmia esimerkiksi syntaksin suhteen, ilmenevät ne tehokkaasti jo tässä vaiheessa. H7 lisää vielä, että myös packages on

automatisoitu. H4, H5 ja H6 kertovat yksikkötestien olevan käytössä ja toimivan hyvin. H6 ja H7 kuitenkin valittelevat integraatiotestien puutteita ja H5 end-to-end-testien ongelmista. H3, H8 ja H9 kertovat myös, että testeissä on kokonaisuutena vajavaisuuksia ja parantamisen varaa, mutta H8 lisää, ettei tämä vielä heijastu lopputuotteen laatuun. Ilmaisusta kuitenkin ilmeni huolestuneisuus ja se, että testaamista on pakko kehittää tulevaisuudessa, jotta lopputuotteen laatu ei lähde laskuun. Pelkän Azure DevOpsin käyttö ei takaa täydellisiä testejä. H7 ja H4 kertovat UI-testeistä, jotka huolehtivat lopputuotteen käytettävyydestä asiakkaan näkökulmasta. Testit klikkailevat käyttöliittymää automaattisesti ja varmistavat, että palvelu toimii asiakkaan päätteellä. H4 mainitsee vielä rasitustesteistä, jotka testaavat järjestelmän vakautta kuormituksessa. H5, H7 ja H9 korostavat automaation roolia kehitysympäristöön julkaisussa. Käytännössä julkaisu vaatii vain yhden napin painalluksen ja automatiikka hoitaa loput. H2, H5, H6 ja H9 kertovat, että tuotantoon julkaisu on manuaalista ja vaatii arvioinnin oikealta ihmiseltä. H6 lisää tämän olevan tarpeellista, sillä täysi automaatio tuotantoon julkaisulle tuntuu jopa vaaralliselta. Tuotantoon menevät julkaisut näkyvät asiakkaille välittömästi. Jos julkaisussa pääsee läpi virhe, voi sillä olla suuria vaikutuksia. H4 lisäsi myös, että automaattisen testaamisen määrä riippuu ympäristöstä ja julkaisusta: ”Esimerkiksi UI-testejä ei tarvitse tehdä kaikille pull requesteille. Lisäksi Staging-ympäristöön ei ole tarvetta testata yhtä kattavasti kuin tuotantoon. Pitkät testit vain hidastavat julkaisua”.

Automaatio ylipäätään nähtiin erittäin tärkeänä ja suorastaan edellytyksenä nykypäivän ohjelmistokehityksessä. H4 kertoo, että ilman automaatiota julkaisun prosessit on ymmärrettävä ja osattava erittäin hyvin. H1, H2, H3, H5, H6 ja H7 korostavat, että manuaaliset ja toistuvat prosessit ovat tärkeintä automatisoida. H6 lisää, että ne koetaan monesti jopa tylsäksi ja kaikki, mikä voidaan automatisoida, on automatisoitava. Kaikki haastateltavat korostivat, että automaatio vähentää virheitä, mikä näkyy lopputuotteen parantuneessa laadussa. H3 ja H5 kertovat yhdeksi syyksi sen, ettei automaatio tee inhimillisiä virheitä, kuten unohduksia. H1, H2, H4, H5 H7 ja H8 kertovat myös automaation näkyvän positiivisesti resurssien säästössä ja aikataulun toteutumisessa.

Kuitenkin automaatiosta ilmeni, ettei sen rakentaminen ole helppoa. Esimerkiksi H3 vertaa automaatiota investoinniksi sanomalla seuraavasti: ”Automaation rakentaminen on investointi. Esimerkiksi skriptit pitää luoda. Lopussa manuaalisten prosessien korvaaminen automatiikalla kuitenkin palkitsee”. H8 kertoo, että tiimin kannalta olisi parasta, jos yksi henkilö olisi jatkuvasti kehittämässä testausta ja automaatiota. H8 kertoi, että näin pyritään seuraavassa rekrytoinnissa tekemäänkin.

7.3 Kulttuuri

DevOps-toimintamalli sisältää avoimuuden kulttuurin. Tällä tarkoitetaan sitä, että kaikki näkevät projektin etenemisen ja roolinjaon. Tietoa myös jaetaan tiimin keskuudessa ja epäonnistumisista opitaan. (Humble & Molesky, 2011.) Myös

Azure DevOps pyrkii toteuttamaan avoimuuden kulttuuria. Esimerkiksi Azure Boards näyttää projektin vaiheet reaaliajassa ja sisäisessä Wikissä voidaan jakaa omaa osaamistaan (Microsoft, 2022). Haastattelussa pyrittiin tuomaan ilmi, miten tiimin keskuudessa totutetaan avoimuuden ja läpinäkyvyyden ilmapiiriä. Haastateltavaa pyydettiin myös mainitsemaan esimerkkejä. Lisäksi selvitettiin, miten haastateltavat kokevat avoimuuden ja läpinäkyvyyden vaikuttavan projektin laatuun.

H1, H2, H7 ja H8 mainitsivat esimerkkinä päivittäiset aamupalaverit, joissa kukin kertoi omista työtehtävistään ja ilmaisi, jos tarvitsi tekemistä. H1, H5 ja H7 kertoivat retrospektiiveistä kolmen viikon välein, jossa muun muassa todetaan, mikä onnistui ja mikä meni pieleen. Ylipäätään tiimissä koettiin vallitsevan avoimen keskustelun henki, jossa ketään ei jätetty pulaan. H4 mainitsi, kuinka on parasta tietää niin paljon kuin mahdollista ja myöntää, jos ei tiedä jotain.

Azure DevOpsin koettiin tukevan avoimuuden kulttuuria hyvin. H2, H3, H4, H5, H6, H8 ja H9 kertoivat Azure DevOpsin Boards-välilehden kertovan projektin etenemisestä ja tehtävänjaosta. H5 kertoi, että Boardsilta voi valita oman osaamisen mukaan sopivan tehtävän. H9 kuitenkin tarkensi, että Boards näyttää mitä tapahtuu, mutta ei tarkemmin esitä, mitä kukin tekee. H1, H3, H8 ja H9 mainitsivat yhdeksi avoimuuden työkaluksi Work Itemit. Niiden avulla voidaan seurata yksittäisen tehtävän etenemistä ja kommenttikenttä mahdollistaa arvioinnin ja avun pyytämisen. H4 nosti vielä Azure DevOpsista esiin Roadmapit ja Backlogit avoimuuden tukijoina.

Samoin kuin automaation kohdalla, avoimuus ja läpinäkyvyys koettiin projektin laadun kannalta yksimielisen positiiviseksi asiaksi. H3 nosti tekijöiksi sen, että virheistä ilmoitetaan avoimesti ja palautetta annetaan pienellä kynnyksellä. Kukaan ei elä pelossa tai epätietoisuudessa. H4 sanoi, että useampi ihminen löytää helpommin virheet kuin yksi ja H8:n mukaan avoin keskustelu varmistaa, että kaikki ymmärtävät muutoksen tavoitteen. H9 kertoi havainneensa, kuinka avoin keskustelu ennen uusien muutoksien tekemistä on tärkeää. Jos aletaan vain koodaamaan suoraan, voi lopputulos olla vääränlainen ja usean viikon työ menee hukkaan. H5 tiivisti avoimuuden edut yksinkertaisesti: ”Kaikki tietävät, mitä tapahtuu ja mitä kukin on tekemässä”.

7.4 Azure DevOps

Viimeinen teema käsitteli haastateltavien suhtautumista Azure DevOpsiin. Teemassa pyrittiin tuomaan ilmi, pidetäänkö Azure DevOpsia yleisesti hyvänä ja toimivana palveluna ja parantaako se projektin laatua. Lisää näkökulmia haettiin myös kysymällä, mitä haasteita Azure DevOpsin kanssa on ja mitä ominaisuuksia ja työkaluja haastateltavat kaipaisivat lisää.

Kokonaisuutena Azure DevOpsiin suhtauduttiin yhtä haastateltavaa lukuun ottamatta positiivisesti. H9 kehui, kuinka Azure DevOps kattaa koko tuotteen elinkaaren: ”Azure DevOps sisältää kaiken tarvittavan. Projekti alkaa work itemistä

tai asiakastarinasta, dokumentaatio löytyy palvelusta, testiautomaatio tukee integraatiota ja koko Pipeline voidaan julkaista kerralla. Kaikki hoituu samassa palvelussa". H7 kertoi, kuinka palvelu lisää kehittäjien välistä yhteistyötä ja toimii hyvin pilvessä, mikä parantaa saatavuutta. H4 lisäsi saatavuuteen, että käyttökatkokset ovat ikäviä, mutta hyvin harvinaisia. H3, H8 ja H9 kertoivat, kuinka kehitettävä tuote on rakennettu Microsoft Azure -ympäristön päälle, mikä tekee Azure DevOpsin käytöstä saumattomampaa. H5 kehuu, kuinka kaikki työkalut löytyvät valmiiksi, eikä niitä tarvitse hakea muualta. Kaikkien mielestä Azure DevOps on myös helppo oppia. H2 kuitenkin lisäsi, että Azure DevOps on laaja kokonaisuus, jonka hahmottamisessa voi mennä hetki.

Kokonaisuutena Azure DevOpsin koettiin parantavan projektin laatua. Kattavien työkalujen, ominaisuuksien ja testien myötä itse lopputuote on parempi. Kattavalla testaamisella löydetään virheitä ja kun päivitykset ovat pieniä mutta tiheitä, löydetään ongelman aiheuttajat selvästi. Nopeiden toimituksien ja automaation myötä Azure DevOps myös nopeuttaa projektin kehitystä. H4 kertoi, että palvelu ja sen automatiikka vapauttaa aikaa itse työskentelyyn. Yhtenä aikataulun nopeutuksen tapana H9 mainitsi, kuinka work itemeiden luominen on todella helppoa ja ne voidaan linkittää lähdekoodiin, jotta nähdään heti, mitä muutoksia on tehtävä. Myös budjetin kannalta Azure DevOps koettiin positiiviseksi ja laatua parantavaksi. Toisaalta haastattelussa ei voinut verrata palvelua suoraan muihin ratkaisuihin, koska ei ollut tietoa, kuinka paljon ne olisivat maksaneet organisaation käytössä. H8 ja H9 kertoivat palvelun hinnoittelun olevan järkevä ja H9 lisäsi vielä, että kun palvelu sisältää valmiiksi kaiken tarvittavan, ei työkaluja tarvitse ostaa muualta. Budjetin lisäksi tämä etu näkyy aikataulussa. H4 painotti vielä, että palvelu ei ole isoille yrityksille edullinen, mutta maksaa kuitenkin itsensä takaisin. Työkalut, tiimin hallinta ja turvallisuus on hankittava tavalla tai toisella ja Azure DevOps sisältää kaiken samassa paketissa.

Myös haasteita ja parannettavaa löytyi. H3:n mielipide Azure DevOpsia kohtaan oli muita kriittisempi ja arvosanaksi tämä kertoi antavansa palvelulle 5-6/10. GitHubin hankinnan myötä Microsoft muokkasi GitHubin toimintoja osin huonompaan suuntaan. Myös H6, H7 ja H9 olivat huolissaan GitHubin tulevaisuudesta. Esimerkiksi H6 kuvaili seuraavasti: "Olen ollut pitkään huolissani GitHubin ostosta. Pelkään, että tulevaisuudessa Azure DevOps ja GitHub yhdistyvät täysin ja palvelusta tulee huonompi". H3 jatkoi, kuinka perustyökalut toimivat hyvin, mutta monimutkaisempien ja kehittyneempien asioiden tekeminen on paikoin haastavaa ja integrointi muiden kuin Microsoftin palveluiden kanssa ei onnistu aina. Myös Feature branching, Pipelines ja Work Item -hallinta ovat osin vanhentuneita. Work Itemit pitäisi voida yhdistää kalenterin kanssa ajanhallinnan parantamiseksi. Muiltakin haastateltavilta tuli mainintoja Azure DevOpsin parannusehdotuksista ja ongelmista, mutta enemmän yksittäisellä tasolla. H1 mukaan Pull Requesteissä olisi parantamisen varaa ja H9 kaipaisi taas parannuksia testisuunnitelmiin ja Support-tikettien näkymiseen Boards-välilehdellä. H4 kaipaisi lisää selkeyttä kauppapaikalle ja H5 taas konfliktien selvittämiseen. H6 nostaa esiin joidenkin asioiden konfiguroinnin puutteen ja vaatimattoman dokumentaation ja kuinka GitHub ja Cubernetics tarvitsisivat lisää työkaluja. H7:aa ovat häirinneet bugit, joita on kyllä korjattua osin, mutta löytyy käyttöliittymästä vieläkin.

H7 kaipaisi myös lisää statistiikkaa, kuten listaa kaikista aktiivisista pull requesteista. Myös testausta pitäisi parantaa ja migraatiossa on ollut ongelmallista. H8 mainitsi vielä, että Azure DevOps ei ihan istu SAFe-viitekehyksen kanssa. Vaikka kokonaisuus on kunnossa, oli parannettavaa yksittäisissä asioissa vielä paljon.

8 JOHTOPÄÄTÖKSET

Luvussa palataan tutkimuskysymyksiin ja niiden määrittelyyn. Molempiin tutkimuskysymyksiin vastataan johtopäätöksien avulla tiivistetysti ja kattavasti. Lisäksi verrataan kirjallisuuskatsauksen tuloksia ja huomioita haastatteluiden tuloksiin. Huomiota kiinnitetään erityisesti siihen, miten pelkästään DevOps-toimintamallin kirjallisuus tukee tutkimusta Azure DevOpsista. Lisäksi arvioidaan toteutuneen tutkimuksen luotettavuutta. Lopuksi arvioidaan vielä tutkimuksen luotettavuutta.

8.1 Ensimmäinen tutkimuskysymys

Ensimmäisenä tutkimuskysymyksenä oli ”Miten DevOps-toimintamallin käytänteitä voidaan toteuttaa PaaS-palveluna?”. Lisäksi apukysymyksenä oli ”Miten Azure DevOps toteuttaa niitä?”. Kysymykseen vastaamisen edellytyksenä oli ensin määritellä DevOpsin toimintaperiaatteet. Keskeisimmiksi ominaisuuksiksi katsottiin automaatio, jatkuva toimitus ja jatkuva integraatio (Rossel, 2017). Lisäksi DevOpsiin sisältyi avoimuuden kulttuuri, joka on perusedellytys toimintamallin toteutuksen onnistumiselle (Humble & Molesky, 2011). Ensimmäisen tutkimuskysymykseen voidaan vastata, että DevOpsia toteuttavan PaaS-palvelun pitää tukea jatkuvaa toimitusta, integraatiota ja sisältää automaatiota ja läpinäkyvyyttä.

Käytännössä automaatio edellyttää tekoälyn ja koneoppimisen hyödyntämistä PaaS-palvelussa. Manuaalisia prosesseja on pystyttävä automatisoimaan mahdollisimman pitkälle. Automaation on tärkeää olla eri työkaluissa läsnä ja kuitenkin selkeästi ja helposti käytettävissä. Azure DevOps sisältää lukuisia automaatiota ja testaamista tukevia ominaisuuksia ja työkaluja, jotka hyödyntävät tekoälyä ja koneoppimista (Microsoft, 2022). Kirjallisuudessa Azure DevOpsin automaatio sai kehuja esimerkiksi Vuppalapatilta ym. (Vuppalapati ym., 2020). Haastatteluissa tuli ilmi, että etenkin manuaalisia ja toistuvia prosesseja pyrittiin

automatisoimaan mahdollisimman pitkälle. Kaikki haastateltavat kokivat automaation erittäin tärkeäksi ja totesivat sen parantavan projektin laatua.

Testaamisen työkalujen on oltava kattavia, jotta automaatiosta saadaan kaikki irti. Mitä parempia testit ovat, sen parempaa on myös projektin laatu. Tämä ilmeni sekä kirjallisuudessa että haastatteluissa. Mohammad (2017) korosti testaamisen merkitystä virheiden löytämisessä ja projektin etenemisessä (Mohammad, 2017). Myös haastateltavat päätyivät yksimielisesti samaan tulokseen. Testaamista verrattiin tuottavaan investointiin. Kattavien testien luominen vie aikaa, mutta palkitsee parantuneella tuotteen laadulla ja nopeutuneella aikataululla. Microsoft mainostaa, että Azure Test Plans on hyvin kattava. Tekoäly ja koneoppiminen on mukana testeissä ja lisäksi testien onnistumisesta luodaan käyttäjälle tilastoja. (Microsoft, 2022.). Haastattelussa kuitenkin ilmeni, että myös Azure DevOpsin testaamisen työkaluissa on parantamisen varaa. Vaikka testaamiseen oltiin pääasiassa tyytyväisiä, toi kolme haastateltavaa parannusehdotukseksi testien parantamisen. Lisäksi vaikka testaamisen ja automaation työkalut olisivat hyvin kattavia, on niiden oltava helppoja käyttää. Testien käytettävyys olikin yksi haastatteluissa ilmennyt kompastuskivi.

DevOpsia toteuttaakseen PaaS-palvelun on myös toteutettava jatkuvaa toimitusta ja integraatiota. Kun kehittäjä on luonut koodia, on palvelun tarkastettava sen toimivuus ja luotettavuus nopeasti. Tämän jälkeen koodi tallennetaan säilytyspaikkaan, josta se voidaan tarvittaessa julkaista nopeastikin. On organisaatiosta riippuvaa, kuinka monta eri julkaisuympäristöä on ja voiko julkaisun tehdä välittömästi ilman manuaalista hyväksyntää. Haastatteluissa ilmeni, että Azure DevOpsia käyttävällä organisaatiolla on kolme eri ympäristöä. Kehitysympäristöön pystyi julkaisemaan testien jälkeen välittömästi, mutta tuotantoon vaadittiin manuaalinen tarkistus ja hyväksyntä toiselta henkilöltä. Yksi haastateltava korosti, että Azure DevOpsin avulla päivitykset voidaan jakaa asiakkaille jo valmiiksi, mutta aktivoida vasta myöhemmin. Tämä on yksi keino toteuttaa jatkuvaa integraatiota ja toimitusta. Päivityksen toimintaa monitoroidaan jatkuvasti ja tarvittaessa korjaukset voidaan julkaista myös nopeasti.

Azure DevOpsin käyttö oikean organisaation kohdalla noudattaa hyvin Mohammedin (2018) jatkuvan integraation ja toimituksen periaatteita. Päivityksen koko elinkaari on Azure DevOpsissa alusta loppuun asti. Yksi haastateltava korostikin hyvänä ominaisuutena elinkaarenhallintaa. Koontityökalut varmistavat koodin toiminnallisuuden ja tiheät päivitykset voidaan jaella ilman katkoksia palvelun saatavuudessa. Yhtenä kriteerinä oli myös se, että usea henkilö pystyy työskentelemään saman koodin kanssa samaan aikaan. (Mohammad, 2018.) Azure DevOpsin branchit varmistavat tämän mahdollisuuden. Tosin haastattelussa koettiin joskus haastavaksi useamman branchin käyttö ja niiden väliset riippuvuudet.

Läpinäkyvyys ja avoimuus katsottiin DevOps-kulttuurin edellytykseksi. Tiimin työskentelyn on oltava saumatonta. DevOps-sisältääkin useita eri tapoja edistää avoimuutta, kuten erilaiset toistuvat palaverit, tiedon jakaminen keskenään ja projektin seuranta. (Humble & Molesky, 2011.) Myös DevOpsia toteuttavan PaaS-palvelun on oltava toiminnaltaan läpinäkyvä ja avoin. Tehtävänjaon,

projektin seurannan ja keskustelun on oltava kaikille näkyvää, jotta tiedetään, missä vaiheessa projekti etenee ja minkä tehtävän parissa kukin on. Yleiskatsauksen antava Kanban-tyylinen näkymä koettiin tehokkaaksi toteutukseksi haastatteluissa. Palvelun on tärkeä sisältää Wiki, johon voidaan koota tietoa. Muiden töitä on myös päästävä kommentoimaan ja arvioimaan, sillä DevOpsiin kuuluu palautteen antaminen. Kaksi henkilöä löytää paremmin virheet kuin yksi.

Azure DevOpsin koettiin toteuttavan avoimuutta ja läpinäkyvyyttä varsin hyvin. Boards antaa kattavan yleiskuvan projektista. Toisaalta Boards-välilehdelle kaivattiin lisää tietoa siitä, mitä kukakin tekee ja myös Support-tiketit haluttiin näkyviin. Work Itemit antoivat tehokkaan seurannan yksittäisille työtehtäville ja tarjosivat arviointi- ja kommentointimahdollisuuden. Azure Roadmap ja Backlog antavat myös kattavan kuvan projektin eri vaiheista ja etenemisestä ja Wikissä voidaan jakaa tietoa tiimin keskuudessa.

PaaS-palvelun kohdalla ei pidä unohtaa pilvipalvelun periaatteita. Kirjallisuuskatsauksessa esitettiin viisi ominaispiirrettä pilvipalvelulle: pyynnöstä itsenäinen palvelu (on-demand self-service), laaja saavutettavuus verkosta (broad network access), resurssien jako (resource pooling), nopea joustavuus (rapid elasticity) ja mitattavat palvelut (measured service). Jotta DevOpsia toteuttava PaaS-pilvipalvelu toimii luotettavasti ja onnistuneesti, on sen täytettävä nämä piirteet. Azure DevOpsin katsottiin myös toteuttavan periaatteet kaikilta osin jo kirjallisuuskatsauksessa. Haastatteluissa tämä korostui. Esimerkiksi yksi haastateltava kehui saavutettavuutta verkosta käsin mistä päin tahansa. Samoin toinen kehui, kuinka käyttökatkoksia on hyvin harvoin. Lisäksi ilmeni, että palvelu skaalautuu hyvin eri kokoisten tiimien tarpeisiin ja on helppo mukauttaa itsenäisesti.

Pilvipohjaisen DevOps-palvelun on siis tuettava DevOps-toimintamallin periaatteita. Toisaalta haasteena voi olla, miten perusedellytykset toteutetaan työkaluina. Kehittäjät joutuvat miettimään kaikkia eri vaiheita tuotteen elinkaaresta aina tiimin väliseen viestintään. Edellytykset eivät täyty, jos esimerkiksi päivitykset voidaan integroida ja julkaista helposti, mutta muut eivät näe, mitä julkaisu sisältää. Käyttöliittymän on myös oltava selkeä etenkin suuren työkalumäärän kohdalla. Tämä ilmeni haastatteluissa esimerkiksi Azure DevOpsin testaamisen työkaluissa. Microsoft mainosti työkaluja kovasti ja kirjallisuudessa ne olivat saaneet kehuja. Käytännössä kuitenkin testityökalut vaatisivat edelleen lisää kehitystä ja muitakin parannusehdotuksia ilmeni paljon. Parin haastateltavan mukaan Azure DevOpsin rajat tulevat herkästi vastaan, jos sillä tekee jotain erityisen monimutkaista. Ongelmista huolimatta Azure DevOps toteuttaa DevOpsia varsin hyvin. Siihen integroitu, joskin huolia aiheuttava GitHub, takaa kattavan versionhallinnan. Päivityksiä voidaan integroida ja julkaista helposti ja useampi henkilö voi työskennellä branchin ansiosta saman koodin parissa samaan aikaan. Automaatio auttaa testeissä ja esimerkiksi Boards, Work Itemit ja sisäänrakennettu Wiki tukevat avoimuutta.

8.2 Toinen tutkimuskysymys

Tutkielman toinen tutkimuskysymys oli seuraava: ”Miten DevOpsiin pohjautuvat pilvipalvelut auttavat ohjelmistoyrityksiä projektien laadussa?”. Koska ohjelmistoyrityksien projektien laatu on laaja käsite, määriteltiin se tutkielmaa varten. Kirjallisuuskatsauksessa luotiin oma viitekehys, jonka pohjana toimi McCallin laatumalli (McCall ym., 1977.). Vaikka apukysymyksenä ei tässä tutkimuskysymyksessä ollut Azure DevOpsia, on se mitattavana DevOps-pilvipalveluna mukana.

Laadun viitekehyksen ensimmäinen mittari oli aikataulu. Haastatteluissa oltiin yksimielisiä siitä, kuinka Azure DevOps nopeuttaa projektin etenemistä ja auttaa aikataulussa pysymisessä. Syitä tälle voi olla useita. Esimerkiksi kaikkien työkalujen koettiin olevan valmiina samassa paketissa, joten niitä ei tarvitse hankkia eri paikoista. Lisäksi yhtenä tekijänä on automaatio. Etenkin manuaalisia ja toistuvia prosesseja haluttiin automatisoida mahdollisimman pitkälle. Sen lisäksi että ne vievät aikaa, vähentävät ne myös työn mielekkyyttä. Automaation myös koettiin parantavan projektin laatua ja se ja testaaminen katsottiin nykypäivänä perusedellytyksiksi. Tämä oli linjassa kirjallisuuden kanssa. Esimerkiksi Avinash (2022) korosti, kuinka manuaalisten prosessien korvaaminen automaatiolla mahdollistaa kehittäjien täyden keskittymisen työhönsä. Lopputuloksena asiakastyytyväisyys parantuu ja projekti valmistuu nopeammin. (Avinash, 2022.) Myös Khan M. ym. (2020) totesivat automaation nopeuttavan projektia ja vähentävän syntyviä virheitä (Khan, M. ym., 2020).

Jatkuva toimitus ja julkaisu ovat merkittävä tekijä projektin aikataulussa. Pieniä päivityksiä voidaan julkaista nopeaan tahtiin, mikä on positiivista myös asiakkaan näkökulmasta. Azure DevOpsissa useampi henkilö voi työskennellä saman päivityksen parissa samaan aikaan, mikä nopeuttaa sen valmistumista. Samoin päivitykset voitiin julkaista jo etukäteen ja aktivoida vasta myöhemmin. Jatkuva integraatio taas mahdollistaa sen, että säilytyspaikkaan julkaistava koodi on aina luotettavaa ja tehokasta (Rossel, 2017). Lisäksi kehitettävään tuotteeseen voidaan tehdä jatkuvasti muutoksia ilman katkoksia saatavuudessa. Näin projektin aikataulu ei sotkeudu ja tuotteesta tulee paremmin asiakkaan toiveita ja vaatimuksia vastaava.

Aikataulun lisäksi automaatiolla on siis positiivinen vaikutus myös tuotteen toiminnallisuuteen. Lisäksi testaaminen koettiin haastatteluissa edellytykseksi projektin onnistumiselle. Kun testit ovat kattavia, on tuotteessa vähemmän virheitä. Tällä on vaikutusta ennen kaikkea tuotteen toiminallisuudelle, kun tuote toimii tehokkaasti ja luotettavasti. Testaaminen oli myös yksi McCallin laatumallin piirre tuotteen tarkistuksessa (McCall ym., 1977). Jo projektin alkuvaiheessa luodut testit palvelevat myös myöhemmässä vaiheessa elinkaarta. Haastattelussa verrattiinkin testien luomista tuottoisaksi sijoitukseksi. Toimivan PaaS-palvelun kattavat testiominaisuudet ja automaatio parantavat siis projektin laatua myös tuotteen tarkastuksen näkökulmasta. Lopulta tuotteen siirtymä kattaa liikuteltavuuden, yhteentoimivuuden ja uudelleenkäytettävyyden.

Kokonaisuutena aiemmassa vaiheessa hyvin tehty ohjelmisto palvelee tehokkaasti myös siirtymässä. Esimerkiksi Azure DevOpsin testit ja tarkastukset huolehtivat koodin tehokkuudesta. Lisäksi Azure DevOpsin yhteensopivuus useiden eri työkalujen kanssa parantaa tuotteen yhteentoimivuutta jatkossakin.

Budjettiin vaikuttaa moni tekijä. Azure DevOps oli haastateltavien mielestä järkevästi hinnoiteltu. Se ei välttämättä ole halpa suurille organisaatioille, mutta katsottiin silti rahanarvoiseksi sijoitukseksi. Hankintakustannus maksaa itsensä lopulta takaisin, kun työkaluja ei tarvitse hankkia usealta eri toimijalta. Projektin aikataulussa eteneminen vaikuttaa lisäksi budjettiin. Azure DevOpsin koettiin nopeuttavan projektin etenemistä useasta eri syystä. Kun projekti etenee nopeasti, on aikataulussa helpompi pysyä. Jokainen työtunti maksaa organisaatiolle. Tuotteen laatu näkyy osaltaan myös budjetissa ja aikataulussa. Mahdolliset ongelmat voivat vaatia korjauksia ja heikentävät tuotteen tehokkuutta ja luotettavuutta. Toiminnallisuuden lisäksi myös tuotteen tarkastuksessa ja siirtymässä esiin nousevat ongelmat voivat vaikuttaa välillisesti budjettiin.

DevOpsin yksi piirre oli avoimuus ja läpinäkyvyys. Sen katsottiin olevan haastatteluissa tärkeää projektin onnistumisen kannalta. Myös DevOps-pilvipalvelun on tärkeää toteuttaa avoimuutta ja läpinäkyvyyttä. Azure DevOps auttaa kehittäjiä ymmärtämään, missä vaiheessa projekti on. Sen avulla voi myös arvioida muiden töitä, antaa palautetta tai jakaa tietoa esimerkiksi Wikissä. Näillä on joko välillisesti tai välittömästi vaikutusta projektin laatuun. Epätietoisuus lisää odottelua, mikä hidastaa aikataulussa etenemistä. Kun ympäristö on avoin, epätietoisuus vähenee. Haastatteluissa korostettiin myös lopputuotteen bugien määrän vähentymistä läpinäkyvyyden avulla. Useampi henkilö löytää virheet paremmin kuin yksi henkilö. Lisäksi tiedon jakaminen ja hankkiminen johtaa kykyyn tehdä parempia ohjelmistoja. Näin avoimuus ja läpinäkyvyys näkyvät myös tuotteen toiminnallisuudessa, tarkastuksessa ja siirtymässä. Kirjallisuus oli samassa linjassa haastatteluiden kanssa. Esimerkiksi Humble ja Molesky (2011) mainitsivat avoimuuden eduiksi bugien, viiveiden ja turhautumisien vähenemisen (Humble & Molesky, 2011).

Tehokkaalla DevOps-toimintamallin toteuttamisella Azure DevOps tehostaa projektin etenemistä ja parantaa kehittäjien työtehoa. Näillä on positiivinen vaikutus kaikkiin laadun viitekehityksen mitattaviin asioihin. Yhtä tekijää parantava vaikutus näkyy myös muissa tekijöissä. Laadun parantaminen on varsin kiistatonta. Vaikka tutkimuskysymyksessä ei suoraan ilmennyt kyseessä olevan Azure DevOps, voidaan tuloksia yleistää myös muita vastaavia PaaS-tuotteita kehittäessä. Toisaalta myös joitakin ongelmia ilmeni haastatteluissa. Erilaiset ristiriidat ja bugit palvelussa vievät aikaa ja heikentävät kehittäjien työtehoa. Lisäksi monissa Azure DevOpsin työkaluissa todettiin olevan parantamisen varaa.

8.3 Tutkimuksen luotettavuus ja vaikutukset

Teemahaastattelu oli tehokas haastattelumenetelmä tutkielman kannalta. Haastateltavien joukossa oli paljon kokeneita ohjelmistokehittäjiä, joilla oli runsaasti

kerrottavaa. Puolistrukturoidussa rakenteessa haastateltava sai kertoa vapaasti huomioitaan aiheesta. Monelta haastateltavalta tulikin huomioita, jotka eivät suoranaisesti liittyneet tutkimuskysymyksiin, mutta olivat tutkielman kannalta hyviä huomioita. Haastateltavaksi valittu joukko oli myös tutkielman kannalta varsin onnistunut. Nykyisessä tehtävässä kaikilla oli Azure DevOps käytössä ja vähintään vuoden verran kokemusta siitä. Lisäksi monella oli jo aiempaa, pidempää kokemusta DevOpsista. Tämä paikkasi rajoitetta siitä, että haastateltavana oli vain yksi tiimi, joka toimi yhden tuotteen kehityksessä. Monella oli kokemusta myös muista yrityksistä ja erilaisista projekteista. Haastatteluiden pituudessa oli vaihtelua ja osa oli varsin lyhyitä. Toisaalta kokeneet haastateltavat pystyivät esittämään tietoa nopeasti ja ilman merkittäviä pohdintataukoja.

Hirsjärvi ja Hurme (2022) esittelivät, kuinka tutkimusta voidaan arvioida reliabiliteetin ja validiteetin avulla. Tutkimus on luotettava, jos samalla tutkimusmenetelmällä samasta aiheesta päädytään samaan tulokseen. Validiteetti taas kertoo, tutkiiko tutkimusmenetelmä todella sitä, mitä sen on ajateltu tutkivan. (Hirsjärvi ja Hurme, 2022.) Käytännössä reliabiliteetin toteaminen vaatisi sen, että aihetta on tutkittu aiemminkin. Lisäksi puoliavoimessa haastattelussa keskustelu elää herkästi ja sitä on vaikeaa tai lähes mahdotonta toistaa uudestaan tismalleen samalla tavalla. Tutkimuksen luotettavuudesta kuitenkin antoi viitteitä se, että haastateltavien lausunnot aiheesta olivat varsin samanlaisia. Lisäksi haastattelut olivat linjassa kirjallisuuden kannalta. Käytännössä vanhempikin DevOps-kirjallisuus ja tutkimukset jatkuvasta integraatiosta ja julkaisusta olivat linjassa nykyajan työelämän kanssa. Validiteetin kannalta yhtenä riskinä oli se, että haastattelut keskittyivät osin DevOpsiin, eikä joissakin kysymyksissä mainittu Azure DevOpsia ollenkaan. Toisaalta tutkielmasta ilmeni, että DevOpsiin liittyvät opit pätevät hyvin pitkälti myös Azure DevOpsiin. DevOpsiin liittyvästä tutkimuksesta voidaan vetää johtopäätöksiä myös Azure DevOpsiin liittyen. Litteroinnin haasteena oli puoliavoin haastattelu ja siitä kertyneen datan suuri määrä. Yksittäisetkin huomiot pyrittiin nostamaan esille. Osa niistä oli merkittäviä tutkielman kannalta. Nauhoitteita litteroidessa riskinä on, että keskittyminen herpaantuu ja jotain jää huomaamatta. Vastaavissa tilanteissa kuunneltiin kohta kaiken varalta vielä toiseen kertaan.

Yleistämisen kannalta tutkielman tulosten etuna on se, että tulokset ovat linjassa aiemman kirjallisuuden kanssa. Lisäksi haastateltavien lausunnot olivat hyvin samankaltaisia. Tutkielmassa luotu oma laadun viitekehys kuitenkin vaatii yleistettävyyttä arvioidessa huomiota. Viitekehyksen sisältämät viisi mittaria voivat pudottaa joitakin näkökulmia pois tutkimuksesta. Samoin pelkästään viiden mittarin tarkastelu ei välttämättä ota huomioon laajempaa kontekstia, kuten organisaatiota ja sen kultuuria ja liiketoimintaa. Toinen haaste tutkielman kannalta on se, että DevOpsia voidaan toteuttaa eri organisaatioissa eri tavalla. Lisäksi DevOpsia toteuttavia PaaS-palveluita on muillakin kuin Microsoftilla ja niiden toteutustapa DevOpsille voi olla erilaista. Eri palveluntarjoajaa käyttävää organisaatiota haastatellessa tulokset voisivat olla hieman erilaisia. Toisaalta myös näissä tilanteissa jatkuva toimitus, integraatio ja automaatio ovat keskeisessä

osassa, mikäli DevOpsia halutaan toteuttaa oikein. Siksi tutkimustuloksia voidaan varauksin yleistää myös muihin palveluntarjoajiin.

Rajoitteina tutkielmassa olikin haastateltavien joukko ja tutkittava tuote. Haastateltavat olivat kaikki samassa organisaatiossa ja työskentelivät saman tuotteen parissa. Jatkotutkimuksissa olisi hyödyllistä laajentaa haastateltavien joukkoa myös muihin organisaatioihin. Lisäksi mahdollisissa jatkotutkimuksissa olisi hyvä tarkastella muidenkin palveluntarjoajien tuotteita. DevOps-työkaluja tarjoavat esimerkiksi Google ja Amazon AWS. Tällä keinolla saataisiin tutkimuskysymyksiin lisää vastauksia siitä, miten DevOpsia voidaan toteuttaa pilvessä ja mitä vaikutuksia sillä on. Muiden palveluntarjoajien toteutukset voivat olla käytännössä erilaisia, vaikka niidenkin on toteutettava DevOpsin toimintatapoja.

Tutkielma hyödyttää organisaation esihenkilöitä ja tiiminvetäjiä. Se antaa kuvauksen siitä, miten DevOpsin käytänteitä kannattaa toteuttaa ja miten avoimuutta ja läpinäkyvyyttä voidaan lisätä organisaatiossa. Esimerkiksi Azure DevOpsia harkitseva organisaatio saa tutkielmasta kattavaa ja hyödyllistä tietoa. Ylipäätään tutkielma paikkaa aukon Azure DevOpsiin liittyvästä tutkimuksesta ja tuloksia voidaan yleistää jossain määrin myös muiden palveluntarjoajien palveluihin. DevOpsin toimintaperiaatteet ovat samat muillekin. Lisäksi tutkielma antaa kuvauksen siitä, miten DevOpsia kannattaa toteuttaa pilvessä. DevOpsin suuren työkalu- ja periaatemäärän vuoksi yhden kaiken kokoavan palvelun luominen ei ole helppoa. Tutkielma antaa suuntaa sille, mitä tavalliset ohjelmistokehittäjät haluavat ja mitkä asiat ovat tärkeitä huomioida. Valtava työkalujen määrä ei ole välttämättä paras vaihtoehto, jos se tarkoittaa hankalasti opittavaa kokonaisuutta ja suurta bugien määrää.

9 YHTEENVETO

Haasteena tutkielman kannalta oli se, että Azure DevOpsista tai vaihtoehtoisesti muista DevOpsia toteuttavista PaaS-pilvipalveluista oli vasta vähän aiempaa tieteellistä tutkimusta. Tutkielmaan otettiin mukaan perinteiseen DevOps-toimintamalliin kohdistuvaa tutkimusta. Osa tutkimuksista oli tehty kauan ennen kuin Azure DevOps saavutti nykyisen nimen ja toiminnot. Tätä perusteltiin sillä, että esimerkiksi jatkuva toimitus ja integraatio, automaatio ja avoimuuden ja läpinäkyvyyden kulttuuri ovat merkittävänä tekijänä myös DevOpsia toteuttavissa PaaS-palveluissa. DevOpsia verrattiin perinteisiin sovelluskehitysmalliin ja sen vaikutusta organisaation kulttuuriin käytiin läpi. Samalla tuotiin ilmi tutkielman kannalta keskeisiä termejä, jotka ovat tärkeä tietää myöhemmässäkin vaiheessa. Lisäksi koska Azure DevOps on PaaS-pilvipalvelu, käytiin pilvipalveluiden ominaisuuksia ja luokitteluita läpi. Ensimmäisessä tutkimuskysymyksessä pohdittiin DevOpsia toteuttavan PaaS-pilvipalvelun luomista. Vaikka tutkielma pyörii pitkälti DevOpsin ympärillä, on kyseisen PaaS-pilvipalvelun myös täytettävä pilvipalvelun piirteet ja vaatimukset.

Haastatteluvaiheessa ilmeni, että aiempi DevOps-kirjallisuus on hyvin linjassa nykyajan työelämän kanssa. Haastatteluissa ilmeni paljon samanlaisia huomioita kuin kirjallisuudessa ja DevOpsin työkaluja hyödynnettiin tehokkaasti. Ylipäätään DevOpsin periaatteet koettiin erittäin positiiviseksi. Testaus ja automaatio todettiin suorastaan elinehdoksi nykyajan ohjelmistokehitysyritykselle. Myös jatkuva integraatio, toimitus ja julkaisu mahdollistivat tehokkaan, joustavan ja sujuvan julkaisutahdin. DevOpsin yleistymiselle vaikutti olevan selkeät syyt. Ei ole ihme, että perinteisestä vesiputousmallista ja siiloutuneesta organisaatiorakenteesta on siirrytty eteenpäin. Edut ovat selviä kaikille osapuolille suurimmassa osassa ohjelmistokehitysprojekteja. Kirjallisuuskatsauksessa koottiin yhteenvedoksi Azure DevOpsin hyötyjä ja haasteita osin pelkän DevOps-kirjallisuuden pohjalta. Se kokoaa yhteen valtavan valikoiman työkaluja, tukee lukuisia eri ohjelmointikieliä ja lisäosia, hyödyntää tehokkaasti tekoälyä ja koneoppimista, tukee testaamisessa ja tekee projektin toteuttamisesta läpinäkyvää. Haastatteluissa päästiin samaan lopputulokseen myös Azure DevOpsinhyötyjen kohdalla. Myös haasteissa oli osittaista yhteneväisyyttä. Esimerkiksi haastatteluissa

ilmeni, että Azure DevOpsin hahmottaminen ja opetteleminen vie aikaa. DevOps-toimintamalli sisältää niin paljon työkaluja, että niiden toteuttaminen pilvessä saa aikaan ison kokonaisuuden. Hahmottamisen lisäksi ongelmana massiivisessa palvelussa voi olla myös bugit, joiden korjaamisessa kestää pitkään. Tämä vaivasi osaa haastateltavista. Lisäksi alun perin Sonin (2017) tutkimuksen pohjalta todettiin, ettei Azure DevOps toimi saumattomasti kaikkien viitekehyksien kannalta. Haastatteluissa tulikin ilmi käytännön esimerkkinä SAFe-viitekehys, jota haastateltavassa organisaatiossa käytetään, mutta joka ei ihan sovi Azure DevOpsin kanssa yhteen. Kirjallisuudessa myös todettiin, että nopea julkaisutahti saa kehittäjät helposti hätiköimään työssään. Haastatteluissa kaksi henkilöä toi saman haasteen ilmi, mutta kokonaisuutena riski katsottiin pieneksi. Sen sijaan kirjallisuudessa vastaan tullutta muutosvastaisuutta ei haastatteluissa ilmennyt. Tämä johtuu todennäköisesti siitä, että Azure DevOps oli ollut jo pitkään käytössä. Lisäksi monella oli aiempaa kokemusta DevOpsista ja Azure DevOps oli tälle luonnollinen jatkumo. Käytännössä yhtenä yhteenvetona tutkielmasta voisi nostaa sen, että DevOps-toimintamalliin keskittyvää kirjallisuutta voidaan hyödyntää lähdemateriaalina myös Azure DevOpsia tai muita vastaavia DevOpsiin keskittyviä PaaS-pilvipalveluita tutkiessa. Samoin voidaan todeta, että DevOpsia toteuttava PaaS-pilvipalvelu parantaa projektin laatua, tekee tiimin työskentelystä läpinäkyvämpää ja mielekkäämpää ja voi nopeuttaa projektin valmistumista merkittävästi. Sen on oltava verkossa saavutettavisssa mistä vain ja käyttökatkoksien on oltava erittäin harvinaisia. Azure DevOps onnistuu tässä.

Käytännössä tutkielma antoi katsauksen siitä, miten DevOpsia ja sen työkaluja ja käytänteitä toteutetaan nykyajan työelämässä. Lisäksi se toi ilmi, miten tärkeäksi monet käytänteet, kuten automaatio ja jatkuva toimitus ja integraatio katsottiin. Lisäksi siiloutuneen kulttuurin voidaan katsoa olevan suurilta osin takana päin, sillä avoimuus ja läpinäkyvyys katsottiin haastatteluissa yksimielisen hyväksi asiaksi. Tutkielma hyödyttää organisaation esihenkilöitä ja tiiminvetäjiä. Se antaa kuvauksen siitä, miten DevOpsin käytänteitä kannattaa toteuttaa ja miten avoimuutta ja läpinäkyvyyttä voidaan lisätä organisaatiossa. Esimerkiksi Azure DevOpsia harkitseva organisaatio saa tutkielmasta kattavaa ja hyödyllistä tietoa. Ylipäätään tutkielma paikkaa aukon Azure DevOpsiin liittyvästä tutkimuksesta ja tuloksia voidaan yleistää jossain määrin myös muiden palveluntarjoajien palveluihin. DevOpsin toimintaperiaatteet ovat samat muillekin. Lisäksi tutkielma antaa kuvauksen siitä, miten DevOpsia kannattaa toteuttaa pilvessä. DevOpsin suuren työkalu- ja periaatemäärän vuoksi yhden kaiken kokoavan palvelun luominen ei ole helppoa. Tutkielma antaa suuntaa sille, mitä tavalliset ohjelmistokehittäjät haluavat ja mitkä asiat ovat tärkeitä huomioida. Valtava työkalujen määrä ei ole välttämättä paras vaihtoehto, jos se tarkoittaa hankalasti opittavaa kokonaisuutta ja suurta bugien määrää.

LÄHTEET

Akbar, M. A., Sang, J., Khan, A. A., Fazal-E-Amin, Nasrullah, Shafiq, M., Hussain, S., Hu, H., Elahi, M., & Xiang, H. (2018). Improving the Quality of Software Development Process by Introducing a New Methodology–AZ-Model. *IEEE Access*, *6*, 4811–4823. https://doi.org/10.1109/ACCESS.2017.2787981

Almeida, F., Simões, J., & Lopes, S. (2022). Exploring the Benefits of Combining DevOps and Agile. *Future Internet*, *14*(2), Article 2. https://doi.org/10.3390/fi14020063

Amazon. (2023). *DevOps – Amazon Web Services (AWS)*. Amazon Web Services, Inc. https://aws.amazon.com/devops/

Angara, J., Prasad, S., & Sridevi, G. (2020). DevOps Project Management Tools for Sprint Planning, Estimation and Execution Maturity. *Cybernetics and Information Technologies*, *20*(2), 79–92. https://doi.org/10.2478/cait-2020-0018

Arora, T. (2016). *Microsoft Team Foundation Server 2015 Cookbook*. https://web-s-ebscohost-com.ezproxy.jyu.fi/ehost/ebookviewer/ebook/ZTAwMHh3d19fMTE2MzgzN19fQU41?sid=2ba07b76-7b0b-4201-9b48-4a649f957cc1@redis&vid=0&format=EB&lpid=lp_v&rid=0

Avinash, G. (2022). APPLYING AZURE to AUTOMATE DEVOPS FOR SMALL ML SMART SENSORS. *International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/IRJMETS32238

Azure DevOps Labs. (2023). *Azure DevOps Hands-On Labs*. https://azuredevopslabs.com/default.html

Braglia, M., Gabbrielli, R., & Marrazzini, L. (2020). Rolling Kanban: A new visual tool to schedule family batch manufacturing processes with kanban. *International Journal of Production Research*, *58*(13), 3998–4014. https://doi.org/10.1080/00207543.2019.1639224

DevOps. (2023). *DevOps on Google Cloud Platform – Complete Guide*. https://www.xenonstack.com/insights/devops-google-cloud-platform

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, *33*(3), 94–100. https://doi.org/10.1109/MS.2016.68

Eurostat. (2023). *Cloud computing – Statistics on the use by enterprises*. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Cloud_computing_-_statistics_on_the_use_by_enterprises

Gehman, C. (2018). *What Is Team Foundation Server?* Perforce Software. https://www.perforce.com/blog/vcs/what-team-foundation-server

Ghanbari, H. (2018). Omission of Quality Software Development Practices: A Systematic Literature Review. *ACM Computing Surveys*, *51*(2), 1–27. https://doi.org/10.1145/3177746

Goldkuhl, G. (2019). The Generation of Qualitative Data in Information Systems Research: The Diversity of Empirical Research Methods. *Communications of the Association for Information Systems*, *44*, 28. https://doi.org/10.17705/1CAIS.04428

Guthrie, S. (2021). *MLOps with Azure Machine Learning*. Microsoft Resources.

Hirsjärvi, S., & Hurme, H. (2022). *Tutkimushaastattelu: Teemahaastattelun teoria ja käytäntö*. Yliopistopaino.

Hoda, R., Salleh, N., & Grundy, J. (2018). The Rise and Evolution of Agile Software Development. *IEEE Software*, *35*(5), 58–63. https://doi.org/10.1109/MS.2018.290111318

Hoffman, D. (2020). *Comparing Waterfall vs. Agile vs. DevOps methodologies | TechTarget*. Software Quality. https://www.techtarget.com/searchsoftwarequality/opinion/DevOps-vs-waterfall-Can-they-coexist

Humble, J., & Molesky, J. (2011). *Why Enterprises Must Adopt Devops to Enable Continuous Delivery*. *Cutter IT Journal VOL. 24, NO. 8*

Jacobson, I. (2022). Better Scrum through Essence. *Software, Practice & Experience*, *52*(6), 1531–1540. https://doi.org/10.1002/spe.3070

Khan, M., Jumani, A., Mahar, F., Siddique, W., & Shaikh, A. (2020). Fast Delivery, Continuously Build, Testing and Deployment with DevOps Pipeline Techniques on Cloud. *Indian Journal of Science and Technology*, *13*, 552–575. https://doi.org/10.17485/ijst/2020/v13i5/148983

Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review. *IEEE Access*, *10*, 14339–14349. https://doi.org/10.1109/ACCESS.2022.3145970

Krill, P. (2018). Microsoft Azure Devops Services puts devops in the cloud. *Networks Asia*. https://www.proquest.com/docview/2102074334/citation/69601872D5364DF3PQ/1

Laine, K. (2021, elokuuta 31). *Tilastokatsaus: Pilvipalveluiden käyttö Suomessa ja maailmalla*. FiCom. https://ficom.fi/ajankohtaista/uutiset/tilastokatsaus-pilvipalveluiden-kaytto-suomessa-ja-maailmalla/

Laukkarinen, T., Kuusinen, K., & Mikkonen, T. (2018). Regulated software meets DevOps. *Information and Software Technology*, *97*, 176–178. https://doi.org/10.1016/j.infsof.2018.01.011

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). *Software Development Life Cycle AGILE vs Traditional Approaches*. nternational Conference on Information and Network Technology (ICINT 2012) IPCSIT vol. 37 (2012) © (2012) IACSIT Press, Singapore

Lee, M.-C. (2014). *Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance*. National Kaohsiung University of Applied Sciences.

Leite, L. (2020). A Survey of DevOps Concepts and Challenges. *ACM Computing Surveys*, *52*(6), 1–35. https://doi.org/10.1145/3359981

McCall, J., Richards, P., & Walters, G. (1977). *Factors in Software Quality. Volume I. Concepts and Definitions of Software Quality*. Administration and Management Computer Programming and Software. https://apps.dtic.mil/sti/citations/ADA049014

McConnell, S. (1996). *Rapid development: Taming wild software schedules*. Microsoft Press.

Mell, P., & Grance, T. (2010). The NIST Definition of Cloud Computing. *Communications of the ACM*, *53*(6), 50–50.

Microsoft. (2023a). *Application lifecycle management (ALM) with Microsoft Power Platform – Power Platform*. Https://Learn.Microsoft.Com/En-Us/Power-

Platform/Alm/. https://learn.microsoft.com/en-us/power-platform/alm/

Microsoft. (2023b). *Azure DevOps Services | Microsoft Azure*. https://azure.microsoft.com/en-us/products/devops

Mohammad, S. M. (2017). *DevOps automation and Agile methodology*. *International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.5, Issue 3, pp.946-949, August-2017, Available at SSRN: https://ssrn.com/abstract=3655581*

Mohammad, S. M. (2018). *Streamlining DevOps automation for Cloud applications*. *International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.6, Issue 4, pp.955-959, October-2018, Available at SSRN: https://ssrn.com/abstract=3655574*

Montesdioca, G. P. Z., & Maçada, A. C. (2015). *Quality Dimensions of the DeLone-McLean Model to Measure User Satisfaction: An Empirical Test on the Information Security Context*. *2015*, 5010–5019. https://doi.org/10.1109/HICSS.2015.593

Noorani, N., Zamani, A., Alenezi, M., Shameem, M., & Singh, P. (2022). Factor Prioritization for Effectively Implementing DevOps in Software Development Organizations: A SWOT-AHP Approach. *Axioms*, *11*(10), 498. https://doi.org/10.3390/axioms11100498

Palonen, M., & Kylmä, J. (2022). Avoin haastattelu ja teemahaastattelu aineistonkeruumenetelminä laadullisessa hoitotieteellisessä tutkimuksessa. *Hoitotiede*, *34*(4), 281–294.

PeerSpot. (2023). *What is your primary use case for Microsoft Azure DevOps?* PeerSpot. https://www.peerspot.com/questions/what-is-your-primary-use-case-for-microsoft-azure-devops

Redhat. (2020). *What is application lifecycle management (ALM)?* Https://Www.Redhat.Com/En/Topics/Devops/What-Is-Application-Lifecycle-Management-Alm. https://www.redhat.com/en/topics/devops/what-is-application-lifecycle-management-alm

Riddle, W. E., & Fairley, R. E. (2012). *Software Development Tools*. Springer Science & Business Media.

Rossberg, J. (2019). *Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics*. Apress.

Rossel, S. (2017). *Continuous Integration, Delivery, and Deployment: Getting Started with the Processes and the Tools to Continuously Deliver High-Quality Software*. Packt Publishing, Limited. http://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?docID=5117840

Schaller, A. (2016). *DevOps transformation challenges facing large scale legacy systems*. https://www.proquest.com/openview/cbc3c0fc15cd39d0d56b92c5169c95bc/1?pq-origsite=gscholar&cbl=18750

Seremet, Z., & Rakic, K. (2021). Best Approach to Security in Azure Devops. Teoksessa B. Katalinic (Toim.), *DAAAM International Scientific Book* (1. p., Vsk. 20, ss. 223–230). DAAAM International Vienna. https://doi.org/10.2507/daaam.scibook.2021.18

Shah, S. K., & Corley, K. G. (2006). Building Better Theory by Bridging the Quantitative–Qualitative Divide*. *Journal of Management Studies*, *43*(8), 1821–1835. https://doi.org/10.1111/j.1467-6486.2006.00662.x

Shahin, M., Ali Babar, M., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, *5*, 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

Soni, M. (2017). *Implementing DevOps with Microsoft Azure: Accelerate and Automate Build, Deploy, and Management of Applications to Achieve High Availability*. Packt Publishing. https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=1513358&site=ehost-live

Thomas, G. (2008). Success in IT projects: A matter of definition? *International Journal of Project Management*, *26*(7), 733. https://doi.org/10.1016/j.ijproman.2008.06.003

Tilastokeskus. (2022, joulukuuta 20). *Pilvipalveluita käytti 81 % yrityksistä vuonna 2022 – Tilastokeskus*. https://www.stat.fi/julkaisu/cktvztyy82z790b55dz6j23q3

Trudova, A., Dolezel, M., & Buchalcevova, A. (2020). Artificial Intelligence in Software Test Automation: A Systematic Literature Review: *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering*, 181–192. https://doi.org/10.5220/0009417801810192

Turetken, O., Stojanov, I., & Trienekens, J. J. M. (2017). Assessing the adoption level of scaled agile development: A maturity model for Scaled Agile Framework. *Journal of Software: Evolution and Process*, *29*(6), e1796. https://doi.org/10.1002/smr.1796

Tüzün, E. (2019). Adopting integrated application lifecycle management within a large-scale software company: An action research approach. *The Journal of Systems and Software*, *149*, 63–82. https://doi.org/10.1016/j.jss.2018.11.021

Vuppalapati, C., Ilapakurti, A., Chillara, K., Kedari, S., & Mamidi, V. (2020). Automating Tiny ML Intelligent Sensors DevOPS Using Microsoft Azure. *2020 IEEE International Conference on Big Data (Big Data)*, 2375–2384. https://doi.org/10.1109/BigData50022.2020.9377755

Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). DevOps and Its Practices. *IEEE Software*, *33*(3), 32–34. https://doi.org/10.1109/MS.2016.81

Zyphur, M. J., & Pierides, D. C. (2017). Is Quantitative Research Ethical? Tools for Ethically Practicing, Evaluating, and Using Quantitative Research. *Journal of Business Ethics: JBE*, *143*(1), 1–16. https://doi.org/10.1007/s10551-017-3549-8