Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2021

Sanna Herrala

Pilvipalveluiden tuotekehitysprosessi

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2021 | 44 sivua

Sanna Herrala

Pilvipalveluiden tuotekehitysprosessi

Erilaisten pilvipalveluiden määrä on kasvanut valtavasti, mutta niiden tuotekehitystä tarkastelevaa tutkimusta on vain vähän. Tässä opinnäytetyössä tutkittiin, miten pilvipalvelun tuotekehitys eroaa perinteisemmästä konesaliin kehittämisestä. Aihetta lähestyttiin case-esimerkin kautta tutustumalla sote-alalle pilvipalveluna kehitetyn palvelun tarvearvioinnin (PATA) ratkaisun kehitykseen. Työssä haastateltiin kehitystyössä mukana olleita henkilöitä ja vastaukset analysoitiin pilvipalveluiden ja tuotekehityksen lähdekirjallisuuden perusteella. Vastauksissa kuvailtuja kehitysmetodeja verrattiin myös kirjallisuudessa pilvipalveluiden kehitykselle kuvailtuihin hyviin työtapoihin, niin kutsuttuihin pilvinatiiveihin kehitystapoihin. Työn on tarkoitus antaa selkeä kuvaus pilvipalveluiden kehityksen ominaispiirteistä, jota voidaan käyttää tulevaisuuden kehitystyössä. Työ keskittyy alustaratkaisun kehittämiseen Microsoftin pilvipalveluja käyttämällä sekä aihealueisiin, jotka saattavat olla erityisen haastavia pilvikehitystä vasta aloitteleville organisaatioille

Työn lopputulos oli, että pilvi- ja konesalikehityksessä eri kehitysvaiheiden välillä ei ole suurta eroa. Pilvinatiivit kehitystavat tähtäävät kehitysprosessin automatisaatioon ja itsenäisiin ohjelmistomoduuleihin, jolloin pilvikehitys hyötyy erityisesti ketteristä kehitysmenetelmistä. PATA-ratkaisun kehityksessä on käytetty ketterää kehitystä, mutta pilvinatiivien työtapojen käyttöön ei ole vielä siirrytty. Pilvipalveluilla todettiin olevan erityisiä etuja skaalautuvuudessa ja infrastruktuurin hallinnoinnin tarpeen vähyydessä. Toisaalta pilvikehitykseen siirtymisen todettiin vaativan uusien asioiden opettelemista, ja määrittely sekä lisenssien hankinnan tarkka suunnittelu nousivat tärkeiksi aiheiksi.

Asiasanat:

pilvipalvelut, tuotekehitys, ohjelmistokehitys, pilvinatiivi

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2021 | 44 pages

Sanna Herrala

Product development process of cloud services

While the amount of different cloud services available has risen, there is a lack of studies on the product development process of these services. The objective of this thesis was to study how the product development of cloud services differs from that of on-premises ones. To achieve this objective, a case study of a cloud platform service called PATA for the healthcare and social welfare sector was studied. People involved in the development process of the service were interviewed and their answers were analysed in the light of literature on cloud services and product development. The development methods described in the interviews were also compared to cloud native design patterns, a set of development tools aimed at being especially suited for the development of cloud services. This thesis aims to give a clear description of the special features of cloud development that can be used to aid in future cloud product development.

The conclusion of this thesis is that cloud- and on-premises development do not significantly differ in the flow between different stages of the development process. Cloud native design pattens aim to automate much of the development process and to produce software composed of independent modules. Because of this, agile development is especially suited for the development of cloud products. Agile development has been used in the development of the PATA-platform, but the production team has not yet taken up cloud native design. Cloud services were found to be beneficial due to their scalability and reduced need for infrastructure management. On the other hand, the move into cloud development was found to require a steep learning curve. Crafting requirements and planning licencing were identified as especially important parts of the process.

Keywords:

cloud services, product development, software development, cloud native

Sisältö

Liitteet

Käytetyt lyhenteet

AD	Active directory, Microsoftin Windowsin käyttäjätietokanta ja hakemistopalvelu (Tolvanen, 2011)
AWS	Amazon web service, pilvipalveluiden tarjoaja
CI/CD	Jatkuva integraatio ja jatkuva toimitus (engl. continuous integration and continuous deployment). Sovelluskehitystapa, jossa käytetään automaattisia testejä ja julkaisua nopeuttamaan sovelluusjulkaisujen tekoa (Vettor & Smitth, 2021).
DevOps	Sovelluskehityksen työtapa, jolla pyritään lyhentämään aikaa kehityksen ja julkaisun välillä (Erich ym., 2017).
IaaS	Infrastructure-as-a-service (infrastruktuuri palveluna), pilvipalveluiden palvelumalli, jossa palvelun ostaja voi hallinnoida monia resursseja (Ranger, 2018).
IaC	Infrastructure-as-code (infrastruktuuri koodina) tapa automatisoida pilvipalveluiden resurssien käyttöönottoa ja tuhoamista (Vettor & Smitth, 2021).
PaaS	Platform-as-a-service (alusta palveluna), pilvipalveluiden palvelumalli, jossa palvelun ostaja voi hallinnoida osaa resursseista (Ranger, 2008).
PATA	Palvelun tarvearviointi ja palveluun ohjaus
SaaS	Software-as-a-service (ohjelmisto palveluna), pilvipalveluiden palvelumalli, jossa käyttäjälle tarjotaan käyttöoikeus pilvessä sijaitsevaan sovellukseen, mutta ei hallinnointioikeuksia (Mell & Grance, 2011).

1 Johdanto

Käytettävät verkkopalvelut sijaitsevat yhä useammin pilvessä. Microsoft, Google ja Amazon ovat nousseet suuriksi pilvipalvelujen tarjoajiksi, ja yhä useammat yritykset ovat siirtämässä kehitystä omista konesaleistaan näille pilvialustoille. Tässä työssä on tarkoitus case-esimerkin kautta tutkia pilvipalvelun tuotekehitysprosessia. Opinnäytetyön toimeksiantaja on sosiaali- ja terveyspalveluiden kehityksen ratkaisuja tuottava 2M-IT. Case-esimerkkinä on 2M-IT:n kolmelle sosiaali- ja terveysalan toimijalle kehittämä palvelun tarvearviointi ja palveluun ohjaus -ratkaisu (PATA).

Pilvipalveluista sekä tuotekehityksestä löytyy kattavasti kirjallisuutta, mutta varsinaista pilvipalveluiden tuotekehitystä ei ole tutkittu paljon. Pilvipalveluiden kehityksestä kertova kirjallisuus keskittyy usein viitekehyksiin, joilla voitaisiin automatisoida kehitysprosessi. Kao ym. (2014) esittelivät viitekehyksen pilvisovellusten kehitykseen ja testaukseen, mutta eivät näyttäneet, miten viitekehyksen komponentit toteutettaisiin, tai osoittaneet, että viitekehys toimisi käytännössä. Benfenakti ym. (2017) ovat myös esitelleet metodologian, joka kattaa pilvipalveluiden kehityksen koko elinkaaren. Julkaisussa keskityttiin erityisesti palvelun vaatimusmäärittelyyn ja esiteltiin järjestelmä, joka automaattisesti vaatimusmäärittelyn perusteella ehdottaa parhaita ratkaisuja palvelulle sekä automatisoi osan kehitysprosessista. Kumpikaan työ ei kuitenkaan anna kuvaa pilvipalvelun tuotekehitysprosessista, vaan molemmat keskittyvät kehityksen teknisien osien automatisointiin.

Tämän opinnäytetyön tarkoitus on kuvata pilvipalvelun tuotekehitysprosessi. Työn on tarkoitus vastata kysymyksiin:

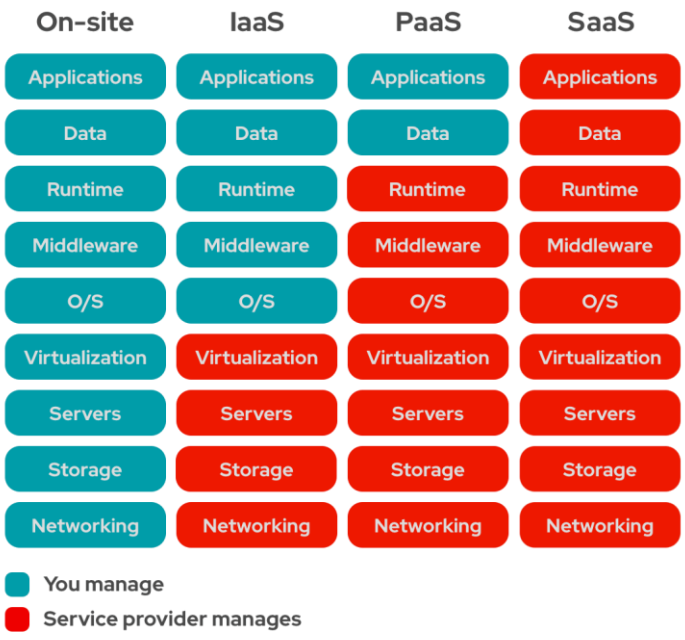
1. Eroaako pilvipalvelun tuotekehitysprosessi konesaliin kehitettävien palveluiden tuotekehitysprosessista?
2. Mitä erityispiirteitä pilvipalvelun tuotekehityksessä on verrattuna perinteiseen tuotekehitysprosessiin?

Työn on tarkoitus tuottaa kokonaisvaltainen kuva tuotekehitysprosessista, jossa käytetään Microsoftin pilvipalveluita. Toimeksiantaja voi käyttää tuotekehitysprosessin kuvausta pohjana tulevaisuuden kehitystöille. Aihetta tutkitaan tutustumalla ensin pilvipalveluista ja tuotekehityksestä kertovaan kirjallisuuteen, minkä jälkeen haastatellaan PATA-hankkeessa mukana olleita henkilöitä palvelun kehitystyöstä. Käytettävä case-esimerkki asettaa työlle tiettyjä rajauksia. Koska PATA-ratkaisu ei ole vielä tuotannossa, rajautuu ratkaisun julkaiseminen ja julkaisun jälkeinen aika työn ulkopuolelle. Pilvipalveluita käsiteltäessä painotetaan erityisesti Microsoftin tuottamia pilvipalveluita, sillä ne ovat käytössä PATA-ratkaisussa.

Työn luvuissa 2 ja 3 tehdään katsaus pilvipalveluiden eri tyyppeihin sekä Microsoftin pilvipalveluihin. Luku 4 käsittelee tuotekehitystä, minkä jälkeen luvussa 5 käydään läpi ohjelmistokehitystä sekä ketterää kehitystä. Luvussa 6 käsitellään pilvipalveluiden tuotekehitykselle ominaisia työtapoja, minkä jälkeen luvussa 7 esitellään case-esimerkki PATA-ratkaisu sekä tutkimuksen metodit. Työn lopussa luvuissa 8 ja 9 esitellään tutkimuksen tulokset ja niistä tehdyt johtopäätökset sekä esitetään tutkimuksen rajoituksia ja jatkomahdollisuuksia luvussa 10.

2 Pilvipalvelut

Perinteisesti IT-palvelut ja sovellukset ovat sijainneet yritysten omissa palvelinkeskuksissa ja laitteiston sekä ohjelmiston päivitys on ollut yrityksen omalla vastuulla. Pilvipalveluissa tietoa ja ohjelmia voidaan tallentaa palveluntarjoajien palvelimille yrityksen palvelimien tai tietokoneiden sijaan. (Kangasniemi & Lintulahti, 2017.) Mell ja Grance (2001) ovat eritelleet pilvipalveluille ominaisia piirteitä. Näitä piirteitä ovat, että palvelut ovat jatkuvasti saatavilla, niiden ominaisuudet ovat käytettävissä erilaisilla laitteilla internetin kautta ja resursseja tarjotaan käyttäjille dynaamisesti perustuen tarpeeseen. Isoimpia pilvipalvelujen tarjoajia ovat Googe Cloud Platform, Amazon web services (AWS) sekä Microsoft Azure. Pilvipalvelut voidaan jakaa erilaisiin palvelumalleihin sen perusteella, kuinka paljon hallinnoitavaa käyttäjällä on (**Error! Reference source not found.**). Myös pilvipalveluiden toimitusmallit voidaan jakaa kolmeen kategoriaan: julkinen, yksityinen ja hybridi. (Ranger, 2018.)



Kuva 1. Yhteenveto pilvipalveluiden eri palvelumalleista (RedHat, 2020).

Yrityksille pilvipalveluiden joustava skaalautuvuus on suuri etu. Yritysten ei tarvitse panostaa omiin konesaleihin, vaan ne voivat ostaa resursseja pilvipalvelujen tuottajilta. Nämä resurssit voivat olla esimerkiksi sovelluksia, muistia tai palvelintilaa. Yritykset maksavat vain käyttämistään resursseista, ja jos palvelun kysyntä nousee, on lisäresurssien hankita helpoimmillaan automaattista. (Ranger, 2018.)

Pilvipalveluissa on myös huonot puolensa. Jos palvelun tarvitsemien resurssien määrä on tasainen ja ennustettavissa, saattaa olla halvempaa käyttää omaa infrastruktuuria. Jo olemassa olevien sovellusten vienti pilveen saattaa olla myös monimutkaista ja kallista. (Ranger, 2018.) Mellin ja Grancen (2001) mukaan riippuen palvelusta käyttäjällä ei välttämättä ole tietoa tai mahdollisuutta vaikuttaa siihen missä hänen datansa on tallennettuna. Hänellä saattaa kuitenkin olla mahdollisuus vaikuttaa datan sijaintiin yleisemmällä tasolla valitsemalla tietty maa tai datakeskus. Myös palveluiden saatavuus ainoastaan internetyhteyden kautta saattaa aiheuttaa ongelmia.

2.1 Pilvipalveluiden palvelumallit

Pilvipalveluiden palvelumallit voidaan jakaa kolmeen kategoriaan sen perusteella, kuinka paljon palveluntarjoaja hallitsee palvelua verrattuna palvelun hankkijaan (**Error! Reference source not found.**). Ratkaisuissa, joissa palvelu sijaitsee yrityksen omassa konesalissa, yritys on vastuussa kaikesta hallinnoinnista.

2.1.1 Infrastruktuuri palveluna (IaaS)

IaaS (Infrastructure-as-a-Service) on palvelumalleista laajin: se tarjoaa mahdollisuuden vuokrata tietojenkäsittelyn resursseja kuten palvelintilaa, virtuaalikoneita sekä prosessointia. Käyttäen näitä resursseja yritykset voivat julkaista omat ratkaisunsa pilveen. (Ranger, 2018.) Käyttäjät eivät voi vaikuttaa kaikkeen pilvi-infrastruktuuriin, mutta he voivat hallinnoida ostamiaan resursseja

(Mell & Grance, 2011). IaaS-malli mahdollistaa testaus- ja tuotantoympäristöjen pystyttämisen helposti ja ostettuja palveluita on mahdollista skaalata tai lopettaa nopeasti tarpeen vaatiessa. Kääntöpuolena IaaS-mallissa on infrastruktuurin resurssien jako palvelun muiden käyttäjien kanssa, palveluntarjoajan mahdolliset turvallisuusongelmat sekä palvelukatkot. (RedHat, 2020.) Isot pilvipalveluiden tarjoajat, kuten AWS, Google Cloud Platform ja Microsoft Azure ovat esimerkkejä IaaS-palveluista.

2.1.2 Alusta palveluna (PaaS)

PaaS (Platform-as-a-Service) -ratkaisuissa käyttäjälle tarjotaan työkaluja, kuten tietokantojen hallintaa, väliohjelmistoja ja kehitystyökaluja sovelluksien rakentamista varten (Ranger, 2018). Käyttäjä ei voi hallinnoida resursseja niin kuin IaaS-ratkaisuissa, mutta voi hallinnoida julkaisemiaan sovelluksia. PaaS-malli on erityisesti hyödyllinen sovelluskehittäjille. (Mell & Grance, 2011.)

2.1.3 Ohjelmisto palveluna (SaaS)

SaaS (Software-as-a-Service) -ratkaisut ovat palveluntarjoajan pilvessä sijaitsevia sovelluksia, joita käyttäjä voi käyttää, mutta ei hallinnoida (Mell & Grance, 2011). SaaS-ratkaisut voivat olla saatavilla selaimen tai sovelluksen kautta, eikä niitä asenneta tietokoneelle. Erilaisia SaaS-ratkaisuja on valtavasti ja se onkin yleisin pilvipalvelumalleista. (Ranger, 2018.) Esimerkiksi monelle tuttu Netflix on SaaS-sovellus, joka sijaitsee Amazon web servicessä (Izrailevsky ym. 2016).

2.2 Pilvipalveluiden toimitusmallit

Ranger (2018) toteaa pilvipalveluiden kolmen yleisimmän toimitusmallin olevan julkinen pilvi, yksityispilvi sekä hybridipilvi. Julkinen pilvi on suuren yleisön käytössä ja mahdollistaa IaaS-, PaaS- ja SaaS-ratkaisujen ostamisen internetin välityksellä. Palveluntarjoajilla on yleensä paljon resursseja, joita se jakaa

useille eri käyttäjille. Resurssien suuri määrä antaa käyttäjille mahdollisuuden skaalata ratkaisujaan nopeasti tarpeen vaatiessa ilman, että palvelu rasittuu. Palvelu sijaitsee fyysisesti palveluntarjoajan tiloissa (Mell & Grance, 2011).

Vastakohtana laajasti saatavilla olevalle julkiselle pilvelle yksityinen pilvi on vain yhden organisaation käytössä. Organisaatio saattaa omistaa, ylläpitää ja operoida pilveä itse, tai nämä toiminnot voidaan ulkoistaa toiselle osapuolelle. (Mell & Grance, 2011.) Ranger (2018) kertoo yksityisen pilven eduista verrattuna julkiseen pilveen. Yksityisessä pilvessä on mahdollista hallinnoida täysin, missä palvelun sisältämä data säilytetään ja miten pilvi-infrastruktuuri rakentuu, mikä tekee mallista turvallisemman. Kääntöpuolena Ranger (2018) toteaa, että yksityisen pilven tekijöillä on harvemmin käytössään saman mittakaavan resursseja, kuten esimerkiksi Amazonilla, Googlella tai Microsoftilla.

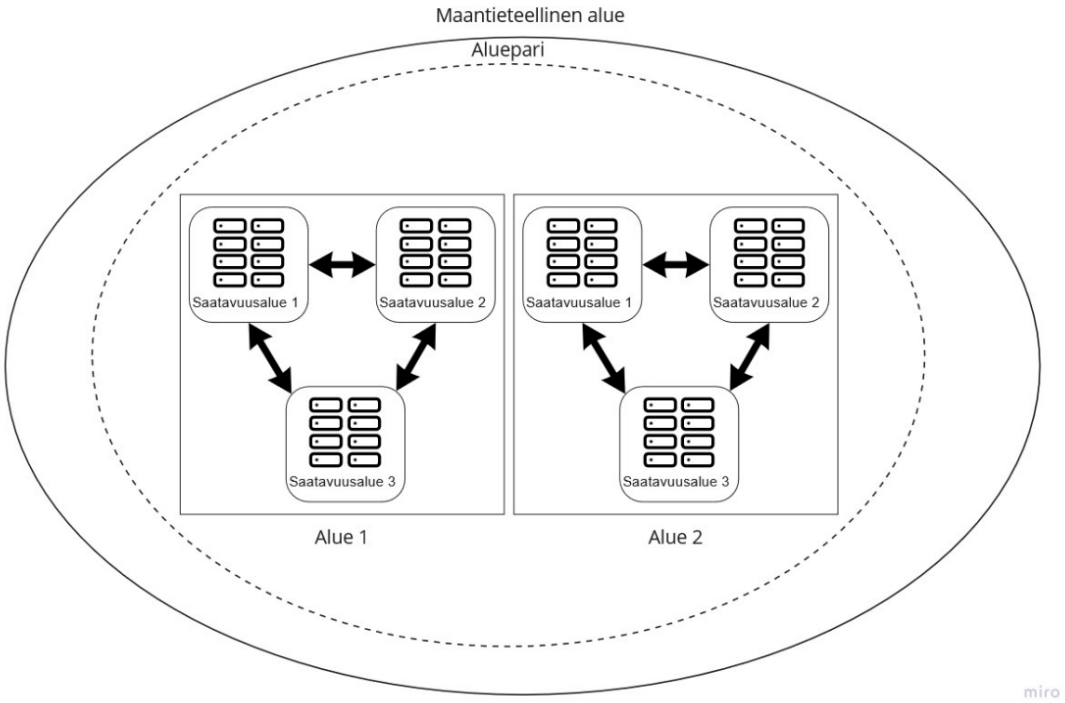
Hybridipilvi on yhdistelmä kahdesta tai useammasta pilvipalvelusta. Mellin ja Grancen (2011) mukaan palvelut pysyvät itsenäisinä kokonaisuuksina, mutta dataa ja sovelluksia voidaan siirtää niiden välillä. Hybridipilvi voi yhdistää esimerkiksi julkisen- ja yksityisen pilven palveluita. Osa toiminnasta voi tapahtua käyttäen julkisen pilven laajempia resursseja, mutta arempi tieto voidaan pitää yksityisessä pilvessä.

3 Microsoftin pilvipalvelut

Microsoftin pilvipalveluiden laajin kokonaisuus on Azure pilvipalvelu, joka tarjoaa SaaS-, PaaS- ja IaaS-ratkaisuja. Azuren lisäksi Microsoftilla on myös muita SaaS- ja PaaS-ratkaisuja. Microsoftin eri pilvipalvelut on myös mahdollista liittää toisiinsa.

3.1 Microsoft Azure -pilvialusta

Microsoft Azure on pilvialusta, joka voi toimia olemassa olevien sovellusten palvelimena tai tehostaa uusien sovellusten kehitystä. Azure sisältää lukuisia eri palveluita sovellusten kehitystä, testausta ja julkaisua varten. (Microsoft, 2021a.) Fyysisesti Azure sijaitsee yli 200:ssa ympäri maailmaa sijaitsevassa datakeskuksessa. Datakeskukset on jaoteltu maantieteellisesti alueisiin, joiden sisällä tietoliikenteen viive on minimoitu. (Microsoft, 2021b.) Kuva 2 näyttää Azuren datakeskusten ja alueiden suhteet toisiinsa.



Kuva 2. Azuren datakeskusten ja alueiden suhteet.

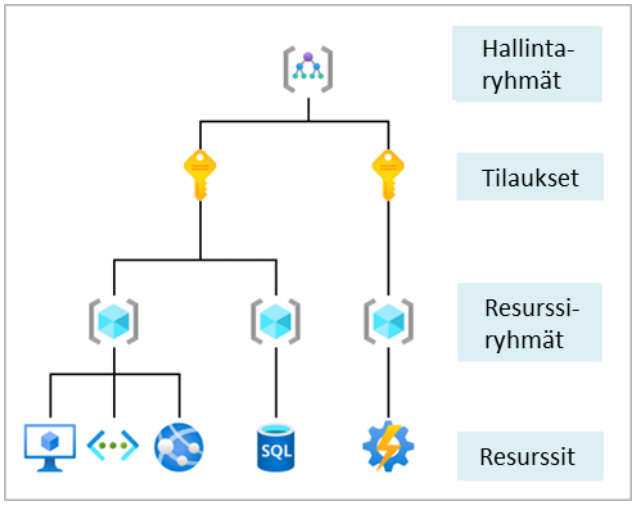
Datakeskukset ovat ympäri maailmaa sijaitsevia rakennuksia, joissa palvelimet sijaitsevat. Ryhmä datakeskuksia tietyssä maantieteellisessä osassa muodostaa alueen (engl. region). Datakeskukset alueen sisällä on kytketty toisiinsa pienen viiveen verkon kautta. Eri markkinat on jaoteltu yhden tai useamman alueen sisältäviin maantieteellisiin alueisiin (engl. geography). (Microsoft, 2021b.)

Saatavuusalueet (engl. availability zone) sijaitsevat alueiden sisällä ja ne suojelevat sovelluksia ja dataa yksittäisen datakeskuksen pettämiseltä. Saatavuusalueen muodostaa yksi tai useampi datakeskus, joista jokaisessa on itsenäinen voimanlähde, jäähdytys ja tietoliikenne. Alueen sisäiset saatavuusalueet ovat kaukana toisistaan, mikä myös suojelee sovelluksia ja dataa kokonaiseen laitokseen vaikuttavilta uhilta. (Microsoft, 2021b.)

Kaksi samassa maantieteellisessä alueessa sijaitsevaa aluetta voivat muodostaa alueparin (engl. region pair). Alueparien tarkoitus on taata palvelun jatkuvuus ja vikatiloista palautuminen. Suunnitellut päivitykset vaikuttavat

kerrallaan vain toiseen parin alueista, ja jos katko vaikuttaa useisiin alueisiin kerralla, ainakin toisen alueparin alueista palautus priorisoidaan. Aluepareja voidaan myös käyttää estämään datan menetys kopioimalla data automaattisesti toiseen alueeseen siltä varalta, että ensisijainen alue ei vastaakaan. (Microsoft, 2021c.)

Käyttäjän luomia palveluita, kuten virtuaalikoneita tai tietokantoja, kutsutaan Azuressa resursseiksi. Resurssit on Azuressa jaettu neljään hierarkkiseen kategoriaan: hallintaryhmät, tilaukset, resurssiryhmät ja resurssit (kuva 3). Eri resursseja voidaan yhdistää niiden käyttötarkoituksen mukaan ryhmiksi. Kun uusi resurssi otetaan käyttöön, se julkaistaan sille sopivaan resurssiryhmään. Tilaukseen kuuluu käyttäjätilejä sekä näiden käyttäjätilien luomat resurssit. Tilauksille asetetaan rajoituksia siihen, paljonko resursseja ne voivat luoda ja käyttää, mikä auttaa organisaatioita hallitsemaan kuluja. Eri tilaukset ryhmitellään hallintaryhmiin. Kaikki hallintaryhmän kuuluvat tilaukset perivät hallintaryhmän asetukset. (Microsoft, 2021d.)



Kuva 3. Azuren resurssien hierarkia (mukaillen Microsoft, 2021d).

3.2 Microsoftin muut pilvipalvelut

Microsoft 365 on Microsoftin pilvipalveluista kenties tunnetuin. Microsoft 365 on ohjelmistopaketti, josta on erilaisia versioita henkilökohtaiseen-, oppilaitos- ja työkäyttöön. Pakettiin kuuluu mm. tekstin- ja tietojenkäsittelyn työkaluja,

säilytystilaa sekä sähköpostipalveluja. Office-sovelluksista on olemassa omalle tietokoneelle asennettava työpöytäversio, mutta kaikkia palveluita tarjotaan myös SaaS-ratkaisuina julkisessa pilvessä. (Microsoft, 2021e.)

Microsoft Power Platform on neljästä osasta koostuva kehitysalusta, jonka osat voi yhdistää muihin Microsoftin palveluihin kuten Dynamics 365: een tai Azureen sekä myös Microsoftin ulkopuolisiin sovelluksiin. Power Platformin osat on esitelty taulukossa 1. Alustan osien on tarkoitus olla helposti käytettäviä ilman ohjelmointitaitoja ja ne on suunnattu vastaamaan erilaisiin liiketoiminnan tarpeisiin. (Microsoft, 2021f.)

Taulukko 1. Microsoft Power Platform -alustan osat ja niiden tarkoitukset (Microsoft, 2021f).

Palvelu	Tarkoitus
Power BI	Datan analyysi ja kuvantaminen
Power Apps	Yritysten toimintaa helpottavien sovellusten ja nettisivujen rakentaminen ilman koodausta
Power Automate	Työnkulkujen automaatio
Power Virtual Agents	Chattibottien luonti ilman koodausta

Microsoft Dynamics 365 on usean yrityssovelluksen kokoelma. Sovellukset on suunnattu palvelemaan eri liiketoiminnan aloja, kuten myyntiä, asiakkuudenhallintaa ja toiminnanohjausta. (Microsoft, 2021g.) Asiakkuudenhallintajärjestelmillä voidaan hoitaa asiakkuuksia, seurata myyntiä sekä hallita markkinointia (Microsoft, 2021h). Toiminnanohjausjärjestelmillä hallitaan liiketoiminnan prosesseja, kuten taloushallintoa, kaupankäyntiä ja raportointia (Microsoft, 2021i).

4 Tuotekehitys

Tuotekehitys on yrityksen liiketoiminnan kannalta tärkeä prosessi, ja sen tarkoitus on parantaa jo olemassa olevia tuotteita tai tuoda uusi tuote markkinoille. Tuotekehityksen kautta yritys voi säilyttää tai parantaa markkina-asemaansa ja varmistaa toimintansa jatkumisen. (Jaakkola & Tunkelo, 1987.) Ilman tuotekehitystä yrityksen tuotteet vanhenevat ja lopulta niiden myynti vähenee ja loppuu (Jokinen, 2001). Jotta tuote menestyy, sen tulee olla kilpailukykyinen ja sopia yrityksen tuotanto- ja jakeluketjuun (Jaakkola & Tunkelo, 1987). Tuotekehitysprosessi on kertaluontoinen ja sille määritellään tavoitteet, kesto ja rajalliset resurssit. Prosessin tavoitteiden ei tarvitse olla tiukasti määriteltyjä alussa, vaan prosessille on ominaista, että ne täsmentyvät prosessin edetessä. (Jaakkola & Tunkelo, 1987.)

Jaakkolan ja Tunkelon (1987) mukaan uutta tuotetta kehitettäessä kehitystiimin ideoilla ja ammattitaidolla on suuri vaikutus prosessin onnistumiseen, kuten myös tiimin käytössä olevilla resursseilla ja ilmapiirillä, jossa kehitystyötä tehdään. He toteavat, että hyvän tuotekehitysprojektin tulee nojata yrityksen osaamisalueisiin ja olla luova sekä järjestelmällinen. Prosessin tulisi ottaa riskejä, olla taloudellisesti kannattava ja nojata yrityksen strategiaan.

Tuotekehitysprosessi on monivaiheinen ja on tyypillistä, että eri vaiheita suoritetaan samaan aikaan. Jo suoritettuihin vaiheisiin saatetaan myös joutua palaamaan, kun prosessin edetessä ilmenee uusia asioita. (Jaakkola & Tunkelo, 1987.) Jokinen (2001) jakaa tuotekehitysprosessin neljään osaan: käynnistäminen, luonnostelu, kehittäminen ja viimeistely. Ohjelmistoja kehitettäessä jotkut osista sulautuvat ohjelmistokehitysprosessin vaiheisiin. Tämä luku tulee käsittelemään Jokisen (2001) esittämistä tuotekehitysprosessin vaiheista käynnistämisen, luonnostelun sekä viimeistelyn. Kehitysosaan kuuluvat prosessit esitellään luvussa 4, joka käsittelee ohjelmistokehitystä.

4.1 Tuotekehitysprosessin käynnistäminen ja luonnostelu

Tuotekehitysprojekti voi saada alkunsa ulkoisesta lähteestä, esimerkiksi asiakkaalta, tai sisäisestä toiminnasta, kuten tutkimuksesta (Jaakkola & Tunkelo, 1987). Tarpeen lisäksi tulee olla mielikuva siitä, miten tarpeeseen voidaan vastata. Ennen kuin projektia aletaan kehittää, tulee selvittää siihen liittyvien kustannusten ja saatavien tuottojen määrä sekä markkinoiden tilanne. (Jokinen, 2001.)

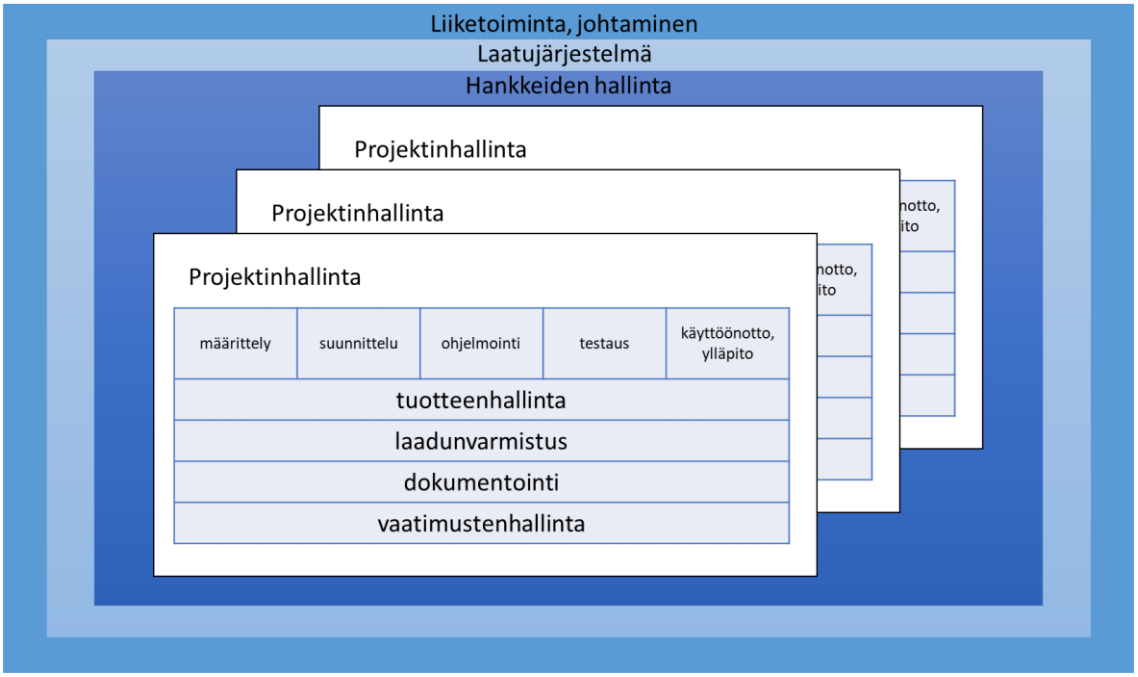
Jokisen (2001) mukaan luonnosteluvaiheessa etsitään erilaisia ratkaisuja kehitettävälle tuotteelle. Tehtävä analysoidaan ja kehitettävälle tuotteelle asetetaan vaatimukset. Jaakkola ja Tunkelo (1987) toteavat, että vaatimusmäärittelyssä voidaan käyttää apuna tuotteen käyttöolojen tutkimista haastattelemalla tulevia käyttäjiä. Onnistuneessa kehitysprojektissa asiakkaan vaatimusten ymmärtäminen ja yhteistyö asiakkaan kanssa on tärkeää (Jaakkola & Tunkelo, 1987).

4.2 Uuden tuotteen markkinoille vienti

Uuden tuotteen vienti markkinoille saattaa vaatia koulutusta uuden tuotteen käyttöön ja julkaistaessa tulee olla varma, että julkaisua ei olla tekemässä liian aikaisin. Tuotekehitykseen liittyy riskejä, kuten uusien teknologioiden tuomat yllätykset, kilpailijoiden pyrkiminen samoille markkinoille tai käyttäjän toimintatapojen tai vaatimusten virhearviointi. Riskejä voidaan pienentää tekemällä taustatyötä ja käyttämällä asiantuntijaosaamista, etenemällä nopeasti ja salaamalla oleelliset tiedot sekä testaamalla tuotetta kattavasti. Jos tuotteesta löytyy virheitä tai puutteita julkaisun jälkeen, ne tulee pyrkiä korjaamaan nopeasti ja asiakkailta tuleviin parannusehdotuksiin tulee suhtautua avoimesti. (Jaakkola & Tunkelo, 1987.)

5 Ohjelmistokehitys

Ohjelmistokehitysprosessissa on Sommervillen (2016) mukaan neljä olennaista osaa: määrittely, kehitys, validointi ja jatkokehitys. Kehityksessä saatetaan käyttää eri tapoja, kuten vesiputousmallia tai ketterää kehitystä, riippuen tuotettavasta ohjelmistosta, mutta nämä osat ovat silti osa kehitystyötä. Nämä vaiheet heijastuvat myös Haikalan ja Märijärven (2004) määrittelemiin ohjelmistokehityksen vaiheisiin. Kuvassa 4 on kuvattu nämä vaiheet sekä erilaisia ohjelmiston elinkaaren läpi kulkevia tukitoimintoja.



Kuva 4. Ohjelmistokehityksen vaiheet sekä ohjelmistotuotannon osa-alueet (mukaillen Haikala & Märijärvi, 2004).

Tämä luku tulee ensin esittelemään vesiputousmallin ja käymään läpi, mitä eri ohjelmistokehityksen vaiheissa tehdään. Tämän jälkeen käydään läpi myös ketterän kehityksen käytäntöjä ja scrum-viitekehys.

Vesiputousmallissa ohjelmistokehityksen eri vaiheista käsitellään erillisinä vaiheina, joista jokainen johtaa seuraavaan. Tarkoitus on, että vaihe ei voi alkaa ennen kuin sitä edeltävä vaihe on loppunut. Käytännössä tämä ei välttämättä

toteudu, sillä prosessin aikana voi ilmetä esimerkiksi uutta tietoa, jonka takia edellisten vaiheiden tuotuoksia on muutettava. (Sommerville, 2016.)

5.1 Määrittely ja suunnittelu

Sommervillen (2016) mukaan määrittelyvaiheessa järjestelmän palvelut, rajoitukset ja toiminnot kuvaillaan yhdessä asiakkaan kanssa. Haikala ja Märijärvi (2004) kertovat ohjelmistovaatimusten kuvaavan ohjelmiston toimintoja, ei-toiminnallisia vaatimuksia sekä rajoituksia. Toimintoja ovat eri toiminnallisuudet, joita ohjelmistolla toteutetaan, käyttöliittymä ja järjestelmän osien välinen kommunikointi. Ei-toiminnallisissa vaatimuksissa kuvataan esimerkiksi ohjelmiston suoritusteho ja vasteaika. Ohjelmistolle voi asettaa rajoituksia esimerkiksi käytössä oleva muistitila tai ohjelmoinnissa käytettävä kieli.

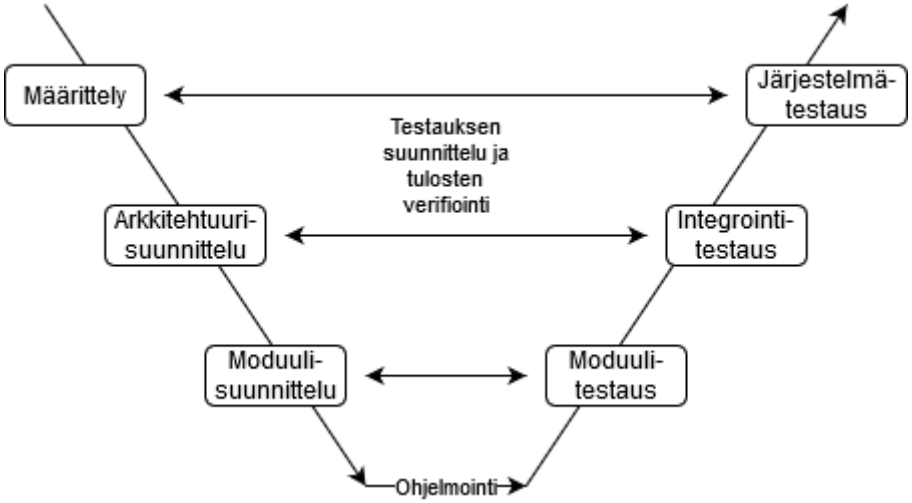
Ohjelmiston toimintojen kuvausten jälkeen niiden toteutus suunnitellaan ja järjestelmä jaetaan mahdollisimman itsenäisin moduuleihin arkkitehtuurisuunnittelussa. Vaiheen seurauksena tuotettavaa dokumenttia kutsutaan tekniseksi määrittelyksi. Kun arkkitehtuurisuunnittelu on tehty, seuraa moduulisuunnittelu, jossa moduulien sisäinen rakenne suunnitellaan. (Haikala & Märijärvi, 2004.)

5.2 Ohjelmointi ja testaus

Kun ohjelmiston arkkitehtuuri ja moduulit on suunniteltu, aloitetaan ohjelmiston ohjelmointi. Vaihe kestää niin kauan, kunnes ensimmäinen virheetön käännös ohjelmistosta on valmis. (Haikala & Märijärvi, 2004.)

Testauksen tarkoitus on löytää virheitä ohjelmistosta. Testaus on mukana ohjelmoinnin jokaisessa vaiheessa, sillä mitä aikaisemmin virheet löytyvät, sitä halvempaa niiden korjaaminen on. Ohjelmistotestaus voidaan jakaa moduuli-, integraatio- ja järjestelmätestaukseen, jotka testaavat eri osia ohjelmistosta. (Haikala & Märijärvi, 2004.) Moduulitestauksessa etsitään virheitä yksittäisistä

moduuleista ja varmennetaan, että ne vastaavat niille asetettuja määrityksiä (Sommerville, 2016). Integraatiotestauksessa varmistetaan, että moduulit toimivat yhdessä ja järjestelmätestauksessa testataan koko järjestelmän toimintoja ja suorituskykyä. Näin jaettua testausta kutsutaan v-malliksi (**Error! Reference source not found.**5). (Haikala & Märijärvi, 2004.)



Kuva 5. Testauksen v-malli (mukaillen Haikala & Märijärvi, 2004).

5.3 Ohjelmiston ylläpito ja laadunvarmistus

Ylläpitovaihe voidaan jakaa kolmeen tyyppiin: korjaava, adaptiivinen ja täydentävä (Haikala & Märijärvi, 2004). Korjaavassa ylläpidossa korjataan ohjelmistosta löytyneitä virheitä ja ratkaistaan asiakkaan ongelmia. Jos ohjelmistoa ympäröivät vaatimukset ovat muuttuneet, voidaan tarvita adaptiivista ylläpitoa, jossa muutetaan ohjelmiston osia. Täydentävässä ylläpidossa ohjelman toiminnallisuuksia parannellaan tai lisätään. (Sommerville, 2016.)

Laadunvarmistus on mukana jokaisessa ohjelmistokehityksen vaiheessa, ja se määrittelee ohjelmistoa kehittävän yrityksen toimintatavat. Laaduntarkastuksella, kuten katselmuksilla ja testauksella, pyritään löytämään mahdolliset virheet mahdollisimman aikaisessa vaiheessa. Katselmuksia pidetään vaiheiden lopussa, ja niissä varmistetaan, että vaiheen aikana on

saavutettu sille asetetut tavoitteet ja vaiheeseen liittyvät dokumentit on tuotettu. (Haikala & Märijärvi, 2004.)

5.4 Ketterä ohjelmistokehitys

Ketterä ohjelmistokehitys perustuu menetelmiin, joissa korostuu ohjelmiston toimivuuden ensisijaisuus, suora viestintä sekä nopea muutoksiin reagointi. Ketterä ohjelmistokehitys tapahtuu palasissa, jotka inkrementaalisesti lisäävät toimivuutta ohjelmistoon. Jokainen osa tehdään ja testataan toimivaksi ennen ohjelmistoon lisäämistä. (Lehtonen ym. 2014.) Ketterän kehityksen viitekehyksiä on monia, joista yksi suosituimpia on scrum.

Agile Alliance –järjestö julkaisi vuonna 2001 manifestin, joka julisti ketterälle kehityksen 12 periaatetta ja neljä tukipilaria. Tukipilareina pidettiin seuraavia asioita:

- Ihmiset ja vuorovaikutus ovat tärkeämpiä kuin prosessit ja työkalut.
- Toimiva ohjelmisto on tärkeämpää kuin kattava dokumentaatio.
- Yhteistyö asiakkaiden kanssa on tärkeämpää kuin sopimusneuvottelut.
- Muutoksiin vastaaminen on tärkeämpää kuin suunnitelmassa pitäytyminen. (Agile Alliance, 2001.)

Ketterässä ohjelmistokehityksessä kehittäjä on vastuussa omasta työstään ja sen suunnittelusta. Kommunikointi tiimin sisällä on tärkeää, sillä tiimi sopii työtehtävistä yhdessä ja työmenetelmiä kehitetään jatkuvasti. Kommunikointi tiimin ulkopuolelle on myös oleellista, sillä asiakas on mukana projektissa kehityksen aikana ja tuloksia esitellään asiakkaalle säännöllisesti. Palautteen saanti läpi kehitystyön on tärkeä osa prosessia ja ajatuksena on, että kun ymmärrys kehitettävästä tuotteesta lisääntyy, niin projektin tavoitteet myös muokkautuvat vastaamaan paremmin asiakkaan tarpeita. Ketterässä kehityksessä projektin alussa tehtävät suunnitelmat eivät ole yksityiskohtaisia ja tarkkoja, vaan ne muokkautuvat projektin edetessä ja vaatimusten muuttuessa. (Lehtonen ym. 2014.)

5.5 Scrum-viitekehys

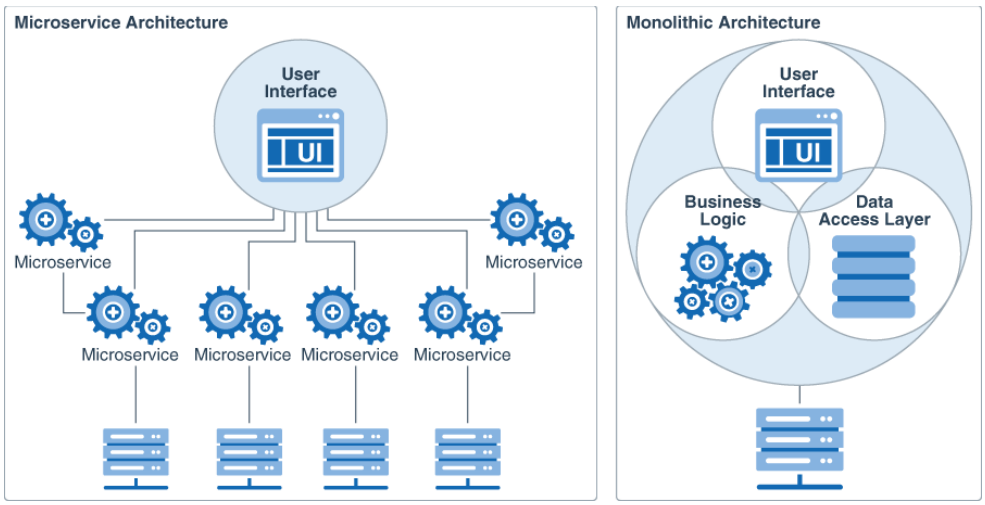
Scrum on ketterän kehityksen viitekehys, joka käyttää iteratiivista (toistavaa) ja inkrementaalista (lisäävää) lähestymistapaa. Työmallin tarkoitus on parantaa ennustettavuutta ja riskienhallintaa. (Schwaber & Sutherland, 2020.) Scrumissa muuntuvuus on tärkeässä asemassa. Projektin tuotoksia katselmoidaan säännöllisin ajoin ja sopeuttaminen muuttuneisiin olosuhteisiin ja vaatimuksiin on osa prosessia. Scrum-tiimissä on määritellyt roolit ja prosessiin kuuluu oleellisesti erilaiset tuotokset. Scrumin perusta on 1–4 viikon pituiset jaksot, joita kutsutaan sprinteiksi. Jokaisen sprintin aikana luodaan scrum-tiimin määritelmän mukaan valmis osa tuotteesta. (Drummmond, 2021.)

6 Pilvipalveluiden tuotekehitys

Pilvipalveluiden kehitykselle on omat toimintamallinsa, jotka ovat osa pilvinatiivia kehitystä. Pilvinatiivin toimintamallin keskiössä ovat kehityksen nopeus ja ketteryys, jotta voidaan vastata kuluttajien vaatimuksiin nopeista ja katkottomasti toimivista ohjelmistoista. Palvelumallilla halutaan välttää suoritusongelmat, toistuvat virhetilat sekä hitaus. (Vettor & Smith, 2021.) Tämä luku käy ensin läpi kaksi pilvinatiiviin toteutukseen kuuluvaa työtapaa, kertoo sen jälkeen pilvinatiivista kehityksestä laajemmin ja lopussa esittelee DevOps-toimintamallin.

6.1 Mikropalvelut

Mikropalveluarkkitehtuuri on vastakohta yleisemmälle monoliittiarkkitehtuurille (**Error! Reference source not found.**6). Vettor ja Smith (2021) kertovat monoliittiarkkitehtuurissa ohjelmiston koostuvan toisistaan riippuvaisista moduuleista. Monoliitteja on yksinkertaista rakentaa, testata ja julkaista, mutta ohjelmiston kasvaessa päivitysten myötä kokonaisuus voi olla vaikeampaa hahmottaa ja muutoksilla voi olla odottamattomia vaikutuksia. Näin ollen monoliittisten ohjelmistojen kasvaessa niiden päivittämisestä ja korjauksesta tulee hankalampaa, ja ne vievät enemmän aikaa ja rahaa.



Kuva 6. Monoliitti- ja mikropalveluarkkitehtuurin vertailu (Oracle, 2021).

Mikropalveluarkkitehtuuria käytettäessä sovellus jaetaan pieniin, itsenäisiin mikropalveluihin. Jokainen mikropalvelu on itsenäinen koodin, datan ja riippuvuuksien kokonaisuus. Yhden koko ohjelmiston kattavan tietokannan sijaan jokaisella mikropalvelulla on oma tietovarastonsa, mikä myös mahdollistaa käytettävän tietovaraston tyypin räätälöimisen palvelun mukaan. (Vettor & Smith, 2021.)

Jokainen mikropalvelu suorittaa tietyn osan suuremmasta palvelusta. Mikropalvelut kommunikoivat keskenään ja yhteenliitettynä palvelut muodostavat ohjelmiston. Mikropalvelut kehitetään autonomisina kokonaisuuksina ja ne voidaan julkaista itsenäisesti. Tämä tuo ketteryyttä ohjelmiston kehitykseen, koska pieni osa voidaan päivittää ilman huolta siitä, miten se vaikuttaa muuhun sovellukseen. (Vettor & Smith, 2021.)

Mikropalveluita voidaan myös kontittaa. Kontitus tarkoittaa ohjelmiston koodin ja riippuvuuksien paketoimista yhteen konttiin. Kontissa ohjelmiston voi suorittaa eri ympäristöissä riippumatta ympäristön käyttöjärjestelmästä. Kontti sulkee ohjelmiston omaan ympäristöönsä irtonaisena ympäröivästä infrastruktuurista. (RedHat, 2021.)

6.2 CI/CD-kehitysputki

CI (engl. continuous integration, jatkuva integraatio) ja CD (engl. continuous deployment/delivery, jatkuva toimitus) on nopea tapa tehdä sovellusjulkaisuja. Jatkuva integraatio tekee koonnin koodista, kun se kirjataan sisään versiohallintaan ja suorittaa koodin automaattisen testauksen. Tämä mahdollistaa mahdollisten virheiden löytämisen ajoissa, ja jos koodi läpäisee testit, se voidaan heti siirtää tuotantoon. Tämä vähentää tehtävän dokumentoinnin määrää. Myös mahdollisuus virheiden tekemiseen vähenee, koska ennen tuotantoon siirtämistä tehtävät tarkastukset tehdään automaattisesti, eikä manuaalisesti. (Vettor & Smith, 2021.)

Jatkuvassa toimituksessa voidaan kirjoittaa komentosarjoja, jotka suorittavat ohjelmiston julkaisuun tarvittavat toiminnot nopeammin ja tarkemmin kuin

ihmiset. Tällaisia toimintoja voivat olla esimerkiksi pilvipalvelujen erilaisten resurssien käyttöönotto. Julkaisut voidaan tehdä testi- tai tuotantoympäristöihin. Tuotantoon julkaistessa on investoitava kattaviin testeihin, jotta voidaan olla varmoja, että mitään ei mene rikki. (Vettor & Smith, 2021.) Testien puute, tarve manuaaliselle laaduntarkastukselle ja byrokraattinen julkaisuprosessi ovat yleisimpiä syitä, miksi yritykset eivät ota jatkuvaa toimitusta käyttöön (Mojtaba ym. 2017).

6.3 Pilvinatiivi kehitys

Mikropalvelut sekä CI/CD-kehitysputki ovat osa pilvinatiivia kehitysmallia. CI/CD-putken automatisaatio nopeuttaa ohjelmiston julkaisusykliä ja käyttämällä mikropalveluarkkitehtuuria voidaan hyödyntää pilvipalveluiden skaalautuvuutta. Pilvi-infrastruktuuria käyttämällä resursseja voidaan ottaa käyttöön minuuteissa, skaalata ja tuhota ohjelmiston tarpeiden mukaan. Tällaista palvelujen automatisoitua käyttöönottoa kutsutaan nimellä infrastruktuuri koodina (engl. infrastructure as code, IaC). (Vettor & Smith, 2021.) Automatisaation käyttöönotto lisää ohjelmiston vikasietoisuutta ja tehokkuutta, sillä automatisoidut prosessit voivat korjata, skaalata ja tehdä julkaisuja nopeammin kuin ihmiset. Koska pilvinatiiviin suunnitteluun kuuluu itsenäisten mikropalveluiden käyttö, voidaan pilvinatiiveja ohjelmistoja korjata automatisoimalla vikatilaan joutuneiden prosessien tuhoaminen ja uusien, toimivien palvelujen käyttöönotto. Sama periaate toimii myös skaalautuvuudessa. Kun kysyntää on paljon, voidaan palveluita ottaa automaattisesti lisää käyttöön vastaamaan tarpeeseen. Kun kysyntä laskee, voidaan ylimääräiset palvelut tuhota, mikä säästää kustannuksia. Järjestelmän tilan, terveyden ja käyttöasteen monitorointi ja lokitus voidaan myös automatisoida, mikä antaa kehittäjille kattavamman kuvan ohjelmiston käytöstä. (Grey, 2019.)

Pilvinatiivit järjestelmät ovat riippuvaisia erilaisista taustapalveluista, jotka voivat hoitaa esimerkiksi palveluun tunnistautumista, datan säilömistä tai palvelun tilan monitorointia. Näiden kehitys itse on mahdollista, mutta käyttämällä

pilvipalvelujen tarjoajien taustapalveluja voidaan säästää rahaa ja resursseja, ja vastuu palveluiden ylläpidosta on pilvipalvelujen tarjoajalla. Toisaalta käyttämällä valmiita palveluita saatetaan ohjelmisto lukita tiettyyn pilvialustaan. (Grey, 2019; Vettor & Smith, 2021.)

6.4 DevOps-malli

DevOps nimenä on yhdistelmä englanninkielisistä termeistä Software Development (ohjelmistokehitys) ja IT Operations (ylläpito/tuotanto). Termillä ei ole universaalia määritelmää, mutta sen katsotaan olevan tapa lyhentää aikaa sovelluskehityksen ja sovelluksen julkaisun välillä ilman, että sovelluksen laatu kärsii. (Erich ym 2017.) Yritykset ovat siirtyneet käyttämään DevOps-mallia, sillä nopea julkaisusykli antaa niille kilpailuedun (Vettor & Smith, 2021).

Erich ym. (2017) tutkivat DevOpsin määritelmää suorittamalla kirjallisuusanalyysin sekä haastattelemalla DevOpsia työssään käyttäviä kehittäjiä. Tutkimuksen mukaan DevOpsille ei ole yksiselitteistä määritelmää, mutta haastattelujen mukaan yrityskulttuurin sekä automaation nähtiin olevan keskeinen osa sitä. Koska DevOpsilla ei ole yhtenäistä määritelmää, eikä sitä ole tutkittu kattavasti, sen tehokkuudesta ei ole tieteellistä näyttöä. DevOps toimintamalli on yhteensopiva ketterän kehityksen sekä pilvinatiivin toimintamallin kanssa.

Mikropalveluarkkitehtuuri on osa DevOpsia. DevOps-työtapa tuli ennen mikropalveluita ja osittain mahdollisti niiden kehittämisen tekemällä useiden sovellusten julkaisusta ja hallinnoinnista tuotannossa helpompaa. (Vettor & Smith, 2021.)

Microsoft Azuressa on DevOpsille omistettuja työkaluja mm. projektin- ja lähdekoodin hallintaan, koodin kokoamiseen, testaukseen ja julkaisuun. Toinen, myös Microsoftin omistama palvelu, jota voidaan käyttää ohjelmistojulkaisujen automatisaatioon on GitHub actions. GitHub actions on maksullinen CI/CD-työkalu, joka mahdollistaa koodin automaattisen julkaisun eri pilvipalveluissa. (Vettor & Smith, 2021.)

7 Case-esimerkki PATA

Tässä työssä tutkittiin pilvipalveluiden tuotekehitystä käyttämällä case-esimerkkinä PATA-hanketta. Työssä tarkasteltiin 2M-IT:n PATA-hanketta haastattelemalla hankkeessa mukana olevia henkilöitä.

7.1 PATA-hanke

PATA (palvelun tarvearviointi ja palveluun ohjaus) -hanke on 2M-IT:n sekä kolmen julkisen sosiaali- ja terveydenhuollon organisaation yhteistyössä kehittämä alusta sote-sektorin neuvontaan ja asiakasohjaukseen. Tarkoitus on, että sosiaali- ja terveyspalveluiden asiakkaita voidaan palvella keskitetysti monien eri kanavien välityksellä tavoitettavissa olevan asiakaspalvelukeskuksen kautta. Asiakaskeskuksessa ammattilainen ohjaa asiakkaan oikeaan jatkopaikkaan (kuva 7). (2M-IT, 2021.)

Kuva 7. PATA-ratkaisun kuvaus (2M-IT, 2021).

PATA koostuu kolmesta osasta: teknologia-alustasta, PATA-asiakaspalvelukeskuksen toimintaa kuvaavasta toimintamallista sekä

alueellisista muutoshankkeista mallin käyttöönottamiseksi (2M-IT, 2021). Tämä työ käsittelee näistä osista ainoastaan teknologia-alustaa.

7.2 Tutkimuksen toteutus

Opinnäytetetyön materiaali kerättiin haastattelemalla kahta PATA-hankkeessa keskeisesti mukana olevaa henkilöä käyttämällä puolistrukturoituja teemahaastatteluja. Puolistrukturoiduissa haastatteluissa ei ole kiinteää kysymys- ja vastauslistaa, vaan haastateltavan annetaan vastata omiin sanoin. Teemahaastatteluissa käsiteltävä aihe jaetaan erilaisiin teemoihin, joista kysyttävät kysymykset ammennetaan. Teemahaastattelussa haastattelijalla ei ole tarkkoja haastattelukysymyksiä, eikä kysymyksiä aina esitetä samassa muodossa tai järjestyksessä. (Hyvärinen ym. 2017.) Tämä haastattelutapa valittiin, sillä sen katsottiin antavan haastateltaville parhaan mahdollisuuden vastata vapaasti ja osittain ohjata haastattelua heidän mielestään tärkeisiin osa-alueisiin.

Haastattelun teemat rakentuivat aiemmin tässä työssä käsiteltyjen aiheiden, kuten pilvipalveluiden ja tuotekehityksen, ympärille. Liitteessä 1 ovat tutkimuskysymykset ja teemat. Jokaisesta teemasta mietittiin etukäteen, mitä olisi tärkeä saada selville tutkimuskysymyksiin vastaamista varten ja haastattelun aikana keskustelua ohjattiin vastaamaan näihin kysymyksiin.

Haastattelut suoritettiin Microsoft Teamsin välityksellä videopuheluina. Haastattelut tallennettiin ja molemmat haastattelut kestivät noin tunnin. Haastateltaville kerrottiin haastattelun alussa haastattelun rakentuvan teemojen ympärille, ja että heille ei-olennaiset teemat voidaan ohittaa. Teemojen käsittelyjärjestys ja liikkuminen teemojen välillä myös ohjautui haastateltavien puheen mukaan. Hyvärinen ym. (2017) painottavat, että haasteltavalle on hyvä esittää haastattelun alussa melko avoimia lämmittelykysymyksiä, jotta haastateltava tottuisi heti alussa vastaamaan pidemmillä puheenvuoroilla. Molemmat haastattelut aloitettiin kysymällä, mikä haastateltavan rooli PATA-hankkeessa on ollut. Samalla vastaus antoi haastattelijalle paremman kuvan

siitä, minkä teemojen kysymyksiä haastateltavan kanssa olisi hyvä painottaa. Haastateltaville annettiin myös haastattelun lopussa mahdollisuus tuoda esille heidän mielestään tärkeitä asioita teemoista, joita ei ollut haastattelun aikana vielä käsitelty. Haastatteluiden jälkeen tallenteet katsottiin ja niistä tehtiin muistiinpanot, joiden perusteella luvun 8 tulokset kirjoitettiin.

8 Tutkimustulokset

Haastattelujen vastaukset on koottu eri teemojen alle. Osa teemoista vastaa liitteessä 1 eriteltyjä teemoja, joiden ympärille haastattelut rakentuivat ja osa teemoista nousi esille haastatteluiden aikana. Ensin käsitellään enemmän sitä, mistä PATA-alusta koostuu ja millainen pilvipalvelu se on. Tämän jälkeen käydään läpi, miksi ratkaisu päädyttiin tuottamaan Microsoftin pilvessä. Tuloksissa käydään myös läpi haastateltavien ajatuksia ketterästä kehityksestä sekä lyhyesti ajatuksia turvallisuudesta.

8.1 PATA-ratkaisu

Tällä hetkellä PATA-ratkaisu on kaksiosainen. Siihen kuuluvat ammattilaisen työpöytä, jonka kautta asiakaskommunikaatio suoritetaan, sekä Patalogi-niminen palvelukatalogin ja ratkaisupankin yhdistelmä. Ammattilainen voi lukea Patalogia ollessaan asiakaskontaktissa ja löytää työhönsä tukea, kuten toimintaohjeita, tai sitä, miten neuvoa asiakasta. Patalogin ohjeiden ylläpitäminen on PATA-ratkaisua käyttävien organisaatioiden vastuulla. Ratkaisu ei tällä hetkellä ole lääkinnällinen laite, mutta lääkinnällisen laitteen määritelmän täyttäviä toiminnallisuuksia, kuten tekoälyn tekemää terveydentilan arviointia, olisi mahdollista lisätä tulevaisuudessa.

PATA-alusta on palvelumalliltaan asiakkaiden näkökulmasta SaaS-ratkaisu, jossa 2M-IT tarjoaa ohjelmiston, ylläpidon, lisenssit ja kehityspalvelut asiakkaille yhdessä paketissa. Palvelun rakentamiseen on käytetty Microsoftin Azure -pilveä ja alustaan kuuluu Microsoft Dynamics 365 -tuote, johon asiakaspalvelijan työpöytä on rakennettu sekä Power Appina tuotettu palvelukatalogi Patalogi. Alustan kehityksessä on siis käytetty SaaS- ja PaaS-ratkaisuja. PATA-ratkaisun arkkitehtuurin ei haastatteluissa nähty olevan mikropalveluarkkitehtuuri, vaikka sellaisen toteuttamisen Azuressa todettiin olevan mahdollista. Automatisaatio on ratkaisun osalta suunnitteilla, mutta toistaiseksi kehityksessä käytetään Azure DevOpsia vain testaukseen ja

sprinttien suunnitteluun. Haastattelujen mukaan PATA-ratkaisun Azure sekä Dynamics 365 -osat sijaitsevat julkipilvessä. Palvelusta on myös integraatioita muihin palveluihin, tehden PATA-alustasta hybridipilvessä sijaitsevan ratkaisun.

Alustaratkaisun hyödyn koettiin olevan valmiissa osissa. Esimerkiksi käyttäjänhallintaan, käyttäjäoikeuksiin ja lokituksiin oli jo olemassa valmiit osat, jotka kelpasivat lähes sellaisinaan. Käyttöön otettavien palveluiden osalta koettiinkin, että jos mitään ei olisi voitu ottaa sellaisenaan käyttöön, olisi tehty väärä valinta.

Haastatteluissa kerrottiin, että hankkeen alussa käytettiin paljon aikaa markkinakartoituksen tekemiseen, koska haluttiin muodostaa kuva siitä, miten ratkaisu olisi parasta tehdä. Alustateknologian käyttö oli alusta alkaen selkeä valinta, mutta alustan valinta haluttiin tehdä huolella, eikä pilvipalvelun tuottaminen ollut itsestään selvää. Pilvipalveluun päädyttiin lopulta, koska sen nähtiin olevan helpoin tapa tarjota palvelu asiakkaille ja laajentaminen olisi helpompaa, kuin omaan konesaliin tehdessä. Haastatteluiden mukaan selvitysvaiheessa kävi myös ilmeiseksi, että pilvipalvelut tulevat korvaamaan vanhoja teknologioita ja ovat tulossa myös sote-alalle.

Juuri Microsoftin pilvipalvelun käyttämiseen kerrottiin vaikuttaneen osittain jo olemassa oleva suhde Microsoftiin Office 365 -palveluiden ylläpidon muodossa. Myös nimenomaan Dynamics 365 -alustan päälle rakentamiseen päätyminen vaikutti päätökseen. Vaikka Dynamicsia olisi mahdollista käyttää toisessa pilvipalvelussa kontittamisen kautta, on sen ajaminen Azuressa tehokkain vaihtoehto. Erilaisten valmiiden osien liittäminen palveluun on helpompaa, jos kaikki tapahtuu Azuren pilvessä. Esimerkiksi jos palvelussa haluttaisiin käyttää tekoälyä, on helpompaa ottaa käyttöön Azure AI -palvelu kuin joku kolmannen osapuolen tekoälykomponentti. Tämä luo synergiaa ja kasvunvaraa palvelulle. Valtavana yrityksenä Microsoftilla on jo monia palveluita, joita voi hyödyntää ja lisää kehitellään jatkuvasti. Eri palveluista löytyy myös paljon julkista materiaalia, jota voi hyödyntää. Kääntöpuolina nähtiin, että Microsoftin valmiissa palveluissa, kuten Azuressa ja Dynamics 365:ssä, on usein vain yksi tapa tehdä asioita, ellei vie Azureen kontissa omia komponentteja ja lisenssihankinnat on

suunniteltava huolella. Alussa tehdyillä valinnoilla saattaa olla suuri vaikutus kustannuksiin myöhemmin, joten perusteellinen suunnittelu projektin alussa koettiin tärkeäksi.

8.2 Ketterä kehitys

PATA-ratkaisun kehityksessä on käytetty ketterää kehitystä ja scrum-viitekehystä. Haastatteluissa ketterän kehityksen todettiin vastanneen hyvin kehitysprojektin tarpeisiin ja olleen oikea suunta sovelluskehityksessä.

Vaatimusten määrittely nousi esiin tärkeänä aiheena tuotekehityksestä puhuttaessa. Ketterässä kehityksessä ei alussa ole tarkkoja määrittelyitä, vaan laajempia toiminnallisuuskuvauksia ja määrittelyjä syntyy kehityksen edetessä. Tämän koettiin olevan hyvä toimintatapa, sillä kaiken suunnittelu alussa olisi ollut hyvin työlästä ja vaatimukset olisivat luultavasti ehtineet muuttua ennen kuin kehitystyö olisi ollut valmis. Asiakkaan pitäminen mukana kehitystyön aikana koettiinkin hyväksi asiaksi. Asiakkaiden todettiin voivan eläytyä tuotteeseen paremmin, kun heille esitellään prototyyppejä. Asiakkaan kanssa kehityksen aikana tehtävällä yhteistyöllä nähtiin olevan suuri etu projektissa, koska suuntaa voitiin tarkistaa jokaisen kahden viikon sprintin välein. Täysin ongelmatonta määrittelytyö asiakasorganisaatioiden kanssa ei kuitenkaan ollut. Kehitystiimin ja asiakasorganisaatioiden aikataulujen kerrottiin välillä olleen haastavaa sovittaa yhteen. Kehitystyön kuluessa tarpeet tietynlaiselle asiantuntemukselle määrittelyn tekemisessä saattoivat nousta nopeammalla aikataululla, kuin hoitoalan henkilöstön työvuorosuunnittelu tapahtui. Siten varsikin hoitohenkilökunnan irrottaminen määrittelyjen tekoon oli vaikeaa.

PATA-ratkaisussa ensimmäiset ominaisuudet ovat menossa tuotantoon loppuvuodesta 2021, mutta palvelusta puuttuu vielä isoja osia. Ketterän kehityksen mukaisesti kehityksessä on edetty ominaisuus kerrallaan.

Pilvessä tapahtuvan kehityksen kerrottiin osittain helpottaneen kehitystyötä, koska infrastruktuurista, pääsyhallinnasta, palvelinkapasiteetista tai muistin riittävyydestä ei ole tarvinnut huolehtia. Toisaalta, vaikka infrastruktuurin

kerrottiin olevan helpompaa hallinnoida pilvessä, vaatii sen tekeminen silti paljon opiskelua. Esimerkiksi Azuren AD -lisenssejä on monia eri tasoja, ja palvelulle optimaalisimman selvittämisen kerrottiin vaatineen työtä.

Pilvipalveluiden todettiin kuitenkin molemmissa haastatteluissa olevan oikea suunta ohjelmistokehitykselle. Ketterän kehityksen katsottiin myös olevan oikea viitekehys pilvikehitykselle, vaikka vesiputousmallillekin todettiin vielä olevan paikkansa. Nykyistä tilannetta kuvailtiin nivelvaiheeksi, jossa on elementtejä molemmista kehitystyyleistä ja jopa hieman vastarintaa siirtymää kohtaan. Tottumus on ollut, että pitää olla selkeä suunnitelma ennen kuin lähdetään toteuttamaan kehitystä. Samoin kehityksen etenemistä on voitu ennustaa pitkälläkin aikavälillä. Koska ketterässä kehityksessä nämä asiat eivät välttämättä toteudu, on luottamuksen rakentaminen uuteen kehitystapaan ollut välillä haastavaa. Haastatteluissa korostui kuitenkin, että ketterä kehitys ei tarkoita, etteikö suunnittelua tehtäisi, vaan sitä tehdään jopa enemmän. Tämän todettiin vaativan yhtiön johdoltakin luottamusta ja ymmärrystä. Vesiputousmallillekin nähtiin silti olevan tarvetta modernissa sovelluskehityksessä tapauksissa, joissa ei olla tekemässä jotain uutta vaan jotain määrämuotoista.

8.3 Turvallisuus

Haastatteluissa todettiin, että turvallisuusvaatimukset ovat erittäin tärkeitä sote-alalla. Julkipilvessä olevan tiedon tulee olla tallennettuna ETA-alueella. Myös lokitukselle on erityisiä vaatimuksia ja lokeja tulee säilyttää viisi vuotta. On myös harkittava, miten palvelussa käsitellään tietoja. Tämän todettiin olevan enemmän kiinni siitä, miten sovellusta muokataan, kuin suoraan Microsoftista. Tietojen tallentamisen pilveen kerrottiin vaativan luottamusta Microsoftiin, mutta samalla todettiin myös, että isona yrityksenä Microsoftilla on paljon resursseja ylläpitää tietoturvaa sekä suorittaa valvontaa ja tietoturvan poikkeuksien paikkauksia.

9 Pohdinta

Haastatteluissa saatiin hieman erilainen näkökulma kehitystyöhön, kuin tutkimuksen alussa odotettiin. Tarkasti ohjelmiston tuotekehityksen eri vaiheisiin sidottujen oivallusten sijaan esille nousikin laajempia asiayhteyksiä sekä kattavampi käsitys PATA-ratkaisun osista ja sen pilvipalvelumallista sekä siitä, miksi hankkeessa päädyttiin rakentamaan nimenomaan pilvipalvelua.

PATA-ratkaisun kehittämisessä ei toistaiseksi ole otettu käyttöön monia pilvinatiivin kehityksen työkaluja. Tämä osoittaa, että ne eivät ole pakollinen osa pilvipalveluiden kehitystä. Mikään ei myöskään estä ottamasta esimerkiksi CI/CD-kehitysputkea käyttöön myöhemmin kehityksen aikana. Pilvinatiivit toimintatavat vaativat opettelua ja voikin olla, että niiden käyttöönotto projektissa, jossa oli muutenkin paljon uutta, ei olisi toiminut. Samoin pilvipalvelun arkkitehtuurilla on vaikutus pilvinativen toimintatapojen tarpeellisuuteen. Rakentaessa alustaratkaisua ei esimerkiksi mikropalveluarkkitehtuuri ollut tarpeen. Toisaalta pilvinatiiviin kehitykseen kuuluu myös erilaisten taustapalveluiden käyttäminen, joita on myös hyödynnetty PATA-ratkaisussa. Lisäksi palvelumallilla saattoi olla vaikutusta, sillä PaaS- tai IaaS-ratkaisussa pilvinatiivit kehitystavat olisivat voineet olla tärkeämpiä ottaa mukaan. PATA-ratkaisun kehityksessä on tarkoitus ottaa käyttöön DevOps -malli. Tällä hetkellä kehitystyössä käytetään vain osaa Azuren DevOps -palveluista.

Haastatteluiden tulokset mukailivat paljon luvuissa 2 ja 6 käsiteltyjä pilvipalveluiden hyviä puolia. Kehitystyö ei vaatinut suurta investointia infrastruktuuriin heti projektin alussa ja pilvipalvelun skaalautuvuutta pidettiin helppona. Microsoftin alustan ja palveluiden käyttämisessä nähtiin myös paljon hyvää. Yksi etu oli erityisesti laaja katalogi valmiita palveluita, joita voitiin hyödyntää parhaimmillaan ilman suuria muutoksia. Toisaalta huonompi puoli verrattuna omaan konesaliin itse koodattuihin palveluihin on, että pilvessä kaikesta veloitetaan maksu ja kuluja tulee seurata, jotta käynnissä ole yhtään turhia palveluita. Microsoftin palveluiden käyttäminen tarkoittaa myös sitä, että

tekijän on oltava valmis paneutumaan lisensointiin, sillä jokainen palvelu tarvitsee oman lisenssinsä. Lisenssienhankinta vaatii tarkkaa suunnittelua jo aikaisessa vaiheessa tuotekehitysprosessia, jotta kulut eivät myöhemmin kasva liian suuriksi. Microsoftin palveluiden käyttäminen vaatii myös luottamusta Microsoftiin varsinkin, kun pilveen tuodaan potentiaalisesti arkaluontoisia tietoja. Kehittäjien on luotettava Microsoftin sanaan siitä, että tietoja ei luovuteta kolmansille osapuolille ja pidettävä huoli siitä, että tieto tallennetaan Euroopassa sijaitsevaan datakeskukseen. Vastineeksi luottamukselle Microsoftin kokoisella yrityksellä on paljon enemmän resursseja käyttää tietoturvaan kuin monella muulla yrityksellä, ja kehittäjän ei tarvitse huolehtia palvelinten asioista kuten päivityksistä tai sähkön saannista. Palveluissa, joissa saavutettavuus on erityisen kriittinen, on myös mahdollisuus saada enemmän vikasietoisuutta kuin useimmissa konesalissa esimerkiksi alueparia käyttämällä. Useilla yrityksillä ja asiakkailla on myös jo käytössä joitain Microsoftin palveluita, jolloin kehitettävät palvelut voidaan yhdistää esimerkiksi AD:hen ja näin tehdä käyttäjänhallinnasta yksinkertaisempaa kehittäjille ja palveluun kirjautumisesta helpompaa käyttäjille. Tämä osoittaa osaltaan myös sen, että pilvipalveluista on hyötyä myös loppukäyttäjille eikä vain kehittäjille.

Määrittely on tärkeää heti tuotekehitysprojektin alusta lähtien, mutta kaikkea määrittelyä ei kannata tehdä heti alussa, sillä kehitystyön edetessä asiakkaan tarpeet luultavasti kirkastuvat ja muuttuvat. Ketterä kehitys soveltuu hyvin pilvipalvelun tuotekehitykseen, osittain koska määrittelyä voi tehdä asiakkaan kanssa läpi kehityksen. Toisaalta aikataulujen sovittaminen yhteen asiantuntijoiden kanssa määrittelyjä varten voi myös tuoda haasteita. Niin sanotussa päivittäisessä kehitystyössä ei PATA-ratkaisussa ole koettu olevan eriäväisyyksiä vesiputouskehitykseen, joskin asiaan saattaa vaikuttaa myös se, että pilvinatiivia suunnittelua on käytetty vain vähän.

10 Lopuksi

Työ käsitteli pilvipalvelukehittämisen tuomia muutoksia tuotekehitysprojektiin ja antoi erityisesti näkökulman pilvipalvelukehittämiseen siirtymisen tuomista haasteista. Pilvi- ja konesalikehitysten vaiheiden kulun välillä ei todettu suurta eroa, mutta pilvikehityksen todettiin hyötyvän erityisesti ketterän kehityksen metodien käytöstä. Muita eroja löydettiin tarvittavasta infrastruktuurista ja linsoinnista. Määrittelytyön ja lisensoinnin suunnittelun todettiin olevan erityisen tärkeää pilvipalveluita kehitettäessä. Case-esimerkin arkkitehtuuri tai kehitystavat eivät juuri vastanneet pilvinatiiveja kehitystapoja, koska ratkaisu on ensimmäinen organisaation kehittämä pilvipalvelu, ja koska ratkaisussa päädyttiin käyttämään alusta-arkkitehtuuria. Microsoftin alustan ja palveluiden käyttämisen todettiin myös tuovan etuja kehitystyöhön.

Tutkimuksessa korostui taustatyön tekemisen tärkeys sekä ketterän kehityksen edut. Pilvipalveluilla nähtiin olevan monia etuja verrattuna konesaliin kehitykseen, mutta myös haasteita, kuten lisensointi.

10.1 Työn rajoitteet

Työn suurin puute on, että se ei seurannut tuotekehitysprosessia loppuun asti. Koska case-esimerkkiä ei ole vielä julkaistu, tulokset eivät kerro pilvipalvelun tuotantoon viemisestä. Näin myös markkinoille vientiä tai ylläpitoa koskevat asiat jäivät käsittelemättä.

Toinen rajoite on suoritettujen haastatteluiden rajallinen määrä. Haastattelemalla useampia henkilöitä olisi voitu saada laajempi näkökulma hankkeeseen. Toisaalta 2M-IT:n PATA-tiimi ei ole kovin suuri ja kahdellakin haastattelulla saatiin kattavasti tietoa kehitystyöstä.

10.2 Jatkokehitys

Mielenkiintoinen työn jatkokehitysmahdollisuus voisi olla haastattelujen suorittaminen yrityksissä, jotka ovat keskittyneet pilvipalveluiden kehitykseen. Tällaisen yrityksen työtavat voisivat muodostaa mielenkiintoisen vertailukohteen tälle työlle. Myös IaaS-mallilla tuotettujen palvelujen tuotekehitys voisi olla hyvä tapa jatkaa työtä.

Toinen jatkotapa olisi suorittaa haastatteluja 2M-IT:llä muutaman vuoden päästä uudelleen, kun yhtiö on kehittänyt pilvipaluita enemmän. Tällainen tutkimus voisi antaa näkemystä siihen, mitä pilvipalveluiden tuottamiseen siirtyminen pidemmällä aikavälillä vaatii organisaatioilta, ja myös siihen, miten PATA-ratkaisun kehitys jatkuu, kun se siirtyy tuotantoon. Samoin tämä antaisi mahdollisuuden seurata, miten pilvinatiivin kehityksen tapoja otetaan ratkaisun kehityksessä käyttöön.

Lähteet

2M-IT, 2021. *Pata-hanke on asiakaspalvelun yhteinen ratkaisu.* [www-sivu]
Saatavissa: https://2m-it.fi/wp-content/uploads/2021/03/Esittely-2MIT-Pata-ratkaisu-04032021.pdf
[Viitattu 07.11.2021].

Agile Alliance, 2001. *Manifesto for Agile Software Development.* [www-sivu]
Saatavissa: https://agilemanifesto.org/
[Viitattu 11.10.2021].

Benfenatki, H. ym., 2017. MADONA: a method for automated provisioning of cloud-based component-oriented business applications. *Service Oriented Computing and Applications,* 11(1), pp. 87-100.

Drummmond, C., 2021. *Atlassian.* [www-sivu]
Saatavissa: https://www.atlassian.com/agile/scrum
[Viitattu 11.10.2021].

Erich, F. M. A., Amrit, C. & Daneva, M., 2017. A qualitative study of DevOps usage in practice. *Journal of Software: Evolution and Process,* 29(6), pp. e1885.

Grey, T., 2019. *5 principles for cloud-native architecture—what it is and how to master it.* [www-sivu]
Saatavissa: https://cloud.google.com/blog/products/application-development/5-principles-for-cloud-native-architecture-what-it-is-and-how-to-master-it
[Viitattu 15.10.2021].

Haikala, I. & Märijärvi, J., 2004. *Ohjelmistotuotanto.* 10. toim. Hämeenlinna: Talentum.

Hyvärinen, M., Nikander, P. & Ruusuvuori, J., 2017. *Tutkimushaastattelun käsikirja.* 1. toim. Tampere: Vastapaino.

Izrailevsky, Y., Vlaovic, S. & Meshenberg, R., 2016. *Completing the Netflix Cloud Migration.* [www-sivu]
Saatavissa: https://about.netflix.com/en/news/completing-the-netflix-cloud-migration
[Viitattu 8.10.2021].

Jaakkola, J. & Tunkelo, E., 1987. *Tuotekehitys – ideoista markkinoille.* 1. toim. Espoo: Weilin + Göös.

Jokinen, T., 2001. *Tuotekehitys.* 6. toim. Helsinki: Otatieto Oy.

Kangasniemi, H. & Lintulahti, M., 2017. *Mikä on pilvipalvelu?.* [www-sivu]
Saatavissa: https://elisa.fi/ideat/mika-on-pilvipalvelu/
[Viitattu 30.9.2021].

Kao, C. H., Liu, S. T. & Lin, C. C., 2014. Toward a Cloud Based Framework for Facilitating Software Development and Testing Tasks. *IEEE/ACM 7th International Conference on Utility and Cloud Computing,* pp. 491-492.

Lehtonen, T. ym., 2014. *Sulautettujen järjestelmienketteräkäsikirja.* 1. toim. Turku: University of Turku, Technology Research Center.

Mell, P. & Grance, T., 2011. *The NIST Definition of Cloud Computing.* [www-sivu]
Saatavissa:
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf
[Viitattu 30.9.2021].

Microsoft, 2021a. *Get started guide for Azure developers.* [www-sivu]
Saatavissa: https://docs.microsoft.com/en-us/azure/guides/developer/azure-developer-guide
[Viitattu 13.11.2021].

Microsoft, 2021b. *Azure global infrastructure.* [www-sivu]
Saatavissa: https://azure.microsoft.com/en-us/global-infrastructure/
[Viitattu 13.11.2021].

Microsoft, 2021c. *Business continuity and disaster recovery (BCDR): Azure Paired Regions.* [www-sivu]
Saatavissa: https://docs.microsoft.com/en-us/azure/best-practices-availability-paired-regions
[Viitattu 22.10.2021].

Microsoft, 2021d. *Overview of Azure subscriptions, management groups, and resources.* [www-sivu]
Saatavissa: https://docs.microsoft.com/fi-fi/learn/modules/azure-architecture-fundamentals/overview
[Viitattu 13.10.2021].

Microsoft, 2021e. *Microsoft 365 ja Office-sovellukset.* [www-sivu]
Saatavissa: https://www.microsoft.com/fi-fi/microsoft-365/
[Viitattu 13.11.2021].

Microsoft, 2021f. *Business Appliation Platform Microsoft Power Platform.* [www-sivu]
Saatavissa: https://powerplatform.microsoft.com/en-us/
[Viitattu 13.10.2021].

Microsoft, 2021g. *What is Dynamics 365?.* [www-sivu]
Saatavissa: https://dynamics.microsoft.com/en-us/what-is-dynamics365/
[Viitattu 13.10.2021].

Microsoft, 2021h. *Mitä on CRM?.* [www-sivu] Saatavissa:
https://dynamics.microsoft.com/fi-fi/crm/what-is-crm/ [Viitattu 2.12.2021]

Microsoft, 2021i. *Mikä ERP on ja miksi sitä tarvitaan .* [www-sivu] Saatavissa: https://dynamics.microsoft.com/fi-fi/erp/what-is-erp/ [Viitattu 2.12.2021]

Mojtaba, S., Babar, M. A., Mansooreh, Z. & Zhu, L., 2017. Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges. *International Symposium on Empirical Software Engineering and Measurement,* pp. 111-120.

Ranger, S., 2018. *What is cloud computing? Everything you need to know about the cloud explained.* [www-sivu]
Saatavissa: https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/
[Viitattu 30.9.2021].

RedHat, 2020. *IaaS vs PaaS vs SaaS.* [www-sivu]
Saatavissa: https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas
[Viitattu 12.10.2021].

RedHat, 2021. *What is containerization?.* [www-sivu] Saatavissa: https://www.redhat.com/en/topics/cloud-native-apps/what-is-containerization
[Viitattu: 13.11.2021]

Schwaber, K. & Sutherland, J., 2020. *Scrum guides.* [www-sivu]
Saatavissa: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100
[Viitattu 11.10.2021].

Sommerville, I., 2016. *Software Engineering.* 10. toim. Harlow: Pearson Education, Limited.

Tolvanen, P., 2011. *Käsitteet ojennukseen: Active Directory (AD), LDAP, SSO ja identiteetinhallinta.* [www-sivu]
Saatavissa: https://intranet-ostajanopas.fi/2011/04/29/kasitteet-ojennukseen-active-directory-ad-ldap-sso-ja-identiteetinhallinta/
[Viitattu 13.11.2021].

Vettor, R. & Smith, S., 2021. *Architecting Cloud Native .NET Applications for Azure.* 1.0.3 (2021-07-12) toim. Washington: Microsoft Developer Division, .NET, and Visual Studio product teams.

Haastatteluiden teemat ja kysymykset

Taulukko 2. Opinnäytetyön toteutuksessa tehdyn teemahaastattelun teemat ja esimerkkikysymyksiä teemoihin liittyen.

Teema	Kysymys
PATA	Voitko kertoa, mikä on ollut roolisi PATA-hankkeessa?
Pilvi	Miten ratkaisussa päädyttiin pilviratkaisuun?
	Onko PATA asiakkaalle SaaS- ja tekijöille PaaS-ratkaisu?
	Onko PATA julkisessa- vai hybridipilvessä?
	Mitä muita pilviratkaisuja olet tehnyt (itse tai 2M-IT:ssä)?
	Onko pilviratkaisun valinta vaikuttanut sisällön tuottoon, UI-suunniteluun, tai muuhun, mitä ei ehkä odottaisi?
	Miksi juuri Microsoft ja Azure?
	Mitä Azuren palveluja PATA-ratkaisussa on käytössä?
	Mitä muita Microsoftin palveluja PATA-ratkaisuun kuuluu?
	Entä muiden palveluntarjoajien pilvipalvelut, onko niitä mukana?
	Mikä on ollut haastavaa Microsoftin pilven kanssa?
	Miten lisensointi on toiminut? Kuinka aikaisin tuli ottaa huomioon?
Tuotekehitys	Onko ketterä kehitysmalli ja Scrum tukenut kehitystä?
	Kuinka paljon kaaviomainen kehitysmaali ylipäätään kulkee "peruskehityksessä" mukana?
Pilvikehitys	Voidaanko PATA-ratkaisun ajatella käyttävän mikroarkkitehtuurimallia?
	Onko automatisaatiota käytetty kehityksessä, tai onko sitä tarkoitus käyttää tulevaisuudessa esim. palvelun päivittämiseen?
	Onko DevOps-mallia käytetty?
Kaikki muu	Onko jotain muuta, mikä olisi hyvä muistaa tulevaisuudessa?
	Onko jotain muuta, mitä haluaisit mainita?