Suvi Hela

”Sä et ole enää mikään kummallinen alien, vaan jotain tuttua”

Ohjelmistotuotannon menetelmien soveltuvuus tekniseen viestintään

Markkinoinnin ja viestinnän akateeminen yksikkö
Teknisen viestinnän pro gradu -tutkielma
Teknisen viestinnän maisteriohjelma

Vaasa 2023

VAASAN YLIOPISTO
Markkinoinnin ja viestinnän akateeminen yksikkö
Tekijä: Suvi Hela
Tutkielman nimi: ”Sä et ole enää mikään kummallinen alien, vaan jotain tuttua”: Ohjelmistotuotannon menetelmien soveltuvuus tekniseen viestintään
Tutkinto: Filosofian maisteri
Oppiaine: Tekninen viestintä
Työn ohjaaja: Niina Nissilä
Valmistumisvuosi: 2023 **Sivumäärä:** 86

TIIVISTELMÄ:

Tekniset viestijät tekevät työtään digitaalisessa murroksessa, jossa uudet teknologiat vaikuttavat työtehtävissä tarvittaviin prosesseihin ja työkaluihin. Erityisesti IT-alalla dokumentaation tuottamiseen saatetaan hyödyntää samoja projektinhallintamenetelmiä, työnkulkuja ja työkaluja kuin ohjelmistokehittämiseen. Tämä ilmiö tunnetaan nimellä *Docs as code*. Tutkielman tavoitteena oli selvittää ohjelmistotuotantoalan käytössä olevien menetelmien, kuten ketterien menetelmien ja *Docs as code* -menetelmän, soveltuvuutta tekniseen viestintään, ja tutkia teknisten viestijöiden toimijuutta IT-alan yrityksissä. Teknisen viestinnän alan kehittämiseksi alalle tarvitaan käytännönläheistä tutkimusta, joka voi parhaimmillaan nostaa ammattikunnan statusta organisaatioissa ja vahvistaa teknisten viestijöiden ammatti-identiteettiä.

Tutkimuksen teoreettisessa osuudessa käsitellään teknologian vaikutusta teknisten viestijöiden työtehtäviin sekä teknisen viestinnän alan projektinhallintamenetelmiä että ohjelmistokehityksessä yleistyneitä ketteriä menetelmiä. Laadullisen tutkimuksen aineiston muodostavat IT-alalla työskentelevien teknisten viestijöiden haastattelut, ja haastattelumenetelmänä on käytetty puolistrukturoitua teemahaastattelua. Aineisto on analysoitu aineistolähtöisen sisällönanalyysin avulla.

Tutkimuksessa selvisi, että samojen projektihallintamenetelmien ja toimintamallien noudattaminen ohjelmistokehitysorganisaatiossa luo hyviä edellytyksiä yhteistoiminnalliselle työskentelylle ja integroi tekniset viestijät paremmin osaksi tuotekehitystiimiä parantaen työn laatua ja viestintää organisaation muiden sidosryhmien, kuten tuotejohdon, kanssa. Dokumentaation laatu paranee, kun katselmointi helpottuu ja nopeutuu versionhallintataohjelmistoissa olevien ominaisuuksien avulla. *Docs as code* -menetelmän käyttäminen ja samassa ympäristössä ohjelmistokehittäjän kanssa toimiminen tuo tekniselle viestijälle uskottavuutta ja luo läpinäkyvyyttä teknisen viestijän työpanokseen. Ohjelmistokehittäjille suunnattua dokumentaatiota kirjoitettaessa toimintamallien käyttö luo samalla tekniselle viestijälle ymmärrystä kohdeyleisöstä. Ketterien menetelmien hyödyntämisen haasteena tekniselle viestijälle on työn ulkoistaminen, joka estää tiedon avointa jakamista sekä usealla aikavyöhykkeellä työskentely, jolloin tuotetiimien ketterien menetelmien mukaisiin päivittäisiin tapaamisiin osallistuminen saattaa olla hankalaa ja monien tuotetiimien tapaamisiin osallistuminen saattaa viedä paljon työaikaa. Menetelmien mukainen työskentely vaatii myös lisäkoulutusta tekniselle viestijälle.

AVAINSANAT: tekninen viestintä, tekniset viestijät, dokumentaatio, ketterät menetelmät, projektinhallinta, ohjelmistokehitys

Sisällys

Kuviot

Taulukot

1 Johdanto

Teknologian kehittyminen vaikuttaa useimpien tietotyöläisten arkeen ja pakottaa asiantuntijat sopeutumaan ja etsimään uusia toimivia toimintamalleja työn tekemiseen. Teknisen viestinnän ammattilaiset, tekniset viestijät, ovatkin tottuneet tekemään työtään jatkuvassa digitaalisessa murroksessa, jossa uudet teknologiat vaikuttavat työtehtävissä tarvittaviin prosesseihin ja työkaluihin. Teknologian kehityksen vaikutus korostuu erityisesti IT-alalla, jossa tekniset viestijät tarvitsevat erityisesti teknisiä taitoja käyttäessään ohjelmistotuotantoalalle suunnattuja käytänteitä ja ohjelmistokehittäjille suunnattuja työkaluja päivittäisissä työtehtävissään. Haikalan ja Mikkosen (2011, s. 11) määrittelevät, että ohjelmistotuotannolla (*software engineering*) tarkoitetaan ohjelmistojen tekemiseen tarvittavia tekniikoita, työkaluja, menetelmiä ja periaatteita. Clearyn (2021, s. 128) mukaan ohjelmistotuotantoala on yksityisen sektorin yksi suurimmista teknisten viestijöiden työllistäjistä.

Salmelan ja Isohellan (2021, s. 47) mukaan teknisen viestijän tarvitsemat tekniset taidot liittyvät tietoon dokumentoitavasta aiheesta tai tuotteesta, mutta myös kykyyn ottaa käyttöön uusia teknologioita sekä dokumentaation laatimiseen tarvittavaan tekniseen osaamiseen. Salmela ja Isohella (2021, s.42) määrittelevät teknisen viestinnän toiminnan näkökulmasta, jossa välitetään teknistä tietoa siltä, joka tietää, sille, jolla on tarve tietää. Teknistä viestintää voidaan määritellä myös teknisen viestinnän toiminnan tuloksena olevien informaatiotuotteiden näkökulmasta, joiden pyrkimys on välittää objektiivista tietoa (Salmela ja Isohella, 2021, s. 42). Tässä tutkielmassa teknisellä viestijällä tarkoitetaan ammattiryhmää, jotka tuottavat päätyönään ohjelmistokehitysyrityksissä tarvittavaa dokumentaatiota osana tuotekehitystiimejä.

Salmela ja Isohella (2021, s. 51) huomauttavat, että vaikka teknologia on muuttanut yrityksien tapoja tarjota palveluita asiakkaille, yritykset silti tarvitsevat selkeää ja tehokasta viestintää, jolloin tarvitaan teknisen viestinnän ammattilaisia kehittämään käyttäjiä auttavaa dokumentaatiota. IT-alalla teknisen viestijän tuottama dokumentaatio voi olla esimerkiksi ohjelmiston loppukäyttäjille suunnattu käyttöohje,

versiotiedote tai esimerkiksi ohjelmistorajapintojen käyttöön ohjelmistokehittäjille tarkoitettu ohjeistus.

Ohjelmistotuotteiden vaatimukset ja spesifikaatiot muuttuvat jatkuvasti, ja teknisten viestijöiden tulee sopeutua jatkuvasti muuttuvaan toimintaympäristöön. Ohjelmistotuotantoon käytettävät ketterät menetelmät ovat yleistyneet viime vuosina, joten teknisten viestijät joutuvat sopeutumaan ja toimimaan kehityssyklien vaihtuessa nopeatempoisesti. Tietotekniikan termitalkoot (Sanastokeskus, 2023) määrittelee ketterän ohjelmistokehityksen ”ohjelmistokehitykseksi, jonka tavoitteena on saada nopeasti aikaan toimiva ohjelmisto, viestiä tehokkaasti ohjelmiston tekijöiden ja asiakkaan kanssa sekä reagoida ripeästi muutoksiin”. Toimintaympäristöllä tarkoitetaan määritelmän mukaista nopeatempoista ohjelmistokehityksen toimintaympäristöä, jossa ohjelmistokehitystuotteen kehityssyklit määrittelevät työn aikataulun ja esimerkiksi muuttuneet ohjelmistotuotteen vaatimukset saattavat aiheuttaa Sanastokeskuksen määritelmän mukaisia muutoksia.

Samojen työkalujen ja prosessien käyttö voi integroida teknisen viestijän paremmin osaksi muuta tiimiä. Toisaalta ohjelmistotuotannon tarpeisiin suunnatut prosessit ja ohjelmistokehittäjien työkalut saattavat kuitenkin myös asettaa haasteita teknisen viestinnän työtehtäviin. Lanierin (2018, s. 76) tutkimuksen mukaan teknisen viestinnän alalla esiintyy huolta ohjelmistotuotannon menetelmien soveltuvuudesta teknisen viestijän työtehtäviin, sillä erityisesti menetelmien asettama tiukka aikataulu saattaa vaikuttaa teknisen dokumentaation laatuun.

Jotta tekniset viestijät voivat ammattikuntana selvitä alalla vaikuttavista muutoksista tulisi Andersenin (2014) mukaan tuottaa käytännönläheistä tutkimusta alalla toimivista hyvistä käytänteistä ja siitä miksi ne toimivat sekä miten käytänteitä tulisi parantaa. Teknisen viestinnän alan kehittämisen ja arvostuksen kannalta tutkimus on tärkeä, sillä se saattaa parantaa ymmärrystä alan käytännön työtehtävistä IT-alan yrityksissä. Tutkimus saattaa myös auttaa vahvistamaan alalla työskentelevien ammatti-

identiteettiä ja antaa näkyvyyttä alalla tarvittavaan ammatin statuksen nostamiseen organisaatioiden sisällä. Tutkimusta voidaan mahdollisesti hyödyntää teknisen viestinnän alan käytänteiden kehittämiseen.

1.1 Tavoite

Tutkielman tavoitteena on selventää ohjelmistotuotantoalan käytössä olevien menetelmien soveltuvuutta tekniseen viestintään ja tutkia teknisten viestijöiden toimijuutta IT-alan yrityksissä. Alalla käytetään projektihallintaan esimerkiksi ketteriä menetelmiä (*agile methods*). Project Management Institute (2021, s. 4) määrittelee projektinhallinnan projektin toimintaan liittyvän tiedon, taitojen, työkalujen ja tekniikoiden soveltamiseksi projektin tavoitteiden täyttämiseksi. Projektinhallinnan avulla voidaan ohjata projektin aikaista työtä sovittujen päämäärien tavoittamiseksi (Project Management Institute, 2021, s. 4). Ketterällä menetelmällä tarkoitetaan iteratiivista kehitysprosessia, jossa vaatimukset ja sovellukset kehittyvät jatkuvasti tiimien sisäisen yhteistyön ja sidosryhmien palautteen avulla (ISO/IEC/IEEE 26515:2018). Lisäksi IT-alan yrityksissä saatetaan käyttää teknisen dokumentaation tuottamiseen usein samoja menetelmiä kuin ohjelmistokoodin kirjoittamiseen. Tätä ilmiötä kutsutaan *Docs as code* -menetelmäksi.

Tutkimuksessa selvitetään, miten menetelmiä hyödynnetään teknisen dokumentaation laatimisen välineenä ja mitä vaikutuksia niillä on esimerkiksi ammatin arvostukseen organisaatiossa tai työn laatuun yhteistyökulttuurin lisääntyessä ketterien menetelmien myötä. Tutkimustavoitteen saavuttamiseksi olen laatinut kaksi tutkimuskysymystä.

1. Miten ohjelmistotuotannon menetelmiä hyödynnetään tutkimukseen osallistuneiden teknisten viestijöiden työpaikoilla?

Ensimmäisen tutkimuskysymyksen tavoitteena on selventää, miten ohjelmistotuotannon menetelmiä hyödynnetään tutkimukseen osallistuvien haastateltavien työnantajayrityksissä. Yrityksessä saattaa olla käytössä esimerkiksi

yleisesti ohjelmistotuotantoon soveltuvia projektinhallintamenetelmiä, kuten ketterät projektinhallintamenetelmät tai tarkemmin teknisen dokumentaation tuottamiseen suunnattu menetelmä *Docs as code*. Tutkimuskysymys selvittää mitkä menetelmät ovat käytössä ja miten niitä hyödynnetään teknisen dokumentaation tuottamiseen kohdeyrityksissä.

2. Mitä vaikutuksia ohjelmistotuotannon menetelmien käytöllä on teknisen dokumentaation tuottamiseen?

Toisen tutkimuskysymyksen tavoitteena on selvittää, minkälaisia vaikutuksia ohjelmistotuotannon menetelmien käytöllä on teknisen dokumentaation tuottamiseen. Ketterät projektinhallintamenetelmät korostavat avointa yhteistyökulttuuria ja viestintää, joten pohdin tutkielmassani, onko tällä vaikutusta esimerkiksi teknisen viestijän ammatin arvostukseen, jos tekninen viestijä on osana ketterää tuotekehitystiimiä. *Docs as code* -menetelmässä tekninen viestijä käyttää dokumentaation hallintaan samoja työnkulkuja ja työkaluja, kuin ohjelmistokehittäjät käyttävät ohjelmistokoodin tuottamiseen. Menetelmään liittyviä työtapoja esitellään tarkemmin luvussa 3.3. Tutkimuskysymyksellä selvitetään minkälaisia vaikutuksia menetelmien ja työkalujen käytöllä on tekniseen viestintään ja miten ohjelmistokehitykseen suunnatut menetelmät sopivat teknisen dokumentaation laatimiseen.

1.2 Tutkimusaineisto

Tutkimusaineisto muodostuu viiden teknisen viestijän teemahaastatteluista saadusta materiaalista. Teemahaastattelulla tarkoitetaan aihepiireihin eli teemoihin jaettua haastattelua, jossa haastattelun teemat ovat kaikille haastateltaville samat (Hirsjärvi & Hurme, 2001, s. 48). Puolistrukturoidussa haastattelussa kysymykset ovat kaikille samat, mutta vastausvaihtoehtoja ei ole sidottu valmiisiin vastausvaihtoehtoihin, kuten strukturoidussa lomakehaastattelussa, vaan haastateltavat voivat vastata vapaammin

omin sanoin (Hirsjärvi & Hurme, 2001, s. 47). Hirsjärven ja Hurmeen (2001) mukaan puolistrukturoiduille menetelmille on ominaista, että jokin haastattelun näkökohta on lyöty lukkoon. Tutkimuksessani lukittuja näkökohtia olivat haastattelun aihepiirit, ja haastattelijana pystyin esimerkiksi vaihtamaan ennalta laadittujen kysymysten sanamuotoa haastattelun aikana. Hirsjärven ja Hurmeen (2001) mukaan teemahaastattelun avulla voidaan tuoda tutkittavien ääni paremmin kuuluviin, joten tämä sopii tutkimustarkoitukseeni erittäin hyvin, sillä tutkimuksen tarkoituksena on selvittää käytännön kokemuksia menetelmien toimivuudesta teknisen viestinnän työtehtävissä. Hirsjärven ja muiden (2010, s. 206) mukaan teemahaastattelu sopii hyvin tiedonkeruumuodoksi, sillä haastateltaviksi kaavaillut henkilöt voidaan tällä tutkimusmenetelmällä saada mukaan tutkimukseen. Haastateltavat voidaan myös tavoittaa haastattelujen jälkeen tarpeen mukaan (Hirsjärvi ja muut, s. 206).

Haastateltavien löytämiseksi hyödynsin Suomen teknisen viestinnän yhdistyksen (STVY ry) sosiaalisen median alustoja, kuten esimerkiksi yhdistyksen Facebook ja LinkedIn-sivustoja. Julkaisin haastattelukutsun helmikuussa 2022 edellä mainituilla sosiaalisen median alustoilla. Haastattelukutsu on esitetty liitteessä 2. Etsin yhdistyksen verkostosta IT-alan yrityksissä toimivia teknisiä viestijöitä haastattelua varten. Rajasin haastateltavat tässä tapauksessa IT-alan yrityksissä toimiviin teknisen viestinnän ammattilaisiin, jotka toimivat esimerkiksi teknisen kirjoittajan työtehtävissä. Haastattelut toteutettiin yksilöhaastatteluina noin 0,5–2 tuntia kestävinä videopuheluina Zoom-palvelun avulla. Viidestä haastattelusta kaksi oli suomenkielisiä ja kolme englanninkielisiä. Yksittäisten haastatteluiden kestot on merkitty alla olevaan taulukkoon (Taulukko 1).

Taulukko 1. Teemahaastatteluiden kestot.

Asiantuntija	**Haastattelun kesto (minuuttia)**
H1	111
H2	43
H3	57
H4	73
H5	33
Yhteensä	**317 min.**

Haastateltavat ovat teknisen viestijän työtehtävissä toimivia tai toimineita eri toimialojen IT-alan yrityksistä. Kolmella haastateltavalla oli kullakin vähintään kymmenen vuoden kokemus teknisen viestinnän työtehtävistä, yhdellä haastateltavalla oli yli 20 vuoden kokemus alalta ja viimeisellä haastateltavalla noin 2 vuoden kokemus teknisenä kirjoittajana pelialan yrityksestä. Neljän haastateltavan dokumentaation kohdeyleisönä ovat ohjelmistokehittäjät tai henkilöt, joilla on tekninen tausta.

Kolme haastateltavista toimii teknisen viestijän työtehtävissä, ja yksi näistä kolmesta toimi samalla teknisen dokumentaatiotiimin vetäjänä. Yksi haastateltavista työskentelee ohjelmistokehitystiimin esimiehenä tuoteomistajan roolissa (*Product Owner*), mutta työtehtäviin kuuluu myös teknisen viestinnän tehtävät. Yhdellä haastateltavista oli useamman kymmenen vuoden kokemus teknisen viestijän työtehtävistä, mutta tällä hetkellä hän toimii organisaatiossa DocOps -tiimissä (*Documentation Operations*), jonka vastuualueeseen kuuluu mm. teknisten viestijöiden tiimien työkalujen ja toiminnan mahdollistaminen. Haastateltavat on valittu siten, että heillä on ollut käytännön kokemusta ohjelmistotuotannon ketterässä kehitysympäristöstä toimimisesta sekä mahdollisesti heillä käytännön kokemusta myös *Docs as code* -tyyppisestä työskentelystä. *Docs as code* -menetelmän tuntemus ei kuitenkaan ollut vaatimus tutkimukseen osallistumiselle, sillä tutkimuksen tarkoituksena oli selvittää yleisesti ohjelmistotuotannon käytänteiden soveltuvuutta.

Haastateltavien koulutustausta vaihteli hieman. Kahdella haastateltavalla oli yliopistotasoinen teknisen viestinnän alan koulutus. Kolmella muulla haastateltavalla ei ollut virallista teknisen viestinnän alan koulutusta, vaan he ovat oppineet alan työtä tekemällä. Yhdellä haastateltavalla oli kaksi kandidaatintutkintoa; toinen tietojenkäsittelytieteestä ja toinen englantilaisesta filologiasta. Yhdellä haastateltavalla oli kirjallisuuden alan kandidaatin tutkinto.

Haastattelukysymykset oli jaoteltu yhteensä kuuteen kategoriaan ja niitä on yhteensä 17. Haastattelukysymykset on koottu liitteeseen 1. Kysymykset olivat osittain samoja, joita

on käytetty Virtaluodon (2015) väitöstutkimuksessa. Virtaluodon tutkimuksen ensimmäisen vaiheen (2015, s. 78) kysymykset voidaan jakaa seuraaviin kategorioihin: taustatiedot, teknisen viestinnän koulutus ja työhistoria alalla, tuotteet ja dokumentaatiotyö, dokumentoinnin prosessi, esimiesvastuu, organisaatio, käyttäjätieto ja dokumentaation käytettävyystestit, etätyö ja työmatkat, visualisointi ja kuvitus, työhön liittyvä lisäkoulutus, työn laatu ja arvostus sekä urakehitys ja tulevaisuuden näkymät.

Lisäsin muutamia menetelmien käyttöön ja dokumentaatioprosessin kuvaamiseen liittyviä kysymyksiä. Virtaluodon käyttämät kysymykset on merkitty taulukkoon (Liite 1). Olen jättänyt oman tutkimukseni kannalta epäolennaisempia kysymyksiä pois. Näitä olivat esimerkiksi urapolkuun ja urakehitykseen, tiiminvetäjän ja budjetointivastuuseen, etätyöhön ja työmatkustamiseen liittyvät kysymykset. Tiedon visualisointiin ja kuvitukseen liittyvät kysymykset olen myös rajannut haastattelututkimukseni ulkopuolelle.

Haastattelukysymykset on jaettu seuraaviin pääkategorioihin:

Taustatiedot. Taustatietoihin kuuluvat kysymykset esitettiin haastattelun alussa. Kysymysten tarkoituksena on selvittää haastateltavan ikä, koulutustausta, työhistoria ja mahdollinen tausta muista IT-alan työtehtävistä.

Organisaatio ja tiimi. Kysymykset haastateltavan tiimiin ja sen sijaintiin organisaatiossa on asetettu toiseksi taustakysymysten jälkeen. Kysymysten tarkoituksena oli selvittää haastateltavan sijainti organisaatiossa, eli kuuluuko haastateltava esimerkiksi osaksi tuotekehitystiimiä vai onko haastateltava osana omaa erillistä teknisen sisällöntuotannon tiimiä.

Dokumentaation tuottamisen prosessi. Taustoittavien kysymysten jälkeen selvitin, minkälaista prosessia teknisen dokumentaation tuottamiseen noudatetaan haastateltavan organisaatiossa.

Työkalut. Kysymyksillä selvitettiin sisällöntuotantoon käytettäviä ohjelmistoja. Onko kohdeyrityksissä käytössä esimerkiksi *Docs as code* -menetelmälle tyypillisiä ohjelmistoja käytössä, kuten versionhallinta.

Menetelmät. Menetelmiin liittyvillä kysymyksillä selvitettiin, minkälaisia menetelmiä yrityksissä on käytössä yleisesti projektinhallintaan sekä erityisesti teknisen dokumentaation tuottamiseen liittyen. Tarkoitus oli myös selvittää haastateltavan mielipidettä edellä mainittujen menetelmien soveltuvuudesta teknisen viestinnän työtehtäviin.

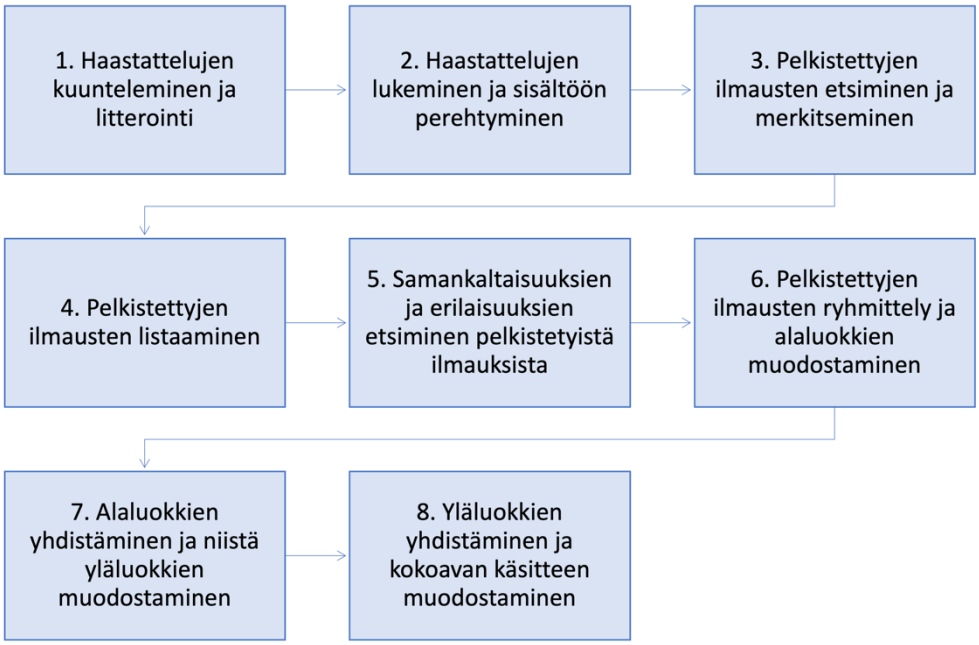
Menetelmien vaikutus työn laatuun ja arvostukseen. Viimeiseen osioon kokosin kysymyksiä menetelmäosaamisen arvostuksesta organisaatiossa ja niiden vaikutuksesta työn onnistumiseen, jolla tarkoitetaan esimerkiksi työn laatua, aikataulua ja työn yleistä sujuvuutta. Kysymyksillä selvitettiin myös sitä, ovatko menetelmät muuttaneet teknisen viestijän työnkuvaa aiempaan verrattuna.

1.3 Tutkimusmenetelmä

Laadullinen aineisto on kerätty teemahaastatteluiden avulla, ja aineiston analyysin tutkimusmenetelmänä käytin aineistolähtöistä sisällönanalyysiä. Haastattelujen jälkeen litteroin haastatteluaineiston, jotta voin paremmin analysoida aineistoa. Litteroin aineiston automaattisella litterointityökalulla, joka on saatavilla Word-ohjelmiston selainversiossa. Automaattisen litteroinnin jälkeen tarkistin ja korjasin tarvittaessa automaation luomia litteroituja aineistoja. Aineiston litteroinnin jälkeen kuuntelin jokaisen haastattelun äänitiedoston ja luin litteroidun aineiston useaan kertaan. Litteroinnin jälkeen toin aineiston laadullisessa analyysissä käytettävään Atlas.ti -

työkaluun tarkempaa analyysiä varten. Käytin analyysin apuna ohjelmiston selainpohjaista versiota (Atlas.ti Web).

Luokittelin aineistoa ja tarkastelin aineistosta mahdollisia löytyviä uusia teemoja. Käytin luokittelun pohjana ensimmäisessä vaiheessa edellisessä luvussa mainittuja haastattelukysymysten aihepiirejä. Luokittelu perustui Tuomen ja Sarajärven (2018, s. 140) kantaan sisällönanalyysin etenemisestä, joka on esitetty kuviossa 1.



Kuvio 1. Aineistolähtöinen sisällönanalyysi.

Kuvio 1 esittää aineistolähtöisen sisällönanalyysin etenemisen (Sarajärvi & Tuomi, 2018, s. 140). Ensimmäisissä vaiheissa tutustuin aineistoon, jonka jälkeen järjestin aineiston koodaamalla. Tämän jälkeen pyrin pelkistämään aineiston tutkimusaihetta kuvaaviksi esimerkeiksi. Muodostin tutkimusaiheeseen liittyviä alaluokkia aineistosta. Muodostuneita alaluokkia olivat esimerkiksi **työmäärän arviointi**, **työn laatu** ja **viestinnän helpottuminen organisaatiossa**. Yhdistin alaluokkia ja muodostin niistä sen jälkeen yhdistäviä yläluokkia. Edellä mainittujen alaluokkien yhdistäväksi yläluokaksi muodostui ketterän kehityksen hyödyt. Muita esimerkkejä löytämistäni tutkimusaiheen kannalta merkityksellisistä aineistolähtöisen analyysin yläluokista olivat ketterän

kehityksen haasteet, *Docs as code* -menetelmän hyödyt ja haasteet tekniselle viestijälle, työnkuvan muuttuminen, organisaation asenteet sekä teknisen viestijän tuottama lisäarvo IT-alan yrityksissä.

2 Teknisen viestinnän rooli ja projektinhallinta

Tämä luku käsittelee teknisen viestinnän roolia organisaatioissa sekä projektinhallintaa dokumentaatioprojekteissa. Esittelen teknisten viestijöiden roolia ja sen muutosta uusien teknologioiden myötä. Käsittelen myös teknisen viestinnän alalle luotuja projektinhallintamalleja. Salmelan ja Isohellan (2021, s. 44) mukaan teknisistä viestijöistä on työelämässä tapahtuneiden muutoksien myötä tullut moniosaajia, jotka hallinnoivat työssään projekteista, prosesseista ja sisällönhallinnasta koostuvia kokonaisuuksia. Tämän vuoksi projektinhallintataidot ovat teknisen viestinnän alalla keskeisessä roolissa tiimityöskentelytaitojen ohella (Salmela ja Isohella, 2021, s. 44).

Teknisen viestinnän ammattilaiset tuottavat dokumentaatiota pääasiassa englanniksi, jonka vuoksi alalla onkin yleisesti käytössä englanninkielisiä ammattinimikkeitä kuten *technical writer, documentation specialist* ja *information designer* (Suojanen, 2018). Virtaluodon (2015) mukaan suuri osa suomalaisista teknisistä viestijöistä työskentelee vientiteollisuuden parissa. Teknisen viestinnän tutkimus on muutamia poikkeuksia lukuun ottamatta pohjoisamerikkalaista. Virtaluoto (2015) toteaa kuitenkin tutkimuksessaan, että alan ongelmakenttä on hyvin pitkälti samanlainen myös Suomessa. Ala kärsii teknisen viestinnän ammatin statuksen puutteesta, alalla vallitsee kuilu akateemisen maailman ja ammatinharjoittajien välissä sekä lisäksi alan tulevaisuudennäkymät vaikuttavat vaikeilta kustannuspaineiden vuoksi.

2.1 Roolin muutos uusien teknologioiden myötä ja teknisen viestinnän status organisaatioissa

Suojasen (2018, s. 40) mukaan tekninen viestintä on jatkuvassa muutoksessa, sillä sitä määrittävä tutkimuskohde, teknologia, muuttuu jatkuvasti. Teknologialla on aina ollut suuri rooli teknisen viestinnän alalla. Ensin teknologia oli merkittävä aihepiiri, josta tekniset viestijät kirjoittivat työtehtävissään, mutta myöhemmin teknologian myötä syntyneet työkalut ovat myös helpottaneet teknisen viestijän työtä (Carliner, 2009). Dicks

(2009) kuvaa muutoksen nopeutta ja merkittävyyttä siten, että monet tekniset viestijät suorittavat nykyään sellaisia alan työtehtäviä, joista eivät olleet kuulleetkaan viisi vuotta sitten.

Carlinerin (2009) mukaan teknisen viestinnän ammattikunnalle muodostuikin 2000-luvulle tultaessa uudenlaisia työtehtäviä. Teknisen viestinnän ammattilaiset saattoivat esimerkiksi suunnitella verkkosivujen käyttäjäkokemusta ja käyttöliittymän toiminnallisuutta. Toisaalta teknisen viestijät suunnittelivat sisältöä ja sen käytettävyysarvioita. Carlinerin (2009) mukaan tämänkaltaiseen työtehtävien ositukseen on syynä työtehtävien ulkoistuksen lisäksi sisällönhallintajärjestelmien (*Content Management System*) kehittyminen, jolloin substanssiasiantuntijat (*Subject Matter Experts*) voivat itse helpommin luoda sisältöä järjestelmien sisältämien pohjien avulla.

Teknisen viestijän rooli on muuttunut 2000-luvulla lineaarisesta työskentelymallista yhteistoiminnalliseksi prosessiksi (Suojanen, 2018, s. 40). Teknologia on siis vaikuttanut vahvasti alan työprosesseihin ja työnkulkuun. Tehostettujen sisällön julkaisuprosessien myötä organisaatiossa on otettu käyttöön uusia työkaluja ja prosesseja esimerkiksi versionhallintaan, joiden avulla voidaan helpottaa lisääntynyttä työmäärää (Carliner, 2009). Clearyn (2021, s. 75) mukaan monet tekniset viestijät toimivat osana monialaisia ketteriä tiimejä, mikä johtaa yhteistyön lisääntymiseen.

Virtaluodon (2015, s. 59) väitöstutkimuksen mukaan teknisen viestinnän ammattilaiset tarvitsevat oman alansa yhteisön tuen selviytyäkseen ammattikuntana, jotta voidaan vahvistaa ammattiin liittyviä taitoja ja teknisen viestinnän tuomaa lisäarvoa organisaatioissa. Tässä tutkielmassa aikomuksenani onkin pohtia voiko se, että tekninen viestijä käyttää samoja työtapoja ja työkaluja verrattuna muuhun organisaation nostaa ammatin arvostusta. Clarkin ja Andersenin (2015, s. 292) mukaan teknisten viestijöiden tulisi tehdä organisaation sisäistä strategiatyötä, jossa teknisen viestijän työtehtävät integroituisivat osaksi yrityksen muuta prosessia, jolloin työtehtävät eivät olisi omassa eristetyssä siilossa. Teknisiltä viestijöiltä puuttuu myös uskottavuutta organisaatioissa,

jonka vuoksi tekniset viestijät eivät saa ääntänsä kuuluviin, kun johto tekee liiketoimintakriittisiä päätöksiä teknisen viestinnän ydinalaan, tiedon hallintaan, liittyvissä kysymyksissä (Clark ja Andersen, 2015, s. 293).

2.2 Teknisen dokumentaation projektinhallintamallit

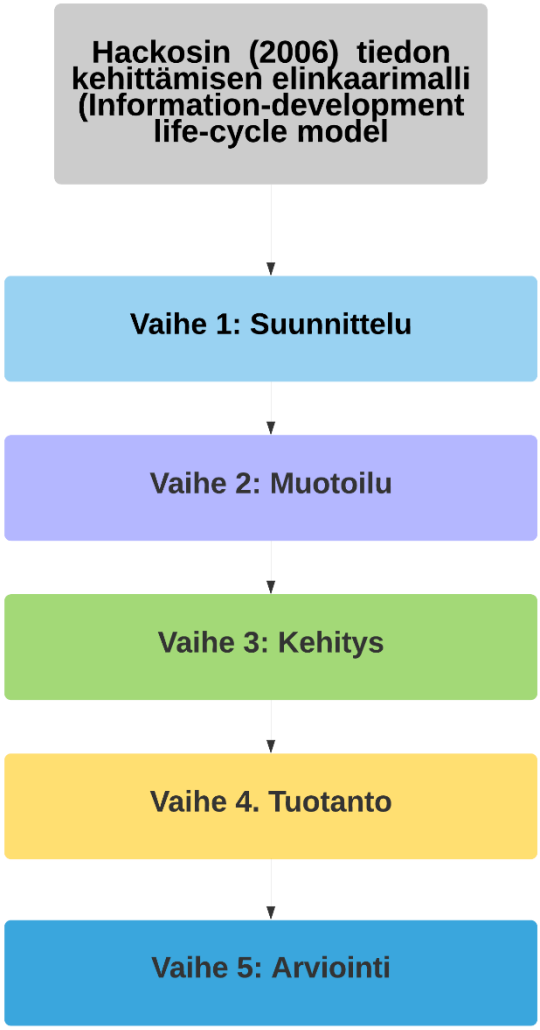
Teknisen viestinnän alalle luodut projektinhallintamallit sisältävät elementtejä ohjelmistotuotannon projektihallinnan malleista. Esittelen teknisen viestinnän alalle luotuja perinteisiä projektinhallinnan prosessimalleja, joista tunnetuin on Hackosin (2006) malli, joka on alun perin esitetty teoksessa *Managing Your Documentation Projects* vuonna 1994. Hackos on tämän projektinhallintamallin lisäksi kehittänyt teknisen dokumentaation tuottamisen prosessin kypsyysmallin (*Information Process Maturity Model, IPMM)* vuonna 1992 (Hackos, 2017, s. 1). IPMM-malli on alunperin luotu täydentämään ohjelmistotuotantoyritysten kypsyysmallia (*Software Engineering Instute, SEI-model*), joka ei eritellyt teknisen viestijän roolia ohjelmistotuotantoyrityksissä (Hackos, 2017, s. 1). Hackosin mallin jälkeen esittelen tässä kappaleessa myös Kisterin (2016) jalostetun iteratiivisen kehitysmallin (*Refined Iterative Development Model*). Käsittelen näitä malleja, sillä ne tuovat taustatietoa teknisen viestijän työtehtävistä projektin eri vaiheissa.

Tässä kappaleessa vertailen Hackosin (2009) ja Kisterin (2016) kehitysmalleja, jotka on suunniteltu teknisen dokumentaation tuottamiseen. Molemmissa malleissa projektin ensimmäisen vaiheen aiheena on projektin suunnittelu, mutta niiden sisältö eroaa hieman toisistaan. Hackosin mallin (2009) suunnitteluvaiheessa luodaan yleinen projektisuunnitelma, mutta Kisterin mallissa (2016) suunnitteluvaiheessa tarkoitetaan yksityiskohtaisempaa projektin rakennetta, jossa voidaan jo määrittää projektissa toimitettavan dokumentaation tarkempaa sisältöä. Molemmat projektinhallintamallit korostavat laadunvarmistuksen merkitystä dokumentaatioprojekteissa, minkä vuoksi laadunvarmistamiseen liittyvät toimenpiteet on kuvattu molemmissa malleissa. Hackos (2009) lisäksi korostaa kehitysmallissaan projektihallinnan merkitystä ja on eritellyt projektipäällikön tehtävät projektin jokaisessa vaiheessa. Kisterin (2016) mallissa

projektihallinnan merkitystä ei ole erityisesti korostettu. Kisterin kehitysmalli sisältää kolme iteratiivista vaihetta, jotka on tarkasti merkitty osaksi projektin peräkkäisesti eteneviä vaiheita. Hackosin malli ei erottele projektin iteratiivisia vaiheita yhtä tarkasti ja liitä niitä projektin vaiheisiin. Virtaluodon ja muiden (2018, s. 192) mukaan Kisterin malli pohjautuu tuotekehitysprojekteihin, mutta sitä voi hyödyntää myös dokumentaation tuottamiseen.

2.2.1 Hackosin malli

Hackosin (2006) luoma malli on viisiportainen malli, joka on rinnastettavissa monien tuotekehitysorganisaatioiden käyttämiin menetelmiin. Hackos (2006, luku 3) huomauttaa, että viisiportainen malli on samankaltainen esimerkiksi Highsmithin (2000) ketterän kehityksen projektinhallinnan viisiportaiseen malliin verrattuna. Hackosin (2006, luku 3) mukaan muuhun organisaatioon verrattuna samankaltaisten projektinhalllintamallien käyttäminen auttaa sen käyttöönotossa, sillä se sisältää tuttuja elementtejä, jolloin niiden tehokkuutta ja monipuolisuutta on helpompi perustella kollegoille. Alla olevassa kuviossa (Kuvio 2) on esitetty Hackosin (2006) tiedon kehittämisen elinkaarimalli.



Kuvio 2. Hackosin (2006) tiedon kehittämisen elinkaarimalli.

Malli on jaettu viiteen eri vaiheeseen. Ensimmäinen vaihe on **tiedon suunnittelu.** Tässä vaiheessa luodaan tiedon tuottamisen projektisuunnitelma, johon määritellään isomman projektin tavoitteet, jonka osana informaation tuottamisen projekti on. Hackosin (2006, luku 3) malli korostaa suunnitteluvaiheen tärkeyttä ja pitää sitä koko projektin kannalta kriittisenä vaiheena. Projektisuunnitelman avulla voidaan varmistua projektin kannalta riittävistä resursseista, voidaan tuottaa asiakkaiden tarpeita vastaava lopputuotos ja ylläpitää tiimin jäsenien työmoraalia. Suunnitteluvaiheessa luodaan ymmärrystä kohderyhmän tarpeista, määritellään projektin laajuus, määritellään

tarpeet lokalisointiin ja kääntämiseen liittyen, määritellään vaatimukset lopputuotteelle, hankitaan projektiin tarvittavat henkilöstöresurssit, suunnitellaan projektivaiheiden riippuvuudet ja riskinhallintaa sekä lopuksi viestitään projektin tavoitteet ja visio kaikille sidosryhmille. Hackos (2006, luku 3).

Toinen vaihe on **tiedon muotoilu.** Hackos (2006, luku 3) huomauttaa, että vaikka tiedon muotoilu on merkitty tähän elinkaarimalliin toiseksi vaiheeksi, se on itseasiassa osa jatkuvan parantamisen prosessia, jossa tavoitteena on ymmärtää kohderyhmien informaatiotarpeet ja organisaation liiketoiminnalliset tavoitteet. Tämän vaiheen työstäminen voi todellisuudessa alkaa edellisen projektin päättymisen jälkeen, kun saadaan palautetta asiakkailta ja muilta organisaation sisäisiltä sidosryhmiltä. Projektikohtainen tiedon muotoilu on kuitenkin määritelty toiseksi vaiheeksi Hackosin (2006) mallissa. Tässä vaiheessa toteutetaan käyttäjäanalyysi ja pyritään ymmärtämään dokumentoitavan tuotteen tai palvelun tavoitteet käyttötapausten avulla, määritetään käytettävyyteen liittyvät tavoitteet, määritellään informaation rakenne ja kirjoitustyyli sekä määritellään projektin lopputuotteet. Tämän vaiheen oheistuotteena syntyy muotoilusuunnitelma (*Design Plan),* rakenteisen dokumentaation *topic*-lista ja kirjoitusohjeita. Hackos (2006, luku 3).

Kolmas vaihe on **tiedon kehittäminen.** Tiedon kehittämisen vaihe on ajallisesti laajin kaikista projektivaiheista. Tässä vaiheessa tuotetaan kaikki sisällöntuotantoon liittyvät työt. Hackosin (2006, luku 3) mukaan sisällöntuotannon ohella tässä vaiheessa suoritetaan laadunvarmistamiseen liittyviä toimenpiteitä, kuten katselmointia ja oikolukua. Tuotteen informaatiosisältöä tulisi validoida organisaation alan asiantuntijan kanssa (*SME*) palautteen saamiseksi. Vaiheen oheistuotteena syntyy viikoittaisia projektin edistymiseen liittyviä raportteja, luonnos projektin lopputuotteesta, viimeistelty rakenteisen dokumentaation *topic*-lista ja päivitetty muotoilusuunnitelma. Hackos (2006, luku 3).

Neljäs vaihe on **tuotanto.** Hackosin (2006, luku 3) mukaan tässä vaiheessa tuotetaan ja toimitetaan projektin lopputuotteet. Yksityiskohtaiset vaiheet tässä projektin vaiheessa riippuvat Hackosin mukaan käytössä olevista työkaluista ja menetelmistä. Mikäli käytössä on esimerkiksi XML-pohjaisia sisällöntuotantojärjestelmiä, jotka mahdollistavat automatisoidun julkaisun, tuotantovaiheeseen ei tarvita Hackosin mukaan juurikaan aikaa. Tässä vaiheessa oheistuotteena syntyy tuotantosuunnitelma, tuotannon testausta luonnosversiolla ja katselmoidun dokumentin hyväksyntä. Vaiheen lopussa tuotetaan projektin lopputuote. Hackos (2006, luku 3).

Viides vaihe on **arviointi.** Arviointivaihe toteutetaan heti, kun lopputuote on valmis toimitettavaksi. Vaiheessa arvioidaan projektin menestyksekkyyttä, haasteita, suunnitellaan parannuksia ja varmistetaan, että projektin lopputuotteena ollut dokumentti vastaa asiakkaan tarpeita sekä tuottaa asiakkaalle lisäarvoa. Hackos (2006, luku 3).

2.2.2 Kisterin malli

Kisterin (2016) luoman kehitysmallin tarkoituksena on parantaa prosesseja informaation kehittämisen projekteissa. Malli on erityisesti suunniteltu helpottamaan nopeaa ja lisäävää (*incremental*) toimittamista laatutekijöistä ja tehokkuudesta tinkimättä. Kisterin (2016) mukaan malli on suunniteltu tukemaan liiketoiminnallisten tavoitteiden saavuttamista.

Kisterin (2016) malli pohjautuu useiden eri alojen käytössä olevien kehitysmallien analyysiin niiden yhtenäisistä piirteistä. Tämän vuoksi malli sisältää piirteitä myös ohjelmistotuotantoalan ketteristä menetelmistä. Kisterin (2016, s. 196) mukaan eri alojen kehitysmallien yhtenäisinä piirteinä voidaan pitää sitä, että ne keskittyvät usein kuvaamaan prosessia, sillä mallit kuvaavat usein projektin eri vaiheita, ne ovat usein myös vaiheittaisia (*sequential*) ja iteratiivisia samaan aikaan, jolloin projekteissa on usein peräkkäin tapahtuvien vaiheiden lisäksi toistuvia iteratiivisia prosesseja jatkuvan

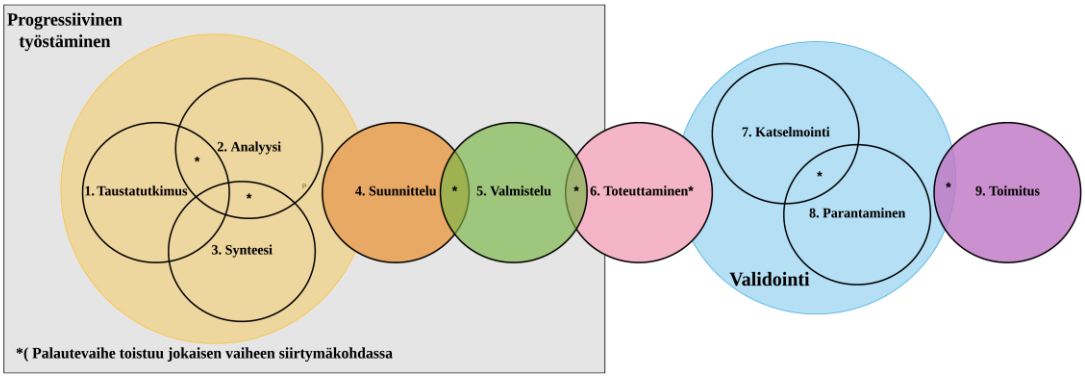
parantamisen varmistamiseksi. Tämän vuoksi Kisterin (2016, s. 193) mukaan projektinhallinnan alalla usein nähtävä vastakkainasettelu perinteisten vaiheittaisten vesiputousmallien ja iteratiivisten ketterien projektihallintamenetelmien välillä on turha.

Kisterin luoman mallin kolme iteraatiota on esitetty alla olevassa kuviossa (Kuvio 2). Keltaisella pohjalla on ensimmäinen iteratiivinen vaihe palautteesta (*Feedback*), joka kuuluu iteratiiviseen vaiheeseen *Progressive Elaboration,* joka on merkitty harmaalle pohjalle. Kolmas iteratiivinen vaihe on sinisellä pohjalla merkitty validoinnin vaihe (*Validation).* Kister on lisännyt malliin uusiksi vaiheiksi synteesi (*Synthesis*), valmistelu (*Preparation*) ja toimitusvaiheen (*Delivery*).

Edellä mainittujen vaiheiden lisäksi Kister (2016) erittelee mallissaan kolme iteratiivista kehitysvaihetta, jotka sisältävät edellä mainittuja projektin vaiheita. Ensimmäinen iteratiivinen vaihe on Kisterin (2016, s. 200) mukaan *Progressive Elaboration*, joka kulkee jatkuvana prosessina läpi projektin ensimmäisten vaiheiden. Projektinhallinnassa progressiivisen työstäminen on menetelmä yleisiin tilanteisiin, jossa projektin alussa aikataulu, budjetti tai vaatimukset eivät ole täysin selvillä, vaan ne täydentyvät projektin aikana (Project Management Institute, 2021, s. 120). Kisterin (2016, s. 200) mukaan tämä iteraatiovaihe on ideaalitilanteessa tullut päätökseensä ennen toteuttamisvaihetta.

Toinen Kisterin (2016, s. 200) mainitsema iteratiivinen vaihe on palautevaihe (*Feedback*), jolla viitataan yhteistoiminnalliseen prosessiin, jonka tavoitteena on jatkuva parantaminen. Tämä iteratiivinen prosessi on toiminnassa koko projektin ajan ja erityisesti vaiheesta toiseen siirtyessä. Kisterin (2016, s. 200) mukaan palautetta voidaan kerätä esimerkiksi kokouksissa, epävirallisten keskusteluiden yhteydessä, verkkokeskusteluforumeilta sekä käytettävyystestin yhteydessä. Kister (2016, s. 200) huomauttaa, että palautevaihe on tärkeä elementti myös edellä mainitussa progressiivisen työstämisen vaiheessa.

Kolmas Kisterin (2016, s. 200) esittelemä iteratiivinen kehitysvaihe on validointivaihe, joka sisältää katselmointiin ja parantamiseen liittyvät vaiheet. Näissä vaiheissa on luotu tärkeitä laadunvarmistamiseen liittyviä työkaluja, kuten standardeja ja ohjeistuksia. Validoinnin vaiheen tarkoituksena on myös mitata näiden työkalujen ja prosessien tehokkuutta ja yhteyttä laadukkaaseen lopputuotokseen.



Kuvio 3. Kisterin (2016) kehitysmalli.

Iteratiivisten vaiheiden lisäksi Kisterin (2016) kehitysmalliin kuuluvat lisäksi seuraavat peräkkäiset vaiheet:

Ensimmäinen vaihe on **taustatutkimus** (*Research*). Ensimmäisen vaiheen tarkoituksena on kerätä projektille olennaista tietoa eri lähteistä organisaation sisältä. Lähteet voivat esimerkiksi koostua haastatteluista aihesisältöön erikoistuneen asiantuntijan ja asiakkaiden kanssa, sekä samankaltaisten tuotteiden käytettävyystestien tuloksista. Markkinoinnista ja asiakastuesta saattaa usein löytyä myös tarpeellista taustatietoa. Tämän vaiheen lopputuloksena syntyy esimerkiksi listoja, jossa eritelty rakenteisen dokumentaation informaatiokategoriat, tietovarastoja, jotka sisältävät tarpeellisia visuaalisia elementtejä (kuvia, diagrammeja) ja muita tarpeellisia resursseja. Kister (2016, s.201).

Toinen vaihe on **analyysi** (*Analysis*). Analyysivaiheessa järjestetään, luokitellaan ja arvioidaan taustatutkimusvaiheessa kerättyä tietoa. Tässä vaiheessa tietoa arvioidaan yksityiskohtaisella tasolla, jotta voidaan varmistua sen tarkkuudesta. Analyysivaiheessa puutteellista tietoa voidaan vahvistaa ja epäolennaiset tiedot jätetään pois. Vaiheen tuotoksena voi olla esimerkiksi rakenteisen dokumentaation tuottamiseen liittyvät informaatioanalyysi, jossa on kategorisoitu tietoa osiin (*topic)* sekä analysoitu osioiden uudelleenkäytettävyyttä (*reuse*). Vaiheen oheistuotteina voi syntyä esimerkiksi uudelleen arvioitu lista informaatiokategorioista, käyttäjän työnkulkua esittäviä työnkulkukaavioita ja projektin seurantaan liittyvä raportti. Kister (2016, s. 202).

Kolmas vaihe on **synteesi** (*Synthesis*). Synteesivaihe keskittyy yhdistelemään analyysivaiheessa koottuja pienempiä elementtejä yhtenäiseksi kokonaisuudeksi. Kisterin mukaan vaihe liittyy *design thinking* -ajatteluun, joka perustuu mm. käyttäjäkeskeiseen suunnittelumalliin pohjautuen ihmisen ja tietokoneen välisen vuorovaikutuksen (*HCI*) tutkimukseen ja ohjelmistokehitykseen. Synteesivaiheessa voidaan esimerkiksi tutustua uusiin tapoihin tuottaa ja julkaista informaatiotuotteita ja hankkia kilpailijatietoa kilpailevien yritysten käyttämistä metodeista. Vaiheessa voidaan pohtia esimerkiksi dokumentin käytettävyyteen ja käyttäjäkokemukseen liittyviä kysymyksiä. Vaiheen oheistuotteena voi syntyä esimerkiksi raportti ja suunnitelma suositellusta toteutuksesta. Kister (2016, s.202).

Neljäs vaihe on **suunnittelu** (*Planning*). Suunnitteluvaiheella viitataan prosessiin, jossa määritetään projektin tavoitteet ja projektin menestystä mittaavat kriteerit, luodaan rakenne lopputuotteelle ja suunnitellaan seuraavat vaiheet, jotka ovat täytäntöönpano, arviointi, parantaminen ja toimitus. Kisterin mukaan lopputuotteen rakennetta voidaan käyttää työnositusuunnitelman (*Work Breakdown Structure*) luomisessa, joka esittää projektin aikataulun ja osittaiset määräajat. Esimerkiksi käyttöohjedokumenttia tuottaessa sisällysluetteloa voidaan käyttää projektin rakenteena ja sisällysluettelon sisältämiä pienempiä elementtejä (*topic*) voidaan käyttää työnositussuunnitelman osina

eli tehtävinä. Suunnitteluvaiheen oheistuotteena voi syntyä työnositussuunnitelman lisäksi projektin aikataulu, hahmotelma dokumentin sisältämistä topic-elementeistä ja mahdollinen varasuunnitelma, mikäli alkuperäisiä määräaikoja ei pystytä noudattamaan. Kister (2016, s. 203-204).

Viides vaihe on **valmistelu** (*Preparation*). Valmisteluvaiheessa luodaan dokumentaatioprojekteille tyypillisiä työvälineitä, kuten tyylioppaita, dokumenttipohjia (*template*) ja suositeltavaan terminologiaan liittyviä listoja. Kisterin mukaan valmisteluvaihe on kriittinen projektin laadunvarmistamisen näkökulmasta, sillä yhdessä luotujen standardien ja ohjeistuksien avulla voidaan tuottaa yhtenäisempiä dokumentteja. Hyvin tehty valmisteluvaihe myös säästää aikaa myöhemmissä vaiheissa. Kister käyttää esimerkkiä ohjelmistotuotannosta, jossa huolellisesti laadittujen vikaviestien tietovarannon luominen valmisteluvaiheessa säästää aikaa ohjelmistokehittäjiltä ja testaajilta, kun heidän ei tarvitse myöhemmissä vaiheissa käyttää aikaa vikaviestien kirjoittamiseen, muotoiluun ja arviointiin. Kister huomauttaa, että laadukkaalla projektin valmistelulla voidaan vaikuttaa positiivisesti myös työntekijän motivaatioon ja siten vähentää myös henkilöstön vaihtuvuutta (*talent retention*). Vaiheen oheistuotteena voi syntyä laadunvarmistamisen työkaluja kuten yleisiä ohjeistuksia, tyylioppaita, terminologiaan liittyviä ohjeistuksia sekä katselmointiin liittyviä ohjeistuksia. Kister (2016, s. 203-204).

Kuudes vaihe on **toteuttaminen** (*Implementation*). Toteuttamisvaiheessa luodaan toimitettavat dokumentit. Valmisteluvaiheessa luodut työkalut nopeuttavat täytäntöönpanovaiheessa tuotantoa. Vaiheen oheistuotteena syntyy lopullinen dokumentti tai sen elementtejä sekä mahdollisesti edellisessä vaiheessa luotujen ohjeistusten ja standardien paranneltuja versioita. Kister (2016, s. 204).

Seitsemäs vaihe on **katselmointi** (*Review*). Katselmointivaihe sisältää prosesseja liittyen testaukseen, validointiin ja laadunvarmistamiseen. Tässä vaiheessa käytetään valmisteluvaiheessa luotuja ohjeistuksia, työkaluja ja standardeja.

Katselmointivaiheessa dokumentteja verrataan ohjeistuksiin ja standardeihin. Vaiheeseen saattaa sisältyä myös käytettävyystestausta. Vaiheen oheistuotteina saattaa syntyä katselmointiin liittyviä kommentteja, suosituksia valmisteluvaiheen työkalujen parantamiseen, paranneltuja dokumentteja sekä erilaisia metriikoita, joilla mitataan sitä, miten hyvin dokumentit vastasivat ohjeistuksia ja standardeja. Kister (2016, s. 204-205).

Kahdeksas vaihe on **parantaminen** (*Refinement*). Parantamisvaihe on myös riippuvainen edellisissä vaiheissa luoduista laadunvarmistamisen työkaluista ja katselmointiprosesseista, joiden tulee tarjota riittävä pohja parantamisen vaiheeseen. Tässä vaiheessa korjataan puutteita ja virheitä dokumentista ohjeistuksien mukaisesti ja tehdään ehdotuksia valmisteluvaiheessa luotujen työkalujen parantamiseen sekä luodaan suosituksia katselmointi- ja parantamisprosesseihin. Tämän vaiheen tuotoksena on viimeinen versio projektin kohteena olevasta teknisestä dokumentista. Kister (2016, s. 205).

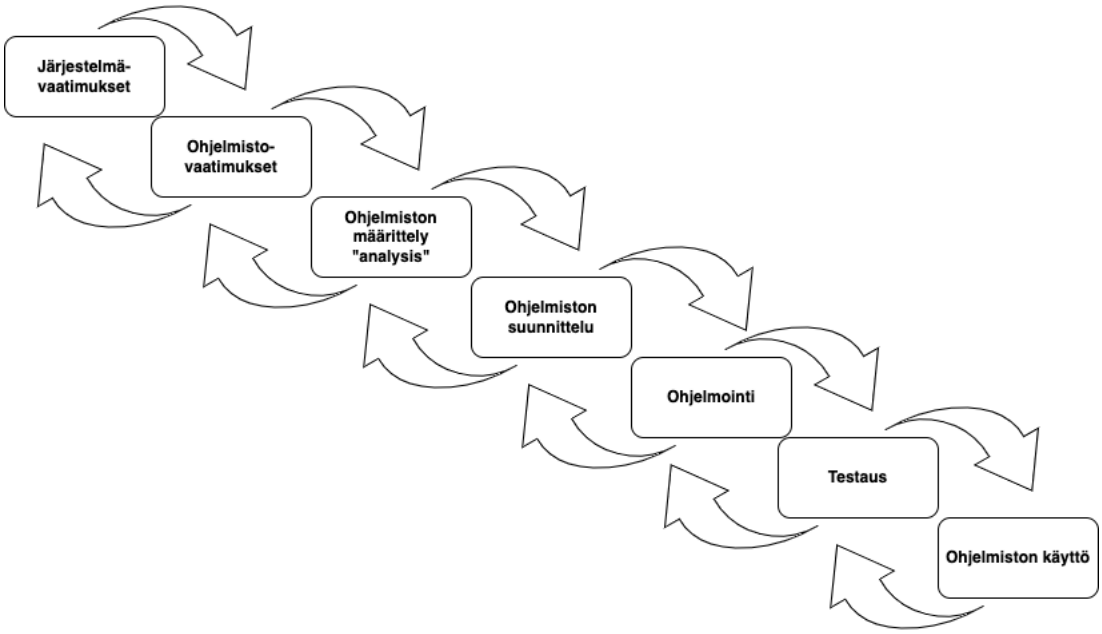
Yhdeksäs vaihe on **toimitus** (*Delivery*). Toimitusvaihe on projektin päätösvaihe, joka auttaa selkeästi viestimään projektin päättämisestä. Toimitusvaiheessa voidaan määritellä projektin päättämiseen liittyviä vaiheita ja toimittaa viimeinen versio dokumentista. Kister huomauttaa, että projektinhallinnan näkökulmasta projektin päättäminen on tärkeä osa projektin prosessia, jossa voidaan käsitellä projektin aikana tulleita oppeja tulevaisuuden varalle, kohdistaa resursseja uudelleen, hoitaa virallisia projektin hyväksyntään liittyviä dokumentteja ja maksuliikennettä. Kisterin mukaan toimitusvaiheessa on vielä mahdollisuus iteratiiviseen parantamisen prosessiin, mutta huomauttaa myös tämän olevan tarpeetonta, mikäli aikaisemmat vaiheet on suoritettu menestyksekkäästi. Toimitusvaiheessa toimitetaan viimeinen versio dokumentista ja toteutetaan projektin päättämiseen liittyviä toimenpiteitä. Kister (2016, s. 205).

3 Ohjelmistotuotantoalan käytänteet

Tässä luvussa käsittelen ohjelmistotuotantoalan käytänteitä projektinhallintaan ja niiden vaikutusta tiedon tuottamiseen, joka on teknisen viestijän ydintehtäviä. Ensimmäiseksi esittelen perinteisen vesiputousmallin ja ketterät menetelmät, jotka ovat käytössä erityisesti ohjelmistotuotantoyrityksissä. Esittelen Highsmithin (2009, alkuperäinen malli vuodelta 2000) ketterän kehityksen projektihallintamallin, josta teknisen dokumentaation tuottamiseen suunnatut edellisessä luvussa esittämäni Hackosin (2006) ja Kisterin (2016) mallit ovat saaneet vaikutteita. Tämän jälkeen pohdin *Docs as code* -menetelmää dokumentaation tuottamisen välineenä. Lisäksi esittelen kansainvälisen standardin ketterän kehityksen suhteesta informaation kehittämiseen (*information development*), jota on selvennetty ISO/IEC/IEEE standardissa (26515-2018).

3.1 Vesiputousmalli

Ensimmäinen julkaistu malli ohjelmistokehitysprosessista oli laajojen sotilasjärjestelmien kehittämiseen tarkoitettu malli, jossa ohjelmistokehitys esitetään vaiheittaisena prosessina (Sommerville, 2016, s. 48). Sommervillen (2016, s. 47) mukaan vesiputousmalli on suunnittelupohjainen malli, jossa tavoitteena on aikatauluttaa ja suunnitella prosessin eri vaiheet ennen varsinaisen työn aloittamista. Mallin mukaisesti yksittäisen vaiheen jälkeen prosessissa siirrytään toiseen vaiheeseen, mutta ohjelmistoja kehittäessä vaiheissa on päällekkäisyyksiä ja vaiheet voivat tuottaa sellaista lisätietoa seuraaville vaiheille, että niiden peräkkäinen tekeminen ei ole mahdollista (Sommerville, 2016, s. 48).



Kuvio 4. Vesiputousmalli.

Kuviossa 4 esitetään alun perin Roycen vuonna 1970 julkaisema vesiputousmalli (Royce, 1970, s. 330). Malli etenee vaiheittain vaiheesta toiseen. Ensimmäisissä vaiheissa selvitetään järjestelmän ja ohjelmiston vaatimukset, jonka jälkeen siirrytään määrittelemään ohjelmisto. Ohjelmistotuote suunnitellaan, ohjelmoidaan ja testataan, jonka jälkeen siirrytään valmiin ohjelmiston käyttöön (*operations*). Roycen (1970, s. 328) mukaan vesiputousmallin tärkeänä ominaisuutena on vaiheiden iterointi taaksepäin edelliseen vaiheeseen, mutta taaksepäin iterointi harvemmin ulottuu edellistä vaihetta edeltävään vaiheeseen. Haikalan ja Mikkosen (2011, s. 37) näkemyksen mukaan uudemmat projektimallit perustuvat vesiputousmalliin ja joidenkin ketterien menetelmien, kuten Scrumin, voidaan nähdä sisältävän sarjan toistuvia lyhyitä vesiputouksia.

3.2 Ketterät menetelmät

Ketterät menetelmät ovat saaneet alkunsa ohjelmistokehitysprojektien viitekehyksenä tuoden monia etuja projektitiimille ja asiakkaille (Stare, 2013). Ketterien menetelmien

ydin on se, että projektin laajuutta ei määritellä yksityiskohtaisesti hankkeen alussa perinteiseen projektinhallintaan verrattuna, jota kutsutaan vesiputousmalliksi. Ketterät projektit toteutetaan pienissä osioissa, joita kutsutaan iteraatioiksi. Ketterät projektinhallintamenetelmät ovat joustavia ja korostavat viestinnän merkitystä tiimin operatiivisessa toiminnassa sekä käyttäjätiedon merkitystä (Stare, 2013).

Ketterien toimintatapojen periaatteet on esitetty niin sanotussa ketterien menetelmien julistuksessa, *Agile Manifestossa*, vuonna 2001. Yksi julistuksen mukainen ketterien menetelmien periaate arvottaa toimivan ohjelmiston kattavan dokumentaation edelle (Agile Manifesto, 2001). Stare (2013) kuitenkin huomauttaa, että ketterät menetelmät ovat kehittyneet merkittävästi julistuksen julkaisun jälkeen. Uikeyn ja muiden (2011) mukaan dokumentaatiolla on suuri merkitys ohjelmistoprojekteissa, sillä sen avulla voidaan nopeuttaa tuotekehitysprosessia ja parantaa projektin sisäistä ja ulkoista viestintää sidosryhmien suuntaan.

Ketterän kehityksen projektihallintamenetelmät syntyivät 1900-luvun loppupuolella tarjoten vaihtoehdon perinteiselle Stage-Gate -projektihallintamallille (Gustavsson & Hallin, 2014, s. 571). Stage-Gate -mallilla tarkoitetaan projektinhallinnan tapaa, jossa projekti jaetaan erilaisiin peräkkäisiin vaiheisiin, joiden välissä on ”portteja”, jotka merkitsevät päätöskohtia, joissa tehdään päätöksiä projektin tulevaisuutta koskien (Gustavsson & Hallin, 2014, s. 572). Gustavssonin ja Hallinin (2014, s. 572) mukaan Stage-Gate -mallin mukaiset perinteiset projektinhallintamallit korostavat erityisesti projektien järjestelmällisyyttä ja hallintaa (*control*).

Ketteriä projektinhallintamenetelmiä alettiin käyttämään erityisesti ohjelmistotuotantoalalla, sillä perinteiset projektihallintamenetelmät tuntuivat sopimattomilta jatkuvasti muuttuvassa liiketoimintaympäristössä (Cohen ja muut, 2004). Ketteriä menetelmiä (*agile methods*) alettiin kutsua tällä termillä vuonna 2001, jolloin joukko yhdysvaltalaisia ohjelmistotuotannon menetelmien asiantuntijoita julkaisivat ketterien menetelmien julistuksen, *Agile Manifeston* (Gustavsson & Hallin, 2014, s. 572).

Ketterän ohjelmistokehityksen julistus sisältää yhteensä neljä arvoa ja kaksitoista ketterien menetelmien periaatetta (Agile Manifesto, 2001). Ketterän ohjelmistokehityksen julistuksen (Agile Manifesto, 2001) mukaan menetelmä arvostaa:

> *”Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja.*
> *Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota.*
> *Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja.*
> *Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa.”*

Ketterän ohjelmistokehityksen suomenkielisen käännöksen on kirjoittanut Lasse Koskela Agile Finland yhteisön avustuksella (Agile Manifesto, 2001). Ketterän ohjelmistokehityksen julistus korostaa ihmisten ja yhteistyön merkitystä projektinhallinnassa sekä ketteriä toimintatapoja, jotta voidaan paremmin vastata muuttuvaan liiketoimintaympäristöön kattavan suunnitelmallisuuden sijasta.

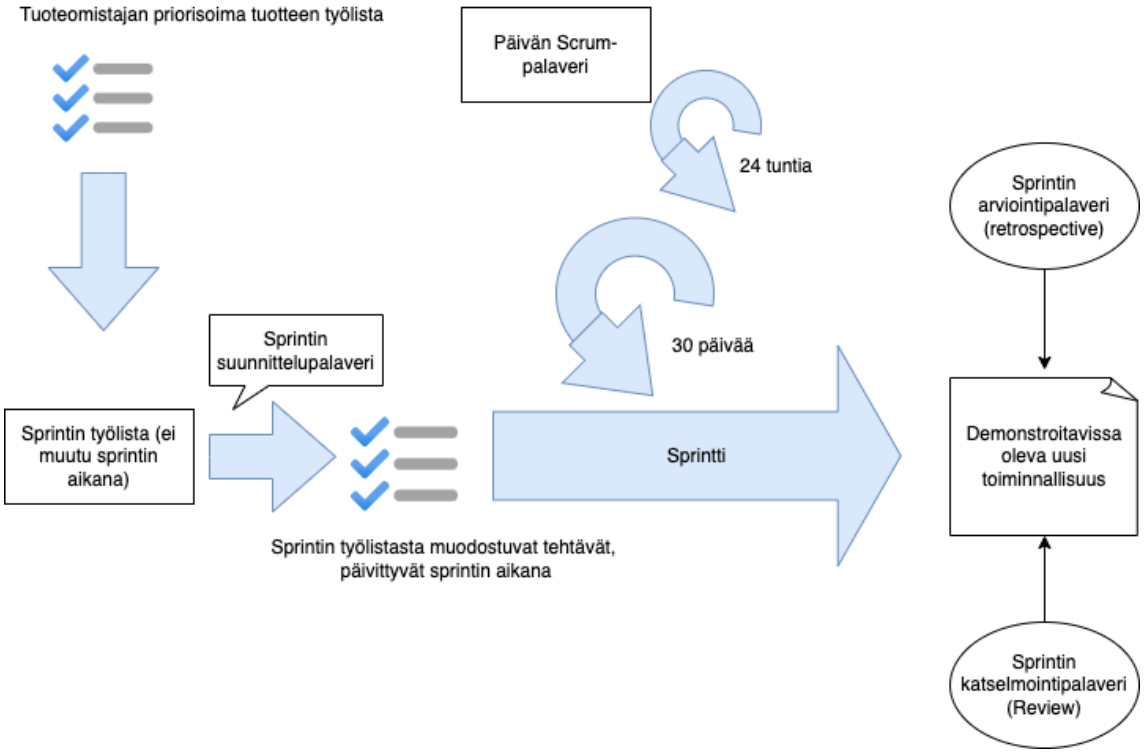
3.2.1 Scrum

Scrum on yksi yleisimmin käytössä olevista ketteristä menetelmistä. Haikalan ja Mikkosen (2011, s. 46) mukaan Scrum on terminä niin yleinen, että sitä pidetään lähes ketterän menetelmän synonyymina. Scrumin keskeiset periaatteet on esitelty jo vuonna 1986 Harward Business Review -lehdessä (Haikala ja Mikkonen, 2011, s. 46). Haikalan ja Mikkosen (2011, s. 47) mukaan Scrum ei ole projektinhallintamenetelmä, vaan tarjoaa tavan organisoida projektin kehityssyklit eli iteraatiot.

Scrum määrittelee kolme roolia, jotka ovat tuoteomistaja (*Product Owner*), Scrum-mestari (*Scrum Master)* ja ohjelmistokehittäjistä, testaajista ja käyttöliittymäsuunnittelijoista muodostuva tiimi (Haikala ja Mikkonen, 2011, s. 48). Haikalan ja Mikkosen (2011, s. 48) mukaan tuoteomistajan rooli vastaa perinteistä tuotepäällikön roolia ja hän on vastuussa projektin taloudellisesta tuloksesta. Tuoteomistaja on myös yhteydessä kaikkiin projektin sidosryhmiin vaatimusten keräämiseksi ja koostaa vaatimukset ylläpitämäänsä tuotteen työlistaan (*product*

backlog) (Haikala ja Mikkonen, 2011, s. 48). Scrum-mestari on vastuussa projektitiimin päivittäisestä operatiivisesta toiminnasta. Haikalan ja Mikkosen (2011, s.49) mukaan Scrum-mestari toimii tiimin projektipäällikkönä ilman valtaa. Scrum-mestarin vastuulla on olla tiimin ja tuoteomistajan mentorina, ja varmistaa, että Scrumin prosessia noudatetaan ja poistaa mahdollisia työn esteitä (*blockers*) (Haikala ja Mikkonen, 2011, s. 49). Haikala ja Mikkonen (2011, s. 52) mainitsevat myös, että Scrum-mestarin vastuuseen kuuluu myös pitää huolta siitä, että ohjelmistokoodin kirjoittamisen lisäksi myös testaus ja tarvittava dokumentaatio on laadittu huolellisesti.

Haikalan ja Mikkonen (2011, s. 49) määrittelevät Scrum-prosessin mukaisen tiimin itseohjautuvaksi noin seitsemästä henkilöstä koostuvaksi ryhmäksi, joka sisältää eri ammattikuntien edustajia (*cross-functional team*), esimerkiksi ohjelmoijia, testaajia tai käyttöliittymäspesialisteja. Teknisiä viestijöitä ei ole tässä yhteydessä mainittu. Haikalan ja Mikkosen (2011, s. 49) määrittelevät tiimin vastuulle työlistan paloittelun tehtäviksi ja tehtävätaulun ylläpitämisen yhdessä Scrum-mestarin kanssa. Tehtävätaulussa on näkyvissä kaikkien tehtävät ja niiden tilat. Monissa ohjelmistoalan yrityksissä on käytössä tehtävien seurantaan kaupallinen Atlassian Softwaren ohjelmisto Jira, jossa ylläpidetään myös tiimien tehtävätauluja (*Scrum board*). Atlassian on ohjelmistokehitysalalla merkittävä järjestelmätoimittaja, jonka tuotteita ovat työnhallintaan käytettävän Jiran lisäksi dokumentaation hallintaan suunnattu Confluence. Haikalan ja Mikkosen (2011, s. 58) mukaan Jira-ohjelmistolla on laaja käyttäjäkunta. Tiimi osallistuu Scrum-mestarin pitämiin päivittäisiin Scrum-tapaamisiin (Haikala ja Mikkonen, 2011, s. 49). Päivittäisiä tapaamisia kutsutaan ketterissä menetelmissä yleisesti myös termeillä *daily* tai *daily standup*.



Kuvio 5. Scrum-prosessi.

Kuviossa 5 on esitetty Scrum-toimintamallin mukainen prosessi (Haikala ja Mikkonen, 2011, s. 48). Kuviossa on näkyvillä Scrumin mukaiset seremoniat eli tapaamiset sprintin eri vaiheissa, jotka tähtäävät avoimeen tiedon jakamiseen ja kommunikaatioon. Sprintin alussa on suunnittelupalaveri ja työn edistymistä seurataan päivittäisissä Scrum-tapaamisissa ja sprintin lopuksi katselmoidaan sprintin aikana valmistuneet tuotokset ja arvioidaan sprintin onnistumista arviointipalaverissa (*retrospective*).

Tuotteen työlistasta (*product backlog*) generoidaan jokaiselle sprintille oma työlistansa, joka käydään läpi sprintin suunnittelupalaverissa. Sprintin työlistalta muodostuu tehtäviä, jotka päivittyvät sprintin aikana. Sprintin kesto on yleensä 2–4 viikkoa, jonka aikana päivittäin pidetään Scrum-tapaamiset, joiden aikana jokainen tiimin jäsen kertoo, mitä on tehnyt viimeisen vuorokauden aikana ja onko työssä mahdollisesti esteitä (*blockers*), joiden poistamiseen tarvitaan muiden tiimin jäsenten apua. Sprintin lopputuloksena on esiteltävissä oleva toiminnallisuus. Sprintin jälkeen pidettäviä seremonioita ovat retrospektiivi, jossa käydään läpi asiat, joissa onnistuttiin ja joita voisi vielä parantaa.

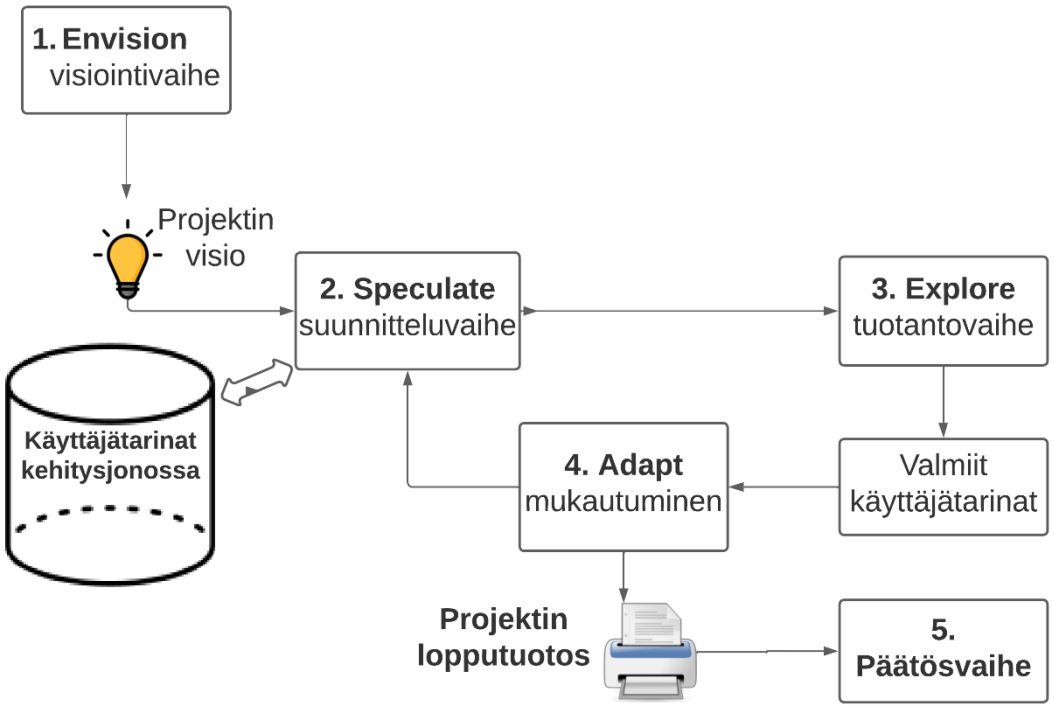
Sprintin katselmointipalaverissa esitellään vielä sprintissä saavutetut tuotokset. Haikalan ja Mikkosen (2011, s. 51) katselmointipalaveriin osallistuu tiimin jäsenten lisäksi Scrum-mestari, tuoteomistaja ja mahdollisesti sidosryhmien edustajia eli asiakkaita.

3.2.2 Highsmithin ketterän kehityksen projektinhallintamalli

Tässä luvussa esittelen ketterän kehityksen viitekehykseen luodun Highsmithin (2000) projektihallinnan mallin. Hackos (2006, luku 3) on maininnut Highsmithin viisiportaisen mallin toimineen referenssinä luomalleen kehitysmallille. Hackosin malli on esitetty luvussa 2.3.1.

Highsmith (2009) on kehittänyt ketterään menetelmään pohjautuvan viisiportaisen projektinhallintamallin, joka pohjautuu hänen vuonna 2000 julkaisemaan malliin, joka on julkaistu teoksessa ”*Adaptive Software Development*”. Highsmithin (2009, luku 5) mukaan ketterässä viitekehyksessä prosesseja ei pidetä yhtä tärkeänä, kuin ihmisten välistä yhteistyötä ja viestintää. Highsmith (2009, luku 5) kuitenkin huomauttaa, että prosessin tulee olla sidottu organisaation liiketoimintatavoitteisiin. Tämän lisäksi Highsmith korostaa, että ketterän projektinhallinnan viitekehyksen tulee tukea *Envision, Explore, Adapt* -kulttuuria. Tämä tarkoittaa Highsmithin (2009, luku 5) mukaan sitä, että mallin tulee tukea itseohjautuvaa tiimiä, olla joustava ja helposti mukautua, tukea prosessien läpinäkyvyyttä ja oppimista sekä sisältää menetelmiä, jotka tukevat projektin eri vaiheita. Projektin viitekehyksen tulisi myös määrittää johdon tarpeisiin suunnattuja tarkastuspisteitä.

Highsmith (2009, luku 5) on luonut ketterän projektinhallinnan mallin, jonka rakenne keskittyy toimitukseen ja projektin mukautumiseen (*adaptation*). Malli on esitetty alla olevassa kuvioissa (Kuvio 6).



Kuvio 6. Highsmithin ketterän kehityksen projektinhallintamalli.

Kuvio 6 esittää Highsmithin (2009, luku 5) viisiportaisen mallin. Kuvio on mukailtu ja käännetty alkuperäisen mallin (*APM Delivery Framework*) pohjalta. Malli sisältää seuraavat vaiheet:

Ensimmäinen vaihe on **visiointi** (*Envision*). Highsmithin (2009, luku 5) mukaan projektin visiointivaihe on kriittinen menestystekijä koko projektin onnistumisen kannalta. Tässä vaiheessa tiimin jäsenien tulee visioida toimitettava tuote ja projektin laajuus, projektin kanssa tekemisissä olevat henkilöt, jotka voivat koostua asiakkaista, tuotepäälliköistä, tiimin jäsenistä sekä muista sidosryhmistä. Tässä vaiheessa myös suunnitellaan miten yhteistyö tullaan toteuttamaan projektin aikana. Highsmith (2009, luku 5).

Toinen vaihe on **suunnittelu** *(Speculate)*. Highsmithin mukaan tässä vaiheessa projektia joudutaan suunnittelemaan perustamalla suunnitelma keskeneräiseen tai osittaiseen tietoon. Highsmithin mukaan *speculate*-termillä viitataan juuri tähän prosessiin, jossa joudutaan tekemään oletuksia vaillinaisen tiedon perusteella. Highsmithin mukaan

suunnitteluvaihe on tärkeä, mutta *speculate*-nimityksellä voidaan realistisemmin kuvata suunnittelua ketterissä projekteissa, joissa on epävarmuutta. Vaiheessa voidaan luoda projektin tavoite ja suunta, mutta jätetään paljon tilaa muutokselle. Tässä vaiheessa kerätään alkuvaiheen vaatimukset tuotteelle, määritetään projektin työmäärä kehitysjonoon (*backlog)*, joka koostuu tuotteen toiminnallisuuksista (*feature*). Vaiheessa luodaan myös iteratiivinen tuotteen toiminnallisuuksiin perustuva julkaisusuunnitelma (*release plan*) ja suunnitellaan strategioita riskinhallinnan varalle. Vaiheessa arvioidaan myös projektin kustannukset ja tuotetaan muuta hallinnointiin liittyvää tietoa. Highsmith (2009, luku 5).

Kolmas vaihe on **tuotanto** *(Explore).* Highsmith (2009, luku 5) kutsuu projektin varsinaista tuotantovaihetta termillä *Explore*. Tässä vaiheessa tuotetaan projektin tuotteen käyttäjätarinat (*user story*), jotka ovat kehitysjonon käyttäjän näkökulmasta luodut tuotteen toiminnallisuudet. Tuotannon lisäksi vaiheessa luodaan yhteistoiminnallinen ja itseohjautuva projektin sisäinen yhteisö. Vaiheessa hallinnoidaan myös viestintää asiakkaiden, tuotepäälliköiden ja muiden sidosryhmien välillä. Highsmith (2009, luku 5).

Neljäs vaihe on **mukautuminen** (*Adapt).* Mukautumisvaiheessa projektin tuloksia katselmoidaan teknisestä, asiakkaiden, ihmisten, prosessien ja projektin statuksen näkökulmasta. Highsmithin (2009, luku 5) mukaan mukautumisvaihe vastaa ketterän kehityksen julistuksen arvoon, jossa arvotetaan muutokseen vastaamista enemmän kuin suunnitelmassa pitäytymistä. Mukautumisvaiheen analyysin tulokset sisällytetään seuraavan tuotekehitysiteraation suunnitteluvaiheeseen. Visiointivaiheen jälkeen Highsmithin malli seuraa silmukan tavoin suunnittelun, tuotannon ja mukautumisen vaiheita, jolloin jokainen iteraatio tähtää tuotteen parantamiseen. Highsmith (2009, luku 5).

Viimeinen vaihe on **projektin päätösvaihe** (*Close).* Päätösvaihe merkitsee projektin päätöstä. Highsmithin (2009, luku 5) mukaan tämän vaiheen tärkein tavoite on projektin

oppien kerääminen ja niiden sisällyttäminen seuraavassa iteraatiovaiheessa tai projektin oppien (*lessons learned*) viestintä seuraavan projektitiimin käyttöön.

3.3 Dokumentaation kehittäminen ketterässä ympäristössä

Tässä luvussa selvennän kansainvälisen standardin mukaisesti ketterien projektihallintamenetelmien suhdetta informaation kehittämiseen. Kansainvälisen standardin (ISO/IEC/IEE 26515:2018) mukainen määritelmä ketterälle ympäristölle (*agile environment*) on, että ketterässä ympäristössä on käytössä ketterää kehitystä tukeva organisaatiokulttuuri, infrastruktuuri ja metodologiat. Standardin (ISO/IEC/IEE 26515:2018) mukaan ketterässä ympäristössä on tärkeää tuottaa informaatiotuotteita käyttäen samoja ohjelmistotuotannon prosesseja. Teknisen dokumentaation sisällöntuotanto tulisi toteuttaa rinnakkain ohjelmistotuotteiden kanssa noudattaen ohjelmistotuoteprojektien elinkaarta. Tämä rinnakkainen toimintamalli tarjoaa informaatiotuotteiden kehittäjälle useita hyötyjä, joita esittelen alla olevassa kappaleessa.

3.3.1 Informaatiotuotteiden toimittaminen jatkuvan julkaisun prosessilla

Standardin (ISO/IEC/IEE 26515:2018) mukaan ohjelmistotuotteiden kehittämiseksi usein käytetään jatkuvan integraation ja julkaisun prosesseja, jotta ohjelmiston sisältämiä uusia toiminnallisuuksia saadaan julkaistua käyttäjille ja asiakkaille mahdollisimman nopeasti. Jatkuvan julkaisun prosessia kuvataan usein osana *DevOps* toimintatapaa (*Development Operations*), jossa ohjelmistokehittäjien tarpeisiin on luotu oma automatisoitu IT-infrastruktuuri. Tällä tavalla julkaisuaikatauluun liittyvät päätökset voidaan tehdä kehitystiimin sisällä eivätkä ne ole riippuvaisia ulkoisen IT-tiimin toiminnasta (ISO/IEC/IEE 26515:2018). *DevOps* termillä tarkoitetaan työtapoja, työkaluja ja työkulttuurin muutosta, jossa ohjelmistokehittäjän ja IT-tiimi toimivat yhteistyössä (Atlassian, 2023a). Standardin mukaan (ISO/IEC/IEE 26515:2018)

informaatiotuotteiden julkaisun kannalta tämä toimintamalli tarkoittaa sitä, että käyttäjille suunnattua dokumentaatiota tulisi luoda, katselmoida, editoida ja julkaista nopealla syklillä. Käyttäjien kannalta tällä toimintamallilla saadaan aikaan nopeampia päivityksiä.

3.3.2 *Docs as code* -menetelmä dokumentaation tuottamisen välineenä

Docs as code -menelmää voidaan pitää yhtenä esimerkkinä ohjelmistotuotannon menetelmien ja työkalujen hyödyntämisestä dokumentaation tuottamisen välineenä. Menetelmä hyödyntää edellä mainittua DevOps toimintatapaa dokumentaation jatkuvaan julkaisuun. *Docs as code* -menetelmästä on kirjoitettu erityisesti ammatinharjoittajien näkökulmasta (Etter, 2016; Gentle, 2017 ja Jitianu, 2021). Menetelmällä tarkoitetaan dokumentaation kehittämistapaa, jonka mukaan dokumentaation kirjoittamiseen käytetään samoja työkaluja ja työnkulkua kuin ohjelmistokoodin tuottamiseen (Jitianu, 2021, s. 22). Clearyn (2021, s. 108) kutsuu *Docs as code* -menetelmää liikkeeksi (*movement*), joka rohkaisee teknisiä viestijöitä kohtelemaan dokumentaatiota samalla tavalla kuin ohjelmistokoodia.

Dokumentaation tuottamiseen tulisi menetelmän mukaan hyödyntää esimerkiksi ohjelmistokoodin versionhallintaan tarkoitettuja työkaluja kuten GitHubia, GitLabia tai Bitbucketia. Versionhallinnan hyötynä dokumentaation tuottamisen näkökulmasta on tallentaa dokumentaatioon tehdyt muutokset ja antaa helppo pääsy dokumentin aiempiin versioihin. Versionhallinnan avulla on mahdollista julkaista dokumentaatioversiot esimerkiksi tuotteen julkaisuaikataulun mukaisesti. Versionhallinnan avulla dokumentteja on myös helpompi työstää yhteistyössä ja katselmoida. (Jitianu, 2021, s. 22–23). Etterin (2016, s. 25) mukaan yksi tärkeimmistä syistä versionhallintajärjestelmien käyttöön ohjelmistotuotantoyrityksissä on se, että ohjelmistokehittäjät käyttävät niitä mielellään. Etter (2016, s. 25) huomauttaa, että sellaisenaan versionhallintajärjestelmät ovat hieman liioiteltuja perinteisessä teknisen dokumentaation työnkulussa, mutta samojen työkalujen käytöllä voidaan viestiä

dokumentaation myötävaikuttajille (*contributor*), ohjelmistokehittäjille, että tekninen viestijä on valmis käyttämään samoja nykypäivän teknologisia ratkaisuja.

Versionhallinnan lisäksi menetelmään kuuluvat ohjelmistotuotannosta tutut prosessit jatkuva integraatio (*Continuous Integration*, CI) ja jatkuva julkaisu (*Continuous Delivery*, CD). Jitianun (2021, s. 23) mukaan jatkuvalla integraatiolla tarkoitetaan dokumentaation tuottamisen näkökulmasta esimerkiksi sitä, että prosessin avulla voidaan esimerkiksi luoda ja automatisoida dokumentin laatutarkastuksia sekä automatisoida dokumentin kääntämiseen liittyviä prosesseja. Jatkuvan julkaisun tavoitteena on, että ohjelmisto on aina julkaisuvalmis ja muutokset ovat toteutettavissa nopeasti. Dokumentaation tuottamisen näkökulmasta tällä tarkoitetaan sitä, että dokumentin julkaisuprosessi on automatisoitu ja muutokset voidaan päivittää järjestelmään koska tahansa napin painalluksella (Jitianu, 2021, 23).

Clearyn (2021, s. 108) mukaan *Docs as code* on lisännyt kevyiden merkintäkielten (*lightweight markup language),* kuten AsciiDocin ja Markdownin, suosiota sisällöntuotannon välineenä. Nämä merkintäkielet ovat helppokäyttöisiä ja ne ovat syntaksiltaan helppolukuisia (Cleary, 2021, s. 108). Taulukossa 2 on esitetty esimerkki Markdown syntaksista verrattuna HTML-syntaksiin. Esimerkissä on käytetty esimerkkinä numeroitua listaa (Markdown Guide, 2023). Clearyn (2021, s. 108) tutkimuksen aineistossa eräs haastateltava mainitsi AsciiDocin olevan ohjelmistokehittäjien suosiossa myös sen vuoksi, että tiedostot voidaan säilyttää ohjelmistokoodin lähellä.

Taulukko 2. Esimerkki Markdown syntaksin mukaisesta numeroidusta listasta.

Markdown	**HTML**	**Renderöity tulos (*rendered output*)**
1. Ensimmäinen 2. Toinen 3. Kolmas	 - Ensimmäinen - Toinen - Kolmas 	1. Ensimmäinen 2. Toinen 3. Kolmas

Taulukon 2 esimerkistä voi huomata, että Markdown on syntaksiltaan yksikertainen ja helppolukuinen verrattuna esimerkiksi HTML-syntaksiin, jossa on käytössä listaelementeille erilaisia tageja, kuten <ol> ja <li>.

3.3.3 *Docs as code* -menetelmän hyödyt

Tässä kappaleessa esittelen *Docs as code* -menetelmän hyötyjä teknisen dokumentaation tuottamisen näkökulmasta. Gentlen (2017, s. 21) mukaan jatkuvan integraation prosessit vapauttavat tekniselle viestijälle aikaa sisällön suunnitteluun ja toteutukseen. Muita *Docs as code* -menetelmän hyötyjä toiminnan tehostumisen lisäksi on esimerkiksi yhteistyön lisääntyminen.

Thomchickin (2019, s. 908) mukaan menetelmästä voidaan hyötyä myös ohjelmistokehittäjille suunnatun rajapintadokumentaation tuottamisen välineenä. Rajapintadokumentaatio on yksi esimerkki dokumentaatiosta, jonka tuottamiseen tekninen viestijä osallistuu ohjelmistoalan yrityksissä. Ohjelmointirajapintoja (*Application Programming Interface, API*) käytetään esimerkiksi yhdistämään SaaS (*Software as a Service*) -ohjelmistoja internetiin (Thomchick, 2019, s. 908). Ohjelmointirajapintojen tarjoajien täytyy dokumentoida rajapinnat riittävällä tasolla, jotta ohjelmistokehittäjät saavat tarvittavan määrän informaatiota pystyäkseen yhdistämään järjestelmiä ja jakamaan dataa sekä toisaalta dokumentaatiota täytyisi pystyä myös tuottamaan tarpeeksi nopeasti ohjelmistojen nopean julkaisuaikataulun vuoksi (Thomchick, 2019, s. 908). Gentlen (2017, s. 13) mukaan SaaS-ohjelmistojen käytön lisääntyessä tarvitaan hyvälaatuista dokumentaatiota myös rajapinnoista käyttöliittymädokumentaation lisäksi.

Gentlen (2017, s. 12) huomion mukaan verkossa olevien ohjelmointirajapintojen määrä on kasvanut räjähdysmäisesti viime vuosina, jonka vuoksi teknisten viestijöiden on välillä vaikea pysyä perässä dokumentoitavien ohjelmointirajapintojen lukumäärästä. Thomchickin (2019, s.909) mukaan *Docs as code* -menetelmällä voitaisiin tiettyjä

työkaluja (esimerkiksi *Swagger UI* -sovellus*)* käyttämällä automatisoida rajapintadokumentaation luominen, joka poistaisi siihen yleisesti liittyvä ongelman sen puuttumisesta kokonaan tai dokumentaation päivittämättömyydestä. Thomchickin (2019, s. 909) mukaan *Docs as code* -menetelmää voitaisiin hyödyntää esimerkiksi integroituna osana ohjelmointiympäristöä, mutta myöntää myös, että aihe vaatisi mahdollisesti lisää jatkotutkimuksia.

4 Ohjelmistotuotannon menetelmien hyödyntäminen teknisessä viestinnässä

Tässä luvussa kuvaan tutkimuksen analyysin tuloksia. Ensimmäiseksi kuvailen analyysin etenemistä ja kappaleesta 4.2 lähtien käsittelen analyysin tuloksia ja annan esimerkkejä tutkimusaineistosta.

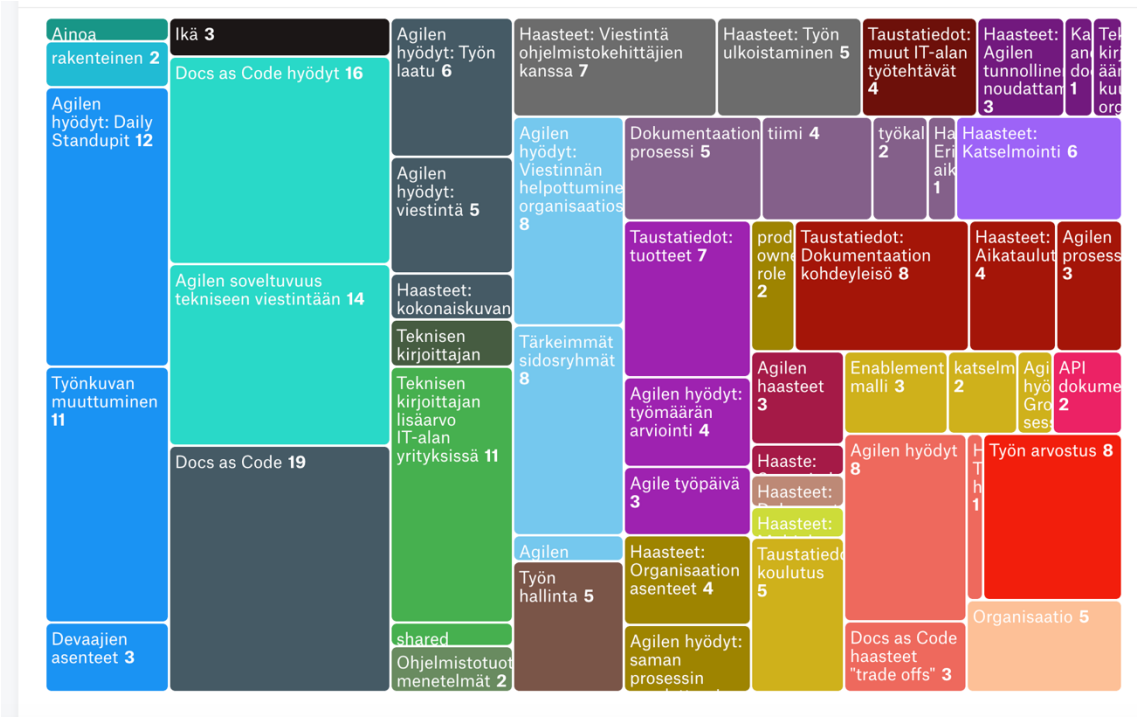
4.1 Analyysin eteneminen ja aineiston käsittely

Analyysivaiheessa tarkastelin ja erittelin litteroituja haastatteluaineistoja. Word-työkalun automaattisen litteroinnin jälkeen olen tarkistanut ja korjannut tarvittaessa automaattisen työkalun puutteita aineistosta. Yksi haastateltavista käytti runsaasti sekä suomen että englannin kieltä haastattelun aikana, joten automaattinen litterointi ei aina pystynyt huomaamaan kielen vaihtumista, joten tämänkin vuoksi oli vielä tärkeää tarkistaa automaattisen litterointityökalun tuottama aineisto. Litteroitua aineistoa oli yhteensä 228 sivua.

Litteroinnin jälkeen toin aineiston Atlas.ti -työkaluun tarkempaa analyysiä varten. Koodasin työkalun avulla aineistosta löytyneitä analyysiyksiköitä aineistolähtöisen sisällönanalyysin avulla, jota olen kuvannut tutkielman alussa luvussa 1.3. Analyysiyksiköt olivat pääasiassa samoja teemoja, joita käytin teemahaastattelun rungossa, mutta analyysin aikana aineistosta löytyi myös uusia teemoja.

Esittelen analyysin tuloksia antamalla esimerkkejä tutkimusaineistosta. Esimerkin alkuun olen merkinnyt asiantuntijan lyhenteen (H1-H5), jotta olisi läpinäkyvää kenen haastateltavan aineistosta kukin sitaatti on peräisin. Olen valinnut esimerkit siten, että ne edustavat parhaiten tutkimuksen analyysiyksiköitä, joita on näkyvillä kuviossa 6. Mikäli aineistossa oli useita samankaltaisia vastauksia johonkin analyysiyksikköön liittyen, valitsin parhaiten kuvaavan esimerkin. Joissakin esimerkeissä olen lyhentänyt esimerkkiä käyttäen hakasulkeita, jotka sisältävät kolme pistettä. Näissä tapauksissa lyhennetty

sisältö ei ole ollut analyysin kannalta olennaista. Olen myös käyttänyt esimerkeissä lihavointia niiden ymmärrettävyyden parantamiseksi, sillä osa esimerkeistä on pitkiä, mutta niiden lyhentäminen ei ollut mahdollista ilman että niistä olisi kadonnut analyysin kannalta olennaista tietoa. Lihavoitu osuus on esimerkin analyysin kannalta olennaisin.



Kuvio 7. Diagrammi Atlas.ti -työkalusta.

Kuviossa 7 näkyy aineiston luokitteluun käytetyt koodit puukartan muodossa. Diagrammissa näkyvän neliön koko on sitä suurempi mitä useimmin kyseistä koodia on käytetty aineiston analyysissä.

Aineiston teemoittelun pääteemat olivat:

- Taustatiedot
- Organisaatio ja tiimi
- Dokumentaation tuottamisen prosessi
- Työkalut
- Menetelmiin (ketterät menetelmät ja *Docs as code*) liittyvät hyödyt ja haasteet sekä niiden soveltuvuus tekniseen viestintään
- Työn laatu ja arvostus

Analyysin aikana löytyneitä uusia teemoja olivat:

- Organisaation ja ohjelmistokehittäjien asenteet teknisiä viestijöitä kohtaan
- Työnkuvan muuttuminen aikaisempaan verrattuna
- Teknisen viestijän tuottama lisäarvo IT-alan yrityksessä

4.2 Ketterien menetelmien hyödyt

Vaikka ketteriä menetelmiä ei ole suoraan suunniteltu käytettäväksi teknisen viestinnän työtehtävissä, on niiden käyttämisestä todettu olevan aineiston mukaan monia hyötyjä. Haastateltavat kokivat, että ketterien menetelmien omaksuminen on helpottanut viestintää organisaatiossa erilaisten sidosryhmien (esimerkiksi ohjelmistokehittäjien ja tuotejohdon) suuntaan, sillä samojen toimintamallien noudattaminen sitoo tekniset viestijät tiukemmin osaksi muuta tiimiä. Viestintä helpottuu, kun kaikki voivat noudattaa samaa yhteistä toimintamallia. Ketterissä menetelmissä hyödynnettävistä seremonioista, kuten päivittäisistä palavereista tai työlistan ominaisuuksien työstöön suunnatuista *Grooming*-sessioista (tunnetaan myös nimellä *backlog refinement*) on hyötyä myös teknisille viestijöille, sillä esimerkiksi päivittäisissä Scrum-palavereissa saa poistettua tehokkaasti työn esteitä. Työmäärän arvioinnin ja työtehtävien priorisoinnin läpinäkyvyyden oli myös koettu parantuneen ketterien menetelmien myötä.

4.2.1 Viestinnän helpottuminen ja seremoniat

Ketterät menetelmät sisältävät elementtejä, jotka parantavat viestintää tiimin jäsenien ja tärkeiden sidosryhmien kesken. Menetelmien noudattaminen tuo teknisen viestijän lähemmäksi tuotekehitystä.

> (1) H2: No ne kyllä siis parantaa sitä viestin kulkemista, kommunikaatiota ja tosiaan että jos sitä nyt noudatetaan säntillisesti ja niin, että se kirjoittaja siinä myöskin on vahvasti toimijana niin mun mielestä tuota oikein hyvä tapa työskennellä, että se just se tuo sen ajantasaisuuden, että kirjoittaja tietää koko aika kun on uusia toiminnallisuuksia tulossa ja sitten. Tuotetiimi osaa sitten ottaa sen kirjoittajan myös siinä huomioon, että kertoo aikatauluja ja suunnitelmia. **Siinä on mahdollista toteuttaa sitä sitten käytännössä sitä ajatusta, että dokumentaatio on osa tuotetta.**

Esimerkin (1) mukaan se, että kaikki toimijat noudattavat samaa yhteistä ketterän kehityksen tarjoamaa toimintamallia parantaa kommunikaatiota tuotejohdon ja teknisen viestijän välillä. Toimintamallit integroivat viestijän osaksi muuta ohjelmistokehitystiimiä, jolloin dokumentaation tuottaminen ei ole erillinen siilonsa, vaan sitä pidetään osana tuotetta. Teknisellä viestijällä on myös parempi mahdollisuus olla mukana tuotteen suunnitteluvaiheesta alkaen ja saada tietoa tuotteeseen tulevista toiminnallisuuksista.

(2) H3: And and also importantly so I can get things to a point where they are blocked. So then **in my next scrum I can ask for help** and say hey, this is as far as I can take this ticket. It truly is blocked now, so now you guys need to pick it up and take a look at your code again and figure out why it's not working correctly. So I think those kind of ways **have really improved our documentation that that sort of like prioritization, organization and the ability to push back and ask for more help**.

Esimerkissä (2) mainitaan ketterien menetelmien halu poistaa työn esteitä mahdollisimman tehokkaasti ja kommunikointi muiden tiimin jäsenten kesken. Haastateltavan mukaan työn esteitä voi tehokkaasti kommunikoida muille tiimin jäsenille päivittäisissä Scrum-tapaamisissa. Ketterät menetelmät auttavat parantamaan dokumentaatiotyön organisointia, priorisointia ja luoneet keinon pyytää apua muilta tiimin jäseniltä.

(3) H1: Meille suuri hyöty tuossa keskimmäisessä työpaikassa oli nimenomaan se, että se tarjosi meille tai ei oikeastaan meille, vaan **se tarjosi niille Product Ownereille selkeän toimintamallin miten toimia meidän kanssa**, meidän konfliktit loppui siihen, että me alettiin tehdä muistaakseni 2 vai 3 viikon scrummia muistaakseni. Tiesi sen jälkeen, että lead time on max 6 viikkoa, että jos meillä sprintti on käynnissä ja se on täynnä niin seuraavan 3 viikon sprinttiin se mahtuu tai me otetaan, jos on tosi urgent. Ja tuota juuri siis se että siitä tuli niinku normaalia ihan samanlaista kommunikaatiota kuin mitä ne teki deve-tiimin kanssa ja ne itseasiassa ne oli vähän ne muut Product Ownerit vähän nihkeitä ensi alkuun, mutta me kuultiin sitten myöhemmin että eihän tietenkään suoraan sitä sano, mutta kuultiin myöhemmin sitten tuota kautta rantain, että ne oli olleet hirveän tyytyväisiä tähän, että niin ne tietää miten lähestyä meitä, että mitä heidän pitää tehdä ja niinku että miten tämä homma menee eteenpäin, että se on ehdottomasti tuota **yks isoimmista eduista kirjoittajalle, että vaikka se on hankalaa, niin. Se muuttaa sut**

> **tämmöisestä uppooudosta vieraasta, jonka kanssa toimista ei oikeastaan niinku oikeesti et. Sä et ole enää mikään kummallinen alien vaan jotain tuttua.**

Esimerkissä (3) todetaan, että Scrum-toimintamallin selkein hyöty tekniselle viestijälle on se, että se tarjosi tuoteomistajille selkeän toimintamallin siihen, miten toimia teknisten viestijöiden kanssa ja viestintäongelmat tuotejohdon suuntaan loppuivat, kun teknisten viestijöiden tiimi otti Scrumin käyttöön. Tuoteomistajat tiesivät, että dokumentaatiotöiden läpimenoaika (*lead time*) olisi maksimissaan 6 viikkoa, jos pyyntö tulisi kesken sprinttiä, sillä kiireellinen dokumentaatiotyö voitiin toteuttaa mahdollisesti seuraavan sprintin aikana. Tekninen dokumentaatiotiimi noudatti samaa toimintamallia ohjelmistokehitystiimien kanssa, joten tuoteomistajat kokivat tämän helpoksi ja teki samalla tekniset viestijät paremmin osaksi muuta organisaatiota, jolloin teknisiä viestijöitä ei koettu enää vieraiksi, kun he noudattivat samaa tuttua toimintamallia muun organisaation kanssa.

Ketterien menetelmien mukaiset seremoniat, kuten esimerkiksi päivittäiset tapaamiset ovat hyvä tilaisuus tiedon jakamiseen ja tuovat vertaistukea pulmatilanteissa.

> (4) H3: My daily life is I sort of logging in. **I go to my team Scrum so my doc team has its own Scrum in the morning it's just us doc writers**. We kind of get together and it's really. It's kind of a not true agile ceremony though 'cause what we'll do is we'll talk to our manager kind of give him our status update and go around the table being like. Here's what I'm working on today. And even though the person sitting next to me probably has no relation to what I'm working on, 'cause we're each servicing different product teams and also going to their scrums. It's an interesting way for us to kind of get together. And be like, hey, I'm also having trouble getting this API to generate are you?

Esimerkissä (4) haastateltavan organisaatiossa järjestetään joka aamu teknisten viestijöiden tiimin sisäinen päivittäinen tapaaminen, jossa jaetaan tietoa työn alla olevista tehtävistä ja haetaan vertaistukea mahdollisiin ongelmiin, esimerkiksi jonkun tietyn työn alla olevan rajapinnan toimintaan liittyen. Tekniset viestijät osallistuvat myös ohjelmistokehitystiimien päivittäisiin Daily-tapaamisiin. Haastateltavan organisaatiossa

teknisille viestijöille on annettu kullekin vastuualueeksi tietyt ohjelmistokehitystiimit, joiden tuotteen dokumentaatiosta viestijä on vastuussa.

(5) H1: Todettiin, että okei tää kuulostaa järkevältä, ei uskota että toi tulee toimimaan, toi vois ehkä toimia. Sit tän teorian pohjalta lähdettiin suunnittelemaan, että mikä kuulostaa järkevältä. Todettiin, että **scrum-sessiot eli siis nää standupit on aivan loistava, koska me työskenneltiin käytännössä saman codebasen kanssa**, meillä ei ollut työkalua, joka olis lukinnut sen codebasen järjestelmänhallintakitissä niin käytännössä siinä oli se mahdollisuus, että jos me satutaan koskemaan samoihin asioihin niin me luodaan konflikti ja sit homma on reisillä.

Esimerkissä (5) haastateltava kertoo, että teknisten viestijöiden tiimissä pohdittiin Scrumiin siirtyessä tekniseen viestinnän työtehtäviin soveltuvia elementtejä ja yksi Scrumin tekniseen viestintään sopiva elementti olivat päivittäiset Scrum-tapaamiset, joiden yhteydessä voidaan jakaa tietoa viestijöiden kesken. Tiimi työskenteli saman koodipohjan kanssa, joka vaati viestintää muiden tiimin jäsenten kesken, sillä jos tekniset viestijät muokkaavat samaa tiedostoa yhtä aikaa, tämä voi luoda konflikteja, kun muutoksia sulautetaan päähaaraan. Haaroituksella (*branching*) tarkoitetaan versionhallintajärjestelmissä yleistä toimintatapaa, jossa keskitetty lähdekoodin koodikanta muodostaa päähaaran ja kehittäjä voi luoda omia haaroja päähaarasta itsenäistä työskentelyä varten (Atlassian, 2023b). Konfliktien ratkaiseminen aiheuttaa turhaa lisätyötä, jonka vuoksi töiden koordinointi tiimijäsenten kesken on tärkeää.

(6) H1: me todettiin että **me ehdottomasti halutaan grooming-sessiot, että niinku yritetään vähän ymmärtää et niinku et ku tulee joku et no tämmöinen feature**, että mitä kaikkea kenelle… kuka se nyt voisi tehdä ja mihin kaikkeen se liittyy ja just niinku et minkälaisiin palasin se pitää pilkkoa meidän kolmen tavallaan pääkomponentin [teknisen kirjoittajan] kanssa ja sitten just se että okei no jos toi on tässä sprintissä niin kuka meistä kolmesta, kenellä on niinku suuri työkuorma että ettei tyyliin tule samalle tyypille kaksi hervottoman isoa featurea samaan sprinttiin ja sitten ne on helisemässä sen kanssa kun en voi saada näitä valmiiksi. [..] Ja sitten se tosiaan tämä, että **me saatiin sitten Product Ownerin näkökulma paremmin grooming-sessiossa, mikä se iso idea tässä taustalla on mikä se, mikä se viesti on, mikä tästä halutaan.** Me saatiin ne sieltä grooming-sessiosta.

Esimerkin (6) mukaan haastateltava piti työlistan työstöön suunnattua *backlog grooming*-palaveria tärkeänä sprintin suunnittelun kannalta. Haastateltava työskenteli tiimissä, joka koostui kolmesta teknisestä viestijästä. *Grooming*-sessiot olivat tärkeitä kahdesta syystä: niissä pystyttiin arvioimaan ja tasoittamaan jokaisen tiimin jäsenen työkuormaa, jotta kunkin sprintin työtehtävät jakautuisivat tasaisemmin tiimin jäsenten kesken. Sen lisäksi tekniset viestijät saivat *grooming*-palavereista tuoteomistajilta tarkempaa tietoa tuotteiden ominaisuuksien viestinnällisestä näkökulmasta. Palavereissa tuotejohto pystyi viestimään teknisille viestijöille tarkemmin, miten kustakin toiminnallisuudesta halutaan viestiä loppukäyttäjille.

4.2.2 Työmäärän arviointi ja priorisointi

Työmäärän arviointia kutsutaan usein ketterissä menetelmissä termillä *T-shirt sizing*. T-paitakokoja ovat XS, S, M, L ja XL. Tällä tarkoitetaan sitä, että kunkin *epici*n tai muutospyynnön kohdalla arvioidaan kuinka monta henkilötyöpäivää kyseinen muutos vie. *Epic* tarkoittaa ohjelmistokehitystyön kokonaisuutta, joka voidaan jakaa pienempiin osakokonaisuuksiin eli tehtäviin, joita kutsutaan käyttäjätarinoiksi (*stories* tai *user stories*) (Atlassian, 2023c). Työmäärän arvioinnista on koettu olevan hyötyä myös teknisen viestijän työn suunnittelussa sekä työmäärän viestinnässä ja priorisoinnissa muun muassa tuotejohdon suuntaan.

(7) H1: **Todettiin, että me halutaan noi arviot. Eli juuri se, että kuinka isosta taskista on kyse.** Joo, ja samaan aikaan me ehdottomasti ei haluttu käyttää absoluuttisia aikoja, koska etenkin meidän Product Ownereiden kanssa se meni helposti siihen, että jos sä sanot että no, mä luulen että saattaisi mennä ehkä viikko ja no sitten tapahtuu jotain yllättävää tai se ei olekaan ihan niin selkeä kuin on annettu ymmärtää tai tulee joku bugi tai vastaava jonnekin, niin sitten se ei välttämättä olekaan se viikko ja se absoluuttinen ajan lupaus kääntyy sitten siihen, että “Mut sä sanoit että hän menee viikko”. Eiku mä sanoin että mä luulen että siihen menee viikko. **Me mentiin T-shirt sizellä ja sitten me käännettiin ne storypointseiksi käytännössä** eli me sovittiin että kaikki siellä oli yks yks. Meillä oli ollut yksi tuota. Meillä oli ollu yksi päivitys lähiaikoina, josta me kaikki oltiin oltu siitä sitä mieltä, että tää on niinkun M-kokoinen T-paita, keskikoko, ja sovitaan, että tää on niin kuin tän verran storypointseja. Me käytettiin sitä aina mittatikkuna, okei tulee uusi muutos, niin mitä me

> luullaan että onko tää nyt niin kuin isompi kuin toi, mikä me ollaan yhdessä sovittu, että on meidän M-koko vai pienempi. Meillä oli noi stepit silleen, että kuinka monta storypointsia mikäkin T-paita koko oli. Se meni tosi hyvin, koska me saatiin ensinnäkin selville, että mikä meidän tiimin velocity on, kuinka paljon me saadaan käsiteltyä sprintissä kamaa läpi noin niin kuin keskimäärin. Ja sit se toisaalta, että se anto meille näyttää sen, että kun meillä on ne pisteet mietittynä ja sit se PO [Product Owner] tulee sieltä – siis ei se meidän hyvä PO vaan ne muut – että mä haluan tän! **Niin no katso, kun me ollaan aika rajoilla tässä nyt, mitäs me otetaan täältä pois, mikä ei ole niin tärkeetä kun tää on? Niin yleensä ne tässä vaiheessa oli silleen, että no seuraavaan sprinttiin?** Joo seuraavaan sprinttiin onnistuu! Tai sit ne oli silleen, että toi pois tää tilalle. Silleen et fine, voidaan hoitaa. Nää me huomattiin tosi hyödyllisiksi.

Esimerkin (7) mukaan työmäärän arviointi on koettu olevan soveltuva elementti teknisen viestinnän työtehtäviin, sillä niiden avulla voitiin kommunikoida tiimin ulkopuolelle, esimerkiksi tuotejohdon suuntaan, tiimin työskentelynopeus (*velocity*), ja samalla on saatu selville kuinka paljon työtä dokumentaatiotiimi pystyy yhdessä sprintissä tekemään. Haastateltavan tiimissä ei ole haluttu käyttää työaika-arvioihin absoluuttisia aikoja työtehtävien heikon ennustettavuuden vuoksi, sillä tehtävään menevää aikaa ei aina pystytä kovin tarkasti arvioimaan ennen työn aloittamista, vaan ne ovat arvioita ja aiheuttavat viestintäongelmia tuoteomistajien kanssa, mikäli alkuperäinen arvio työhön käytettävästä ajasta ei pidä paikkaansa. Tämän vuoksi tehtävien työmäärän arviointiin on käytetty yhteisesti sovittuja t-paitakokoja (*t-shirt sizing)*, jossa eri koot on muutettu tarinapisteiksi (*story point*), jotka ovat ketterissä menetelmissä yleisesti käytössä olevia työmäärän arvioinnin pienempiä yksiköitä. Näiden toimenpiteiden myötä tiimissä huomattiin, että viestintä tuoteomistajien suuntaan parantui ja työtehtäviä pystyttiin priorisoimaan paremmin, sillä kaikkea ei välttämättä aina ehditä tekemään yhden kehityssyklin aikana rajatulla resurssimäärällä. Työtehtävien arviointien myötä tuoteomistajat pystyivät tekemään päätöksiä kiireellisten työtehtävien sisällyttämisestä käynnissä olevan sprintin tehtäviin tai sen siirtämisestä seuraavaan sprinttiin, sillä tarinapisteiden ansiosta tiimin työmäärä ja työskentelynopeus oli tullut läpinäkyväksi.

> (8) H3: I found the Docs as code and the agile thing to be helpful. It really does make things smoother. **It keeps you as a writer, kind of honest too,**

> **or at least it helps me to remember what my priorities are and be like. Yes, OK, I'm gonna get these three things done**, or at least into the Verify column on our board. So that way when I report on in my next scrum, everyone knows where it is.

Esimerkin (8) mukaan *Docs as code* ja ketterät menetelmät auttavat teknistä viestijää työn sujuvoittamisessa. Se helpottaa tehtävien priorisoinnissa, kun voi keskittyä päivän aikana saamaan muutaman Jira-tiketin teknisen viestijän osalta valmiiksi eli tarkastettavaksi (*Verify*) sarakkeeseen Jiran tehtävätaululla. Esimerkissä kerrotaan myös, että tämän kaltainen asia on myös konkreettista kerrottavaa päivän Scrum-palaverissa kollegoille, jossa voi tehdä omaa työtä läpinäkyväksi muille.

4.3 Ketterien menetelmien haasteet

Aineiston mukaan ketterien menetelmien noudattaminen saattaa tuoda haasteita esimerkiksi työn suunnitteluun ja aikataulutukseen sekä eri aikavyöhykkeillä työskentelyyn. Menetelmiä voidaan myös joskus noudattaa liian kirjaimellisesti ja kokonaiskuvan hahmottaminen saattaa muodostua hankalaksi, kun työskennellään pienempien osakokonaisuuksien kanssa. Teknisen viestinnän alalla työn ulkoistaminen ja alihankinta saattaa olla myös esteenä ketterien menetelmien hyödyntämiseen.

4.3.1 Ohjelmistoprojektien työn suunnittelu ja aikataulutus

Ketterät menetelmät on kehitetty ohjelmistotuotantoprojekteihin eivätkä varsinaisesti teknisen viestinnän työtehtäviin. Ohjelmistoprojektien työn suunnittelu ja aikataulutus voi myös asettaa haasteita dokumentaation laatimiseen.

> (9) H1: Me ei oltu hirveän dogmaattisia Scrumin suhteen, koska me todettiin saman tien, että ei sitä ole suunniteltu tekniseen viestintään ja siinä on paljon asioita jotka ei niinku tasan käy järkeen. Se on ehkä tämmöinen *common sense Scrum.* […] Ensimmäinen on tosiaan se, että niitä ei ole suunniteltu dokumentaatiota silmällä pitäen. **Ne on hyvin usein suunniteltu puhtaasti softaprojekteille ja näin ollen, jos mainitaan**

> **dokumentaatio niin, mulla on semmonen kutina, että se on hyvin usein sitten mietitty niin, että joku nimetty devaaja, joka kirjoittaa sen**. Siinä ei tule sitä, että siinä on esimerkiksi minä, kirjoittaja, jonka tietotaso ei riitä ymmärtämään puhtaasti sitä koodia, jolta puuttuu se tekninen tausta. Se on tietotaidollisesti samalla tasolle niiden muiden devaajien kanssa, jolloin se voidaan tavallaan miettiä, että se tulee sen muun ohessa.

Esimerkissä (9) todetaan, että Scrumia ei ole suoraan suunniteltu teknisen viestinnän tehtäviin, joten haastateltavan organisaatiossa tekniset viestijät eivät noudattaneet Scrumin periaatteita tunnollisesti vain ainoastaan niiltä osin, kun se tuntui teknisen viestinnän näkökulmasta järkevältä, haastateltava kutsuu tällaista lähestymistapaa nimellä *common sense scrum*. Esimerkin mukaan dokumentaatio on otettu Scrumin kaltaisissa ketterissä menetelmissä huomioon ainoastaan sillä tavalla, että ohjelmistokehittäjä laatisi dokumentaatiota työnsä ohessa, joten se ei ota teknisen viestijän roolia huomioon.

Ketterien mallien haluttomuus aikatauluttaa ja suunnitella dokumentaatio osaksi ohjelmistotuotantoprojektia aiheuttaa haasteita dokumentaation laatimiseen.

> (10) H3: cause so I think Agile, it's a great… **It's a good way of thinking about how to make software until you have to include documentation as code and as part of that software and you force it to be a Definition of Done**. Like if you force that to be an artifact that has to be there at the end of the day. **Finding time for documentation can be tricky, and I don't think a lot of teams like to do it as part of their Sprint planning**. You know, I think they they already see QA as being problematic to their timeline, and I think adding in documentation can be even more problematic for them.

Esimerkin (10) mukaan ketterät menetelmät toimivat hyvin, kunnes dokumentaatio tulisi laatia osaksi valmista tuotetta. Ketterissä menetelmissä käytetään termiä valmiin määritelmä (*Definition of Done)* kuvaamaan organisaation määrittelemää laatukriteeriä valmiille tuotteelle. Ketterät menetelmät ovat niin keskittyneitä ohjelmistokehitykseen, että jo laadunvarmistamisen aikataulu testaustiimin puolelta saattaa aiheuttaa viivästyksiä tuotteen tai ominaisuuden valmistumiseen ohjelmistokehityksen näkökulmasta puhumattakaan siitä, että dokumentaation voisi suunnitella osaksi

tuotetta jo sprintin suunnittelupalavereissa, jolloin sen laatimiseen tarvittava aika voitaisiin aikatauluttaa ja ottaa huomioon ohjelmistokehitysvaiheessa.

4.3.2 Työskentely eri aikavyöhykkeillä ja ketterien menetelmien tunnollinen noudattaminen

Maailmanlaajuinen etätyö saattaa aiheuttaa haasteita myös teknisille viestijöille. Haasteet nousevat erityisesti siinä tapauksessa, jos viestijöiden tulisi osallistua monien eri tuotetiimien päivittäisiin Scrum-tapaamisiin.

> (11) H3: **Another problem I always encounter with Agile is the time zone thing**. So we've got one team member who's in California, so he's on the Pacific Coast of America. And then we've got team members who are all the way on, like the eastern side of Ukraine and so they're in that furthest kind of like time zone there in Eastern Europe. We can't have a scrum on the same time every day, so Monday, Wednesday, Friday we have scrum at 9:00 AM Chicago time, which is how I think of it. And then Tuesday, Thursday, like today we'll have one almost at noon my time, so that certain team members can join. So our guy in California, he only joins the scrum 2 times a week and then our team members in Ukraine can only join three times a week. **This makes getting the whole team together really challenging and not everyone always gets all the same information via the screen.** So we're not actually doing Agile correctly in that regard.

Esimerkin (11) mukaan eri aikavyöhykkeillä työskentely aiheuttaa haasteita tiimityölle ja päivittäisten palaverien suhteen on jouduttu tekemään kompromisseja aikataulujen suhteen, jotta kukaan ei joutuisi jatkuvasti osallistumaan palavereihin epäinhimilliseen aikaan. Esimerkiksi kauimmaisella aikavyöhykkeellä Yhdysvaltojen Kaliforniassa työskentelevä tekninen viestijä osallistuu tästä syytä Scrum-palaveriin ainoastaan kaksi kertaa viikossa ja Euroopan Ukrainassa työskentelevä tiimin jäsen kolme kertaa viikossa, joten Scrumin mukainen päivittäinen palaveri ei aikaerojen vuoksi täysin toteudu.

> (12) H3: The other problem is that **when you're servicing like 6 product teams, you are sometimes going to like 4 to 6 scrums almost every single day**, and we actually had to scale that back and so we started saying hey, you know, I realize that your scrum, your agile process is important, but could

> we maybe come twice a week to yours or twice a week to yours? So it's been challenging 'cause we're not doing a true agile then at that point, either 'cause you're not able to say hey on that task, I'm not done so a lot of it has become leading comments in your Jira tickets or you know just being very communicative in our Webex chat spaces must be like hey for that ticket here's where I'm at on it. So it's been **really challenging from that perspective of just time zones and multiple scrums, and trying to balance your day 'cause you know. Obviously, if you have that many scrums, you're already eating up about 2 hours of your day every day just going to Scrum, and it's not even that beneficial sometimes.**

Esimerkissä (12) kerrotaan, että eri tuotetiimien päivittäisiin Scrum-palavereihin osallistuminen saattaa viedä tekniseltä viestijältä paljon työaikaa päivittäin, jos viestijä palvelee kuutta eri tuotetiimiä. Esimerkin mukaan haastateltavan organisaatiossa on päädytty jakamaan siten, että tuotekehitystiimien päivittäisiin palavereihin osallistutaan vain muutamina päivinä viikosta. Tämän vuoksi viestintäyhteys saattaa hieman heikentyä, mutta sitä voidaan kompensoida viestimällä muissa kanavissa. Haastateltava antaa esimerkin viestinnästä Jira-tiketin kommenttikentässä tai organisaation käyttämässä Webex-pikaviestipalvelussa. Tekninen viestijä joutuu tasapainottelemaan työn organisoinnin kanssa, jotta aika ei menisi pelkästään palavereissa istumiseen, jotka eivät haastateltavan mukaan aina ole niin hyödyllisiä.

Joskus ketterien menetelmien prosessien noudattamiseen koetaan menevän enemmän aikaa kuin itse työtehtävän tekemiseen.

> (13) H3: I'd say sometimes the only drawback I can think of would be that I spend a lot more time doing bureaucratic stuff like moving tickets or putting comments into tickets. I'm very calm and focused on that kind of thing, so like I want to capture a lot of information in my tasks for whatever reason, so I find myself, you know, spending more time than I would have by being, you know, putting in commentary in my task, saying things like, oh I talked to so and so. Or here's a copy of this email that goes with this, that sort of thing. So I feel like **I'm documenting my work sometimes almost more than documenting the work itself.** I find that true for like really tiny bugs like we had a bug coming where I had to update a link and then so I made the link change. It was nothing. You know merge request made in like you know maybe 10 minutes it took. But then I spent you know, 15 minutes updating the task. Putting all this information in there about where it was, here's a copy to the MR. You

know all that sort of thing, and I'm like so sometimes I find myself looking at my something like that.

Esimerkin (13) mukaan ketterät menetelmät saatetaan kokea hieman byrokraattisiksi ja teknisellä viestijällä menee paljon aikaa oman työn dokumentointiin ja tikettien siirtämiseen sekä kommentointiin, vaikka dokumentaation päivittäminen saattaisi olla nopea toimenpide. Näin saattaa tapahtua, jos kyseessä on vaikka pienen ohjelmavirheen korjaus. Esimerkissä mainitaan, että teknisen viestijän piti päivittää linkki ja tehdä muutoksen sulauttamisesta päähaaraan. Vetopyynnön tekemiseen meni 10 minuuttia, mutta haastateltavalta kului 15 minuuttia tehtävään liittyvän tiketin päivittämiseen Jirassa, jotta siinä olisi kaikki tarvittava tieto ja kopio vetopyynnöstä.

(14) H3: **Sometimes the process can be a little too formal**, I think. Uhm, you know we've had situations where it's kind of an all hands sort of thing where it's like, hey, we just really need to get this bug fix in, you know, we launched with this code 2 weeks later. We found out there was a major breaking bug and the agile process, you know, the scrum master will kind of force us into this agile process, but we'd rather be a little bit more free form with it, you say like hey this bug is the main thing I need to document the updates in the doc, they're gonna adjust this code fix. I need to work on the release notes. **I know what we need to do, developers know what we need to do.** Let's just skip the agile process and I think being flexible like that is the best thing when they sort of force the issue. And they're like no. We have to have this scrum every single day, and it has to be the structure it sometimes. **It's just not valuable, and I sometimes wish that would go away and the other thing I think too is that sometimes trying to fit everything into neat little epics and tasks doesn't always work. Sometimes it staunches the creativity a little bit where I think**. […] business development too, really likes to see Agile because they can kind of plan how much what they can sell further on.

Esimerkin (14) mukaan prosessi voi joskus tuntua hieman viralliselta ja haitata varsinaista työntekoa. Haastateltava mainitsee esimerkin, jossa julkaisuun menevästä tuotteesta löydetään ohjelmavirhe, jolloin ohjelmistokehittäjien tulee tehdä korjaus koodiin ja teknisen viestijän tehtävänä päivittää dokumentaatiota ja versiotiedotetta (*release note*). Vaikka kaikki tietävät omat tehtävänsä, ja työtehtävät voisi hoitaa joustavasti, niin silti aina on noudatettava ketterien menetelmien rakennetta ja mentävä päivittäiseen tapaamiseen. Esimerkin mukaan se ei ole aina hyödyllistä ja syö mahdollisuutta luovalle

työlle. Haastateltava mainitsee myös, että kaikkea työtä ei aina ole mahdollista mahduttaa *epic* tai *task*-muotoon Jirassa.

4.3.3 Kokonaiskuvan puuttuminen

Ketterien menetelmien haasteena tekniselle viestijälle voi myös olla kokonaiskuvan puuttuminen, kun työskennellään pienten osakokonaisuuksien ja toiminnallisuuksien kanssa.

> (15) H1: Käytännössä **huomaa työskentelevänsä niinku paljon pienemmissä fragmenteissa eli sulla on yhtäaikaa paljon pieniä muutoksia**. On siellä sitten kun pojat rakentaa jonkun ison featuren niin onhan siellä sitten semmoinen hervoton köntsä aina välillä, mutta niin kun huomaa juuri sen, että on paljon pieniä branchejä, jotka tulee nopeasti ja sitten menee tonne ja joka asiasta on tiketti. **Joka bugi on oma tikettinsä, niin sitten niitä tavallaan niitä palasia kertyy. Eli jos tämmöisen kokee niin kuin hallitsemattomana tai ahdistavana, että liikaa palasia**, mä en voi hallita tätä kaikkea niin se voi olla tosi haastavaa. Minulle se sopii.

Esimerkissä (15) haastateltava kertoo, että ketteriä menetelmiä noudatettaessa työskennellään yleensä pienten muutoksien kanssa, jotka ovat osana isompaa kokonaisuutta. Tässä tapauksessa tekninen joutuu seuraamaan haaroja (*branch*), joissa on koodimuutoksia, ja jotka mahdollisesti aiheuttavat työtä myös viestijälle. Ohjelmistokoodin virheenkorjauksesta aiheutuvista muutoksista aiheutuu myös mahdollisesti työtä tekniselle viestijälle. Lopputuloksena on paljon pieniä muutoksia aiheuttavia tikettejä, joka voi tuntua hallitsemattomalta joillekin teknisille viestijöille. Haastateltava kuitenkin huomauttaa, että hänelle tämänkaltainen työskentelytapa sopii hyvin.

> (16) H4: ’cause in Agile people can focus so closely on the sort of tactical and operational mechanisms sort of encourage you to think in these two week cycles, if you're doing Sprint planning right? It's easy to forget to stop look up at the horizon and see where you're going.[..] so **it can be difficult to sort of keep an eye on the longer horizon, which is the what the more traditional waterfall planning does**, right? That it's quite good

> at that. But it's that, but as long as you're aware I think that's just something you deal with.

Esimerkissä (16) haastateltavalla on samankaltaisia kokemuksia ketterien menetelmien kanssa toimimisesta. Ketterät menetelmät rohkaisevat työskentelemään esimerkiksi kahden viikon sprinteissä, joiden aikana voi olla haastavaa ajatella lopputulosta tai kokonaiskuvaa, kun työ on jaettu pienempiin osiin. Esimerkin mukaan perinteisessä vesiputousmallissa keskitytään paremmin kokonaiskuvaan ja lopputulokseen. Haastateltavan mukaan asian kanssa voi hyvin tulla toimeen, kunhan tekninen viestijä vain tiedostaa tilanteen ja pyrkii mahdollisuuksien mukaan aina välillä miettimään asioita kokonaiskuvan kannalta.

> (17) H3: they'll [customers] say to us, hey, I'm having a huge issue trying to get this feature to run, and here's exactly why and then our team will absorb it and then they'll come back to me, as the writer be like hey we found this issue in the code block. Can you update all the code blocks and I'm like no problem and get that in the doc so it's kind of a multifaceted way of doing it. **I find this environment to be a little bit hectic sometimes, like there's a lot going on. Lots of tickets to manage.**

Esimerkin (17) haastateltava toimii teknisen dokumentaatiotiimin esimiehenä ja kuvaa tiimin vastuuta ongelmien selvittämisessä ja korjaamisessa. Teknisen viestijän tehtäviin kuuluu ymmärtää mitä koodipäivitys tarkoittaa dokumentaation päivityksen kannalta. Haastateltava kokee työympäristön hektiseksi ja työ vaatii monien tikettien hallinnointia samanaikaisesti.

4.3.4 Työn ulkoistaminen ja alihankinta

Teknisen viestinnän alalla toimii Suomessa ja kansainvälisesti organisaatioita (esimerkiksi Etteplan ja ALTEN), jotka tarjoavat teknisen viestinnän palveluita asiakkailleen ostopalveluna. Tällöin tekninen viestijä toimii alihankintaroolissa ja konsulttina asiakasyrityksissä. Muutaman haastateltavan mukaan tämä kuvio saattaa

lähtökohtaisesti aiheuttaa ongelmia ketterien menetelmien hyödyntämiseen. Ketterät menetelmät korostavat viestinnän tärkeyttä, ja alihankintasuhteessa oleville teknisille viestijöille ei välttämättä haluta jakaa kaikkea asiakasyrityksen ohjelmistotuotantoprosessiin liittyvää tietoa.

(18) H1: Ja sit puolestaan sen ulkoistajaorganisaation puolelta se ongelma on just se mitä mä sanoin aikaisemmin, että jotta Scrumia vois tehdä järkevästi niin käytännössä mun mielestä tarvitaan joku Jiran kaltainen asia – henkilökohtaisesti mä oon suuri Jira-fani – et sä tarvitset jotain mistä näkee missä kama on, mikä on niiden status, kenen pöydällä se istuu. Ja yleensä tässä tulee sit että **se info mitä tarvitaan et se voidaan ns. suorittaa alusta loppuun – tuotekehityspalanen siinä – vaatii niin paljon tavallaan detaljia että helposti tullaan siihen että onko ok antaa tätä infoa eksternaalille?** Etenkin aikaisemmin organisaatiot ja isot organisaatiot, ne voi olla hyvin paranoideja sen suhteen, että minkä verran infoa ne suostuu antamaan, jos sä et oo meidän oma työtekijä. **Ja jopa silloin jolloin tulee käytännössä täysin mahdottomaksi toimivaa Scrummia – etenkin Scrummia – sillee et sulla pitäis olla pääsy Jiraan, jossa on paljon sensitive infoa ja jotenkin pystyä rajaamaan se, että mihin projektiin ne nyt kattoo ja ettei vaan mainita mitään sellaista siellä kommenteissa mitä nää ei ehkä sais nähdä tai vastaavaa niin tota… Niin tuota mä en itseasiassa joo, mä luulen että se saattaa olla käytännössä mahdotonta.**

Esimerkin (18) mukaan ketterässä kehityksessä olennaisena työkaluna on työnhallintaan IT-alan yrityksissä yleisesti käytössä oleva kaupallinen työkalu nimeltään Jira. Haastateltavan mukaan isot organisaatiot eivät välttämättä halua jakaa kaikkea tuotekehityksen liittyvää tietoa ulkoistetuille teknisille viestijöille. Tämä tekee ketterien menetelmien hyödyntämisestä mahdotonta, sillä ne perustuvat avoimeen tiedon jakamiseen, eivätkä ota huomioon tilanteita, joissa tietoa ei voisi jakaa tietyille henkilöille.

(19) H2: Esimerkiksi sellainen tilanne että olin, oli tässä tota niinku juuri tähän SAFe-malliin tuota liittyvä semmoinen Scrum of Scrums, on viikkopalaverit ja siinä on kaikki tuoteomistajat sitten koolla. Ja tota mä olin siinä aluksi mukana et **sit mä olin aina niinku kartalla että mitä mikäkin tiimi suunnittelee**, mutta sitten se tuota, yksikön… **tietohallintoyksikön pomo sano että eihän mun kannata siinä nyt sitten aikaani hukata**, että mä tuun niihin palavereihin. Eli tuota, **eihän hän**

sitten selvästikään ollut sitä ymmärtänyt sitä niinku pointtia että tuota niin tietenkin dokumentoinnin pitäisi olla juurikin tässä palaverissa.

Esimerkissä (19) todetaan, että työn ulkoistaminen voi myös johtaa siihen, että teknistä viestijää ei haluta ottaa mukaan Scrum-toimintamallin mukaisiin palavereihin, jotka tarjoavat teknisille viestijöille tärkeitä tiedonlähteitä tuotekehitykseen liittyen. Haastateltavan mukaan organisaatio piti viikkopalavereita, jossa olivat mukana kaikki tuoteomistajat (*Product Owner)* ja jossa käytiin läpi suunnittelussa olevia toiminnallisuuksia. Tämänkaltaiset palaverit luovat teknisille viestijöille läpinäkyvyyttä mahdollisiin tuleviin dokumentaatioon liittyviin töihin ja selventävät suunnitteilla olevia toiminnallisuuksia.

(20) H2: Että siellä on vähän sellaista niinku **luottamuspulaa tavallaan** että. Että mut on niinku asiantuntijana palkattu ja näin asianmukaisesti, mutta sitten ei oikein kuitenkaan uskalleta päästää irti siitä omasta sisällöstä ja anneta niinku. En tarkoita, että pitäisi vapaat kädet saada, mutta että sellainen niinku **hei antakaa mä nyt teen tämän**. Että sellainen tilanne nyt sitten tässä nykyisessä.

Esimerkissä (20) haastateltava koki työpaikassaan myös tilaajaorganisaation puolelta hieman luottamuspulaa liittyen alihankintasuhteeseen. Organisaatio ei kuitenkaan pystynyt antamaan tekniselle viestijälle toimintaympäristöä, jossa hän voisi hoitaa työnsä riittävän hyvin.

(21) H2: Että mun kun **mä teen konsulttityötä tuntitaksalla ja olen aika kallis**. Niin tuota tavallaan, että se niinku nähdään, että mä oon sellanen joku niinku typewriter, että mä vaan niinku nak nak tuotan. Ja **tuotanto katkeaa, jos menen nyt vaikka tunniksi tai kahdeksi viikossa palaveriin**, etsit siis mun mielestä siellä **ei niinku oikein nähdä sitä isoa kuvaa**, mut tää on mun mielipide, mun vaikutelma asiaan.

Esimerkissä (21) haastateltava koki myös, että hänen korkea tuntihintansa voi olla syynä siihen, minkä vuoksi hänen ei haluta osallistua ketterän toimintamallin mukaisiin palavereihin. Tilaajaorganisaatiossa on hänen mukaansa käsitys, että kaikki heidän tilaamansa tunnit tulisi olla ns. tuottavaa työtä, jonka aikana muodostuu heidän tilaamiansa dokumentaatiotuotteita. Esimerkin mukaan organisaatiossa ei koeta, että

teknisen viestijän olisi hyödyllistä olla mukana palavereissa, vaikka ne voisivat olla tehokas tapa jakaa tietoa ja johtaa dokumentaation laadun paranemiseen.

4.4 *Docs as code* -menetelmä

Kahdella haastateltavalla (H3 ja H4) oli omakohtaista kokemusta *Docs as code* -menetelmän mukaisesta työskentelytavasta ja molemmat haastateltavat työskentelivät organisaatiossa, jossa *Docs as code* on vakiintunut toimintamenetelmä ja kaikkien teknisten viestijöiden oletetaan noudattavan samaa työskentelytapaa.

Yhden haastateltavan mukaan *Docs as code* tarkoittaa menetelmää, jossa ohjelmistokehittäjät kirjoittavat dokumentaatiota työtehtäviensä ohessa ohjelmistokehitykseen tarkoitetuilla työkaluilla ja teknisillä viestijöillä ei ole varsinasta roolia tässä prosessissa. Kahden muun haastateltavan (H3 ja H4) mukaan *Docs as codella* tarkoitetaan kuitenkin sellaista menetelmää, jossa tekninen viestijä kirjoittaa dokumentaatiota ohjelmistokehitykseen suunnatuilla työkaluilla samassa kehitysympäristössä. Toimintamallin määrittelyssä nousi myös esiin rinnakkainen toimintamalli, *enablement model*.

> (21) H4: **I call that enablement model**. So what does that mean? That means that **you have the people who are creating this software product create the documentation for that software product**. **Those are the developers and the writers have a mentoring role there**, right? So you can ask, but what does that mean in practice? That means then that you have to **define the document types** that the product should issue. You should have **templates for each type**. And basically is you don't just give the developers a blank page and say write stuff, because that's not what their training is, right? Yeah. **Task analysis stuff that is very common to us writers is not something they're familiar with**. So what you do in that context is you provide templates you define what documents should be issued, right? You know in the templates. **Basically you also provide boilerplate. You'll put this information here. Here's where you put your example**. When I say example, this is what I mean by the word example, right? Is that so **you give them a structure** into which they can put their

information. **You also are there for the reviews**. And to also, if they ask questions about, well, how do I express this? You help them with that too.

Esimerkissä (21) puhutaan mallista, jota kutsutaan nimellä ”*enablement model*”. Mallilla tarkoitetaan työskentelytapaa, jossa ohjelmistokehittäjät kirjoittavat dokumentaation ja tekniset viestijät mahdollistavat tämän prosessin omalla asiantuntijuudellaan esimerkiksi informaation muotoilusta ja tyypittelystä (*task analysis*), jolloin teknisen viestijän rooli on konsultatiivinen. Tässä mallissa tekniset viestijät voivat esimerkiksi tuottaa ohjelmistokehittäjille dokumenttien pohjia (*template*) ja ovat mukana dokumenttien katselmointivaiheessa (*review*). Tekniset viestijät voivat myös auttaa kielellisessä ilmaisussa. *Enablement*-mallissa teknisen viestijän tehtävänä on mahdollista yhtenäisten dokumenttien tuottaminen ja helpottaa kirjoittamista esituotettujen dokumenttipohjien avulla. Tämän mallin vastakohtana on *Docs as code* -menetelmä, jossa tekniset viestijät on integroitu osaksi ohjelmistokehitystiimiä ja käyttävät samoja työkaluja, joita käytetään ohjelmistokoodin kirjoittamiseen.

4.4.1 *Docs as code* -menetelmän hyödyt tekniselle viestijälle

Tutkimusaineiston mukaan *Docs as code* -toimintamenetelmän noudattaminen tuo tekniselle viestijälle monia hyötyjä. Samassa ympäristössä ohjelmistokehittäjän kanssa toimiminen tuo tekniselle viestijälle uskottavuutta ja tuo muille tiiminjäsenille läpinäkyvyyttä teknisen viestijän työpanokseen. Toimintamallin avulla saadaan luotua myös laadukkaampaa dokumentaatiota, sillä ohjelmistotuotannon työkalujen avulla dokumentaatiota voidaan kommentoida ja työstää koodikatselmointien (*code review*) tapaan.

(22) H4: Right, So what do you get? Yeah, **you get a much quicker velocity on reviews and collaborative authoring.** Right, what do I mean by that is because you're not asking developers to step out of the environment that they're used to working in, and potentially log into some CMS. Yeah, that they don't understand, and then they see like once every six months they at least have to remember how things work. My experience has been

> even developers who value documentation just don't do that. Because they have enough pressures on them. That you know, they find it difficult to switch contexts right? And currently anyway, **the vibe and development world is don't switch contexts, right? They want to do everything in one space**. **Right, so by working in their environment, the collaboration works better.** In my experience, right notice I say better, right? 'cause global, you can't make global statements about this. People are people, right? **The other thing too is it gets you a fair degree of credibility with your technical colleagues.**

Esimerkin (22) mukaan tekninen viestijä saa mallissa nopeammin tapahtuvia katselmointeja ja mallissa toteutuu yhteistoiminnallinen kirjoittaminen teknisen viestijän ja ohjelmistokehittäjän välillä. Tämä toteutuu, kun työskennellään samassa ympäristössä ohjelmistokehittäjien kanssa. Haastateltavan kokemuksen mukaan ohjelmistokehittäjän saattaa olla vaikea siirtyä työskentelemään toiseen ympäristöön, esimerkiksi teknisen viestijän käyttämään sisällönhallintajärjestelmään, ja opetella sen toimintatapoja. Esimerkissä kerrotaan, että vaikka ohjelmistokehittäjä arvostaisi dokumentaation tuottamista, ympäristöstä toiseen siirtyminen ei ole mutkatonta, sillä ohjelmistokehittäjät haluavat työskennellä yhdessä ympäristössä. Haastateltavan mukaan samassa ympäristössä toimiminen parantaa yhteistyötä ja sen avulla voi myös saada uskottavuutta ohjelmistokehittäjien keskuudessa.

> (23) H4: I only do developer documentation. **If I'm using the same tools developers use, but I understand my audience better**. Because I'm literally doing what they do, right? So it gives me insight into audience and audience behaviour.

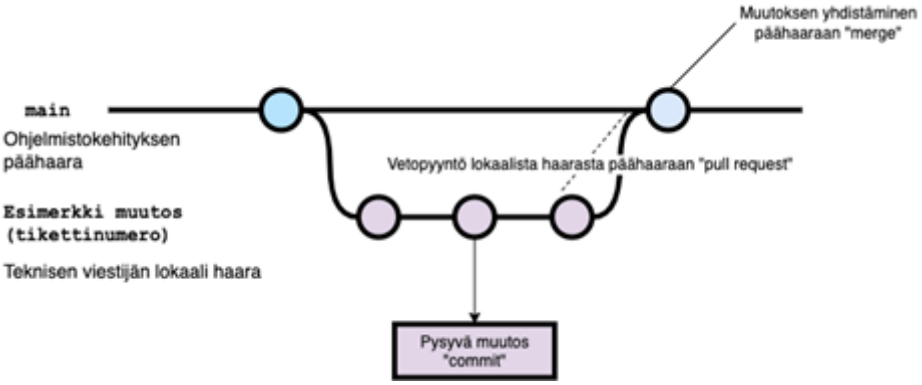
Esimerkin (23) mukaan *Docs as code* -menetelmä soveltuu erityisen hyvin tekniseen, ohjelmoijille suunnatun, dokumentaation tuottamiseen. Haastateltavan mukaan samojen työkalujen käyttö ohjelmistokehittäjien kanssa luo samalla myös ymmärrystä kirjoittajalle dokumentaation kohdeyleisöstä, joka koostuu ohjelmistokehittäjistä haastateltavan tapauksessa. Tämä toimintamalli antaa kirjoittajille myös tietoa ohjelmistokehittäjien käyttäytymisestä.

> (24) H3: It was the kind of it was the kind of company where you would have to print out your documentation and bring it to somebody. They would mark it up with a red pen and that was our way of getting feedback and

> it was a nightmare and it just added so much time to my day, so I think the biggest thing Docs as code was. **Not just the editing process being a lot smoother and having a central location for your doc, but the ability to use those tools to do things like suggest edits,** which I think are great because I mean technical writers nowadays… You know our value isn't in the fact that we can, like you know, say, oh, you're right there needs to be a comma there and then we go in there. Make a commit to the merge request and added comma. That's such a waste of time. So the more we can use these tools be like Oh yeah, you saw that there was a missing comma. You put the suggestion in there and I just hit apply suggestion. Apply suggestion, merge great. Yeah, **it just makes it so much faster and smoother**. And the fact too that you know when there is a change. Like for example, you know, **we find a late breaking bug. We can't roll out this one part of the feature in production because I've broken out every single doc element into unique merge requests for each feature to have, but I didn't commit, I can just dial up that commit number and say roll back this one specific commit from master branch and I can have the same. You know high-level, valuable documentation ready to go in about 30 minutes as opposed to having to stop and manually try to take it out** or manually comment it out. Or all of that stuff. So being able to sort of just, you know, refine the content on the fly. That's the value I think, of Docs as code, yeah?

Esimerkin (24) mukaan samojen työkalujen käyttö sujuvoittaa dokumentaation laatimisen prosessia. Joskus lähellä ohjelmistotuotteen julkaisua löydetään virhe, jonka vuoksi kyseistä toiminnallisuutta ei voida laittaa tuotantoon. Esimerkin haastateltava on tässä tapauksessa laittanut jokaisen dokumentaatioelementin erillisiin vetopyyntöihin (*pull request* tai *merge request*), joista jokainen liittyy omaan toiminnallisuuteen (*feature*). Vetopyyntöjä tehdään, jotta voidaan sulauttaa muutoksia esimerkiksi teknisen viestijän luomasta lokaalista haarasta ohjelmiston päähaaraan. Viestijä voi tässä tapauksessa peruuttaa toiminnallisuuden dokumentaatioon liittyvän pysyvän muutoksen (*commit*) tuotantoympäristön päähaarasta ja tuote on dokumentaation puolelta valmis julkaistavaksi hyvin lyhyessä ajassa. Vetopyyntö tehdään lokaalista haarasta ohjelmiston päähaaraan. Esimerkissä nähdäänkin Docs as code -menetelmä arvokkaana juuri tästä syystä, että sisältöä voi hyvin joustavasti muokata ”lennossa” ja viime hetken dokumentaatiomuutoksiin ei mene liikaa aikaa. Dokumenttien editointi vetopyynnöissä on myös helppoa sisäänrakennetun toiminnallisuuden ansiosta, jossa jokainen voi ehdottaa muutoksia koodiin tai teknisen viestijän tapauksessa viestijän

laatimaan dokumentaatioon, ja viestijä voi hyväksyä muutoksia yhdellä napin painalluksella (*Apply suggestion* -toiminnallisuus). Esimerkissä mainitaan puuttuva pilkku, jonka lisääminen tämän toiminnallisuuden avulla on helppoa ja nopeaa, eikä teknisen viestijän tarvitse tehdä lähdekoodiin uutta muutosta virheen korjaamiseksi.



Kuvio 8. Esimerkki ohjelmistokehityksen versionhallinnan haaroituksesta.

Kuviossa 8 on selvennetty ohjelmistokehitysympäristöjen haaroitusta. Tekninen viestijä luo oman lokaalin haaran päähaarasta, ja kun muutos on valmis, tekninen viestijä puskee muutokset pysyvän muutoksen avulla (*commit*) ja tekee vetopyynnön sisällön yhdistämisestä päähaaraan. Vetopyynnön tarkoitus on vetää (pull) ja yhdistää (*merge*) lokaaliin haaraan puskettu koodi päähaaraan (*main branch*). Vetopyynnön yhteydessä sen sisältö voidaan vielä katselmoida.

(25) H3: if the more invested I think the team is in supplying feedback is always great. **I think my best doc usually is when the merge request is very well commented from other people**, but I think **that kind of follows the Docs as code model** where documentation shouldn't be this thing where somebody is often at corner writing within Microsoft Word anymore.

Esimerkin (25) mukaan *Docs as code* -menetelmä mukainen työskentely parhaimmillaan nostaa dokumentaation laatua, sillä dokumentaation katselmointi ja kommentointi on helppoa GitLabin tai GitHubin kaltaisissa työkaluissa. Tekninen viestijä tekee dokumentaatiosta pysyvän muutoksen, jonka jälkeen hän tekee siitä vetopyynnön, jota kutsutaan nimellä *pull request* tai *merge request* käytettävästä työkalusta riippuen. Vetopyynnön yhteydessä ohjelmistokehittäjät ja muut sidosryhmät voivat katselmoida

ja kommentoida dokumentaatiota. Haastateltavan mukaan hyvin kommentoitu dokumentaatio parantaa sen laatua. Jitianu (2021, s. 23) tuo myös esiin yhteistyön ja katselmoinnin merkityksen dokumentaation laadulle. Jitianu (2021, s. 23) huomauttaakin, että jo niinkin yksinkertainen asia, kuten versionhallintajärjestelmien käyttö ja niiden sisältämien vetopyynnön tekemiseen tarkoitettujen ominaisuuksien käyttö fasilitoi yhteistoiminnallista ympäristöä teknisen viestijän ja muiden sidosryhmien välillä. Katselmointi voidaan liittää myös luvussa mainittuun Kisterin (2016, s. 205-205) projektinhallinnan mallin katselmoinnin ja parantamisen vaiheisiin, jotka ovat tärkeitä dokumentaation laadunvarmistamisen ja validoinnin näkökulmasta.

(26) H4: So if you **think about a pull request, that's actually very effective for quality,** right? Because what that means is. Relevant part people, **people both in the development team and outside the development team can like review the content** right? So you have the more eyes the better. In some ways, right? Because we all come with our own perspectives, so that's incredibly useful.[…] **We know that the later in the process, you catch the problem. The more expensive it is to fix it.**

Esimerkin (26) mukaan haastateltava H4 on samaa mieltä haastateltavan H3 kanssa. Dokumenttien katselmoinnin katsotaan parantavan dokumentaation laatua, kun tärkeät sidosryhmät voivat katselmoida ja kommentoida dokumentaatiota versionhallintatyökalun sisällä teknisen viestijän tehtyä vetopyynnön järjestelmässä. Laatu parantuu, kun useampi silmäpari tarkistaa tekstin. Haastateltava tuo mukaan myös liiketoiminnallisen näkökulman, jonka mukaan mahdollisten virheiden huomaaminen ja korjaaminen varhaisessa vaiheessa on edullisempaa, kun myöhemmin.

4.4.2 *Docs as code* -menetelmän haasteet tekniselle viestijälle

Docs as code asettaa tekniselle viestijälle myös muutamia haasteita. Menetelmän mukaisia työkaluja ei ole optimoitu teknisille viestijöille, vaan ohjelmistokehittäjille, mistä saattaa koitua haasteita esimerkiksi lokalisoinnin hallintaan.

(27) H4: The tools you work with are optimised for developers, not documentation. Doc tools offer you whole suite of things which are very,

very useful for managing documentation and you just don't have those. Yeah, right? So **you're less efficient in your daily work, but you're more efficient in gaining credibility and collaborative efforts with your technical colleagues, right? So to my mind it's just a brutal trade**. Yeah, and my experience has been over the years and I've worked in a number of different models. **It's worth it.**

Esimerkin (27) mukaan yksi *Docs as code* -menetelmän asettamista haasteista on se, että ohjelmistotuotannon työkalut, kuten versionhallinta, on optimoitu ohjelmistokehittäjille eikä dokumentaation laatimiseen. Dokumentaation sisällönhallintajärjestelmät tarjoavat teknisille viestijöille monia hyödyllisiä ominaisuuksia, kuten yksilähteistämisen ominaisuuksia, jotka mahdollistavat sisällön uudelleenkäytön, mitä ohjelmistotuotannon työkalut eivät tarjoa. Se saattaa tarkoittaa sitä, että tekninen viestijä ei ole päivittäisessä työssään niin tehokas, mutta kolikon kääntöpuolena on, että viestijä saa lisää uskottavuutta ja pääsee paremmin yhteistyöhön ohjelmistokehittäjien kanssa. Haastateltavan mukaan tämä on julma vaihtokauppa, mutta se on hänen kokemuksensa mukaan sen arvoista, sillä hän on vuosien aikana tehnyt töitä usealla menetelmällä ja toimintamallilla.

(28) H4: it's a different way of thinking, right? Be kind to yourself, nobody is born knowing. You have to learn this stuff and learning is not just one person telling you quickly in 30 minutes how it's supposed to work right? **You need to have a sort of a muscle memory for these kinds of things, right? That just takes a while. So that's one trade off**. It requires an additional level of learning from you in order to work in this way.

Esimerkin (28) mukaan tekniset viestijät tarvitsevat koulutusta heille mahdollisesti uudessa ohjelmistokehittäjien toimintaympäristössä toimimiseen, sillä se vaatii viestijältä uudenlaisen ajatusmallin. Ohjelmistokehittäjien työkalut, esimerkiksi Gitin ja versionhallinnan ohjelmistot, vaativat käyttäjältä usein lihasmuistia ja monia toistoja, jotta niiden käytön voi todella omaksua. Esimerkin mukaan oppiminen voi usein viedä kauankin aikaa, sillä toimintamallia ei voi oppia kovin lyhyessä ajassa hyvin.

(29) H4: In my personal valuation that makes me less efficient as a writer. But **I gain more than I lose by entering this collaborative space**.

Esimerkin (29) mukaan *Docs as coden* tuottamat hyödyt peittoavat mahdolliset haitat. Esimerkin mukaan käyttämällä samaa toimintamallia ohjelmistokehittäjien kanssa päästään yhteistyöhön (*collaborative space*), jonka tarjoamat hyödyt peittoavat mahdolliset haitat teknisen viestijän työn tehokkuuden heikentymisessä. Tehokkuuden heikentyminen tarkoittaa tässä tapauksessa mahdollisista sisällönhallintajärjestelmistä (CMS) luopumista.

(30) H4: They've **spent the last 20 years building features into their tools to help us manage files and localization and things like that**. Nope, you can do it, but it works totally differently and **you have to have a pretty good grasp on the general principles of the Docs as code** thing, you need to have to learn, and you know actions and things that you do, things where you copy things over and all this other kind of stuff and you have to have fundamental thoughts about where do the files go right?

Esimerkissä (30) kerrotaan, että lokalisoinnin hallinta on *Docs as code* -menetelmässä monimutkaisempaa. Haastateltava kertoo teknisen viestinnän alan sisällönhallintajärjestelmistä, joita on kehitetty ja optimoitu teknisten viestijöiden käyttöön viimeisen kahdenkymmenen vuoden ajan. Lokalisointi on mahdollista myös *Docs as code* -menetelmää noudattamalla, mutta viestijöillä täytyy olla hyvä ymmärrys toimintamallista ja siihen liittyvistä työkaluista sekä tiedostojen hallinnasta.

4.5 Teknisen viestijän tuottama lisäarvo

Teknisiä viestijöitä tarvitaan IT-alan yrityksissä esimerkiksi alentamaan asiakastuesta aiheutuneita kustannuksia. Teknisten viestijöiden ansiosta yritysten tuotteiden käyttö ja käyttöönotto (*adoption*) voi laajentua.

(31) H4: I would encourage every writer that if you use your elevator and the CEO comes in and says, why are you here? What is your purpose? You should have a 10 second answer: because you notice what **was I reduce your support costs and I help you get your products more widely adopted**. […] your core skill set is that user task analysis. Understanding what documents you need to need to produce. What needs to be and

then how that information needs to be organised, communicated. Clearly ensuring people can actually use the product.

Esimerkin (31) mukaan teknisen viestijän rooli on merkittävä myös liiketoiminnan näkökulmasta. Teknisen viestijän ansioista tuotteiden käyttöönotto helpottuu. Helppokäyttöiset tuotteet tarkoittavat laajempaa asiakaskuntaa ja enemmän liikevaihtoa yritykselle. Esimerkissä mainitaankin, että jos sattuisi osumaan yrityksen toimitusjohtajan kanssa samaan hissiin ja teknisen viestijän pitäisi perustella positionsa tärkeyttä hänelle nk. hissipuheen muodossa, hänen tulisi kertoa, että hän alentaa yrityksen tuotetuesta aiheutuvia kustannuksia tuottamalla laadukasta dokumentaatiota, jolloin käyttäjät löytävät vastauksen dokumentaatiosta ja heidän ei tarvitse olla yhteydessä tuotetukeen, joka on yleensä yrityksille kustannuksiltaan iso resurssi ylläpidettäväksi. Teknisen viestijän työ vaikuttaa myös tuotteiden laajempaan käyttöönottoon. Teknisen viestijän ydinosaamiseen kuuluu myös tiedon organisointi, käyttäjän tehtävien analyysi ja ymmärrys siitä, minkälaisia informaatiotuotteita tulisi tuottaa.

(32) H2: Ja etenkin softamaailmasta sitten mun mielestä oleellinen huomio on sellainen, ja mikä on se kirjoittajan lisäarvo. Eli jos… On nyt sitten käyttöliittymä ja siihen täytyy tehdä ohje. Jos kirjoittaja havaitsi, että hän ei tätä ei muuten nyt tätä ohjetta pysty tekemään mitenkään selkeästi kirjoittamaan, palaute voi olla sille suunnittelijalle, että hei mietipäs nyt vielä vähän. Että se **kirjoittaja kyllä niinku pystyy niin kuin käyttäjänä arvioimaan sitä käyttöliittymää sitten niinku sen loppukäyttäjän näkökulmasta**, että se on mun mielestä se on sellainen added bonus kyllä mitä kirjoittaja tuo tiimiin.

Esimerkissä (32) tuodaan esiin teknisen viestijän lisäarvon käytettävyyden arvioijana. Tekninen viestijä kykenee käyttöohjeen kirjoittamisen ohessa arvioimaan käyttöliittymän käytettävyyttä ja kommunikoimaan suunnittelijalle käytettävyyteen liittyviä haasteita loppukäyttäjän näkökulmasta. Tämä on esimerkin mukaan teknisen viestijän tuoma lisäarvo ohjelmistoalan yrityksissä.

(33) H2: Niinku useinhan testaajat testaa sen että tuote tekee sen mitä sen on tarkoitus tehdä. Mutta sehän ei aina välttämättä ollenkaan sen loppukäyttäjän näkökulmasta ole. Niinku mikä se reitti on sinne loppupisteeseen niin sellainen se. Että, että sillä tavalla kyllä näen **nään**

niinku tän oman työni merkityksellisenä tässä… Tunnen kyllä itse semmoista ammattiylpeyttä.

Tekninen viestijä tuo vahvasti loppukäyttäjän näkökulman tuotekehitykseen, jota esimerkiksi testaustiimi (*quality assurance team*) ei välttämättä mieti testatessaan tuotteen teknistä toimivuutta. Tekninen viestijä voi tuoda mukaan loppukäyttäjän näkökulman ja miettiä käyttäjän käyttäjäpolkuja ohjelmistotuotteessa. Esimerkissä tulee esiin haastateltavan ammattiylpeys ja hän kokee työn merkityksellisenä.

4.6 Organisaation asenteet ja arvostus teknisiä viestijöitä kohtaan

4.6.1 Yleiset asenteet

Haastateltavat mainitsevat tuoteomistajien (*Product Owner*) merkityksen. Tuoteomistajat ovat merkityksellinen sidosryhmä teknisille viestijöille ja he voivat vaikuttaa siihen, miten tekniset viestijät nähdään organisaatiossa ja jakaa viestijöille tietoa tuotejohdon ja liiketoiminnan näkökulmasta.

(34) H1: Ja me ruvettiin sitten alusta pitäen niinku miettimään että OK, mitä me mitä me halutaan tehdä. Me on tehty waterfallia ja se ei oikeastaan toimi hirveän hyvin, jos me kokeiltais Scrumia. OK mitäs Scrum on? Se [Product Owner] piti meille ensin parin tunnin koulutuksen mitä Scrum on, koska **meitä ei koskaan oltu kutsuttu yhteenkään Scrum-koulutukseen aikaisemmin tai mihinkään agileen silleen, että ne ei teidän tarvitse tietää kun te ette ole devaajia te olette vaan kirjoittajia**.

Esimerkin (34) mukaan haastateltavan organisaatiossa ei ollut aiemmin huomioitu teknisiä viestijöitä, kun organisaatiossa oli järjestetty koulutuksia Scrum-toimintamallista. Haastateltava kertoo, että organisaation asenne teknisiä viestijöitä kohtaan erosi tässä suhteessa asenteesta ohjelmistokehittäjiä kohtaan. Teknisiä viestijöitä ei mielletty olevan samalla viivalla tai pidetty samanarvoisina tiimin jäseninä.

(35) H3: I think they [Product Owners] kind of **they get the bigger picture** 'cause a lot of times I think team members don't necessarily value documentation even though it's a requirement and deliverable.

Esimerkin (35) haastateltavan mukaan tuoteomistajilla on myös iso rooli kokonaiskuvan kommunikoinnissa, ja he yleensä myös ymmärtävät dokumentaation merkityksen. Haastateltavan mukaan hänen kokemuksensa on, että tiimin jäsenet (ohjelmistokehittäjät) eivät välttämättä arvosta dokumentaation tuottamista, vaikka se on vaatimus ja toimitettava osa tuotetta. Tuoteomistajat voivat nostaa dokumentaation tärkeyttä ja arvostusta organisaatiossa.

(36) H5: **I very clearly felt that as a technical writer, that your role is somehow less than a very senior developer in that company,** so you. Appreciation of course, like if you are doing your job very well. If you have good people working with you, they will appreciate, but I think a technical writer is definitely our perception, social perception and industrial perceptions are it's less important than a developer. […] **It's a hierarchy I would say.**

Esimerkin (36) mukaan haastateltava oli kokenut organisaatiossa, että organisaation yleinen asenne teknisiä viestijöitä kohtaan oli sellainen, että ammattia arvostettiin vähemmän ohjelmistokehittäjiin verrattuna. Haastateltava kokee, että teknisien viestijöiden koetaan olevan organisaation hierarkiassa alempana kuin ohjelmistokehittäjien. Rosselot-Merritt (2020) toteaakin, että tekniset viestijät tarvitsevat enemmän koulutusta organisaation dynamiikasta, ja siitä miten osoittaa ammatin arvoa työpaikalla. Clearyn (2021, s. 188) mukaan teknisen viestinnän alalla sekä tutkijat että ammatinharjoittajat ovat olleet huolissaan ammatin arvostuksesta ja statuksesta organisaatioissa. Tästä syystä teknisen viestinnän alan taloudellisen, sosiaalisen ja kulttuurisen pääoman kasvattaminen olisi strategisesti tärkeää (Cleary, 2021, s. 188). Rosselot-Merrittin (2020) mukaan tekninen viestintä ammattina jatkaa vielä kehittymistään ja tämän ammattimaistumisen (*professionalization*) myötä tekniset viestijät voivat saavuttaa valtaa organisaatioissa. Rosselot-Merritt (2020) argumentoi, että ammattimaistumisen myötä saavutetaan legimiteettiä, joka johtaa arvostuksen ja statuksen kasvamiseen. Hänen mukaansa tämän prosessin myötä teknisestä viestijästä voi tulla arvostettu toimija, joka voi luoda ja vaikuttaa organisaation sisäisiin prosesseihin.

4.6.2 Ohjelmistokehittäjien asenteet ja viestintä

Ohjelmistokehittäjät ovat yksi tärkeimmistä sidosryhmistä dokumentaation laatimisessa IT-alan yrityksissä. Ohjelmistokehittäjät toimivat usein teknisille viestijöille tärkeänä informaation lähteenä.

(37) H3: I've also had a couple of **developers who are a little bit gatekeepy about their code**. Is how I would describe it like they don't want people to understand too much. Or they want to obfuscate things a little bit. And as a documentation person, obviously your whole job is to explain things. It's to make it as clear as possible, and so sometimes it'll be like. Well, what do you mean? it just translates the information. How does it do that? And they'll be like, well, it's too complicated to understand like to to explain and you're like, well, that's that's impossible because you obviously did it and then they get kind of offended. You know, I don't mean to offend you, I'm not saying it wasn't hard to do, but let's make it easy for people to understand 'cause people around the world need to use this product. So sometimes you encounter those personalities.

Esimerkissä (37) haastateltava kertoi kokemuksistaan ohjelmistokehittäjien kanssa työskentelystä. Haastateltavalla oli kokemuksia ohjelmistokehittäjistä, jotka eivät halunneet jakaa tietoa tekniselle viestijälle, tai eivät ymmärtäneet dokumentaation laatimista pakollisena tuotteeseen kuuluvana osana. Esimerkissä kerrotaan tilanteesta, jossa ohjelmistokehittäjä ei ole ollut halukas jakamaan tietoa tai selittämään koodiaan viestijälle, jonka tehtävänä olisi ymmärtää toiminnallisuuden tarkoitus dokumentaation kirjoittamiseksi. Ohjelmistokehittäjä on ollut sitä mieltä, että hänen tuotostaan on liian vaikea selittää tai ymmärtää, ja tällaiset tilanteet saattavat aiheuttaa konflikteja kehittäjän ja teknisen viestijän välille.

(38) H3: There is maybe that **small subset of developers who are either like they don't want you interacting with their stuff, or they don't want to explain their stuff**. I see that personality type sometimes and I'm like… Well, we have to do something we have to produce documentation work with me here.[…] usually it's the kind of people who don't think documentation is important. Like I've definitely had that in the past. We're like well, people are buying our code, our product. **They don't care**

> **about documentation. I'm like, well, that's not true, and in fact, many of the government people who are purchasing our product need that documentation for it.** To even be able to. Be sold so we have to produce this it's. It's an artifact that has to be made like this. This isn't like a fun optional thing we make.

Jotkut ohjelmistokehittäjät eivät ymmärtäneet dokumentaation tärkeyttä ja eivät ymmärtäneet teknisen viestijän tarvetta ymmärtää koodia/tuotetta, jotta siitä voidaan kirjoittaa käyttäjälle ymmärrettävää dokumentaatiota. Haastateltava kuvailee ohjelmistokehittäjän asennetta termillä ”*gatekeepy*”, jolla viitataan ohjelmistokehittäjän haluttomuuteen jakaa tietoa koodista tai muita dokumentaation tuottamiseen tarvittavia yksityiskohtia, sillä dokumentaation tärkeyttä tai teknisen viestijän työtehtäviä ei ymmärretä riittävästi. Esimerkissä kuvaillaan, että ohjelmistokehittäjän asenne dokumentaatiotyötä kohtaan on hankala ja dokumentaatiota ei pidetä tärkeänä. Haastateltava on joutunut tilanteisiin, jossa hän on joutunut puolustelemaan dokumentaation tuottamisen tärkeyttä ja roolia osana tuotetta. Hän mainitsee dokumentaation pakollisuuden esimerkiksi osana julkishallinnon projekteja, jolloin dokumentaation tuottaminen ei ole vapaaehtoista.

> (39) H1: Niin tosiaan se **suurin haaste kirjoittajan kannalta on saada se deve-puoli ymmärtämään sun homman, että miten se sun hommasi toimii** ja että mikä kaikki siihen vaikuttaa koska ne. Kun deve-puolen ihmiset tuppaa yleensä ajattelemaan vaan sitä koodia ja parhaimmillaan vaan niinku Backend-koodia, että sitä unohtaa, jos tuotteessa on UI.

Esimerkissä (39) kerrotaan, että suurin haaste on saada ohjelmistokehittäjät ymmärtämään dokumentaation laatimisen realiteetit. Nopeasyklisessä ketterässä toimintaympäristössä toimiminen saattaa aiheuttaa aikataulupaineita tekniselle viestijälle, kun tuotteen julkaisuaikataulu on yleensä tiukka, ja ohjelmistokehittäjät työskentelevät nopeatempoisesti sprintin viimeiseen päivään saakka. Teknisen viestijän työ vaatii läheistä viestintää ohjelmistokehittäjien kanssa, jotta dokumentaatio saadaan kirjoitettua samassa aikataulussa koodin kanssa.

> (40) H1: Ja tän niinku **tän kommunikointi silleen että ymmärrättekö että paljonko tähän pitää varata aikaa että ette te voi heittää tätä minulle viimeisenä päivänä ennen rellua, tässä menee vähän pidempään**, niin

> tämä on tämä on selkeä haaste näissä koska käytännössä hyvin usein sillä puolella, joka pyörittää sitä sitä Agile prosessia tai scrummia tai mitä metodia käytetäänkään, niin **niiltä puuttuu sitten puolestaan se teknisen dokumentaation ymmärrys, että mitä mitä kaikkea siellä pitää ottaa huomioon, eikä ne toisaalta niitä ei myöskään kiinnosta**…[…] Toinen on osittain tähän liittyen on myös se, että mikä on iso ja mikä on pieni muutos. Eli se mikä saattaa olla tosi **pieni muutos koodissa voi olla ihan älytön muutos dokumentaatiossa**. Et silleen et OK te vaihdatte bittiä tossa ja mä kirjoitan puolet dokumentaatiosta uusiksi.

Esimerkin (40) mukaan ohjelmistokehittäjiltä puuttuu joskus ymmärrys teknisen viestijän vaatimasta aikaresurssista, jolloin viimeisenä päivänä ennen tuotteen julkaisua (*release*) saattaa tulla jokin suuritöinen dokumentaatiotyö, jolloin sitä on mahdotonta toteuttaa samassa aikataulussa ohjelmistotuotantosyklin kanssa. Haastateltava mainitsee myös seikan, että pieni muutos koodissa voi tarkoittaa suuritöistä muutosta dokumentaatiossa. Tällaiset asiat vaativat tarkkaa viestintää ohjelmistokehittäjän ja teknisen viestijän välillä.

4.7 Työnkuvan muuttuminen

Teknisen viestijän työnkuva on muuttunut teknologioiden kehittyessä ja projektihallintamenetelmien yleistyessä. Digitalisaation myötä viestijöiltä vaaditaan aiempaa enemmän teknistä kyvykkyyttä.

> (41) H3: And there's kind of an interesting shift there where I'm like, oh **my job description now seems to be a little bit more computer science focused** like I have to know a couple different markup languages. I have to understand GitLab. I had to use the command prompt. You know I've had to learn how to work with Ant and you know Apache Maven. You have these different sort of like code processing tools.

Esimerkissä (41) haastateltava kertoo työnkuvan muuttuneen siten, että nykyään teknisen viestijän (ainakin IT-alan yrityksissä) olisi hyvä osata muutamaa merkintäkieltä ja ymmärtää versionhallintaan liittyvien työkalujen toimintaa (esimerkiksi GitLab). Haastateltava mainitsee myös opetelleensa ohjelmistokoodin työskentelyyn käytettäviä työkaluja kuten Antin ja Apache Mavenin, jotka ovat ohjelmistotuotteen koosteen

(*software build*) automatisointiin käytettäviä työkaluja. Nissilän ja Nuopposen (2018, s. 179) tutkimuksen mukaan teknisen viestinnän alan työpaikkailmoituksissa edellytetään teknistä osaamista ja ohjelmistoihin liittyvää ymmärrystä.

> (42) H3: Picking and choosing your branches. Having a master branch, all that stuff sort of facilitates good documentation, I think. And you know, we try to strive for that a lot. At my company right now, we actually still have teams who are producing things using FrameMaker, and it's all **stuck on one guys computer. It's not backed up anywhere**, and it's like those kind of moments for Doc people are very like alarming because it's like well, what if something happens to your computer. What are we going to do with the doc? We don't have that anywhere. I'd have to recreate from the ground up and they're just kind of like… **So we were really striving to have that sort of like treat your docs just as important as your code, have them live together in a system. Treat it like that.** So that's my definition of it. That's kind of how we hope to do it.

Esimerkissä (42) haastateltava vertaa uuttaa *Docs as coden* mukaista työskentelytapaa aikaisemmin suosittuun sisällöntuotannon Adoben FrameMaker-työkaluun, joka oli asennettuna aina jokaisen käyttäjän työasemalla, ja ohjelmisto ei ollut sidottu versionhallintaan, jolloin kaikella dokumentaatiotyöllä oli vaarana hävitä, jos työasema menisi rikki, sillä varmuuskopioita ei ollut saatavana. Tämänkaltainen tilanne olisi dokumentaation kannalta erittäin vahingollinen, sillä pahimmassa tapauksessa työn joutuisi aloittamaan uudelleen sen hävitessä. Versionhallintajärjestelmät tarjoavat aina varmuuskopion dokumentista. Esimerkissä kerrotaan, että haastateltavan organisaatiossa pyritään kohtelemaan dokumentaatiota samanarvoisena kuin ohjelmistokoodia ja niitä ylläpidetään tästä syystä samassa järjestelmässä. Clearyn (2021, s. 105) FrameMaker-työkalua pidettiin teknisen viestinnän alan standardina sisällöntuotantoon muutama vuosikymmen sitten, mutta nykypäivänä alan työkaluja on enemmän eikä varsinaista standardia ole enää olemassa.

> (43) H1: tosiaan silloin meillä ei ollut varsinaisesti prosessia. **Me tehtiin ehkä lähimpänä waterfallia** ja silleen, että kaikki meni seuraavaan, sulla oli deadline, jota vasten työskenneltiin, että tossa pitää olla kama valmista ja tässä on noi tuotteet tai noi prototyypit, softat mitä ne meinaa suunnitella tänne ja täältä täältä tulee tätä infoa, että mitä muutoksia on suunniteltu, päivitä toi kama sillain että se vastaa sitä ja tuolla on deadline

> ja meillä ei ollut siis tosiaan **ei ollut oikeastaan hirveästi mitään seurantamiittejä. Ei ollut Jiraa, ei ollut tämmöisiä mitään seremonioita**, jotka sitten taas näissä kahdessa viimeisessä työpaikassa on ollut hyvin keskeisessä roolissa, niin siinä mielessä tämä ensimmäinen toimi ehkä lähinnä kontrastina, historiatietona siitä, että minkälaista tai mikä, mikä ainakin se tilanne silloin oli Suomessa. Miten se nähtiin, koska **mun kutina on tämä aika harvassa paikassa 2010-luvun alussa tehtiin teknisen viestinnän puolella minkäänlaista metodia käyttäen mitään.**

Esimerkin (43) mukaan alalla on tapahtunut suuri muutos viimeisen kymmenen vuoden aikana. Tänä aikana haastateltavan mukaan on siirrytty vanhanaikaisesta vesiputousmallista ketteriin menetelmiin myös teknisen viestinnän alalla. Haastateltava oli projektin aluksi saanut tiedot dokumentoitavasta ohjelmistotuotteesta ja päivämäärän, jolloin dokumentaation tulisi olla valmis. Haastateltavan mukaan kymmenen vuotta sitten teknisillä viestijöillä ei ollut käytössä työnhallintaan tarkoitettuja järjestelmiä, kuten Jiraa, joka luo läpinäkyvyyttä työn hallintaan ja parantaa töiden seurattavuutta. Esimerkin mukaan myös projektin aikainen viestintä osapuolten kesken puuttui. Viestintä on ketterän kehityksen seremonioiden ja projektin aikaisten tapaamisten myötä parantunut.

4.8 Analyysin yhteenveto

Ketterien menetelmien hyödyntämisestä on monia hyötyjä myös tekniselle viestijälle. Sellaisenaan monet ketterät menetelmät eivät kuitenkaan sovellu suoraan tekniseen viestintään, vaan ne ovat hyödyllisiä, kun niistä hyödynnetään sopivia elementtejä. Sopivia elementtejä ovat esimerkiksi ketterien menetelmien seremoniat, kuten viestintää helpottavat päivittäiset Scrum-palaverit tai työlistan työstöön suunnatut *backlog refinement* -tapaamiset. Myös ketterien menetelmien työaika-arvioiden koettiin parantavan viestintää esimerkiksi tuoteomistajien suuntaan, sillä työaika-arvioita käyttämällä saatiin selville tiimin työskentelynopeus yksittäisen sprintin aikana, mikä mahdollisti työtehtävien paremman priorisoinnin ja organisoinnin. Parhaimmillaan viestintä paranee organisaatiossa, kun käytetään samoja toimintamalleja samassa

työympäristössä toimivien sidosryhmien, kuten tuotejohdon tai ohjelmistokehittäjien, kesken. Samalla teknisen viestijän arvostus paranee ja tekninen viestijä on osa tiimiä ja dokumentaatio nähdään osana tuotetta, eikä erillisenä palasena.

Docs as code -menetelmän on todettu sopivan erityisesti ohjelmistokehittäjille suunnatun dokumentaation tuottamiseen. Kun tekninen viestijä tuottaa dokumentaatiota samassa ympäristössä ohjelmistokehittäjien kanssa, se tuottaa viestijälle samalla ymmärrystä kohdeyleisöstä. Tekninen viestijä voi myös helpommin saada katselmointien yhteydessä palautetta ja korjausehdotuksia ohjelmistokehittäjiltä, mikä johtaa parempaan dokumentaation laatuun. Aineiston mukaan dokumenttien katselmointi vetopyyntöjen yhteydessä tapahtuu yleensä myös nopeammin ja muutoksien toteuttaminen versionhallintaohjelmistoissa olevien ominaisuuksien kautta onnistuu helposti. Samassa ympäristössä ohjelmistokehittäjien kanssa toimiminen tuo tekniselle viestijälle uskottavuutta ja tuo teknisen viestijän työn näkyväksi.

Ketterien menetelmien hyödyntämisen haasteena tai jopa esteenä koetaan teknisen viestinnän alalla yleinen alihankinta-asetelma. Alihankintatilanteissa tilaaja-asiakas saattaa olla halumaton jakamaan tuotekehitykseen liittyvää tietoa vastoin ketterän toimintamallin käytänteitä, jossa tiedon jakaminen on keskeistä. Työn hektisyys ja pienissä palasissa työskentely saattavat tehdä kokonaiskuvan hahmottamisen hankalaksi jatkuvasti muuttuvassa työympäristössä. Kansainvälisissä organisaatioissa työskentelevät tekniset viestijät kokivat päivittäisiin palavereihin osallistumisen hankalaksi, jos tiimin jäsenet olivat levittäytyneet ympäri maailmaa, jolloin eri aikavyöhykkeillä työskentelevät tekniset viestijät eivät aina päässeet osallistumaan tuotetiimien päivittäisiin tapaamisiin.

Docs as code -menetelmän haasteena voidaan aineiston mukaan pitää sitä, että ohjelmistokehittäjien työkaluja ei ole optimoitu dokumentaation kirjoittamiseen, ja esimerkiksi lokalisoinnin hallinta on monimutkaisempaa. Menetelmän mukainen työskentely vaatii tekniseltä viestijältä myös perehtymistä ja koulutusta. Menetelmien

hyödyntäminen dokumentaation kirjoittamiseen saattaa myös tarkoittaa sitä, että tekninen viestijä ei ole päivittäisessä työssä niin tehokas kuin esimerkiksi sisällönhallintajärjestelmiä käyttäessään, sillä ohjelmistokehittäjien työkaluista puuttuu sisällönhallintajärjestelmien teknisen dokumentaation laatimiseen optimoituja ominaisuuksia, kuten sisällön uudelleenkäyttöön suunnattuja ominaisuuksia. *Docs as code* -menetelmän avulla työskentely tuo kuitenkin tekniselle viestijälle lisää uskottavuutta ohjelmistokehittäjien keskuudessa, jolloin yhteistyö ohjelmistokehittäjien ja teknisten viestijöiden välillä on sujuvampaa.

Aineistosta kävi myös ilmi muun organisaation huonot asenteet dokumentaatiota ja teknistä viestijää kohtaan. Haastateltavien vastauksissa toistui hankalat asenteet viestijöitä kohtaan, jos asenne dokumentaation tuottamista kohtaan ylipäätään oli huono, se heijastui huonona asenteena myös teknisiin viestijöihin, sillä joskus ohjelmistokehittäjät olivat haluttomia selittämään kirjoittamaansa koodia. Teknisten viestijöiden olisi tärkeä jatkaa organisaatioissa sisäistä strategiatyötä ammatin nostamiseksi ja nostaa työnkuvan tunnettuutta.

Virtaluodon (2015) tutkimustuloksiin verrattuna tutkimukseni osoittaa, että tekniset viestijät ovat nyt aiempaa paremmin integroituneet osaksi tuotekehitystiimiä. Virtaluoto (2015, s. 68) esittääkin osana väitöskirjansa tutkimustuloksia, että tekniset viestijät tarvitsevat vahvemman yhteyden tuotekehitystiimiin laadukkaan dokumentaation tuottamiseksi. Virtaluoto (2015, s. 68) erittelee tutkimustuloksissaan, että tekniset viestijät eivät hänen tutkimuksensa mukaan olleet osana tuotekehitystiimejä, ja että tiimin rakennetta tulisi muuttaa siten, että tekniset viestijät pääsisivät helpommin käsiksi tuotetietoon ja viestintä alan substanssiasiantuntijoiden (*SME*) kanssa olisi helpompaa. Tutkimukseni osoittaa, että ohjelmistotuotannon menetelmien käyttö on poistanut Virtaluodon (2015, s. 68) mainitsemia haasteita teknisiltä viestijöiltä, sillä tuotekehitystiimissä toimiminen tuo tekniselle viestijälle ajankohtaista tuotetietoa projektien aikana ja viestinnän kynnys esimerkiksi tuoteomistajan kanssa madaltuu. Virtaluoto (2015, s. 68) esittää huolen tässäkin tutkimuksessa mainitusta alihankinta-

asetelmasta, joka saattaa osaltaan estää teknistä viestijää osallistumasta samalla tavalla tuotekehitystiimin toimintaan, sillä työn ulkoistaminen tuo haasteita avoimeen tiedon jakamiseen. Tutkimukseeni osallistuneista kuitenkin lähes kaikki olivat haastatteluhetkellä teknisen viestinnän *in-house* -roolissa. Kaiken kaikkiaan tutkimukseni luo positiivisempaa kuvaa teknisen viestinnän nykytilasta Virtaluodon väitöskirjaan verrattuna. Tutkimukseni haastateltavat suhtautuivat positiivisesti ja intohimoisesti teknisen viestinnän alan tulevaisuuteen ja alan käytänteiden kehittämiseen, kokivat työnsä merkityksellisenä ja olevansa arvostettuja kollegoita työpaikoillaan.

Luvussa 2.2 esittelemäni Hackosin (2009) ja Kisterin (2016) kehitysmallit on suunniteltu ainoastaan teknisen dokumentaation tuottamiseen, ja vaikka ne sisältävät joitakin elementtejä ketteristä menetelmistä, ne eivät ota kantaa siihen miten teknisen dokumentaation tuottaminen on yhteydessä laajempaan tuotekehitystiimin prosessiin, jossa ohjelmistoa kehitetään inkrementaalisesti ja iteratiivisesti. Mikäli tutkielmassa esitellyt ketterät menetelmät yhdistettäisiin Hackosin (2009) ja Kisterin (2016) kaltaisiin malleihin, voitaisiin laadukkaan teknisen dokumentaation tuottamista tehostaa ohjelmistokehitysprojekteissa.

5 Johtopäätökset

Tutkimuksen tarkoituksena oli selvittää, miten ohjelmistotuotannon menetelmiä hyödynnetään tutkimukseen osallistuneiden teknisten viestijöiden työpaikoilla IT-alan yrityksissä ja mitä vaikutuksia menetelmien käytöllä on teknisen dokumentaation tuottamiseen. Tutkimuksen aineisto kerättiin haastattelemalla IT-alalla työskenteleviä teknisiä viestijöitä. Tavoitteen saavuttamiseksi analysoin tutkimusaineiston aineistolähtöisellä sisällönanalyysillä.

Tutkielmaa aloittaessani oletin, että ohjelmistotuotantoon luodut käytänteet asettavat haasteita tekniselle viestijälle, sillä niitä ei ole suoraan luotu käytettäväksi teknisen viestinnän työtehtävissä, vaan ne ovat alun perin syntyneet viitekehykseksi nopeatempoiseen ohjelmistokehityksen maailmaan, jossa työtahdin ja aikataulun sanelevat ohjelmistotuotteen kehitys ja mahdollisesti muuttuvat vaatimukset. Seuraavaksi käyn läpi tutkimustuloksia tutkimuskysymyksittäin.

Ensimmäinen tutkimuskysymys: Miten ohjelmistotuotannon menetelmiä hyödynnetään tutkimukseen osallistuneiden teknisten viestijöiden työpaikoilla?

Tutkimukseen osallistuneet tekniset viestijät toimivat pääasiassa samassa ketterässä viitekehyksessä tuotekehitystiimien kanssa ja osallistuivat esimerkiksi tuotekehitystiimien ketterien menetelmien mukaisiin seremonioihin, kuten päivittäisiin palavereihin, tuotekehitystiimien kanssa. Tekniset viestijät toimivat läheisessä yhteistyössä ohjelmistokehittäjien ja tuotejohdon kanssa. Teknisen viestijän työn luonne oli kaikkien haastateltavien organisaatioiden tapauksessa palvella useita tuotekehitystiimejä samanaikaisesti, jolloin tekninen viestijä oli niin kutsuttu *shared service* useille tuotekehitystiimeille. Tämän kaltainen asetelma pakottaa teknisen viestijän joustamaan ja vaihtamaan kontekstia päivittäisessä työssään ja monen tuotekehitystiimien ketterän kehityksen mukaisiin tapaamisiin saattaa viedä paljon aikaa teknisen viestijän työpäivästä. Tutkimuksen mukaan kuitenkin teknisen viestijän

työtehtävien priorisointi ja organisointi on helpompaa ketterien menetelmien käytössä olevien työnhallintaan tarkoitettujen ohjelmistojen avulla.

Viidestä haastateltavasta kolme teknistä viestijää työskenteli *Docs as code* -menetelmää hyödyntäen ohjelmistokehityksen työkaluja, kuten versionhallintaa, päivittäisessä työssään. *Docs as code* menetelmää hyödyntävät kirjoittivat dokumentaation esimerkiksi versionhallintaan kytketyn XML-pohjaisen sisällönhallintajärjestelmän tai käyttäen koodieditoria ja kevyitä merkintäkieliä kuten Markdownia. Teknisen viestijän tuottama sisältö katselmoitiin kahden haastateltavan tapauksessa GitHub -järjestelmässä vetopyynnön yhteydessä, jonka jälkeen käynnistyivät dokumentin julkaisuun tähtäävät mekanismit.

Toinen tutkimuskysymys: Mitä vaikutuksia ohjelmistotuotannon menetelmien käytöllä on teknisen dokumentaation tuottamiseen?

Tutkimuksessa selvisi, että sekä ketterien menetelmien että *Docs as code* menetelmän hyödyntäminen tuo tekniselle viestijälle monia hyötyä ohjelmistotuotantoyrityksissä.

Kuvio 9. Ohjelmistotuotannon menetelmien hyödyt.

Kuviossa 9 esitetään tutkimuksen tuloksena ohjelmistotuotannon menetelmien hyödyt tekniselle viestijälle. Menetelmien käyttö tuo teknisen viestijän osaksi tuotekehitystiimiä, mikä johtaa parempaan dokumentaation laatuun ja viestinnän helpottumiseen. Yhteistoiminnallinen kirjoittaminen ohjelmistokehittäjien ja teknisien viestijöiden välillä helpottuu ja katselmoinnit ovat helpompia ja nopeampia. Teknisen viestijän työmäärän arviointi ja priorisointi helpottuu sekä muille tiimin jäsenille muodostuu näkyvyyttä teknisen viestijän työpanokseen.

Tutkimusta voisi laajentaa koskemaan laajemmin IT-alan yritysten tuotejohtoa organisaatioiden käytänteiden ja asenteiden selvittämiseksi. Laajentamismahdollisuus

olisi myös kattavampi tutkimus, jossa selvitetään tarkemmin parhaita teknisen viestinnän alan käytänteitä ketterässä toimintaympäristössä toimimiseen tai teknisen viestinnän roolia esimerkiksi Scrum-mallissa, sillä nykyisellään Scrum ei määrittele teknisen viestijän roolia, vaikka muut tiimin roolit, kuten Scrum-mestarin, tuoteomistajan ja ohjelmistokehittäjän roolit, ovat tarkkaan määritelty. Ketteristä menetelmistä tulisi kehittää laajempi malli, joka ottaa huomioon myös muita ohjelmistokehitykseen liittyviä osa-alueita, kuten käyttöliittymäsuunnittelun tai teknisen viestinnän.

Tutkimukseni rajoitteena oli haastateltavien pieni määrä. Toisaalta koin, että pienelläkin otoksella pystyin saavuttamaan nk. saturaatiopisteen ja saman suuntaiset vastaukset toistuivat useilla haastateltavilla. Tutkimukseni tuo ajankohtaista tietoa IT-alalla työskentelevien teknisten viestijöiden toiminnasta, kipukohdista ja toisaalta myös toimivista käytänteistä. Ohjelmistokehitykseen suunnatuista ketteristä menetelmistä voidaan poimia teknisille viestijöille soveltuvia elementtejä, jotka voivat parantaa esimerkiksi työn laatua ja yhteistyötä organisaation muiden sidosryhmien kanssa. Teknisen viestijän kannattaa asemoida itsensä niin lähelle tuotekehitystiimiä kuin mahdollista ja osallistua tiimin työskentelyyn sekä ketterän kehityksen seremonioihin, kuten sprintin suunnittelupalavereihin, tehtävälistan työstöön ja päivittäisiin palavereihin. Tämä helpottaa tiedon jakamista ja integroi teknisen viestijän osaksi muuta tiimiä. Samojen toimintamallien hyödyntäminen siirtää tekniset viestijät aiempaa työnkuvaan verrattuna erillisestä siilostaan paremmin osaksi muuta tiimiä ja nostaa teknisen viestijän ammatin statusta tasa-arvoiseksi kollegaksi ja teknisen viestinnän alan tunnettuutta.

Lähteet

Agile Manifesto. (2001). Ketterän ohjelmistokehityksen julistus. Noudettu 2021-10-13 osoitteesta: https://agilemanifesto.org/iso/fi/manifesto.html

Andersen, R. (2014). Rhetorical Work in the Age of Content Management: Implication for the Field of Technical communication. *Journal of Business and Technical Communication*.28(2),115-157. https://doi.org/10.1177/1050651913513904

Atlassian. (2023a). *What is DevOps?* DevOps. Noudettu 13.4.2023 osoitteesta: https://www.atlassian.com/devops

Atlassian. (2023b). *Branching strategy: a path to greatness*. Branching. Noudettu 14.4.2023 osoitteesta: https://www.atlassian.com/agile/software-development/branching

Atlassian. (2023c). Agile epics: definition, examples, and templates. Noudettu 13.4.2023 osoitteesta: https://www.atlassian.com/agile/project-management/epics

Cohen, D., Lindvall, M., Costa, P. (2004). An Introduction to Agile Methods. Advances in Computers. *Elsevier.* Volume 62, p. 1-66. https://doi.org/10.1016/S0065-2458(03)62001-2

Clark, D. & Andersen, A. (2005). Renegotiating with Technology: Training Towards More Sustainable Technical Communication. *Technical Communication*. 52. 208-301.

Cleary, Y. (2021). *The Profession and Practice of Technical Communication.* Routledge.

Etter, A.(2016). *Modern Technical Writing: An Introduction to Software Documentation.* Amazon Digital Services LLC.

Gentle, A. (2017). *Docs like Code.* Just Write Click.

Gustavsson, T. & Hallin, A. (2014). Rethinking dichotomization: A critical perspective on the use of “hard” and “soft” in project management research. *International Journal of Project Management*. Volume 32, Issue 4, Pages 568-577.

Hackos, J. (2006). *Information Development: Managing Your Documentation Projects, Portfolio and People*. John Wiley & Sons.

Haikala, I. & Mikkola, T. (2011). *Ohjelmistotuotannon käytännöt.* Talentum Media Oy.

Highsmith, J. (2009). *Agile Project Management*. Addison-Wesley Educational Publishers Inc.

Hirsjärvi, S. & Hurme, H. (2001). *Tutkimushaastattelu: Teemahaastattelun teoria ja käytäntö*. Yliopistopaino Helsinki.

Hirsjärvi, S., Remes P., Sajavaara, P. (2010). *Tutki ja kirjoita*. Kustannusosakeyhtiö Tammi.

ISO/IEC/IEEE international standard. (2018). Systems and software engineering -- developing user information in an agile environment. (26515:2018).

Jitianu, A. (2021). Improving your documentation project with "Docs as Code". *TC World magazine, July 2021*. Tcworld GMBH.

Kister, T. (2016). Improving the Information Development Process: A Refined Iterative Development Model. *Technical Communication*, Volume 63, Number 3, pp. 186-211(26).

Markdown Guide. (2023). Basic syntax. Noudettu 18.3.2023 osoitteesta: https://www.markdownguide.org/basic-syntax/#html

Nissilä, N. & Nuopponen A. (2018). Teknisen viestinnän taidot ja tehtävät työpaikkailmoituksissa. Teoksessa L. Kääntä, M. Enell-Nilsson, & N. Keng. (toim.), *Työelämän viestintä: VAKKI-symposiumi XXXVIII*. S. 225-232.

Project Management Institute. (2021). *A Guide to the Project Management Body of Knowledge (PMBOK Guide).* Seventh Edition and the Standard for Project Management. Project Management Institute.

Rosselot-Merrit, J. (2020). Fertile grounds: What interview of working professionals can tell us about perceptions of technical communication and the viability of technical communication as a field. *Technical Communication*, Volume 67(1), 38-62.

Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques. *Proceedings of the 9th international conference on Software Engineering*. IEEE Computer Society Press, 328-338.

Salmela, E. & Isohella, S. (2021). YouTube-videoista helppokäyttöisiin verkkosivuihin – käyttäjän ymmärtäminen teknisen viestinnän ytimenä. Teoksessa T. Waaramaa, L. Kääntä, M. Koskela & S. Isohella (toim.), *Monialainen viestintä: puheesta peleihin* (s. 1–99). Vaasan yliopisto.

Sarajärvi, A. & Tuomi, J. (2018). *Laadullinen tutkimus ja sisällönanalyysi: Uudistettu laitos*. Kustannusosakeyhtiö Tammi.

Sanastokeskus. (2023). Ketterän ohjelmistokehityksen määritelmä. Noudettu 18.3.2023 osoitteesta: https://sanastokeskus.fi/tsk/fi/termitalkoot/hakemistot-267.html?page=get_id&id=ID744&vocabulary_code=TSKTT

Stare, A. (2013). Agile Project Management – A future approach to the management of projects? *Dynamic Relationships Management Journal*, 44-54.

Sommerville, I. (2016). *Software Engineering: Tenth edition*. Pearson.

Suojanen, T. (2018). *Suomalaista teknistä viestintää: Sinä- ja me-asenne kotoistamisstrategioina kodinkoneidein käyttöohjeissa 1945-1995* [väitöskirja, Tampereen yliopisto]. Tampere University Press.

Thomchick, R. (2019). Improving access to API documentation for developers with Docs-as-Code-as-a-service. *Proceedings of the Association for Information Science and Technology*. https://doi.org/10.1002/pra2.2018.14505501171

Uikey, N., Ugrasen, S. & Ramani, K. (2011). A Documented Approach in Agile Software Development. *International Journal of Software engineering*, Volume 2.

Virtaluoto, J. (2015). *Technical communication as an activity system: A practitioner's perspective* [väitöskirja, Oulun yliopisto]. University of Oulu.

Virtaluoto, J., Suojanen, T. & Isohella, S. (2018). Minimalismiin pohjautuvan dokumentointiprosessimallin kehittäminen. VAKKI Publications, 9, 187-200.

Liitteet

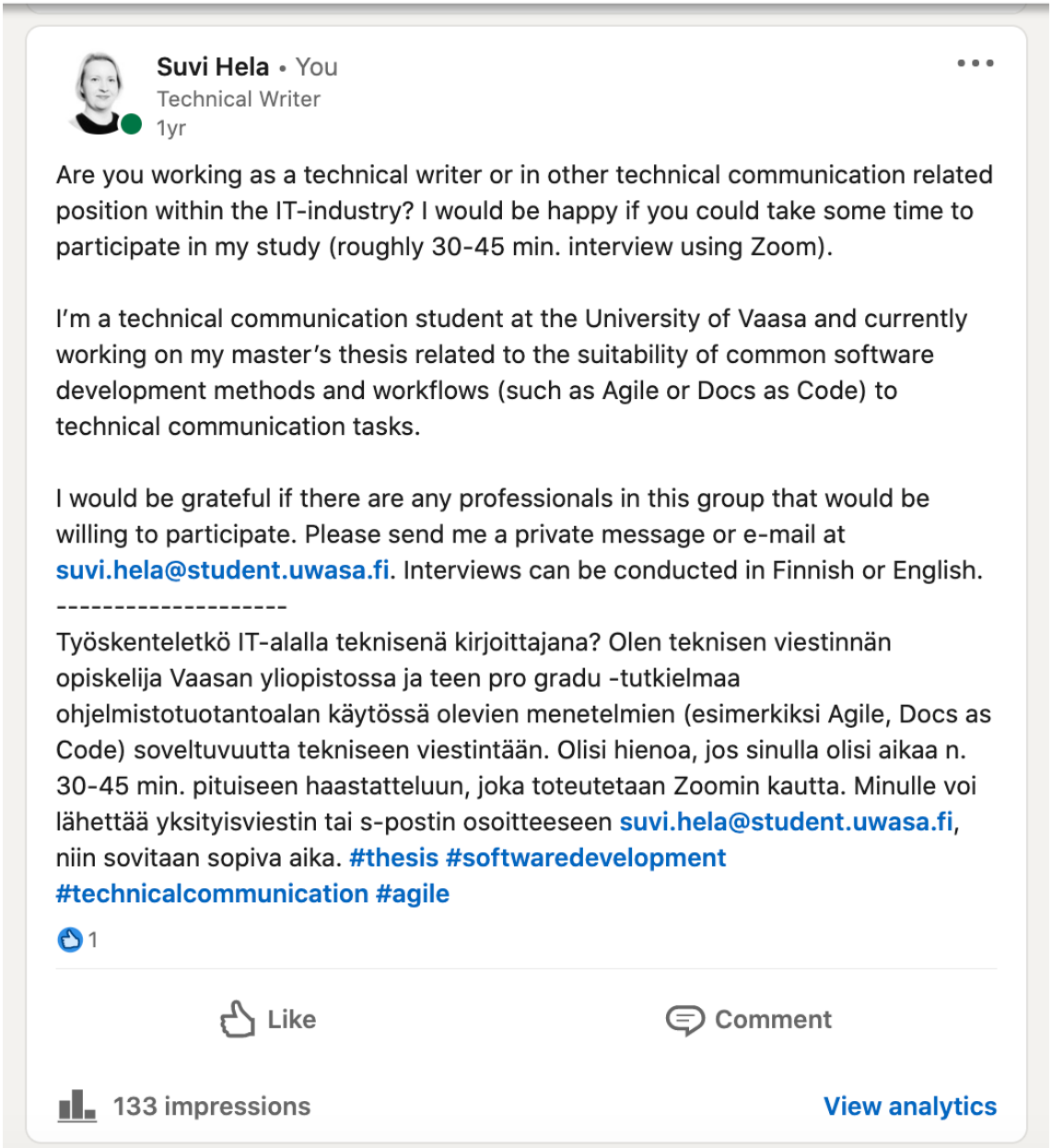
Liite 1. Tutkimuksen haastattelukysymykset suomeksi

Tähdellä (* merkityt kysymykset poimittu Virtaluodon (2015) tutkimuksesta.

Teema	**Haastattelukysymykset**
Koulutustausta, työhistoria, tuotteet ja dokumentit	
1.	Ikä (*
2.	Koulutustausta (valmistumisvuosi, pää- ja sivuaineet, ammatillinen koulutus) (*
3	Työhistoria; kuinka monta vuotta nykyisessä työssä? (*
4.	Mitä muita IT-alan töitä olet tehnyt? (*
5.	Millaisten tuotteiden parissa toimit? (*
Organisaatio, tiimi:	
6.	Kuulutko johonkin tiimiin, mihin? (*
7.	Mikä on tiimin sijainti organisaatiossa – oma osasto, markkinointi, testaus, tuotekehitys? (*
Prosessi:	
8.	Minkälainen prosessi dokumentaation tuottaminen on?
Työkalut:	
9.	Mitä työkalua työpaikallasi käytetään sisällöntuotantoon?
10.	Käytetäänkö jotakin merkintäkieltä? (esimerkiksi Markdown)
Menetelmät:	
11.	Minkälaisia menetelmiä yrityksessä on käytössä ohjelmistotuotantoon (Agile, DevOps…)?
12.	Minkälaisia menetelmiä on käytössä dokumentaation tuottamiseen (Docs as code)?
13.	Miten koet menetelmien (Agile, Docs as code) sopivan teknisen dokumentaation tuottamiseen?
Työn laatu ja arvostus:	
14.	Vaikuttaako menetelmien käyttö työn onnistumiseen (laatuun, aikatauluihin, työn sujuvuuteen?
15.	Ovatko menetelmät muuttaneet työnkuvaa aiempaan verrattuna?
16.	Arvostetaanko menetelmien osaamista (Agile, Docs as code)?
17.	Olisiko työnkuvassa & menetelmien käytössä jotain, mitä haluaisit muuttaa tai parantaa, jotta jälki olisi laadukkaampaa tai työ sujuisi tehokkaammin?
18.	Saatko mielestäsi arvostusta työssäsi? Mitä muuttaisit tai parantaisit? (*

Liite 2. Tutkimuksen haastattelukutsu

Tutkimuksen haastattelukutsun ruutukaappaus LinkedIn-palvelusta. Haastattelukutsu julkaistiin Suomen teknisen viestinnän yhdistyksen ryhmässä helmikuussa 2022.



Suvi Hela • You
Technical Writer
1yr

Are you working as a technical writer or in other technical communication related position within the IT-industry? I would be happy if you could take some time to participate in my study (roughly 30-45 min. interview using Zoom).

I'm a technical communication student at the University of Vaasa and currently working on my master's thesis related to the suitability of common software development methods and workflows (such as Agile or Docs as Code) to technical communication tasks.

I would be grateful if there are any professionals in this group that would be willing to participate. Please send me a private message or e-mail at suvi.hela@student.uwasa.fi. Interviews can be conducted in Finnish or English.

Työskenteletkö IT-alalla teknisenä kirjoittajana? Olen teknisen viestinnän opiskelija Vaasan yliopistossa ja teen pro gradu -tutkielmaa ohjelmistotuotantoalan käytössä olevien menetelmien (esimerkiksi Agile, Docs as Code) soveltuvuutta tekniseen viestintään. Olisi hienoa, jos sinulla olisi aikaa n. 30-45 min. pituiseen haastatteluun, joka toteutetaan Zoomin kautta. Minulle voi lähettää yksityisviestin tai s-postin osoitteeseen suvi.hela@student.uwasa.fi, niin sovitaan sopiva aika. #thesis #softwaredevelopment #technicalcommunication #agile

1

Like Comment

133 impressions View analytics