DEVOPS-MALLI YRITYKSEN ICT-YKSIKÖSSÄ

TyTA-kokonaisuus

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinnan korkeakoulukeskus
Tietojenkäsittelyn koulutusohjelma

Syksy, 2019

Henri Klemetti

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Hämeenlinnan korkeakoulukeskus

Tekijä Henri Klemetti **Vuosi** 2019

Työn nimi DevOps-malli yrityksen ICT-yksikössä, TyTA-kokonaisuus

Työn ohjaaja Lasse Seppänen

TIIVISTELMÄ

Opinnäytetyön tavoitteena on luoda ja kuvata DevOps-malli LähiTapiolan ICT -yksikössä TyTA–järjestelmäkehitysalueelle. TyTa-alueella työskentelee asiantuntijoita vakuutus- ja korvausjärjestelmien sekä raportoinnin parissa. Kyseisellä järjestelmäalueella tapahtuu tuotannonhoitoa ja kehitystä jatkuvasti, joten alueeseen sopii hyvin DevOps-periaatteet. Opinnäytetyössä tarkastellaan näitä periaatteita TyTA DevOps-tiimin muodostamisen alkuvaiheen kautta.

Opinnäytetyössä seurataan asioita, joista tiimin muodostaminen lähti liikkeelle ja ensimmäisiä askeleita kohti DevOps-käytäntöjen omaksumista. Tiimin muodostamista seurattiin observoimalla, eli havainnoinnin avulla. Prosessin aikana saatiin toimintaan TyTA DevOps -tiimi, joka hoitaa tuotannonhoitoa ja kehitystä yhdessä jakaen vastuualueita ristiin.

Hyötyinä koettiin muun muassa läpinäkyvyyden parantaminen sekä tehostunut priorisointi tehtävistä töistä. Haasteitakin toki koettiin alkumetreillä. Esimerkiksi ajan puute sekä kulttuurin muutos olivat osittain hidasteena tiimin kehittämiselle. Toimintamallit ovat herättäneet kiinnostusta muillakin järjestelmäalueilla yrityksessä, ja käytäntöjä voikin ottaa malliksi muiden alueiden prosessien suunnittelussa. Tässä opinnäytetyössä käydään läpi TyTA-tiimin alkuvaihetta. DevOps-mallin kehittäminen jatkuu opinnäytetyön jälkeenkin ja pyrkimyksenä on jatkuva kehittäminen.

Avainsanat DevOps, ketteryys, automaatio, kulttuuri

Sivut 39 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna University Centre

Author Henri Klemetti **Year** 2019

Subject DevOps model in the company's ICT unit, TyTA area

Supervisors Lasse Seppänen

ABSTRACT

The aim of this thesis is to create and describe DevOps-model in the Local-Tapiola ICT-unit for TyTA -systems area. There are experts working with insurance, claims and reporting systems in this work area. This system area is undergoing continuous production management and development therefore DevOps is well suited to this.

This thesis monitors how we started the team building and the first steps towards adopting DevOps practices. For the method of following the formation of the team, observation was used. During this process, the TyTA DevOps team launched and team handles production management and development works together by sharing responsibilities.

The team experienced that the benefits were improved transparency and prioritized work. There were also some challenges. For example, lack of time and cultural change were some of the obstacles to team's development. Operational models have also attracted interest in other areas of the systems within the company. These practices can be modeled as well on process planning in other system areas. This thesis goes through the first stages of the TyTA team. In any case, the development of the DevOps model goes on and the aim is continuous development.

Keywords DevOps, agile, automation, culture

Pages 39 pages

KÄSITTEET

AGILE	Ketterän kehityksen toimintamalli, johon kuuluu erilaisia prosesseja ja työkaluja.
AUTOMAATIOTESTAUS	Testitapausten automaattinen testaus, joka vähentää manuaalityötä.
BACKLOG	Lista tulevista töistä, joita priorisoidaan.
BUG BOUNTY -OHJELMA	Hakkerit voivat osallistua etsimällä järjestelmistä ja sovelluksista virheitä palkkioiden toivossa.
CONFLUENCE	Atlassian-ohjelmistoyrityksen tarjoama wikiohjelmisto organisaatioille dokumentointia varten.
DEV	Lyhenne englanninkielisestä sanasta development, joka tarkoittaa suomen kielessä kehitystä.
DEVOPS	Tulee sanoista Development ja Operations, eli kehitys ja ylläpito. Toimintamalli, joka kannustaa tuotantoa ja kehitystä työskentelemään tiiviissä yhteistyössä kohti samaa päämäärää. Joukko käytäntöjä, joka vähentää aikaa järjestelmämuutosten ja tuotantoon siirron välillä.
DEVSECOPS	Tulee sanoista Development, Security ja Operations, eli kehitys, turvallisuus ja ylläpito. Tuo tietoturva-asiat mukaan DevOps-käytäntöihin.
ICT	Information & Communications Technology, joka tarkoittaa suomen kielessä tieto- ja viestintätekniikka.
ITERAATIO	Scrum-mallissa tehdään töitä kiinteän pituisissa sprinteissä, jotka tarkoittavat iteraatiokierrosta.
JATKUVA INTEGRAATIO	Menetelmä löytää ohjelmistojen virheet mahdollisimman aikaisessa vaiheessa kehitysjaksoa ja varmistaa, että kaikki alustan osat toimivat keskenään oikein.

JATKUVA TOIMITUS	Menetelmät toimivan ja testatun ohjelmiston viemiseen tuotantoon pienissä erissä jatkuvasti.
JIRA	Atlassian-ohjelmistoyrityksen tarjoama tehtävienhallintaohjelmisto.
KANBAN	Kanban-taululla esitetään meneillään olevat työt korttien avulla. Kanbanin päämääränä on antaa tiimin jäsenille juuri tarpeeksi työtä, jotta tiimi työskentelee jatkuvasti kykyjensä mukaan.
KANPLAN	Hybridiversio Kanbanin ja Scrumin sekoituksesta. Sisältää Kanban-taulun ja erillisen töiden backlogin.
LEAN	Johtamisfilosofia, jonka ydinajatuksena on parantaa jatkuvasti valmistettavaa tuotetta sekä vähentää tuottamatonta toimintaa valmistusprosessin aikana. Pyritään parantamaan asiakastyytyväisyyttä, laatua, lyhentämään läpimenoaikoja tuotannossa sekä pienentämään toiminnan kustannuksia.
OPS	Lyhenne englanninkielisestä sanasta operations, joka tarkoittaa suomen kielessä tuotantoa.
REGRESSIOTESTAUS	Pyritään löytämään ohjelmistoregressioita. Ohjelman toimiva osa lakkaa regressiossa toimimasta. Regressiotestauksella varmistetaan, ettei mitään nykytoimintoja mene rikki, kun uusia ominaisuuksia lisätään.
SCRUM	Scrum-mallissa tehdään töitä kiinteän pituisissa sprinteissä. Sprintteihin aikataulutetut työt ovat tiimin tärkein prioriteetti.
SCRUMBAN	Hybridiversio Kanbanin ja Scrumin sekoituksesta. Sisältää Kanban-taulun ja erillisen töiden backlogin, sekä sprintit.
SKRIPTI	Lyhyt ajon aikana tulkattava ohjelma. Voidaan kutsua myös komentosarjaksi.
STORY POINTS	Arvioidaan työn koko arviointiasteikon avulla.

TESTAUSYMPÄRISTÖ	Testausympäristössä testataan järjestelmämuutoksia ennen tuotantoon vientiä.
TYTA	Lyhenne työtapaturma- ja ammattitautivakuutus sanoista. Työtapaturma- ja ammattitautivakuutus kuuluu suomalaiseen sosiaalivakuutusjärjestelmään ja on osa työsuhteeseen liittyvää sosiaaliturvaa. Tässä työssä TyTA sanaa käytetään myös järjestelmäalueen nimenä, johon opinnäytetyö tehdään. TyTA-alueeseen kuuluu vakuutus-, korvaus-, raportointi- sekä verkon sovelluksia.
TyTAL	Työtapaturma- ja ammattitautilain säädöskokoelmanumero on 495/2015. Lyhenne TyTAL.
USER STORY	Käyttäjätarinassa kerrotaan järjestelmän toiminnallisuutta ilman teknisiä termejä. Kirjoitettu selkeästi sillä tavalla, että asiakas ymmärtää asian.
VELOCITY	Kuvaa tiimin työskentelynopeutta.
WIP-RAJA	Keskeneräisten töiden määrää rajoitetaan jokaisessa työnkulun vaiheessa Kanban-taululla. Tästä käytetään lyhennettä WIP-raja (work in progress).

SISÄLLYS

1 JOHDANTO 1

2 DEVOPS 2

2.1 Historia 2
2.2 Jatkuva integraatio ja jatkuva toimitus 4
2.3 Kulttuuri työyhteisössä 5
2.4 Jatkuva dokumentaatio 8
2.5 Mittaaminen 9
2.6 DevSecOps 11

3 BACKLOGIN KÄYTTÄMINEN DEVOPSISSA 13

3.1 Suunnitelma suunnittelemattomille töille 13
3.2 Kanban 14
3.3 Scrum 15
3.4 Scrumban 16
3.5 Kanplan 16

4 AUTOMAATIOTESTAUS 17

4.1 Testiautomaation suunnittelu 17
4.2 Automaation kehittäminen ja käyttöönotto 18
4.3 Testiautomaation toteutus ja ylläpito 18

5 YRITYKSEN, AIHEEN JA METODOLOGIAN ESITTELY 19

5.1 Organisaation esittely 19
5.2 TyTA-järjestelmäalue 19
5.3 Opinnäytetyön aihe ja metodologia 20

6 DEVOPS-TIIMIN MUODOSTAMINEN JA KÄYTÄNTEET 22

6.1 Istumapaikat 22
6.2 Starttipäivä ja käytänteistä sopiminen 23
6.3 Dokumentaatio Confluenceen 25
6.4 Backlogien käyttöönotto Jirassa 26
6.5 Mittaristo ja metriikka 27
6.6 Testausautomaation kehittäminen 29
6.6.1 Regressiotestaus ja tuotannon varmentaminen 29
6.6.2 Hyväksymistestaus 29

7 JOHTOPÄÄTÖKSET DEVOPSISTA 31

7.1 Hyödyt 31
7.2 Haasteet 32
7.3 SWOT-analyysi 33
7.4 Tietoturvan huomioiminen 35
7.5 Jatkokehityskohteet 36

8 YHTEENVETO 37

LÄHTEET 38

1 JOHDANTO

DevOps on tällä hetkellä IT-alalla suosiota kasvattava toimintamalli. Termi tulee sanoista Development (kehitys) ja Operations (tuotanto). Tämä tarkoittaa, että ylläpito ja kehitys työskentelevät yhtenä tiiminä, eivätkä enää hoida siilomaisesti erillään omia työtehtäviään. Tämän opinnäytetyön tarkoituksena on kuvata, miten DevOps-malli otetaan käyttöön työtapaturma- ja ammattitautien (TyTA) järjestelmäalueella. Työskentelen LähiTapiola Palvelut Oy:ssa järjestelmäkehityksen ja ylläpidon parissa, johon teen myös opinnäytetyöni.

TyTA tulee sanoista työtapaturma- ja ammattitautivakuutus, joka on työsuhteisilla työntekijöillä osa sosiaaliturvaa ja suomalaista sosiaalivakuutusjärjestelmää. Korvattavat vahinkotapahtumat, vakuutuksesta maksettavat etuudet, järjestelmän toimeenpano sekä vakuutusten hinnoittelun periaatteet on säädetty laissa. Työtapaturma- ja ammattitautilain säädöskokoelmanumero on 495/2015 ja lyhenteenä käytetään TyTAL-sanaa. TyTAL-laki tuli voimaan 1.1.2016 ja sitä ennen sattuneisiin työtapaturmiin ja ammattitauteihin sovelletaan edeltäviä lakeja. (Tapaturmavakuutuskeskus, 11.03.2019)

LähiTapiolassa on jo lähdetty tekemään ketterää kehitystä projekteissa. Ketteryydestä tavoitellaan suurempia hyötyjä yhdistämällä tuotannonhoito ja kehitys yhdeksi tiimiksi DevOps-periaatteita noudattaen. Tähän asti järjestelmäkehitys ja tuotannonhoito on hoidettu omissa organisaatioissaan, mutta jatkossa TyTA-alueella toimisi yksi tiimi DevOps-mallin mukaan ja hoitaisi koko järjestelmäkokonaisuuden ylläpidon ja kehityksen. TyTA-järjestelmäkokonaisuuteen kuuluu vakuutus- ja korvausjärjestelmiä, raportointia sekä verkkopalveluiden töitä. Kyseinen järjestelmäkokonaisuus sopii DevOps-malliin hyvin, koska siinä tapahtuu koko ajan sekä kehitystä, että ylläpitoa. TyTA-alueelle tulee vaatimuksia liiketoiminnan lisäksi lainsäädännöstä ja eri viranomaistahoilta. Työ rajataan koskemaan DevOps-mallin käyttöönottoa ja tiimiytymisen alkuvaihetta ICT -organisaatiossamme.

Tutkimuskysymyksinä opinnäytetyössäni ovat:

Pääkysymys

- Millä keinoilla DevOpsia voidaan toteuttaa organisaatiossa TyTA-järjestelmäalueella?

Alakysymykset

- Mitkä ovat DevOps-mallin hyödyt ja haitat verrattuna perinteiseen ohjelmistokehitykseen?

- Miten erilaisia hallintajärjestelmiä voidaan hyödyntää DevOps-mallissa?

2 DEVOPS

DevOps on joukko käytäntöjä, jotka pyrkivät vähentämään aikaa järjestelmän muuttamisen ja tuotantoon siirtämisen välillä. Samalla on tarkoitus varmistaa järjestelmätöiden korkea laatu. Näihin DevOpsin käytäntöihin sisältyy neljä pääpiirrettä, jotka ovat nopeampi muutosten tuotantoon siirtäminen, virheiden löytäminen automaatiotestauksen avulla, virheiden vähentäminen käyttöönoton aikana sekä järjestelmän vikojen nopea löytäminen ja korjaaminen. (Bass, 2017, s. 8-10)

DevOpsissa kehittäjät osallistuvat yhdessä koko palvelun elinkaareen suunnittelusta kehitysprosessiin ja siitä tuotannon tukemiseen. DevOpsille on ominaista, että tuotannonhoidon henkilöt käyttävät samoja tekniikoita kuin kehittäjät heidän järjestelmätöissään. DevOps on moniulotteinen malli, joka kattaa paljon eri asioita. DevOps voidaan määritellä olevan muun muassa kehittäjien ja tuotannonhoidon yhteistyötä, automaatiota, Kanbanin käyttämistä, uusia työkaluja tai kulttuuria. (Ernest, 2010)

Tässä luvussa on tarkoitus avata DevOps-termiä tarkemmin. Luvussa kerrotaan yleisesti, mitä DevOps tarkoittaa sekä historiaa, miten DevOps-menetelmät syntyivät ja mihin tarpeisiin. Luvussa kerrotaan myös, mitä tarkoittaa jatkuva integraatiota ja jatkuva toimitus, sekä esitellään, mitä kulttuuri merkitsee DevOpsissa.

2.1 Historia

DevOpsin edelläkävijöitä ovat Lean-menetelmät ja ketterä kehitys (Agile). Lean-menetelmiä käytettiin alun perin autoteollisuudessa ensin Henry Fordin ja myöhemmin Toyotan toimesta. Lean-menetelmän ydinajatuksena oli parantaa jatkuvasti valmistettavaa tuotetta sekä vähentää tuottamatonta toimintaa valmistusprosessin aikana. Vuonna 2001 julkaistiin ketterän ohjelmistokehityksen julistus (Agile Manifesto). Sen julkaisivat 17 merkittävää ketterän kehityksen puolestapuhujaa. Tarkoituksena oli päästä pois jäykästä, vesiputousmallisesta sekä raskaasti dokumentoidusta ohjelmistokehityksestä, joka johti usein ohjelmistoprojektien myöhästymiseen, budjetin ylittymiseen ja ongelmiin virheiden vuoksi. (Sharma, 2017, s. 2-6)

Sharman (2017) mukaan ketterästä kehityksestä tuli keskeinen tekijä DevOps-käytäntöjen tarpeelle. Kehittäjien alettua toimittaa ketterän kehityksen keinoilla koodia nopeammin, koodi piti myös testata nopeammin. Asentaminen kehitys- ja testausympäristöihin piti tehdä myös aiempaa nopeammin ja viedä entistä useammin myös tuotantoon. Tuotantotiimit eivät olleet vielä valmiita näin nopeaan kehittämiseen, mikä johti siihen, että kehitystöiden siirtämisestä testausympäristöihin tuli merkittävä pullon-

kaula ohjelmistokehityksessä. Oikeita testausympäristöjä ei ollut käytettävissä tarvittaessa, ja mikä vielä pahempaa kehitystöiden tuotantoon vienti ei onnistunut milloin tahansa, vaan tuotantoon viennin piti tapahtua tiettyyn aikaan julkaisuviikonloppuisin. Nopeampi koodin kehittäminen lyhyissä iteraatioissa toi tarpeen kehitys- ja tuotantotiimien yhteistyölle.

DevOps-suuntaus alkoi John Allspaw'n ja Paul Hammondin (työnantaja tuolloin Flickr/Yahoo) pitämästä mullistavana pidetystä keskustelusta Velocity 2009 -konferenssissa O'Reillyssä. Puheen otsikko oli "10+ Deploys Per Day: Dev and Ops Cooperation at Flickr". Tuohon aikaan yli kymmentä käyttöönottoa päivässä ei pidetty mahdollisena. (Sharma, 2017, s. 2-6)

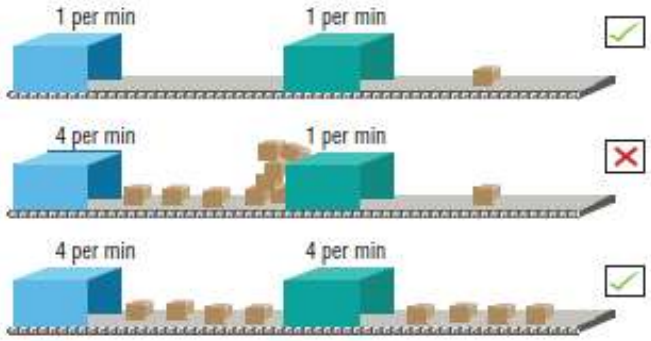
Samana vuonna 2009 Patrick Debois piti ensimmäisen aiheeseen liittyvän DevOpsDays-konferenssin Gentissä Belgiassa. Tämän jälkeen käsite, jolle oli nyt nimi, alkoi saada valtavaa kiinnostusta muissakin maissa. Patrick Deboisin mielestä DevOps syntyi reaktiona olemassa olevista käytännöistä johtuviin siiloihin ja joustamattomuuteen. Muun muassa automaation kasvaminen, lisääntyneet työkalut ja ohjelmistot, ketterät prosessit ja kehityksen sekä tuotannonhoidon yhteistyötarve vaativat uusia toimintamalleja vanhojen tilalle. (Ernest, 2010)

Alun perin DevOps-menetelmät herättivät kiinnostusta Start Up -yrityksissä sekä erityisesti organisaatioissa, jotka toimittivat verkkosovelluksia. Nämä sovellukset oli luotu kehittäjien toimesta (Dev), jotka toimittivat muutokset ja päivitykset verkkosovelluksiin hyvin nopealla tavalla. Suurin este nopealla käyttöönotolla oli tuotantopuoli (Ops), joka oli hidas muutosten käyttöönotossa, koska siellä oli jäykät ja tiukat muutostenhallintaprosessit. DevOps-suuntauksen ideana oli korjata tämä hidaste kehityksen ja tuotannonhoidon välillä ja edistää viestintää, yhteistyötä ja luottamusta. Kehityksen ja tuotannonhoidon välinen kulttuurinmuutos oli DevOpsin ydinajatus, jotta sovellusten toimitus olisi nopeampaa, tehokkaampaa ja jatkuvaa. Start Up -hankkeet ja uusien innovaatioiden kärjessä olevat yritykset käyttivät DevOps-käytäntöjä ilman suuria, monimutkaisia vanhoja järjestelmiä ja niiden ylläpitoa. (Sharma, 2017, s. 2-3)

Toimintamalli ei levinnyt heti suurten yritysten keskuuteen. Suuret yritykset kuitenkin näkivät mitä aloittelevat yritykset saavuttivat DevOpsilla ja alkoivat miettiä, kuinka voisivat itsekin hyötyä menetelmistä. Todellinen kasvu DevOpsille suuryrityksien parissa alkoi vuonna 2012, kun yritykset, kuten IBM, tulivat mukaan heidän ensimmäisellä, vaikkakin lyhytikäisellä jatkuvan toimituksen kokeilullaan. Monet konsulttifirmat, kuten IBM ja ThoughtWorks alkoivat myös tarjota konsultointipalveluita DevOpsista etenkin suuryrityksille, jotka halusivat omaksua DevOps-toimintatavat. (Sharma, 2017, s. 2-3)

2.2 Jatkuva integraatio ja jatkuva toimitus

Epäonnistuneet muutosten tuotantoon siirrot toivat esiin tarpeen tarjota kehittäjillekin pääsy tuotantomaisiin ympäristöihin. Suurin ongelma aikaisemmassa toiminnassa oli tehostaa vain yhtä osaa prosessista eli kehityspuolta. Tämä loi merkittäviä pullonkauloja testaus- ja tuotantopuolelle. Jos ajatellaan sovelluskehitys- ja toimitusprosessia tehtaan kokoonpanolinjastona, vain yhden linjaston kohdan nopeuttaminen laitteiden tuottamisessa ei auta, jos linjaston myöhemmät kohdat toimivat edelleen hitaammin (kuva 1, keskimmäinen linjasto). Linjaston loppupäähän kasautui enemmän töitä. Tämä kuvaa sitä, että kehitystiimi tuottaa paljon uutta koodia ohjelmistoon, mutta tuotantotiimi ei kerkeä viedä muutoksia aiempaa nopeammin tuotantoon. Aiemmin vesiputousmallisessa ohjelmistokehityksessä tämän kanssa ei ollut ongelmaa, kun kehitettiin yksi kokonaisuus kerrallaan ja se vietiin tuotantoon kerralla (kuva 1, ylin linjasto). Kuvassa 1 alimmalla linjastolla tuotantopuoli on kehittynyt nopeammaksi kuin aikaisemmin ja se pystyy toimittamaan kehityspuolen tuottamia muutoksia tuotantoon jatkuvasti eli pullonkaulat on saatu poistettua. (Sharma, 2017, s. 6-7)



Kuva 1. Jatkuvan toimittamisen pullonkaulat. (Sharma, 2017, s. 6)

Painopiste siirtyi minimoimaan toimitusjaksoihin kuluvaa aikaa. Alkaen vaatimuksesta tai käyttäjätarinasta (user story) kykyyn saada muutos mahdollisimman nopeasti integroituna, testattuna ja valmiina asiakkaan käyttöönottoon. Tämä johti DevOpsin kahden jatkuvan ydinominaisuuden kehittymiseen: jatkuvaan integraatioon (periytyi ketterästä kehityksestä) ja jatkuvaan toimitukseen. (Sharma, 2017, s. 6-7)

Jatkuvassa integraatiossa kehittäjä tekee koodimuutoksen ja sille ajetaan joukko automaatiotestejä ennen kuin koodi jaetaan muiden kehittäjien ja testaajien kanssa ja yhdistetään muun koodin kanssa. Testit sisältävät myös regressiotestejä, jotka varmistavat, ettei mitään nykytoimintoja mene rikki, kun uusia ominaisuuksia lisätään. Tämä toimintamalli kannustaa kehittäjiä tekemään pieniä muutoksia kerrallaan ja yhdistämään ne

usein olemassa olevaan koodiin. Jatkuva integraatio tapahtuu ennen jatkuvaa toimitusta tai jatkuvaa käyttöönottoa. (Boström, 2019)

Jatkuvalla toimituksella varmistetaan, että uudet koodimuutokset saadaan julkaistua asiakkaalle mahdollisimman nopeasti ja vakaasti. Automaattisen testauksen lisäksi koko julkaisuprosessi on automatisoitu ja käyttöönotto voidaan tehdä milloin tahansa nappia painamalla. Teoriassa jatkuvan toimituksen avulla muutoksia voidaan julkaista päivittäin, viikoittain, joka toinen viikko tai mikä tahansa sopiikaan parhaiten liiketoiminnan tarpeisiin. Pittetin (2019) mukaan parhaat hyödyt jatkuvasta toimittamisesta saadaan kuitenkin viemällä muutoksia tuotantoon niin pian kuin mahdollista. Näin varmistetaan, että julkaistaan pieniä eriä muutoksia ja joiden viat on helpompi selvittää ongelmien ilmetessä. (Pittet, 2019)

Jatkuva käyttöönotto menee jatkuvaa toimitusta vielä askeleen pidemmälle. Tässä käytännössä kaikki muutokset, jotka läpäisevät automaattiset testaukset ja vaiheet, julkaistaan tuotantoon asiakkaille käytettäväksi. Ihmisten väliintuloa ei ole, vaan ainoastaan epäonnistunut testitapaus estää muutoksen viemisen tuotantoon. Jatkuva käyttöönotto on hyvä tapa nopeuttaa palautteen antamista asiakkaan kanssa ja vähentää tiimiltä paineita, kun erillisiä julkaisupäiviä ei enää ole. Kehittäjät voivat keskittyä ohjelmistojen rakentamiseen ja he näkevät työnsä tuloksen tuotannossa lähes saman tien, kun ovat saaneet työnsä tehtyä. Jatkuvan toimituksen ja jatkuvan käyttöönoton erona on siis se, että jatkuvassa toimituksessa käyttöönotto hyväksytään manuaalisesti. Jatkuvassa käyttöönotossa tämä tapahtuu automaattisesti. (Pittet, 2019)

2.3 **Kulttuuri työyhteisössä**

Uusien työkalujen käyttöönotto ja yhdistäminen on helpompaa kuin organisaation kulttuurin muuttaminen. DevOps tarkoittaa pohjimmiltaan kulttuurinmuutosta. Organisaatioilla on luontainen vastustus muutoksille. Muutos ei ole helppoa, etenkään suurissa organisaatioissa, joissa tietynlainen kulttuuri on kehittynyt vuosien saatossa satojen tai tuhansien työntekijöiden keskuuteen. Työntekijät yksilöinä saattavat käsittää DevOps-käytäntöjen arvon, mutta yhteisön tapojen muuttaminen tuo hitautta muutokselle. Muutoksen hitauden voittaminen DevOps-kulttuurin omaksumiselle avainasia. Kulttuurinen hitaus tulee esiin muun muassa kuvassa 2 esitettyjen tutun kuuloisten lauseiden kautta. Lauseet ovat kärjistyksiä tietynlaisesta muutosvastaisesta puhetavasta, eivät suoria lainauksia.

Kuva 2. Kulttuurillista hitautta kuvaavia lauseita. (Sharma, 2017, s. 36-37)

Ajan myötä organisaatiot kehittävät käyttäytymistapojaan. Tiimit jakautuvat ajan saatossa keskenään ja vastuutkin jakautuvat organisaatiorakenteiden mukaisesti. Erilaisia prosesseja on olemassa, mutta kukaan ei välttämättä tiedä miksi – ne vain ovat olemassa. Saatetaan tuottaa raportteja, joita kukaan ei enää lue, mutta kukaan ei halua poistaa niitä. Aiemmin on saattanut tapahtua virheitä ja niiden seurauksena yritetään varmistaa, että samat virheet eivät toistu. Kaikki nämä käytännöt rakentavat hitautta organisaation kulttuurin muuttamisessa. (Sharma, 2017, s. 36-37)

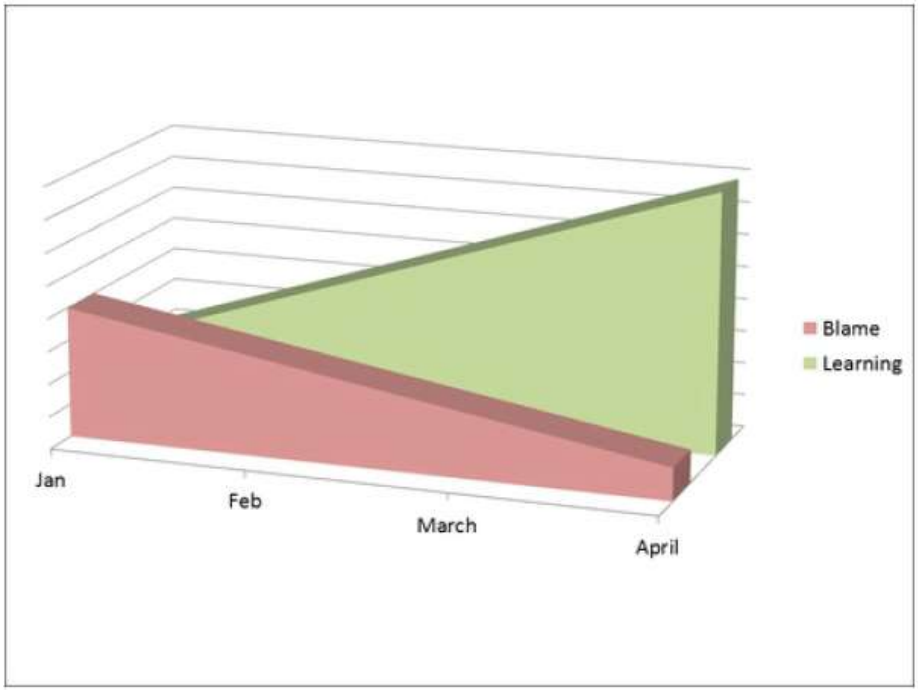
Millaista kulttuuria DevOps-käytäntöjen omaksumisessa tarvitaan? Sharman (2017, s. 36-37) mukaan ainakin luottamusta, viestintää ja yhteistyötä. Kulttuurinen hitaus voidaan voittaa käsittelemällä kolmea aluetta, jotka ovat näkyvyys, tehokas viestintä ja yleiset mittaukset. Tiimit ja työntekijät tarvitsevat näkyvyyttä koko toimitusputkeen. Luottamusta lisää kaikkein eniten asioiden täydellinen näkyvyys. Esimerkiksi testaajalla on näkyvyys siihen, miten kehittäjä on hoitanut yksikkötestauksen ja kehittäjä tietää, että hän ei voi tehdä koodia ilman asianmukaista yksikkötestiä. Sharman (2017, s. 36-37) mukaan viestinnän on oltava suoraa, ei sähköpostitse, tikettien tai hallinnon kautta. Jokaisen pitäisi pystyä kommunikoimaan kenen tahansa kanssa, ilman hierarkiaa. Reaaliaikaisten viestintäkanavien tulisi korvata sähköposti. Kaikista eniten hitautta aiheuttaa oikeiden mittareiden puuttuminen tiimeissä ja työntekijöillä. Työntekijät eivät muuta heidän käyttäytymistään, ellei tapa, jolla heitä mitataan, vastaa uutta haluttua käyttäytymistä. (Sharma, 2017, s. 36-37)

Luottamuksen ja yhteisöllisyyden tunne on kaiken perusta DevOps-käytäntöjen onnistumiselle ja menestymiselle. Kaikki alkaa siitä, kuinka ihmiset ymmärtävät toisiaan. Onko yrityksellä ”me vs. he” -kulttuuri vai ”me”-kulttuuri? DevOps keskittyykin jakamisen käsitteeseen: jaetaan ideoita, kysy-

myksiä, prosesseja, työkaluja ja tavoitteita. DevOps tiimissä jäsenet työskentelevät yhteistyössä saavuttaakseen yhteiset tavoitteet. Sopiva tiimin koko on korkeintaan kymmenen jäsentä. Ryhmän optimaalinen koko riippuu tehtäväalueesta ja sen monimutkaisuudesta. Liian monta jäsentä monimutkaistaa tiimin rakentamista ja lisää väärinkäsityksiä. (Hüttermann, 2012, s. 66; Swartout, 2012, s. 99-100)

Tiimin käyttäytymiseen vaikuttaa monet asiat, kuten millainen fyysinen ympäristö on, istumapaikat, seinät toimistojen välissä tai jakavatko työntekijät saman äidinkielen. Avoin tila toimii yhdistävänä tekijänä. Ihanteellinen työyhteisö on sellainen, jossa on mahdollisuus mennä keskustelemaan suoraan ihmisille, joiden kanssa tekee yhteistyötä. Kaikille tämä ei ole mahdollista ja tiimit saattavat olla fyysisesti erillään toisistaan eri toimistoilla, lentomatkan päässä tai jopa eri aikavyöhykkeillä. Tarkoituksena on poistaa fyysiset ja virtuaaliset esteet ihmisten väliltä. Esimerkiksi yhteiset kokoontumiset kahvittelun merkeissä, palaverit, muuttaminen toimistolla ja juttelu ilman sähköpostia ovat hyviä käytäntöjä. Jokaisen työyhteisön tulisi itse miettiä parhaat käytännöt, jotka sopivat heille. (Hüttermann, 2012, s. 66; Swartout, 2012, s. 99-100)

Swartoutin (2012, s. 105-107) mukaan kulttuurin muuttamisessa teot puhuvat enemmän kuin sanat. Syyttelykulttuuri voi nopeasti heikentää hyvää työtä, joka on tehty avoimuuden, rehellisyyden, yhteistyön, innovaatioiden ja vastuullisuuden kulttuurin edistämiseksi. Jos työpaikalla osoitellaan sormella työntekijöitä, kun asiat menevät pieleen, on vaikeaa luoda työskentely-ympäristö, jossa ihmiset ovat valmiita laittamaan itsensä likoon ja kokeilemaan uusia asioita. Ihannetilanne on työympäristö, jossa virheiden tapahtuessa, yksilöitä rohkaistaan oppimaan virheistä ja miettimään toimenpiteitä, ettei samat virheet toistu ja jatkamaan eteenpäin. Ihmisiä on myös rohkaistava puhumaan kokemuksistaan ja havainnoistaan muiden kanssa. (Swartout, 2012, s. 105-107)



Kuva 3. Syyttely vähenee ja virheistä oppiminen kasvaa. (Swartout, 2012, s. 107)

Kuvassa 3 on havainnollistettu sitä, että kun syyttelemisen ilmapiiriä saadaan kitkettyä, niin oppiminen alkaa suhteessa kasvaa merkittävästi. Ihmisten ei tarvitse enää pelätä virheitä, joita sattuu kaikille.

2.4 **Jatkuva dokumentaatio**

Tuottavuus on avainasemassa nykyaikaisessa ohjelmistojen toimitusputkessa, DevOpsissa. IT-dokumentointia ei kuitenkaan perinteisesti liitetä tuottavuuteen. Monet ajattelevat, että dokumentaatio vain hidastaa DevOpsia ja tämänkaltaisia metodologioita. Tuotantopuoli vaatii dokumentaatiota, mutta usein dokumentaatio on kirjoitettu myöhemmin ja sitä on hallittu vain projektin käyttöönoton alussa ja lopussa. Dokumentaation luominen on perinteisesti tapahtunut vesiputousmallin tapaisesti suurina määrinä kerrallaan. Tällöin dokumentaatio saattaa jäädä jälkeen kehityksestä, kun seuraava päivitys tehdään vasta pitkän ajan kuluttua. Vesiputousmalliseen ohjelmistokehitykseen tällainen tapa sopi paremmin, koska ohjelmistokin luotiin hyvissä ajoin ennen käyttöönottoa, eikä se muuttunut enää kovin usein. Tällöin dokumentaatiokin pysyi ajan tasalla paremmin. Ketterässä kehityksessä, jossa kehitystä tehdään jatkuvasti pieninä palasina, tämänkaltainen dokumentaatio tuottaa haasteita. (Riley, 2016)

Vaikka perinteinen dokumentaatio ei vastaakaan nykyaikaisen kehityksen vaatimuksia, dokumentoinnista luopuminen on mahdotonta. Käynnissä

olevat, automatisoidut prosessit kietovat modernin dokumentaation DevOps-käytäntöihin ja estävät dokumentaatiota tulemasta pullonkaulaksi nopeassa julkaisurytmissä. Kuten perinteinen dokumentaatio mukautui vesiputousmallin mukaisiin julkaisuihin, tekee moderni dokumentaatio saman DevOpsissa. Se mukautuu sovellusten jatkuvaan toimittamiseen ja käyttöönottoon ”jatkuvana dokumentaationa”. Moderni dokumentaatio tulee aktiiviseksi osaksi ohjelmiston toimitusputkea ja parantaa valvontaa, mitattavuutta ja reagointikykyä. (Riley, 2016)

Ketterässä toimintatavassa dokumentaation tekijät liitetään mukaan projektiin heti alusta alkaen. Tätä kutsutaan jatkuvaksi dokumentaatioksi. Uusissa tuotteissa, missä dokumentaatio täytyy aloittaa tyhjästä, jatkuva dokumentaatio varmistaa, että suuri määrä dokumentaatiota kirjoitetaan, katselmoidaan ja toimitetaan ajoissa. Kun kirjoittajat ja kehittäjät työskentelevät yhteistyössä iteratiivisessa prosessissa, he voivat oppia toisiltaan ja tehdä koko prosessin tehokkaammaksi. Esimerkiksi, jos dokumentaation kirjoittajalla on paljon kysymyksiä jostain sovelluksesta, kehitystiimi voi vastata näihin kysymyksiin saman tien välttääkseen asioiden pompottelua edestakaisin. Dokumentaation tekijöiden varhainen osallistuminen työhön on myös loistava tapa saada palautetta suunnitelmista. Jos dokumentaatioryhmä ei tajua suunnitelmia, todennäköisesti asiakaskaan ei niitä tajua. (Thompson, n.d.)

Helpoin osuus on määritellä, että dokumentaatio tulisi kirjoittaa ketterillä menetelmillä. Vaikeampi osuus onkin varata tähän työhön resursseja, budjettia ja aikaa tarpeeksi. (Thompson, n.d.)

2.5 **Mittaaminen**

Monesti organisaatiot keskittyvät DevOps-muutoksissa teknologiaan ja kulttuurisiin asioihin. Mittaus on DevOps-muutoksissa kuitenkin merkittävä osa-alue, kun se tehdään oikealla tavalla. (Forsgren, n.d.) Mittareiden tulisi olla samat kaikilla tiimin jäsenillä, jotta saadaan aikaan todellista yhteistyötä ja tunne yhdestä ryhmästä, joka työskentelee kohti samoja tavoitteita siilojen yli. Kehitys, testaus, ja tuotanto tarvitsevat yhteiset tai ainakin samanlaiset metriikat, joilla heidän toimintaansa mitataan. (Sharma, 2017)

Oikeiden mittareiden ja mittaustapojen avulla voidaan parhaiten hyödyntää tiimin oivalluksia ja kokemuksia sekä muuttaa tarvittaessa suuntaa. Pienet parannukset ovat avain isompiin saavutuksiin ja kun mittarit ovat kunnossa, ollaan valmiita hyödyntämään näitä parannuksia. Kun lähdetään miettimään sopivia mittareita, kannattaa miettiä ensin mitä halutaan mitata. On tunnistettava yhdessä tiimin kanssa tavoitteet esimerkiksi kuukaudelle tai vuosineljännekselle. Näitä tavoitteita käytetään tunnistamaan

lopputulokset ja ne asiat, mitkä tulisi johtamaan toivottuun tulokseen. Seuraavaksi mietitään mittarit, jotka mittaavat näiden asioiden toteutumista. Tämä voi olla aluksi vaikeaakin, mutta tulee helpottumaan kokemuksen kautta. Tiimille selkeytyy samalla mihin työskentely tähtää ja miksi ja kuinka sitä mitataan. (Forsgren, n.d.)

Jatkuvan kehittymisen ja parantamisen todistaminen on vaikeaa ilman kerättyä dataa. Suorituskyvyn mittaamiseen on olemassa paljon erilaisia työkaluja ja tekniikoita, kuten kuinka kauan käyttäjät käyttävät järjestelmää, tuottiko jokin asia myyntiä ja kuinka usein kriittiset hälytykset ilmestyvät lokille. Vaikka kaikkea voisikin mitata, ei ole tarkoitus mitata kaikkea mahdollista, vaan perusasioista on hyvä lähteä liikkeelle. Kuinka kauan kesti kehitystyön aloittamisesta tuotantoon siirtoon? Kuinka usein virheitä tai häiriöitä ilmenee ohjelmissa? Kuinka kauan järjestelmän korjaamisessa menee aikaa häiriön jälkeen? Kuinka monta ihmistä käyttää järjestelmää juuri nyt? Kuinka monta käyttäjää saavutit tai menetit tällä viikolla? (Atlassian, 2019c)

Jatkuva mittaaminen johtaa jatkuvaan prosessien parantamiseen, kuten DevOps-käytännöt suosittavat. On hyvä pitää tiettyjä raja-arvoja mittareissa, milloin pitää tulla hälytys tai tiimin pitää parantaa toimintaansa. Mittausprosessit kannattaa ottaa käyttöön asteittain ja miettiä aluksi, joku sopiva mittari, jonka voi ottaa helposti käyttöön. (Philippot, 2017)

Mittareiden avulla saadaan kokonaisvaltainen näkymä nykytilasta, mihin ollaan menossa, mihin voidaan mennä ja kuinka näihin tavoitteisiin päästään. DevOps-mittarit mittaavat yleensä muun muassa kehittämisen, käyttöönottojen ja asiakasvastausten nopeutta, käyttöönottojen määrän tiheyttä, virheitä, korjausaikaa, korjausten määrää sekä näiden mittareiden muutosnopeutta. DevOps-mittareiden kohteena on yleensä käyttöönotot, tuotanto ja tukitoiminnot, koska suurin osa DevOpsiin liittyvistä töistä tapahtuu näillä alueilla. Useimmat DevOpsiin liittyvät mittarit voidaan jakaa kolmeen taulukossa 1 kuvattuun luokkaan eli ihmisten, prosessien ja tekniikan mittaamiseen. Monet DevOps mittaristot sisältävät enemmän tai vähemmän näitä kaikkia kolmea luokkaa. (Riley, 2015)

Taulukko 1. DevOps-mittareiden jakaminen. (Riley, 2015)

IHMISET	PROSESSIT	TEKNIIKKA
Ihmiset ovat olennainen osa mitä tahansa DevOps-prosessia. Ihmislähtöinen mittari mittaa esimerkiksi vaihtuvuutta, suorituskykyä ja vasteaikaa. Ihmiset ovat vaikein osa mitattavuuksissa ja heidän vaikutustaan toimintoihin on joskus vaikea havaita.	Jollain tavalla kaikki DevOpsiin liittyvä on prosessia. Jatkuvat käyttöönotot, tuotannonhoito ja tukitoimintojen syklit ovat jatkuva sarja erilaisia prosesseja. Mutta jotkut mittarit ovat selvästi prosessiin liittyvämpiä kuin toiset, kuten jatkuva toimittaminen, reagointi ja korjaustoimenpiteet. Esimerkiksi kehityksestä käyttöönottoon kuluva aika on suurelta osin prosessia, kuten myöskin käyttöönottojen tiheys ja vasteaika. Prosessimittarit voivat mitata nopeutta (missä voi olla pullonkauloja, vai onko prosessi itsessään pullonkaula?), tarkoituksenmukaisuutta (onko kaikki vaiheet merkityksellisiä?), tehokkuutta (ovatko kaikki vaiheet ihanteellisessa järjestyksessä?).	Teknologisilla mittareilla on suuri rooli DevOps:ssa. Mittaamalla muun muassa käyttöaikaa (kuinka monta prosenttia ajasta järjestelmä on käytössä?) ja epäonnistumisastetta (montako prosenttia epäonnistuneita käyttöönottoja ja muutoksia on ollut?).

2.6 **DevSecOps**

Tietoturva jää helposti huomioimatta, samalla kun kyky uusien sovellusten ja ominaisuuksien käyttöönottoon nopeutuu ja laajentuu. Turvallisuutta ei voida kuitenkaan ottaa huomioon vasta kehityksen loppuvaiheessa. Tietoturva tulee huomioida DevOpsissa integroimalla tietoturvatyökalut ja prosessit DevOpsin toimitusputkeen. Tärkeimmät tietoturvaan liittyvät tehtävät on liitettävä jo varhaisessa vaiheessa ohjelmistokehityksen elinkaareen. Tietoturvavikoja seurataan ja korjataan koko sovelluksen elinkaaren ajan sekä kehitys- että ylläpitotöissä. (Moore & Bovoso, 2018)

DevSecOps vastaa näihin tarpeisiin sisällyttämällä tietoturvan kehitys- ja tuotantoprosesseihin. DevSecOps-lähestymistavassa tietoturvaan liittyvät toimintamallit integroidaan osaksi DevOps-prosessia. DevSecOps edistää joustavaa yhteistyötä kehityspuolen (Dev), tietoturvan (Sec) ja tuotannon (Ops) välillä. DevSecOpsin päätavoite on kaventaa ja lopulta poistaa mahdolliset kuilut ohjelmistokehityksen ja tietoturvan väliltä varmistaen samalla koodin nopean toimituksen ja toteutuksen turvallisella tavalla.

Turvallisuuden ei pidä olla vain tietoturvaosastojen tai tietoturvatiimien vastuulla. Projektien elinkaaren aikana monissa vaiheissa tulisikin tunnistaa tietoturvaan ja yksityisyyteen liittyvät perusvaatimukset etenkin, jos asiat ovat säänneltyjä. Esimerkiksi miten käsitellään asiakastietoja, milloin käytetään vahvaa tunnistautumista? Mitä tietoa ja mitkä tapahtumat voi olla kirjattuna lokiin ja mitkä pitää olla lokilla? Luettelo on pitkä. Projekteissa onkin hyvä olla erityinen rooli tietoturvan varmistamista varten. (LocalTapiola, 2019)

Aito siirtyminen DevSecOps-käytäntöihin edellyttää kulttuurimuutosta organisaatiossa. Innovaatioiden estämisen sijaan turvallisuuden korostaminen voi nopeuttaa innovaatioiden toteutumista. Turvallisuusasiat voivat nimittäin vähentää todennäköisyyttä joutua korjaamaan haavoittuvuuksia elinkaaren myöhemmissä vaiheissa. (Matthews, 2018)

3 BACKLOGIN KÄYTTÄMINEN DEVOPSISSA

Suunnittelematon työ on arkipäivää, jota jokainen tiimi kohtaa. Suunnittelemattomalla työllä on vaikutusta tiimin tuottavuuteen. Vakiintuneilla prosesseilla ja selkeällä priorisoinnilla backlogien avulla, Dev (kehitys) ja Ops (tuotanto) -tiimit voivat hallita suunnittelematonta työtä paremmin, keskittyen samalla jo suunniteltuun käynnissä olevaan työhön.

3.1 Suunnitelma suunnittelemattomille töille

Suunnittelemattoman työn siirtäminen ja priorisointi eri tiimeissä ja järjestelmissä on tehotonta ja häiritsee meneillään olevaa työtä. Lisääntyneen näkyvyyden ja ennakoitavuuden avulla tiimit voivat paremmin ennakoida ja jakaa suunnittelematonta työtä keskenään. (Atlassian, 2019c)

DevOps-yhteisössä ketterät toimintatavat tuntevat työntekijät tietävät, että Scrum on hyödyllinen tapa suunnitellun työn seuraamiseen. Osa tuotannonhoidon töistä voidaan suunnitella, esimerkiksi ison järjestelmämuutoksen julkaisu tai järjestelmäpäivitysten suorittaminen. Suuri osa tuotannonhoidon töistä on kuitenkin suunnittelematonta, kuten suorituskyvyn nousut, järjestelmän katkokset ja turvallisuuden vaarantuminen. Nämä tapahtumat vaativat välitöntä reagointia. Ei ole aikaa odottaa näiden priorisointia backlogilla tai seuraavassa sprintin suunnittelussa. Tästä syystä monet DevOps-mallilla työskentelevät tiimit käyttävät Scrumin lisäksi myös Kanbania. Tämä auttaa heitä seuraamaan molemman tyyppisiä töitä ja auttaa ymmärtämään niiden välistä vuorovaikutusta. Tiimit voivat myös omaksua hybridilähestymistavan, jota kutsutaan usein Scrumbaniksi tai Kanplaniksi. Näitä termejä ja käytäntöjä esitellään tässä luvussa myöhemmin (Ian, n.d.)

Oli kyse sitten Scrum-, Kanban-, Scrumban- tai Kanplan -välineestä, jokaiselle työlle voidaan määrittää työn koko verrattuna muihin töihin. Tämä helpottaa arviointia, kuinka paljon tiimi pystyy tekemään töitä jokaisessa sprintissä. Scrum-mallissa tämä tunnetaan tiimin työskentelynopeutena (velocity). Työn tai käyttäjätarinan pisteyttäminen (story points) on yleisin tapa määritellä työn koko scrum-tiimissä. Arviointiasteikkoja on useita erilaisia, esimerkiksi 1-12 tai 1-5. Tärkeintä on käyttää aina samaa laskentamenetelmää samassa tiimissä. Työskentelynopeutta (velocity) voi sprinteittäin seurata, eli katsoa mitä töitä oli suunniteltu tehtäväksi ja mitä saatiin toteutettua. Yleensä tiimin nopeus kasvaa ajansaatossa, kun työntekijöiden kokemus ja osaaminen kasvaa. Kun tiedetään tiimin työskentelynopeus, on mittaustavasta riippumatta helpompi arvioida karkeasti, paljonko aikaa menee kaikkien backlogilla olevien töiden tekemiseen. (Atlassian, 2017)

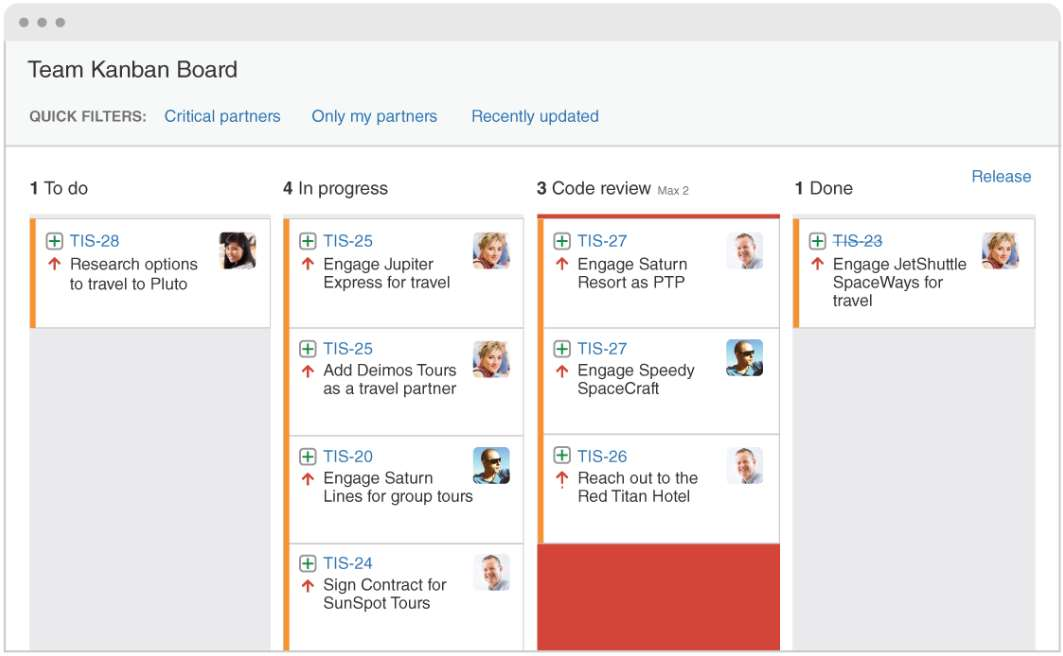
3.2 **Kanban**

Kanban on yksi suosituimmista ohjelmistokehitysmenetelmistä, joita ketterät tiimit käyttävät nykypäivänä. Kanban tarjoaa useita lisäetuja työn suunnitteluun ja suorituskykyyn kaiken kokoisille tiimeille. Päätarkoitus on esittää tiimin työt korttien avulla Kanban-taululla, jotta tiimin jäsenet voivat seurata työn etenemistä työnkulun kautta erittäin visuaalisesti. Kanban-kortit sisältävät kaikki tarpeelliset tiedot kyseisestä työstä ja antavat koko tiimille täydellisen näkyvyyden siitä, kuka vastaa kyseisestä työstä, lyhyt kuvaus suoritettavasta työstä, kuinka kauan kyseisen työn arvioidaan kestävän ja niin edelleen. Antamalla tiimin jäsenten nähdä jokaisen työn tilan milloin tahansa, samoin kuin kaikki siihen liittyvät yksityiskohdat, varmistetaan parempi keskittyminen, täydellinen jäljitettävyys ja esteiden ja riippuvuuksien nopea tunnistaminen. (Atlassian, 2019a)

Kanbanin päämääränä on antaa tiimin jäsenille juuri tarpeeksi työtä, jotta tiimi työskentelee jatkuvasti kykyjensä mukaan. Kanbania käyttävät tiimit hyötyvät joustavasta suunnittelusta, selkeämmästä fokusoinnista ja täydellisestä läpinäkyvyydestä töiden tärkeysjärjestyksessä. Dalyn (n.d) mukaan Kanban on loistava tuotantotiimeille, jotka ovat keskittyneet jatkuvaan toimitukseen muuttuvilla prioriteeteilla. (Daly, n.d.)

Keskeneräisten töiden määrää rajoitetaan jokaisessa työnkulun vaiheessa Kanban-taululla. Tästä käytetään lyhennettä WIP-raja (work in progress). Rajoitetun työmäärän avulla pullonkaulat tiimin toimitusputkessa on selkeästi näkyvillä Kanbanilla ennen kuin tilanne menee huonoksi. (Atlassian, 2019a)

Kuvassa 4 on esitetty esimerkkinä Jira-ohjelmistolla luotu Kanban-taulu ja siinä olevat tiimin työt. Taululla näkyy tehtävälistalla yksi työ, meneillään olevissa töissä neljä työtä, koodin katselmoinnissa kolme työtä ja valmiina yksi työ. Eri tiimeillä voi olla erilaisia Kanban-tauluja, jotka sisältävät tarpeen mukaan eri määrän sarakkeita ja voivat olla eri nimisiä. (Atlassian, 2019a)



Kuva 4. Jira Kanban-taulu ja siinä olevia tehtäviä. (Atlassian, 2019a)

3.3 **Scrum**

Scrum-mallissa tehdään töitä kiinteän pituisissa iteraatioissa eli sprinteissä. Sprintteihin aikataulutetut työt ovat tiimin tärkein prioriteetti. Tuotantotiimit, joilla on töiden suhteen selkeä etenemissuunnitelma ja suuri määrä töitä priorisoituna, hyötyvät tyypillisesti eniten Scrumista. (Daly, n.d.)

Kuva 5. Scrum työskentelytavan kuvaus. (Permana, 2015)

Kuten kuvassa 5 nähdään, tuotteen backlogilta otetaan töitä sprinttien backlogille ja niitä priorisoidaan tehtäväksi tässä tapauksessa kahden viikon sprinteissä ja sprintistä valmistuu valmiita tuotoksia. Tiimit pitävät päivittäin palavereja, joissa käydään töiden tilannetta läpi. (Permana, 2015)

3.4 **Scrumban**

Tiimille voi olla hyödyllistä käyttää myös Scrumin ja Kanbanin yhdistelmää. Ratkaisuna tiimille voi tällöin olla Scrumban. Scrumban-mallia voidaan käyttää eri tavoin tiimin tarpeiden mukaan. Yleisimpiä suuntauksia Scrumbanissa ovat Scrum-mallin mukaan sprinteissä työskentely, backlogin käyttäminen sekä Kanbanista tulevat WIP-rajat ja Kanban-taulun käyttäminen jakson kestona. Jakson kesto on siis aika, joka kuluu yhden tehtävän suorittamiseen tiimin koko työnkulun eli Kanban-taulun eri vaiheiden läpi. Vaiheita voi olla erilaisia riippuen tiimin työnkulusta, esimerkiksi ”uudet työt”, ”työn alla”, ”testauksessa”, ”valmis”. (Daly, n.d.)

3.5 **Kanplan**

Tiimi ei välttämättä halua työskennellä sprinteittäin, mutta haluavat silti käyttää backlogia apuna töiden hallinnassa. Kanplan voi olla tällöin sopiva ratkaisu. Kuten Scrumban, Kanplan on sekoitus Kanbanin ja Scrumin käytänteistä painottaen enemmän Kanbania. (Daly, n.d.)

Backlogin hallitseminen Kanbanin ensimmäisessä sarakkeessa on helppoa niin kauan kuin siellä on vain muutamia töitä. Kun töiden määrä backlogilla kasvaa, sen selaaminen tulee koko ajan vaikeammaksi. Töitä suunniteltaessa Kanplanin backlog tarjoaa tapauskohtaisesti toteutetun listan tiimin töistä, jotka ovat tärkeysjärjestyksessä. Siksi Kanplanin backlogia käytetään töiden suunnitteluun. Tiimin jäsenet voivat työskennellä Kanbanilla keskittyen meneillään oleviin töihin ilman että työt, jotka ovat vielä backlogilla häiritsevät heidän työntekoaan. (Atlassian, 2019b)

4 AUTOMAATIOTESTAUS

Ohjelmistokehityksessä testaus on iso osa sekä kehitystä että tuotannon häiriöiden korjaamista. Siksi onkin tärkeää tarkastella testausta erillisenä osiona.

Järjestelmien testaaminen on kallista ja vaatii paljon työtä. Ohjelmistotestaus vaatii jopa 50 % järjestelmien kehittämiskustannuksista ja vielä enemmän, jos kyseessä on turvallisuuskriittinen järjestelmä. Yksi ohjelmistotestauksen tavoitteista on automatisoida mahdollisimman paljon ja siten vähentää huomattavasti kustannuksia, minimoida inhimilliset virheet ja tehdä regressiotestauksesta helpompaa. (Ammann & Offutt, 2008, s. 10)

Automaatiotestaus on yksi käytäntö DevOps-mallissa, jolla pyritään vähentämään aikaa järjestelmämuutosten ja tuotantoon viennin välillä. DevOps-malliin liittyvän jatkuvan käyttöönoton ja jatkuvan toimittamisen edellytyksenä on kattavat automaattisesti ajettavat testitapaukset. Kun kehittäjä on luovuttanut koodin versionhallintajärjestelmään, sille suoritetaan automaattisesti useita erilaisia testitapauksia ja sen jälkeen muutokset voidaan asentaa tuotantoon sillä oletuksella, että kaikki testit läpäistään. Nykyaikaisissa järjestelmissä voidaan ottaa käyttöön muutoksia useita kertoja päivässä. Nopea käyttöönotto tarkoittaa, että ihmisten tekemä manuaalinen testaaminen on mahdotonta. Vaikka käyttöönottoja olisi yksi päivässä, testausmäärä kuormittaa ihmisiä huomattavasti, jos testit joudutaan tekemään manuaalisesti. Tämän vuoksi testiautomaatio on kriittisen tärkeää nopeaa käyttöönottoa varten. (Bass, 2017)

Regressiotestaus on yksi ohjelmistotestauksen tyyppi. Regressiotestauksella varmistetaan, että äskettäinen ohjelman tai koodin muutos ei ole vaikuttanut haitallisesti olemassa oleviin ominaisuuksiin. Tarkoituksena on testata, että kaikki vanhat toiminnallisuudet toimivat kuten kuuluukin. (Guru99, 2019)

4.1 Testiautomaation suunnittelu

Hyvä automaatiotestisuunnittelu kertoo, kuinka tietty toiminto tai ominaisuus testataan. Testisuunnittelija ottaa huomioon muun muassa seuraavia asioita. Mitä testataan ja kuinka testi asetetaan? Mitä syötteitä käytetään ja mistä syötteet tulevat? Mitä tarkistetaan ja missä ovat odotetut tulokset? Mitä asioita pitäisi syntyä testauksen tuloksena? Mistä tiedetään, että testitapaus on hyväksytty tai hylätty? Testauksen tuotokset pidetään yksinkertaisena ja hyvin muotoiltuina. (Abdul Rauf & Madhusudhana Reddy, 2015)

4.2 Automaation kehittäminen ja käyttöönotto

Pääasiassa on kehitettävä kahden tyyppisiä asioita testiautomaation kehitysvaiheessa. Nämä kaksi asiaa ovat suoritettavat skriptit eli komentosarjat sekä ympäristö koodille. Jotkut projektit tai kokonaiset moduulit eivät välttämättä ole suoraan saatavissa testaukseen. Tällöin on kehitettävä omia osia ja ohjaimia moduulin simuloimiseksi. Testisuunnittelija on vastuussa testisarjojen luomisesta, työkalun räätälöinnistä ja testausympäristöasetuksista. (Abdul Rauf & Madhusudhana Reddy, 2015)

4.3 Testiautomaation toteutus ja ylläpito

Testausvaiheessa suoritetaan testitapaukset joko manuaalisesti tai automatisoidusti. Automaatiota varten testitapausten valinta tehdään riippuen automaatiostrategiasta. Yleensä kaikki regressiotestaustapaukset testataan automaattisesti toistuvan manuaalisen suorituksen välttämiseksi. Automaattisessa ympäristössä testaaja voi valita tarpeelliset testitapaukset, jotka hän haluaa suorittaa tietyllä alustalla ja tietyissä tapauksissa. Suoritettaessa testauksia, voidaan miettiä kahdesta vaihtoehdosta, kumpi on parempi. Ensimmäisessä vaihtoehdossa testin suorittaminen lopetetaan, jos jokin valituista testeistä epäonnistuu. Toisessa vaihtoehdossa jatketaan testisarjan suorittamista yksittäisestä viasta huolimatta. Suorituksen lopussa nähdään testauksen tulokset ja loki. Tämä antaa yksityiskohtaiset tiedot testauksen onnistumisesta. (Abdul Rauf & Madhusudhana Reddy, 2015)

5 YRITYKSEN, AIHEEN JA METODOLOGIAN ESITTELY

Tässä luvussa esitellään ensin organisaatio ja tiimi, joille opinnäytetyö tehdään. Luvussa pureudutaan myös opinnäytetyön tavoitteisiin ja avataan aihetta tarkemmin. Lisäksi kuvataan opinnäytetyön toteutustapaa ja käytettyä metodologiaa.

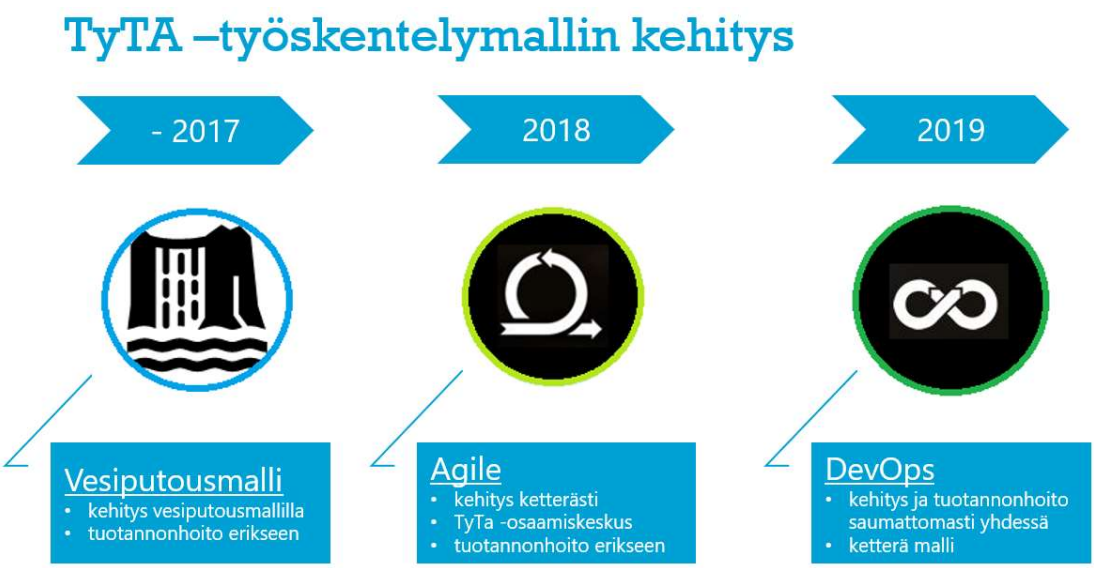
5.1 Organisaation esittely

LähiTapiola Palvelut Oy kuuluu LähiTapiola-ryhmään yhdessä LähiTapiola Vahinkoyhtiön, LähiTapiola Henkiyhtiön, LähiTapiola Kiinteistövarainhoidon, LähiTapiola Varainhoidon, LähiTapiola Kiinteistöpääomarahaston sekä 20 alueyhtiön kanssa. Yhtiöryhmän henkilöstömäärä on noin 3400 työntekijää.

LähiTapiola Palvelut Oy, jolle opinnäytetyö tehdään, tuottaa asiantuntija-, kehitys- ja tukipalveluita LähiTapiola-ryhmään kuuluville liiketoimintayhtiöille ja ryhmän ohjaustoiminnoille. Näitä palveluita ovat esimerkiksi ICT-, talous-, markkinointi- viestintä- ja HR-palvelut. LähiTapiola Palvelut Oy:ssä työskenteli 2018 vuonna 1017 henkilöä. Opinnäytetyö tehdään LähiTapiola Palvelut Oy:n ICT kehitys -yksikössä TyTA-kokonaisuuteen liittyen.

5.2 TyTA-järjestelmäalue

Kehitys ICT -yksikön TyTA-järjestelmäalueella työskentelee asiantuntijoita vakuutus- ja korvausjärjestelmien sekä raportoinnin parissa. TyTA-alueen tehtäviin kuuluu sekä kehitystöitä että tuotannonhoitoa eli häiriönhallintaa. TyTA-järjestelmäalueen tuotannonhoitoa ja kehitystä on tähän asti hoidettu eri organisaatioissa eri asiantuntijoiden toimesta. Töiden hajaantuminen eri organisaatioihin ja paikkoihin tuottaa ongelmia osaamisen laajentamisessa, sijaistamisessa, kokonaisuuden ymmärtämisessä ja tuotannonhoidon sekä kehitystöiden seurannassa. Töiden keskinäinen priorisointi on ollut haastavaa ja töiden jakaantuminen eri henkilöiden kesken epätasaista. Yrityksessä on jo aiemmin toteutettu projekteja ketterän kehityksen keinoin ja ketteryydestä tavoitellaan vielä suurempia hyötyjä, kun yhdistetään tuotannonhoito ja kehitys yhdeksi tiimiksi DevOps–periaatteita noudattaen. Kuvassa 6 on kuvattu, miten TyTA-alueen työskentelymalli on kehittynyt vuosien saatossa.



Kuva 6. TyTA-alueen työskentelytapojen kehitys.

Vuoteen 2017 asti kehitystä tehtiin pääasiassa vesiputousmallin mukaan ja tuotanto hoidettiin erikseen. Vuonna 2018 perustettiin kehityspuolelle TyTA-osaamiskeskus ja kehitystöitä ja projekteja alettiin tehdä ketterien menetelmien mukaan (Agile). Tuotannonhoito tehtiin edelleen erillään kehityksestä. Vuoden 2019 syksystä alkaen TyTA-alueella on yksi tiimi, joka hoitaa tuotannonhoidon ja kehityksen DevOps-mallin mukaisesti.

5.3 Opinnäytetyön aihe ja metodologia

Opinnäytetyön tavoitteena on kuvata ja luoda DevOps-malli LähiTapiolan kehitys ICT -yksikössä TyTA–järjestelmäkehitysalueelle. Kyseisellä järjestelmäalueella tapahtuu tuotannonhoitoa ja kehitystä jatkuvasti, joten DevOps-periaatteet sopivat alueeseen hyvin. Työt ovat sekä liiketoiminnan kehittämiseen liittyviä töitä, että viranomaisvaatimuksien kautta tulevia töitä. Tämä johtuu toimialan lakisääteisyydestä.

TyTA-tiimi toimii tiiviissä yhteistyössä liiketoiminnan ja järjestelmätoimittajien kanssa. Aihe rajautuu koskemaan vain tiettyä organisaation järjestelmäkehitysaluetta ja DevOps–tiimin luomista kyseiselle alueelle. Tutkielmassa ei kuvata yleismaailmallisesti mallia, joka voidaan ottaa käyttöön missä tahansa järjestelmäkokonaisuudessa, vaan keskitytään tietyn tiimin luomiseen ja DevOps-periaatteiden noudattamiseen kyseisessä TyTA-tiimissä. DevOps on moniulotteinen malli, joka kattaa paljon eri asioita ja DevOps-käytäntöjä voidaan käyttää eri organisaatioissa ja tiimeissä eri tavoilla riippuen tarpeista. Tiimi muodostetaan opinnäytetyöprosessin aikana ja työssä kuvataan tiimin perustamisen alkuvaihetta ja käytäntöjä, joista päätetään lähteä liikkeelle.

Opinnäytetyön toteutustapa on toiminnallinen ja se toteutetaan observoimalla eli havainnoimalla. Observoinnissa kerätään tutkittavasta ilmiöstä tietoa sitä seuraamalla ja tekemällä samalla havaintoja kyseisestä asiasta. Havaintotietojen keräämismenetelmä luokitellaan osallistavaksi tutkimukseksi, koska tutkijan on upotettava itsensä tilanteeseen vastaajien kanssa ja tekemällä samalla muistiinpanoja tai nauhoituksia. (Methodology, n.d.)

Tiimin muodostamisen ja suunnitelmien vaiheista kerättiin tietoa päiväkirjamaisesti. Tämä oli luontevin tapa seurata uuden tiimin muodostamista. Seuraavissa luvuissa käydään tarkemmin läpi näitä vaiheita ja suunnitelmia.

6 DEVOPS-TIIMIN MUODOSTAMINEN JA KÄYTÄNTEET

Aluksi organisaatiossa kartoitettiin ketkä asiantuntijat ja mitkä järjestelmäalueet kuuluvat TyTA-kokonaisuuteen. Kun nämä oli selvitetty, pidettiin työtunnit tulevien tiimiläisten kanssa ja kartoitettiin mitä työtehtäviä ja vastuualueita kenellekin kuului sillä hetkellä. Samalla saatiin selville, miten paljon resursseja on varattuna mihinkin työhön.

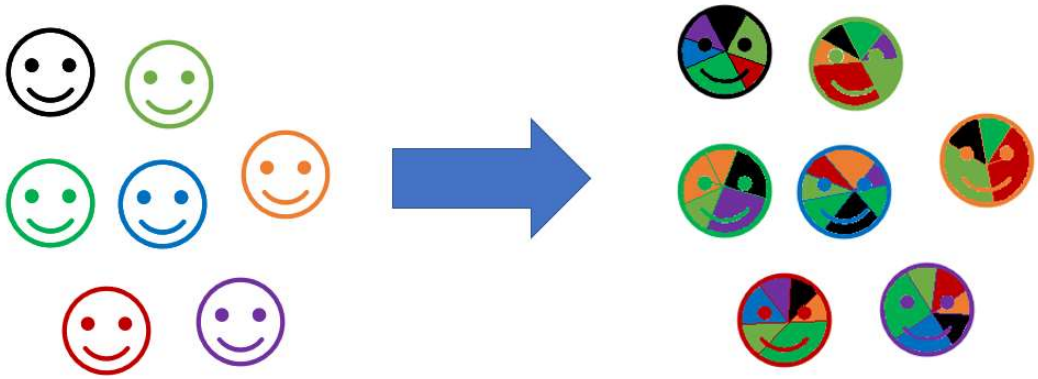
Työtunneilla kartoitettiin myös mihin muihin järjestelmien osa-alueisiin asiantuntijat haluaisivat tulevaisuudessa perehtyä osaamisen laajentamisen näkökulmasta. Tarkoituksena on DevOps-käytäntöjen mukaan myös asiantuntijoiden osaamisen laajentaminen ja tiedon jakaminen muiden asiantuntijoiden kanssa. Työtuntien perusteella muodostui selkeä kuva tiimin nykyisistä vastuista ja työtehtävistä sekä asiantuntijoiden ammatillisista kehitystoiveista.

6.1 Istumapaikat

Ensimmäinen askel kohti tiimin muodostamista oli tulevien tiimiläisten siirtyminen istumaan samaan projektihuoneeseen. Asiantuntijoiden työskenteleminen mahdollisimman paljon samassa paikassa on perusedellytys tiimin toiminnalle. Täysin tämä ei onnistunut, koska tiimiin kuuluu myös toisella paikkakunnalla työskentelevä asiantuntija ja osa tiimiläisistä tekee noin pari päivää viikossa etätöitä. Etätyöpäiviä yritetään kuitenkin mahdollisuuksien mukaan sovittaa niin, että asiantuntijat olisivat mahdollisimman paljon samoina päivinä etätöissä tai toimistolla. Tarkoitus on poistaa virtuaaliset ja fyysiset esteet tiimiläisten väliltä. Muualla työskentelevien tiimin asiantuntijoiden kanssa yhteistyö saadaan sujumaan kivuttomasti nykyteknologian avulla. Yhteyttä pidetään puheluin, yhteisten online-palaverien avulla, chat-keskusteluilla sekä nähdään säännöllisesti toimistolla. Toki kasvokkain työskentely on näistä tehokkain vaihtoehto.

Kuvaan 7 on havainnollistettu osaamisen jakamista TyTA-tiimiläisten kesken ajan saatossa. Kaikille tulee uusia opittavia osa-alueita vanhojen vastuiden lisäksi. Syksyyn 2019 asti tiimiläiset istuivat pitkälti eri paikoissa ja vakuutus-, korvaus- sekä raportointialueiden asiantuntijat tekivät itsenäisesti omia töitään siilomaisesti. Tavoitteena on, että osaaminen laajenisi tiimin sisällä eri osa-alueisiin. Lisäksi tiimiläiset pystyisivät auttamaan muita paremmin esimerkiksi loma-aikoina ja suuntamaan resursseja niihin alueisiin, missä eniten tarvitaan apua milloinkin. Vähintäänkin kaikkien järjestelmäalueiden pääpiirteet ja tehtävät olisi hyvä jokaisen tiimiläisen tuntea, jotta he pystyisivät tekemään töitä ristiin mahdollisimman hyvin. TyTA-tiimin asiantuntijat ovat nähneet samojen istumapaikkojen hyödyiksi ainakin kommunikaation parantumisen, töiden jakamisen helpottumisen,

ryhmähengen parantumisen, osaamisen jakamisen sekä backlogilla olevien töiden hallinnan parantumisen.



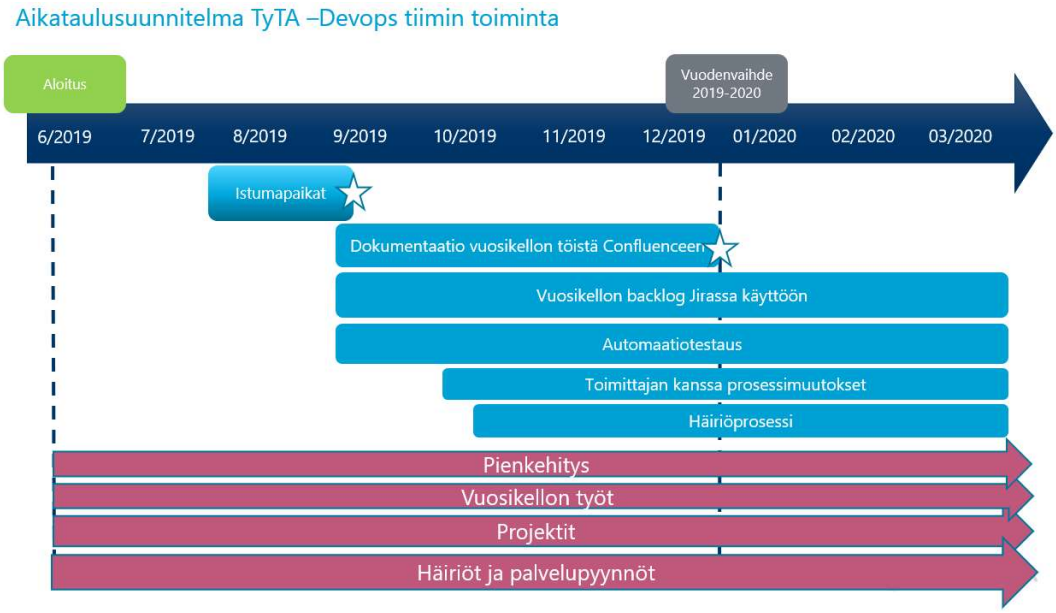
Kuva 7. Osaaminen jakaantuu tiimiläisten kesken eri osa-alueille.

6.2 Starttipäivä ja käytänteistä sopiminen

Seuraavaksi tiimin kanssa pidettiin starttipäivä, jossa toiminta aloitettiin ”virallisesti”. Kyseessä oli kolmen tunnin kick-off -tyylinen tilaisuus. Ennen starttipäivää tulevia tiimiläisiä pyydettiin ennakkotehtävänä listaamaan nykyiset toistuvat palaverit, jotta saadaan tietää mitä palavereja kenelläkin on ja ettei jatkossa tulisi päällekkäisiä uusia palavereja samoista aiheista. Tiimiläisiä pyydettiin myös miettimään odotuksia tulevaan tiimiin liittyen sekä tarpeellisia yhteisiä palaverikäytäntöjä tiimin kesken.

Starttipäivän alussa tiimille esiteltiin yleisellä tasolla mitä DevOps tarkoittaa. Tässä teoriaosuudessa kerrottiin DevOps-käytännöistä, puhuttiin tuotannon, kehityksen sekä laadunvarmistuksen yhteistyöstä, DevOpsin tavoitteista, haasteista ja hyödyistä, automaatioasteen nostosta sekä kulttuurista.

Alun teoriaosuuden jälkeen käytiin läpi aiheita, joista tiimi voisi lähteä liikkeelle DevOps-käytäntöjen omaksumisessa TyTA-alueella. Kuvassa 8 on esitetty tiimin aikataulusuunnitelmaa alkuvaiheeseen liittyen. Istumapaikkamuutosten jälkeisiksi askeleiksi kohti DevOps-käytäntöjä päätettiin edetä loppuvuoden 2019 osalta dokumentaation keräämisestä yhteen paikkaan, uusien backlogien käyttöönotosta, automaatiotestauksen lisäämisestä, toimittajan kanssa prosessimuutoksista sekä häiriöprosessin kehittämisestä. Lisäksi starttipäivän aikana puhuttiin jatkokehitysideoina muun muassa mittaristosta ja metriikasta sekä muusta automaatiosta. Samaan aikaan tiimi työskentelee kehitystöiden, projektien ja tuotannonhoidon parissa, joten kaikkia käytäntöjä ei ole tarkoitus ottaa kerralla käyttöön, vaan lisätä niitä vaiheittain.



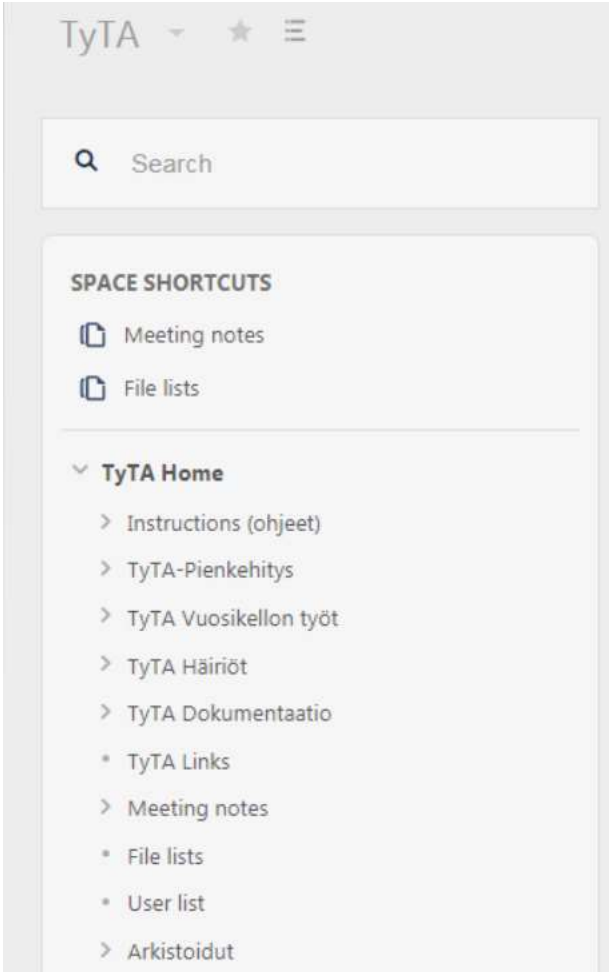
Kuva 8. Suunniteltu aikataulu TyTA-tiimin alkuvaiheesta.

Starttipäivän lopuksi tehtiin ryhmätöitä, joiden tarkoituksena oli suunnitella yhdessä tiimin toimintamalleja. Yhtenä ryhmätyön aiheena oli tiimin yhteiset palaverikäytännöt, joita tiimiläiset olivatkin miettineet jo etukäteen ennakkotehtävässä. Ryhmätyö toteutettiin keskusteluna ja tiimi mietti yhdessä mitä yhteisiä palavereja olisi tarpeellista pitää.

Tiimi koki hyvänä käytäntönä yhteisiksi palavereiksi viikon aloituspalaverin maanantaiaamuisin, sekä joka toinen viikko pidettävän TyTA refinement -palaverin. Viikon aloitus palaverissa käydään läpi kehitys- ja tuotannonhoidon backlogien tilanteet ja katsotaan töiden etenemistä sekä tulevan viikon työtehtäviä ja vastuita. Refinement-palaverissa on tarkoitus suunnitella tiimin prosesseja, jakaa uusia ideoita, kouluttaa uusia osa-alueita muille ja käydä läpi muita tiimin ajankohtaisia asioita. Palaverikäytäntöjen avulla pyritään lisäämään tiimissä avoimuutta, yhdessä keskustelua, suunnittelua, priorisointia ja käydä läpi tärkeitä ajankohtaisia asioita. Näillä yhteisillä palavereilla on tarkoitus lähteä liikkeelle ja muokata palaverikäytäntöjä tarvittaessa.

6.3 **Dokumentaatio Confluenceen**

Kehityksessä on käytetty Confluencea dokumentaation hallintajärjestelmänä erilaisissa projekteissa. Tiimille luotiin TyTA-alueelle oma projektisivu Confluenceen (kuva 9) ja tämän sivuston alle on tarkoitus kerätä dokumentaatiota tuotannonhoitoon, vuosikellontöihin ja kehitystöihin liittyen. Samaan paikkaan kerätään dokumentaatiota sekä vakuutus- ja korvausjärjestelmistä että raportoinnista.



Kuva 9. TyTA-tiimin Confluence sivuston puurakenne.

Aiemmin dokumentaatiota on kerätty eri paikkoihin riippuen asiantuntijasta ja järjestelmäalueesta. Kohdistamalla kaikki dokumentaatio samaan paikkaan saadaan tiimille todella suuria hyötyjä. Kaikkien on tällöin helppo etsiä tarvitsemansa tiedot yhdestä paikasta ja uusimmat versiot dokumenteista tulee aina samaan paikkaan.

Dokumentaatiota tehdään DevOps-mallissa jatkuvana toimintana. Tarkoitus ei ole pelkästään kerätä vanhoja ohjeita samaan paikkaan, vaan päivit-

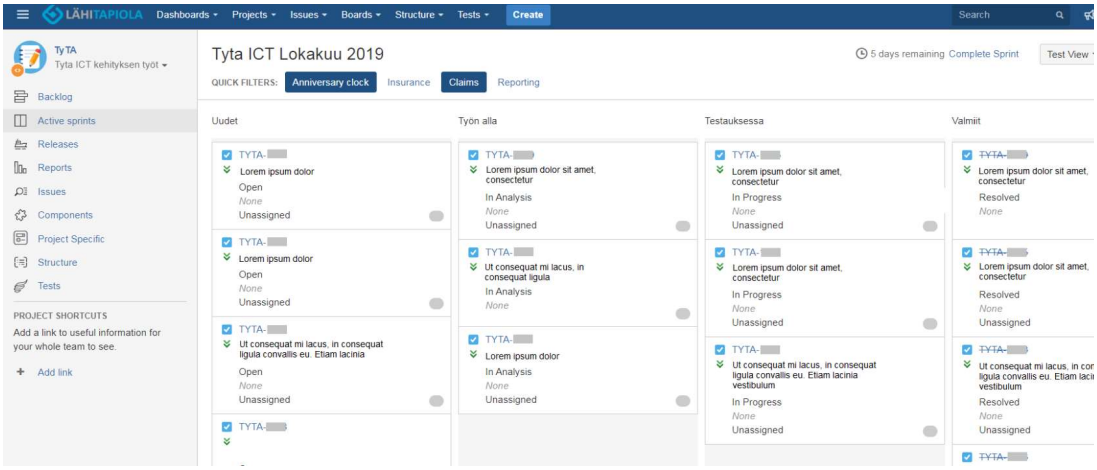
tää niitä jatkuvasti eri kehitystöiden ja projektien edetessä. Myös tuotannonhoidon dokumentaatiota päivitetään jatkuvasti prosessien ja vaatimusten muuttuessa. Tarkoitus on kuvata jokainen järjestelmien vuosikellon työ Confluenceen rautalankamallimaisesti, niin että kuka tahansa tiimin jäsenistä voisi hoitaa työn ohjeen avulla. Tämä auttaa osaamisen jakamista merkittävästi tiimin kesken. Tavoitteena on, että jokaisella työllä on selkeät ohjeet samassa paikassa riippumatta järjestelmästä tai siitä onko kyseessä korvaus, vakuutus vai raportoinnin työtä.

6.4 **Backlogien käyttöönotto Jirassa**

Aiemmin TyTA-alueen kehityspuolella on käytetty Jiran backlogia erilaisissa projekteissa ja pienkehitystöissä. Tiimi mietti, miten saisimme muutkin työt läpinäkyväksi. Tuotannonhoidon töissä voidaan käyttää myös erilaisia backlogeja apuna. Tarkoituksena on saada työt näkyviksi ja seurata Jiran kautta sekä tuotannonhoidon että kehityksen töitä ja ymmärtää näiden vuorovaikutusta.

Tiimi päätti kokeilla vuosikellon töiden hallinnassa Scrumin ja Kanbanin yhdistelmää Scrumbania. Vuosikellon työt ovat ennustettavia ja niitä esiintyy eri määrä eri vuodenaikoina. Kuukausittaiset sprintit päätettiin ottaa kokeiluun vuosikellon töissä. Tämä sprinttiajattelu tulee Scrum-mallista. Sprintit voitiin luoda etukäteen koko vuodelle ja sprintteihin laitettiin jo valmiiksi töitä backlogilta, joiden tiedetään tulevan työn alle tiettynä aikana. Näin saatiin läpinäkyvyyttä koko vuoden ajalle vuosikellon töille ja tiimi pystyy varautumaan ruuhka-aikoihin etukäteen paremmin miettimällä resursointia ja tekemällä valmistelevia tehtäviä eri töille. Vuosikellon töitä on sen verran paljon, että niiden priorisointi Kanbanin ensimmäisessä sarakkeessa olisi ollut mahdotonta ja sekoittanut meneillään olevia töitä. Tämän vuoksi backlogin käyttö tulikin välttämättömäksi. Scrumbanissa käytetään backlogin ja sprinttien ohessa Kanban-taulua. Kanban-taululle tulee nykyisessä sprintissä olevat työt, eli TyTA-tiimin kohdalla meneillään olevan kuukauden työt.

Kuvassa 10 on havainnollistettu, millaista Scrumbania TyTA-tiimissä lähdettiin kokeilemaan. Vasemmalta navigaatiosta pääsee katsomaan backlogia tai siirtymään nykyisen sprintin Kanban-taululle. Kuvassa on havainnollistettu lokakuun 2019 sprintin töitä. Aluksi kaikki työt ovat sarakkeessa ”Uudet”. Kun työ otetaan työn alle, siirtää asiantuntija sen ”Työn alla” sarakkeeseen. Testaukseen tulevat työt ovat ”Testaus” -sarakkeessa ja valmistuneet työt siirretään lopuksi ”Valmiit”-sarakkeeseen. Tiimi on pitänyt vuosikellon töiden pitämistä backlogilla hyvänä käytäntönä, joka parantaa priorisointia, lisää läpinäkyvyyttä ja ennakoitavuutta.



Kuva 10. Havainne kuva TyTA-tiimin vuosikellontöiden backlogilta.

Opinnäytetyöprosessin aikana ehdittiin ottamaan backlog käyttöön, mutta jatkossa backlogin käytöstä olisi tarkoitus saada vielä suuremmat hyödyt, kun jokaiselle sarakkeelle mietitään tietyt WIP-rajat. Jatkossa olisi tarkoitus arvioida jokaiselle työlle koko (story points). Näin saadaan tiedettyä myös tiimin työskentelynopeus (velocity). Tämän jälkeen on mielenkiintoista seurata, miten tiimin tehokkuus kasvaa, kun osaaminen laajenee ja kokemus karttuu. Myös mittaaminen helpottuu, kun pystyy tarkastelemaan työskentelynopeutta ja töiden kokoja. Tiimi on kartoittanut myös häiriöiden ja palvelupyyntöjen saamista samalle Jiran backlogille käsittelyyn, mutta prosessi on vielä kesken opinnäytetyön teon aikaan.

6.5 **Mittaristo ja metriikka**

Mittaristo ja metriikka on osa-alue, jota tiimin on tarkoitus lähteä suunnittelemaan tarkemmin. Alkuvaiheessa tiimille ei ole vielä selvää mitä halutaan mitata ja millä keinoin mittausta lähdetään tekemään. Tiimin mittaamien asioiden lisäksi pitää kartoittaa mitä johto tai organisaatio pitää tärkeinä ja mitattavina asioina. Näiden asioiden miettiminen onkin lähtökohta mittaamisen aloittamiselle.

Mittareiden tavoitteet voidaan asettaa esimerkiksi kuukaudelle, vuosineljännekselle, puolelle vuodelle tai vuodelle. Myös liiketoiminnalle on tarkoitus tarjota mittareiden kautta tilannetietoa ja lisätä läpinäkyvyyttä tekemiseen. Saadakseen aikaan todellista yhteistyötä ja tunteen yhdestä ryhmästä, pitäisi menestyksen mittareiden olla samat kaikilla osapuolilla. TyTA-tiimin perustamisen myötä onkin nähty tarvetta yhteisille mittareille, kunhan ensin mietitään mitkä ovat parhaimmat mittarit tällä hetkellä ja mihin suuntaan tiimin toimintaa halutaan viedä.

Tälläkin hetkellä tiimiläisten työtä ja tuloksia mitataan. Koska muodostetun TyTA-tiimin asiantuntijat tulevat eri organisaatioista ja tiimeistä, on heillä vielä kuitenkin ollut omat mittarit. Mittarit voivat liittyä kehitystöihin, tuotannonhoitoon tai muuhun asiaan riippuen aiemmasta tekemisestä ja organisaatiosta. Jotta saadaan tiimille yhteiset tavoitteet, ainakin osa mittareista olisi hyvä olla samoja. Tällöin tiimi työskentelee paremmin kohti yhteisiä tavoitteita ja tietää mitä heiltä odotetaan.

Tiimin starttipäivässä käytiin joitakin mahdollisia mittariehdotuksia läpi. Aiheet olivat vasta ideatasolla, eikä opinnäytetyön kirjoittamisen aikoihin ole vielä ehditty miettimään mittareita loppuun asti, koska tiimi on toiminut vasta alle kaksi kuukautta. Kuvassa 11 on esitelty starttipäivässä käytyjä ideoita mittareiksi.

Kuva 11. Mittariehdotuksia TyTA-tiimille.

Kuvatut mittarit ovat melko yleisiä DevOps-mittareita. Niillä voidaan mitata esimerkiksi laatua, automaatiotestauksen läpäisyastetta, löytyneitä virheitä, fiilistä, työskentelynopeutta eli velositeettia, käyttöönottojen tiheyttä ja onnistumista, asiakastyytyväisyyttä tai järjestelmämuutosten nopeutta. Vielä ei käyty tarkemmin läpi, millä keinoilla mitäkin osa-aluetta voisi mitata. Kaikkia mittareita ei tule välttämättä käyttöön, mutta jatkossa pitäisi miettiä millä tavoin mittaus tapahtuu ja millä välineillä. Tarkoituksena on pitää yhteisiä työpajoja, joissa mietitään muutamat avainmittarit, jotka otetaan aluksi käyttöön. Mittareiden käyttöönoton jälkeen arvioidaan tuloksia ja katsotaan, miten valitut mittarit soveltuvat tiimin käyttöön vai tulisiko miettiä muita mittareita.

6.6 **Testausautomaation kehittäminen**

Tässä luvussa kerrotaan testausautomaation lisäämisestä TyTA-tiimin järjestelmiin. Testausautomaatiota on jo käytetty osassa järjestelmistä, esimerkiksi tuotannonvarmennuksessa versioiden aikana. Osa testauksista tehdään edelleen manuaalisesti, mutta mahdollisimman paljon järjestelmien testauksesta olisi tarkoitus automatisoida.

TyTA-tiimin töissä automaatiotestauksen tarkoituksena on saada ajettua nopeasti aiempaa kattavammat testitapaukset. Manuaalisella testauksella työ olisi hidasta ja virhealttiimpaa, kuin automaattisesti ajettavilla testeillä. Testausautomaation uskotaan vähentävän kustannuksia, minimoivan inhimillisiä virheitä ja tekevän regressiotestauksesta helpompaa. TyTA-järjestelmissä lähdettiin toteuttamaan testausautomaatiota niin, että suoritus loppuu virheen sattuessa kohdalle. Tällöin nähdään heti missä kohtaa testi menee vikaan. Järjestelmä kertoo myös millä skriptin rivillä virhe tapahtui ja voidaan tutkia, onko skripti ajan tasalla järjestelmämuutosten jälkeen vai onko kyseessä muutoksesta johtuva virhe tai joku muu syy. Skriptit on tarkoitus viedä samaan paikkaan selkeisiin hakemistoihin riippumatta järjestelmästä. Tällöin kuka tahansa TyTA-tiimistä pystyisi tarvittaessa ajamaan testitapaukset.

6.6.1 Regressiotestaus ja tuotannon varmentaminen

Regressiotestausta tehdään osassa TyTA-tiimin järjestelmistä vielä manuaalisesti ja tiimi lähti suunnittelemaan erilaisia automaatiotestitapauksia skriptien avulla. Regressiotestauksessa ja tuotannon varmentamisessa lähdettiin liikkeelle siitä, että testit olisivat mahdollisimman kattavat, nopeat ajaa ja täysin automaattiset. Manuaalisessa regressiotestauksessa tai tuotannon varmennuksessa versioissa on havaittu välillä liian suppeita testauksia. Virheitä on voinut löytyä esimerkiksi sellaisista kohdista järjestelmää, mihin ei ole tehty muutoksia. Automaattiset testitapaukset onkin tarkoitus kehitellä sellaisiksi, että testataan laajasti koko järjestelmää. Tarkoitus on testata kattavasti ja nopeasti niitäkin osia mihin ei ole tullut muutoksia. Manuaalisesti tällainen työ olisi laajoissa järjestelmissä hyvin hidasta, joten automaatiotestaus säästää paljon aikaa. Regressiotestaus jätetään kiireessä usein taka-alalle. Tarkoituksena on myös saada tuotantoon siirtyvien virheiden määrät mahdollisimman vähäisiksi.

6.6.2 Hyväksymistestaus

Hyväksymistestausta varten on tarkoitus tehdä TyTA-järjestelmille sellaisia automaatiotestitapauksia, joita voidaan järjestelmämuutoksesta riippuen soveltaa paremmin erilaisissa testitapausvariaatioissa. Esimerkkinä testitapaus pysähtyy tietyllä ruudulla ennalta määrättyyn kohtaan ja testaaja voi

syöttää ponnahdusikkunaan haluamansa arvon. Tämän jälkeen testitapaus etenee taas automaattisesti. Näin saadaan tehtyä samalla tavalla toistuvat kohdat automaattisiksi ja nopeiksi ja muuttaa vain niitä kohtia, joita on tarpeen vaihdella eri testitapauksissa. Hyväksymistestauksessa voidaan käyttää hyödyksi myös täysin automaattisia testejä ilman manuaalisia vaiheita riippuen järjestelmätyöstä.

7 JOHTOPÄÄTÖKSET DEVOPSISTA

Tiimi on toiminut vasta alle kaksi kuukautta, joten vielä ei ole kattavia tuloksia DevOpsin hyödyistä ja haasteista tiedossa. Tiimin alkuvaiheen hyödyistä ja haasteista on jo kuitenkin jonkin verran kokemusta. Näistä alun kokemuksista kerrotaan tarkemmin seuraavissa luvuissa. Lisäksi tarkastellaan tietoturvan yhdistämistä DevOps-malliin ja lopuksi mahdollisia jatkokehityskohteita ja DevOpsin tulevaisuutta TyTA-tiimissä.

7.1 Hyödyt

Kuvassa 12 on kuvattu avainsanoja, miksi DevOps-pilottiin päätettiin lähteä TyTA-alueella ja mitä hyötyjä odotettiin. Läpinäkyvyyden ja töiden seurannan odotettiin parantuvan, priorisoinnin odotettiin helpottuvan backlogien käyttöönoton myötä. Myös ryhmähengen odotettiin parantuvan, kun tiimiläiset työskentelevät tiiviissä yhteistyössä kohti samoja tavoitteita. Samoja tavoitteita kohti työskentelyä tukee myös tiimin yhteiset mittarit. Samoin kulttuurin, kokemusten jakamisen, yhteistyön sekä osaamisen laajentamisen nähtiin liittyvän koko tiimiytymisen ydinasioihin ja hyötyihin.

Lisäksi siilomaisuudesta oli tarkoitus päästä pois, kun ylläpidon ja kehityksen väliltä kaadetaan raja-aidat. Ketteryyttä ja sen tuomia hyötyjä haluttiin myös tuotannonhoitoon kehitystöiden lisäksi. Dokumentaation vieminen samaan paikkaan sekä jatkuva dokumentaatio nähtiin hyvänä ja selkeyttävänä käytäntönä, josta hyödytään etenkin työn jakamisessa ja uuden oppimisessa. Lisäksi uudet prosessit, varmempi tuotanto, aikataulutus ja Lean-menetelmät nähtiin tavoiteltavina hyötyinä.

Kuva 12. Mahdollisia DevOpsin hyötyjä TyTA-tiimille.

Alle kahden kuukauden aikana tiimin perustamisesta on huomattu, että osa näistä asioista on jo toteutunut ja tuonut hyötyjä tiimin työskentelyyn. Ensinnäkin kulttuuri on muuttunut, kun suurin osa tiimiläisistä istuu samassa paikassa. Tällöin kaikkien on helppo ilman kynnystä jutella muille ja selvitellä yhteistyössä epäselviä asioita.

Yhteisissä palavereissa käydään töiden tilanteita läpi ja koko tiimi pysyy meneillään olevista töistä ja töiden tilanteista paremmin ajan tasalla. Yhdessä voidaan myös priorisoida tulevia töitä ja kohdistaa resursseja niille alueille missä eniten tarvitaan apua. Osaaminen on jo laajentunut jonkun verran yli aiemman vastuualueen, ja tiimiläiset ovat pystyneet auttamaan toisiaan ristiin eri töissä. Dokumentaatiota on tehty kattavasti Confluenceen, eikä kenenkään tarvitse enää miettiä mistä mikäkin dokumentti löytyy, vaan voi käydä etsimässä dokumentteja keskitetysti samasta paikasta.

Lisäksi aikataulutus ja töiden priorisointi on parantunut selvästi varsinkin vuosikellon töissä, kun niitä seurataan Jiran Scrumbanilla. Tehtävät tulee hoidettua oikeaan aikaan, eikä unohduksia pääse sattumaan, kun kaikki tehtävät on priorisoitu ja esillä.

Tiimiläisten kommenttien perusteella he ovat pitäneet hyvänä sitä, että pääsevät itse vaikuttamaan prosesseihin ja yhdessä tiiminä voidaan miettiä parhaat mahdolliset ratkaisut eri asioihin. Tämä lisää myös ryhmähenkeä tiimin sisällä. Testiautomaatiosta odotetaan tuovan paljon helpotusta tiimin testaamiseen, joka vie paljon aikaa manuaalisella tavalla suoritettuna.

7.2 **Haasteet**

Vaikka hyötyjä on jo käytännössä saatu aikaan, on myös joitain haasteita ollut. Näitä on kuitenkin pyritty käsittelemään yhdessä tiimin kanssa ja löytämään ongelmiin ratkaisuja.

Eniten haasteita tiimille on tuonut ajanpuute asioiden kehittämistä varten. Tiimiläisillä on menossa isoja projekteja, kehitystöitä ja tuotannonhoidon töitä samaan aikaan, kun tiimin toiminta käynnistyi. Uusien asioiden opetteluun, omaksumiseen ja tekemiseen ei ole tarpeeksi aikaa. Tämä hidastaa käytäntöjen käyttöönottoa ja uusien asioiden suunnittelua. Uusia käytäntöjä ja prosesseja täytyykin ottaa pikkuhiljaa käyttöön. Esimerkiksi testiautomaatiotapausten tekeminen on ollut hidasta, johtuen meneillään olevasta työtilanteesta. Testitapausten tekemiseen ja suunnitteluun vaaditaan paljon aikaa ilman keskeytyksiä ja tällaisen ajan löytäminen on ollut haastavaa kiireellisempien töiden vuoksi. Tiimin asiantuntijat ovat kuitenkin pyrkineet löytämään sopivia välejä testitapausten tekemiseen ja priorisoimaan myös tiimiytymiseen liittyviä asioita tärkeiksi.

Kiireellisen työtilanteen lisäksi on myös muutosvastarintaa esiintynyt jonkin verran muutamissa uusissa käytännöissä. Tiimin työskentelyssä onkin huomattu, että kulttuurin muuttaminen vaatii paljon työtä ja läpikäyntiä. Tämä on aikatauluhaasteiden vuoksi ollut välillä vaikeaa. Kuitenkin asioiden tehokkaampi viestintä ja läpikäynti ovat auttaneet myös muutosvastarinnan minimoimisessa ja vähentänyt epäselvyyksiä. Tiimissä on tarkoitus miettiä yhdessä koko tiimin kanssa parhaat käytännöt omille prosesseille ja myös tämä osallistaminen vähentää muutosvastarintaa. Tiimille onkin koottu yhteiset pelisäännöt, joita kaikkien on helppo noudattaa. Tärkeää on myös luoda avointa ilmapiiriä, joka auttaa keskustelemaan ongelmaseikoista jo hyvissä ajoin. Näin voidaan ehkäistä ristiriitoja ja muutosvastarintaa. Vaikka haasteita on kohdattu jonkin verran, on nämä onneksi pystytty selvittämään yhdessä tiimin kanssa. Usein tällaiset tilanteet vahvistavat hyvin läpikäytyinä tiimihenkeä ja tiimi pystyykin varmasti entistä tiiviimpänä tiiminä ratkaisemaan tulevaisuudessakin paremmin ristiriitoja ja haasteita.

7.3 **SWOT-analyysi**

Starttipäivän aikana tiimi teki SWOT-analyysin DevOps-malliin liittyvistä vahvuuksista, heikkouksista, mahdollisuuksista ja uhista. Tässä työssä haluttiin, että tiimi pohtii ryhmissä näitä ensin itse, ja sitten jaetaan näkemykset kaikkien kanssa ja keskustellaan niistä. Tunnistetuille heikkouksille ja uhille pyrittiin löytämään myös ratkaisut ja toimenpiteet, miten ne selätetään. Tiimiläiset miettivät parityönä eri osa-alueita ja lopuksi tulokset koottiin yhteen taulukkoon. Kuvassa 13 esitetään SWOT-analyysin tulokset. Vahvuudet ja heikkoudet ylemmissä sarakkeissa ovat tiimin sisäisiä asioita. Ulkoiset mahdollisuudet ja uhat on kuvattu alemmissa sarakkeissa.

Kuva 13. SWOT-analyysi TyTA-tiimin DevOps-mallin käyttöönotosta.

Tiimin sisäisinä vahvuuksina nähtiin töiden läpinäkyvyys, osaamisen laajentaminen, töiden jakaminen, yhteistyö, auttaminen, laadun parantaminen järjestelmätöissä, nopeus/tehokkuus sekä dokumentaation keskittäminen ja yhteiset toimintatavat dokumentaation tuottamisessa. Tiimiläiset pystyivät jo etukäteen tunnistamaan vahvuuksia tiimille ja vahvistamaan tiimihenkeä.

Sisäisinä heikkouksina pidettiin sairauspoissaolojen, lomien ja muiden poissaolojen aikaisia sijaistamisia. Tähän ongelmaan päätettiin tehdä avuksi resursointisuunnitelma. Välillä tulee varmasti tilanteita, että joku ei ole paikalla ja toisen pitää pystyä tekemään tarpeelliset työt sillä aikaa. Yhtenä heikkoutena pidettiin sitä, että tiimiläisillä on liikaa erilaisia töitä jatkossa. Tämä tilanne tulee vastaan tuotannon, kehityksen ja eri järjestelmien yhdistyttyä saman tiimin vastuulle. Uuden oppiminen tulee pikkuhiljaa töiden ja perehdytysten kautta ja ei ole tarkoitus lisätä työtaakkaa vaan jakaa sitä tasaisemmin. Lisäksi heikkoutena pidettiin myös sitä, että koko yhtiöryhmätasolla ei ole DevOps-kulttuuria. Tämä on pilotin haaste, mutta samojen käytäntöjen odotetaan leviävän jatkossa muillekin alueille yhtiössä.

Ulkoisina mahdollisuuksina pidettiin läpinäkyvyyttä ja backlogin seurantaa. Läpinäkyvyyttä voidaan lisätä uusilla prosesseilla liiketoiminnalle ja toimittajille päin. Liiketoiminta ja toimittaja eivät ole osa tiimiä, mutta tiimi tekee heidän kanssaan tiivistä yhteistyötä. Ennakointia, töiden priorisointia ja sujuvampaa työntekoa toimittajan ja liiketoiminnan kanssa pidettiin myös uusien käytäntöjen kautta mahdollisuuksina, joita voidaan kehittää. Yksi

prosessiparannus on myös englanninkielisten tilausten ja määrittelyiden lisääminen, koska tiimi työskentelee myös englannin kielellä toimittajien kanssa.

Ulkoisina uhkina pidettiin uusien käytäntöjen omaksumista. Tätä pyritään parantamaan tiedottamalla, ohjeistamalla ja tekemällä tiivistä yhteistyötä toimittajien ja liiketoiminnan kanssa. Nykyistä metriikkaa, joka on erilainen eri tehtäviä tekevillä henkilöillä, pidettiin myös uhkana. Tiimille on tarkoitus jatkossa miettiä omat yhteiset mittarit. Liiketoiminnan vaatimusten selkeyttäminen nähtiin tärkeäksi kehittämiskohteeksi, jotta varsinkin toimittajilla olisi parempi käsitys töiden kokonaiskuvasta. Tähän ongelmaan mietittiin ratkaisuksi jatkossa prosessikaavioiden tekemistä ja selkeämpää ohjeistusta. SWOT-analyysin tuloksista ja niiden purkamisesta jäi kuva, että tiimi näki mahdollisuuksia DevOps-mallista ja uusista käytännöistä. Uhkiin ja heikkouksiin löytyi ratkaisuja, eikä niitä pidetty ylitsepääsemättöminä esteinä.

7.4 **Tietoturvan huomioiminen**

Vaikka opinnäytetyössä ei keskitytä tietoturva-asioihin syvällisemmin, ovat ne osa kehitys- ja ylläpitoprosesseja. LähiTapiolassa tietoturvaan ja siihen liittyviin uhkiin suhtaudutaan vakavasti. Tietoturvaa parannetaan monin eri keinoin ja eri osa-alueilla, kuten verkossa, mobiililaitteilla, vakuuttamis- ja korvausjärjestelmissä, sähköpostissa sekä tietoturva-auditointien avulla.

DevSecOps-käytännöt ovat LähiTapiolassa käytössä ja yhtiö pyrkii kohti parempaa maailmaa missä tietoturva on kokonaisvaltaisesti mukana koko projektien ja järjestelmien elinkaaren ajan. Jokainen projekti tähtää tuotantoon vientiin ja kun projekti on ohi, ei tietoturvaa voida unohtaa. Vastuu tietoturva-asioiden luovuttamisesta kehityspuolelta tuotannonhoitoon ei ole aina suoraviivaista, varsinkin kun yritys kasvaa ja asiat monimutkaistuvat. Tähän on kuitenkin olemassa LähiTapiolassa hyvät prosessit.

LähiTapiolassa on käytössä myös menestyksekäs Bug Bounty -ohjelma. Bug Bounty -ohjelmassa niin sanotut hyvät hakkerit etsivät ja löytävät virheitä sovelluksista, ennen pahoissa aikeissa olevia hakkereita. Löytäessään virheitä, hakkereille maksetaan palkkioita. Bug Bounty -ohjelman kautta on onnistuttu löytämään monimutkaisia virheitä ennen kuin niistä on tullut isompia ongelmia ja tämä on osoittautunut erittäin arvokkaaksi asiaksi GDPR:n jälkeisessä maailmassa. Vuoteen 2019 mennessä LähiTapiola on palkinnut lähes 300 virheilmoitusta ja maksanut yli 100 000 euroa palkkioita.

Myös TyTA DevOps -tiimi ottaa tietoturvan huomioon omassa työssään. Tiimille on annettu tarvittavaa koulutusta, miten tietoturva tulee ottaa

huomioon eri töissä ja niiden vaiheissa. Aina saa myös apua tietoturva-asiantuntijoilta, jos jokin asia epäilyttää. Projekteissa ja kehitystöissä on käytössä usein uhkamallinnukset, joissa katsotaan läpi töiden riskit ja mitä pitää ottaa huomioon tietoturva-asioissa.

7.5 **Jatkokehityskohteet**

Opinnäytetyö keskittyi tiimin muodostamiseen ja tiimin alkuvaiheen käytäntöihin. Kattavat tulokset jäivät vielä puuttumaan, koska tiimi on toiminut vasta alle kaksi kuukautta. Jatkotutkimuksena voisi olla hyvin paljon erilaisia asioita tiimin toimintaan liittyen. Minkälaisia yhteisiä mittareita ja metriikkaa tiimi tulee käyttämään? Minkälaista uutta automaatiota lähdetään suunnittelemaan? Mitä uusia työkaluja tiimi voisi ottaa käyttöön? Miten tiimin osaamisen laajentaminen saavutetaan? Mitkä ovat tiimin seuraavat askeleet ensi vuonna ja sitä seuraavina vuosina? Miten jatkuvaa käyttöönottoa ja toimitusta voidaan parantaa?

Kuten opinnäytetyössä on tullut selväksi, DevOps on jatkuvaa kehittämistä ja aina löytyy uutta mietittävää ja käytäntöjä, joita voidaan kehittää. Tiimi ja tiimin prosessit eivät oikeastaan koskaan tule valmiiksi, mutta se onkin parasta, että voidaan kokeilla uusia asioita ja muuttaa suuntaa tarvittaessa. Näin löydetään parhaat mahdolliset käytännöt tiimin tarpeisiin.

8 YHTEENVETO

DevOps-käytännöt ovat viime vuosina levinneet nopeasti IT-alalla. Uudet teknologiat ja nopeampi kehittäminen antavat myös painetta kehittää prosesseja ja etenkin yhteistyötä tuotannonhoidon ja kehityspuolen välillä. Tarkoitus ei ole tehdä asioita niin kuin aina on tehty, vaan miettiä mikä oikeasti toimii ja omaksua uusia ketterämpiä menetelmiä. Tärkeää on jatkuva oppiminen ja asioiden katsominen uudesta näkökulmasta.

Tutkimuskysymyksiin saatiin vastaukset ja TyTA-järjestelmäalueelle saatiin toteutettua DevOps-käytäntöjen mukaan työskentelevä asiantuntijatiimi. DevOps-mallin käyttöönotossa yrityksessä on hyvä lähteä liikkeelle joistain selkeistä askelmerkeistä ja laajentaa näitä pikkuhiljaa. TyTA-tiimi lähti liikkeelle istumapaikoista, starttipäivästä ja loppuvuoden käytäntöjen aikataulusuunnitelmasta. Tarkoituksena oli tarkastella myös DevOpsiin liittyviä hallintavälineitä sekä teorian valossa että TyTA-tiimissä käytännön tasolla. TyTA-tiimi keskittyy töiden hallinnassa ja dokumentaatiossa erityisesti Jiran ja Confluencen hyödyntämiseen hallintavälineinä. Lisäksi testausautomaatiota kehitetään ja mietitään muitakin uusia työkaluja prosessien helpottamiseksi.

Hyötyjä ja haittoja tuli käytyä läpi kattavasti ja tiimissä on nähty DevOps-käytännöille enemmän hyötyjä ja mahdollisuuksia, kuin uhkia ja heikkouksia. Hyötyinä nähtiin muun muassa osaamisen jakaminen, töiden priorisointi, laadun parantuminen, läpinäkyvyys sekä tiimihenki. Haittoina nähtiin ainakin ajanpuute uusien käytäntöjen opettelussa. Perinteiseen ohjelmistokehitykseen verrattuna prosesseista saadaan ketterämpiä tuotannon ja kehityksen yhdistyessä ja ottaessa käyttöön uusia toimintatapoja ja prosesseja. Jatkokehityskohteet käytiin myös läpi ja DevOps tuleekin olemaan TyTA-tiimille jatkuvaa kehittämistä ja kehittymistä.

TyTA DevOps -tiimin perustaminen on tuntunut kaikin puolin positiiviselta ja kannattavalta. Koko työtapaturma- ja ammattitautialueen kokoaminen yhteen tiimiin ja raja-aitojen kaataminen kehityksen ja tuotannon hoidon välillä on ollut organisaation toiveissa. Siksi olikin erittäin kiinnostavaa lähteä pilotoimaan ja ylipäätään tutustumaan DevOps-käytäntöihin ja jalkauttamaan näitä.

Ennen opinnäytetyön tekemistä DevOps-termi oli minulle varsin tuntematon, enkä kunnolla tiennyt, mitä se tarkoittaa. Opinnäytetyöprosessin ja TyTA-tiimin kehittämisen myötä termi on avautunut ja olen huomannut, miten laaja käsite se on. Se voi tarkoittaa eri tiimeissä ja organisaatioissa eri asioita, mutta ytimessä ovat kulttuurin muutos, automaation kasvattaminen, mittaaminen sekä kokemusten ja osaamisen jakaminen.

LÄHTEET

Abdul Rauf, E. M., & Madhusudhana Reddy, E. (2015). Software test automation: An algorithm for solving system management automation problems. *Procedia Computer Science*, *46*, 949–956. https://doi.org/10.1016/j.procs.2015.01.004

Ammann, P., & Offutt, J. (2008). *Introduction to Software Testing*. Retrieved from www.introsoftwaretesting.com.

Atlassian. (2017). Estimate in story points - Atlassian Documentation. Retrieved 23 October 2019, from https://confluence.atlassian.com/jirasoftwareserver073/estimate-in-story-points-861254346.html

Atlassian. (2019a). Kanban - A brief introduction | Atlassian. Retrieved 13 October 2019, from https://www.atlassian.com/agile/kanban

Atlassian. (2019b). Using your Kanban backlog - Atlassian Documentation. Retrieved 8 October 2019, from https://confluence.atlassian.com/jirasoftwarecloud/using-your-kanban-backlog-856846180.html

Atlassian. (2019c). What is DevOps? | Atlassian. Retrieved 8 October 2019, from https://www.atlassian.com/devops

Bass, L. (2017). The Software Architect and DevOps. *IEEE Software*, *35*(1), 8–10. https://doi.org/10.1109/MS.2017.4541051

Boström, J. (2019). Continuous Integration, Delivery, Deployment: What's the difference? Retrieved 20 October 2019, from https://www.eficode.com/blog/ci-cd-difference

Daly, L. (n.d.). Kanplan: where your backlog meets kanban | Atlassian. Retrieved 8 October 2019, from https://www.atlassian.com/agile/kanban/kanplan

Forsgren, N. (n.d.). How to use metrics, measurement to drive DevOps | TechBeacon. Retrieved 12 October 2019, from https://techbeacon.com/devops/how-use-metrics-measurement-drive-devops

Guru99. (2019). What is Regression Testing? Retrieved 12 October 2019, from https://www.guru99.com/regression-testing.html

Hüttermann, M. (2012). *DevOps for Developers*. Retrieved from www.it-ebooks.info

Ian, B. (n.d.). Agile and DevOps - Friends or Foes? | Atlassian. Retrieved 8 October 2019, from https://www.atlassian.com/agile/devops

LocalTapiola. (2019). *How to adapt information security in agile development?*

Matthews, K. (2018). Understanding DevSecOps in Data Science - Towards Data

Science. Retrieved 15 November 2019, from https://towardsdatascience.com/understanding-devsecops-in-data-science-93b695a0d8ab

Methodology, R. (n.d.). Observation - Research-Methodology. Retrieved 23 October 2019, from https://research-methodology.net/research-methods/qualitative-research/observation/

Moore, M. G., & Bovoso, A. L. (2018). *DevSecOps Embedded Security Within the Hyper Agile Speed of DevOps*. Retrieved from https://www2.deloitte.com/content/dam/Deloitte/global/Documents/About-Deloitte/DevSecOps-Explained.pdf

Permana, P. A. G. (2015). Scrum Method Implementation in a Software Development Project Management. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 6). Retrieved from www.ijacsa.thesai.org

Philippot, O. (2017). Continuous measurement in DEVOPS, or how to easily control the efficiency of its applications - GREENSPECTOR®. Retrieved 12 October 2019, from https://greenspector.com/en/articles/2017-03-23-continuous-measurement-in-devops/

Pittet, S. (2019). Continuous integration vs. continuous delivery vs. continuous deployment. Retrieved 18 October 2019, from Atlassian website: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment

Riley, C. (2015). Metrics for DevOps - DevOps.com. Retrieved 13 October 2019, from https://devops.com/metrics-devops/

Riley, C. (2016). *Documenting DevOps: Agile, Automation, and Continuous Documentation*.

Sharma, S. (2017). ProQuest Ebook Central - Reader. In *The DevOps Adoption Playbook : A Guide to Adopting DevOps in a Multi-Speed IT Enterprise*. Retrieved from https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=4789057

Swartout, P. (2012). *Continuous Delivery and DevOps : A Quickstart guide*. Retrieved from https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=1069746&query=#

Tapaturmavakuutuskeskus. (2019). Työtapaturma- ja ammattitautivakuutus - Tapaturmavakuutuskeskus. Retrieved 21 October 2019, from https://www.tvk.fi/tyotapaturma-ja-ammattitautivakuutus/

Thompson, T. (n.d.). Best documentation practices in agile software development. Retrieved 12 October 2019, from https://techbeacon.com/app-dev-testing/why-agile-teams-should-care-about-documentation