hyväksymispäivä arvosana

arvostelija

# Globaalisti hajautetun ohjelmistokehityksen yhteistyön ja viestinnän haasteiden ratkaisuja ketterässä kehitykssä

Satu Bennert

Helsinki 3.12.2017

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Laitos — Institution — Department |
| --- | --- |
| Matemaattis-luonnontieteellinen tiedekunta | Tietojenkäsittelytieteen laitos |

Tekijä — Författare — Author

Satu Bennert

Työn nimi — Arbetets titel — Title

Globaalisti hajautetun ohjelmistokehityksen yhteistyön ja viestinnän haasteiden ratkaisuja ketterässä kehitysssä

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidoantal — Number of pages |
| --- | --- | --- |
| | 3.12.2017 | 24 sivua + 1 liitesivua |

Tiivistelmä — Referat — Abstract

Globalisaatio ja digitalisaatio antavat mahdollisuuden hajautettuun ohjelmistokehitystyöhön, jossa sovellusta voidaan kehittää tauotta eri aikavyöhykkeillä vuorokauden ympäri. Globaali ohjelmistokehitys käyttää yhä lisääntyvässä määrin ketterää kehitystä, mikä on alkujaan kehitetty pienehkön lokaalin ohjelmistokehitystiimin tarpeisiin. Ketterän kehityksen periaatteita ei voi kaikilta osin sellaisenaan noudattaa hajautetussa ympäristössä. Lisäksi hajautuksesta aiheutuu maantieteellisiä, ajallisia, kielellisiä ja kulttuurisia haasteita. Tässä tutkielmassa etsitäänkin vastausta kysymykseen: Miten ketterän kehityksen käytänteet vastaavat tästä ympäristöstä nouseviin haasteisiin? Erityisesti perehdytään yhteistyön ja viestinnän haasteisiin, koska tehokas viestintä korreloi suoraan projektin onnistumisen kanssa.

Tämän tutkielman tavoite on tuottaa yhteistyön ja viestinnän haasteista yhteenvetotaulukko, jossa on ketterän kehityksen käytänteiden empiirisiä ratkaisuja haasteisiin. Yhteenvetotaulukko on tarkistustyökalu, joka auttaa ohjelmistokehitystiimiä haasteiden havaitsemisessa ja ratkaisemisessa ja sitä kautta edistää projektin onnistumista. Tutkimusmateriaalina on käytetty alan tutkimusta globaalista ohjelmistokehityksestä ketterällä kehityksellä.

ACM Computing Classification System (CCS):
Software and its engineering → Software creation and management → *Software development process management* → Software development methods → **Agile software development**

Avainsanat — Nyckelord — Keywords

Globaali ohjelmistokehitys, Ketterä ohjelmistokehitys, Yhteistyö, Viestintä

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

# Sisältö

# 1 Johdanto

*Globaali ohjelmistokehitys* (engl. Global Software Development (GSD), Globally Distributed Software Development) suosii *ketterää ohjelmistokehitystä* (engl. Agile Software Development) vuosi vuodelta enemmän [3]. Ketterän kehityksen kaksi ydinajatusta kehitystiimin tehokkuudesta ovat kasvokkain kohtaaminen ja asiakkaan päivittäinen läsnäolo. Nämä tavoitteet aiheuttavat ristiriidan globaalissa ympäristössä: tapaamiset kasvokkain eivät ole mahdollisia koko hajautetun tiimin kanssa, eikä asiakkaan edustajia riitä jokaiseen lokaatioon. Lisäksi hajautuksesta aiheutuvat omat haasteensa: maantieteellinen etäisyys sekä aika-, kieli- ja kulttuurierot.

Ketterää kehitystä käytetään, koska sillä on huomattavia etuja monimutkaisen ohjelmistokehityksen hallinnassa. Ketterä ohjelmistokehitys sallii vaatimusmääritysten muutokset, parantaa projektin näkyvyyttä, lisää tiimin tuottavuutta, nopeuttaa toimitusaikoja, pysyy aikataulussa ja budjetissa sekä tuottaa asiakkaalle jatkuvasti lisäarvoa pala kerrallan kasvavan sovelluksen muodossa [2, 3]. Ketterä kehitys sopii tilanteeseen, jossa tiimi samanaikaisesti sekä kehittää itse tuotetta että lisää ymmärrystään kehitettävästä tuotteesta ja sen ympäristöstä.

Tehokkaalla viestinnällä on positiivinen vaikutus maantieteellisesti hajautetun ketterän kehitystiimin menestykseen [6]. Menestyksekkään ohjelmistokehityksen tärkeimmät elementit ovat jäsenten välinen tehokas tiimityöskentely ja viestintä ennemmin kuin tekninen osaamistaso [17]. Viestintä on tunnustettu laajalti yhtenä avainhaasteena maantieteellisesti hajautetussa ketterässä ohjelmistokehityksessä [6].

Edellisissä kappaleissa mainituista lähtökohdista herää tämän tutkielman tutkimuskysymys: Miten ketterän kehityksen käytänteet vastaavat globaalista ympäristöstä nouseviin haasteisiin? Tutkimuskysymys jaetaan kolmeen osatavoitteeseen: 1) löytää ja luokitella tutkimusaineistosta yhteistyön ja viestinnän haasteita, 2) löytää ja kohdistaa haasteisiin ketterän ohjelmistokehityksen empiirisiä ratkaisuja ja 3) luoda tuloksena haasteiden ja ratkaisujen yhteenvetotaulukko. Yhteenvetotaulukko on uusi työkalu haasteiden havaitsemiseen ja ratkaisemiseeen (Taulukko A.1, Liite A). Haasteisiin reagoiminen projektin mahdollisimman aikaisessa vaiheessa edistää projektin onnistumista. Onnistuminen vaatii jatkuvaa tiimin sisäistä ja tiimien välistä yhteistyötä ja viestintää. Ketterällä ohjelmistokehityksellä voidaan lieventää yhteistyön ja viestinnän haasteita.

Tutkimusaineistona on kansainvälinen tutkimus ketterän ohjelmistokehityksen soveltamisesta hajautettuun ohjelmistotyöhön. Tutkimusartikkelit on valittu siten, että mukaan on saatu sekä kvantitatiivista että kvalitatiivista materiaalia. Kvantitatiivista osuutta edustavat systemaattiset kirjallisuuskatsaukset ja kvalitatiivista osuutta tapaustutkimukset.

Tässä tutkielmassa yhdistetään ketterän ohjelmistokehityksen empiirisiä ratkaisuja GSD-tiimin kohtaamiin yhteistyön ja viestinnän haasteisiin ja luodaan tarkistus-

työkalu. Jos tiimissä on ratkaisemattomia yhteistyön tai viestinnän ongelmia, riskinä on projektin viivästyminen, turhan työn tekeminen, resurssien tuhlaaminen, asiakaslupauksen vaarantuminen, luottamuksen menettäminen tai laadun heikkeneminen.

Tämän tutkielman luvussa kaksi perehdytään globaalin ohjelmistokehityksen suosion syihin ja tutustutaan ketterään ohjelmistokehitykseen. Luvussa kolme kerätään GSD:n haasteita ketterää kehitystä käyttävästä ohjelmistotuotannosta. Haasteet luokitellaan yhteistyön ja viestinnän otsikoiden alle yhteenvetotaulukkoa varten. Luvussa neljä etsitään ja kohdistetaan ketterän kehityksen ratkaisuja haasteisiin ja muodostetaan yhteenvetotaulukko (Liite A), joka on tämän tutkielman tuottama työkalu. Yhteenveto sisältää ajatuksia tästä tutkielmasta.

# 2 Globaali ja ketterä ohjelmistokehitys

Tässä luvussa avataan tarkemmin syitä, miksi globaalia ohjelmistokehitystä tehdään. GSD on tämän tutkielman laajin konteksti. Sen sisällä on tämän tutkielman toinen konteksti eli ketterä ohjelmistokehitys. Ketterä kehitys kerrotaan ensin yleisesti ja sen jälkeen tarkemmin Kanban- ja Scrum-käytäntöjen avulla kuvaillen. Ketterän kehityksen ominaispiirteet ja termit on hyvä avata, koska tutkimusmateriaalin haasteet ja ratkaisut perustuvat ketterään kehitykseen. Luvun loppupuolella esitetään ketterän ohjelmistokehityksen skaalautuminen laajaan projektiin ja kehitysehdotuksia ketterälle kehitykselle.

## 2.1 Globalisaation edut

Globaali ohjelmistokehitys on ohjelmistopalvelutuotteen määrittelyä, suunnittelua, kehittämistä, testaamista ja toimittamista asiakkaalle kansalliset rajat ylittävissä tiimeissä. GSD-tiimi voi syntyä ulkoistuksen, alihankinnan tai kumppanuuden seurauksena tai hajautus tehdään innovaatioiden toivossa. Globaali ohjelmistokehitys on erilaisten organisaatioiden, tekniikoiden ja ihmisten vuorovaikutusta.

GSD:n yleisesti tunnetut edut ovat organisaatiotasoisia kuten *kustannussäästöt* ja joustava *työntekijäreservi*. Merkittävin etu on työvoimakustannussäästö silloin, kun ohjelmistokehitys viedään edullisemman työvoiman maahan. Yrityksillä on mahdollisuus laajentaa ohjelmistokehitystoimintaansa, koska ne voivat palkata tarvittavan määrän ammattitaitoista työvoimaa sijainnista riippumatta [23]. Yksi eduista on *resurssien jakaminen*: työntekijäreservi joustaa kausittain tarvittavan työvoiman mukaan ja korkeamman kustannuksen työntekijät voidaan uudelleensijoittaa strategiseen työhön [23]. Yksittäisen huippuasiantuntijan osaamista voidaan siirtää tiimin yksilöille laajentamalla heidän tietotaitoaan ja tehostamalla tiimin yhteistyötä.

*Markkinoiden valtaaminen* ja *innovaatiot* houkuttelevat yrityksiä ulkomaille. Organisaatio hyötyy markkinoiden valtaamisen nopeudesta: eri aikavyöhykkeillä työskentely voi tehokkaimmillaan johtaa tauottomaan kehitykseen ja tuotteen lanseeraukseen [23]. Markkinoiden ja asiakkaan läheisyydestä saadaan *synergiaetua*: tytäryhtiön perustaminen asiakkaan luo lisää markkina-alueen tuntemusta [23]. Uusien työpaikkojen perustaminen voi luoda goodwilliä paikallisten asiakkaiden kanssa, mikä saattaa johtaa uusiin sopimuksiin. Joskus voi olla myös liiketoiminnallisesti välttämätöntä tulla asiakkaan luo, jotta laajentuminen uusille markkinoille ylipäätään onnistuu. Samalla voi löytyä uusia strategisia kumppaneita. Hajautuksesta aiheutuva etu on innovaatioiden ja parhaiden käytäntöjen jakaminen: yritykset voivat hyödyntää luovuutta, innovaatioita ja parhaita käytäntöjä, jotka kumpuavat erilaisista taustoista peräisin olevien ihmisten yhteistyöstä [23]. Hajautetussa ympäristössä yksilöt itseohjautuvat uusiin tuloksiin. Jotkut yritykset hakeutuvat jopa tarkoituksellisesti sinne, missä älykkäimmät ja luovimmat henkilöt ovat. Tällöin motiivina on potentiaaliset innovaatiot, eivät kustannussäästöt.

Kehittynyt ohjelmistoteknologia tukee hajautusta: abstrahointi, modulaarisuus, automaattitestaus ekä integrointi- ja versionhallintatyökalut. GSD:n hajautettu luonne johtaa tehtävien pilkkomiseen itsenäisiksi komponenteiksi. Komponentin riippuvuudet ja *koordinointikustannukset vähenevät* sen koko elinkaaren ajaksi [23]. Jos eri aikavyöhykkeillä työskentelevien tiimien tai henkilöiden työt ovat tiukasti ja lineaarisesti toisistaan riippuvaisia (ohjelmointi ja testaus), toinen voi tehdä oman osuutensa toisen nukkuessa [11]. Tällöin säästytään viivästyksiltä ja hukka-ajan koordinoinnilta. Organisatorinen ja maantieteellinen etäisyys johtaa *tiimin autonomian* kasvamiseen [23]. Autonomia nähdään tarpeellisena, jotta jokaisen tiimin oma työskentelykulttuuri ja työn laatu säilyvät.

Talletettu dokumentaatio ja selkeät prosessit kompensoivat globaalin ohjelmistokehityksen aika- ja etäisyyseroja. *Viestinnän muodollinen dokumentaatio* on etu: maantieteellisistä eroista johtuen reaaliaikainen viestintä on korvattu jollain muulla tavalla, josta säilyy talletettu dokumentaatio kaikille sidosryhmille uudelleen tutkittavaksi [13]. Etuna nähdään *kehittynyt dokumentaatio*: projektin tiedot dokumentoidaan ja jaetaan sähköisesti useammin kuin mitä tehtäisiin, jos tavattaisiin jatkuvasti kasvokkain [23]. Myös *selkeästi määritellyt prosessit* ovat etu: prosessit määritellään tarkemmin hajautetussa ympäristössä kuin vain yhden lokaation projektissa [23].

## 2.2 Ketterä ohjelmistokehitys

Alkujaan ohjelmistoja valmistettiin samalla tavalla lineaarisesti kuin rakennuksia: arkkitehti suunnitteli kokonaisuuden ja rakentajat rakensivat sen kerralla valmiiksi. Tämä ohjelmistotekniikan menetelmä - vesiputousmalli - lakkasi toimimasta, kun ohjelmistot kasvoivat, ympäristöt monimutkaistuivat, eikä sovelluksen kaikkia ominaisuuksia tiedetty etukäteen. Tarvittiin uusi tapa tehdä ohjelmistokehitystä, jossa vaatimusmääritykset saavat muuttua kehityksen aikana ja asiakas on aktiivisesti mukana tuotteen evoluutiossa. *Ketterän ohjelmistokehityksen julistus* (engl. Agile Manifesto) [1] 11.-13. marraskuuta vuonna 2001 määritteli uuden tavan tehdä ohjelmistokehitystä. Julistuksen allekirjoitti 17 ohjelmistokehityksen asiantuntijaa. Ketterän ohjelmistokehityksen julistus on kokoelma arvoja, joiden sisältö on priorisoitu seuraavasti:

- Tiimin jäsenet ovat yksilöitä, joiden kanssakäymistä arvostetaan enemmän kuin käytänteitä ja työkaluja.
- Asiakkaalle toimitettava lopputuote on tärkeämpi kuin täydellinen dokumentaatio.
- Jatkuva yhteistyö asiakkaan kanssa on tärkeämpää kuin sopimusneuvottelut.
- Muutoksenhallinta on tärkeämpää kuin alkuperäisessä kaiken kattavasssa suunnitelmassa pysyminen.

Ketterälle ohjelmistokehitys suosii ohjelmiston kehittämistä *iteratiivisesti* (engl. Iterative) ja *pala kerrallaan* (engl. Incremental). Jokainen iteraatio tuottaa ohjelmistoon uusia ominaisuuksia ja siten lisäarvoa asiakkaalle. Ohjelmistokehitystiimi on muutosmyönteinen ominaisuuksien lisäämisen, arkkitehtuurien ja käytettyjen työtapojen suhteen. Tiimillä on korkea työmoraali. Tiimi on itseohjautuva ja omasta työstään oppiva. Tiimi pyrkii maksimaaliseen ja tasaiseen tuottavuuteen joka iteraatiossa. Ohjelmakoodi on korkealaatuista.

Ketterän ohjelmistokehityksen julistus [1] sisältää kaksi merkittävää ajatusta viestinnästä:

- "Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu."
- "Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan."

Nämä ajatukset ovat ristiriidassa globaalin ohjelmistokehityksen kanssa. Miten järjestetään kasvokkain tapaamiset hajautetun tiimin kanssa? Miten kohdataan asiakas päivittäin joka lokaatiossa?

Eniten kasvokkain kohtaamisista hyötyvät homogeeniset tiimit, jotka käsittelevät tietointensiivistä tehtäviä ja työskentelevät paineen alla [10]. Kasvokkain viestintä on tärkein viestinnän mekanismi, koska suurin osa ihmisen viestinnästä on kehollista. Kehonkieli on usein alitajuista, samoin sen tulkinta. Kun tavataan kasvokkain uudessa ympäristössä, aivot stimuloituvat ja synnyttävät uusia ideoita. Ryhmässä voidaan tuntea yhtä aikaa samoja tunteita: iloa, kiinnostusta, intohimoa ja innostusta, mikä puolestaan luo yhteenkuuluvaisuutta, auttaa yhteistyösuhteiden muodostumista ja poistaa ennakkoluuloja. Kasvokkain kohdatessa keskittyminen on parempaa ja tiiviisssä vuorovaikutuksessa voidaan selvittää väärinkäsityksiä. Kasvokkain viestinnän ja tehokkuuden välillä organisaatiossa on syy-yhteys [10].

Ketterää kehitystä on kritisoitu siitä, että se keskittyy enemmän välittömään toimitukseen asiakkaiden nykyisten tarpeiden mukaan ottamatta huomioon kehityksen pitkän aikavälin vaikutuksia ihmisiin ja yhteiskuntaan kokonaisuutena [18]. Kestävä kehitys ohjelmistoalalla pyrkii kehittämään ja käyttämään ohjelmistoja minimaalisella taloudellisella, yhteiskunnallisella ja ekologisella vaikutuksella. Datan siirto internetin kautta on ekologisempaa kuin ihmisten lennättäminen paikasta toiseen.

## 2.3 Scrum-käytänteitä

Scrum Alliancen tuoreimman raportin [2] mukaan Scrum ja Kanban ovat suosituimmat ketterät metodologiat: 98% vastaajista kertoi käyttävänsä Scrumia ja 43% kertoi käyttävänsä Kanbania. Kolmannella tilalla on hybridimalli (23%), jossa perinteinen vesiputousmalli on yhdistetty johonkin ketterään ohjelmistokehitykseen. Scrum Alliancen tarkoitus on lisätä tietoisuutta Scrum-prosessista, tarjota virallisia Scrum-sertifiointeja kouluttajille sekä ylläpitää käyttäjäryhmiä ja tapahtumia. Scrum Alliancen tuorein raportti - The State of Scrum Report - on julkaistu vuonna 2017. Tutkimukseen vastasi yli kaksi tuhatta jäsentä.

Scrum-käytännössä ohjelmistokehitystyö tehdään iteraatioissa ja sovellusta kasvatetaan kokonaisuus kerrallan kun taas Kanban-käytäntö perustuu jatkuvaan työn virtaukseen. Iteraatiot vaativat jatkuvaa suunnittelua, määriteltyjä rooleja ja projektin tuotoksia. Scrumin käytänteiden lähteenä on käytetty Scrum-käytännön (jälj. Scrum) keksijöiden Ken Schwaberin ja Jeff Sutherlandin kirjoittamaa ja vuonna 2016 päivittämää Scrum-opasta (engl. The Scrum Guide) [19].

Scrumissa on selkeä tehtäväjako *tuoteomistajan* (engl. Product Owner), *scrummasterin* (engl. Scum Master) ja *kehitystiimin* (engl. Development Team) kesken. Tuoteomistaja ylläpitää tuotteen kehitysjonoa ja tarkentaa käyttäjätarinoita niin tarkalle tasolle, että kehitystiimi voi ne toteuttaa. Scrummaster mahdollistaa tiimin toimimisen optimitasolla. Scummaster järjestää palaverit ja huolehtii, että Scrum-käytäntöä noudatetaan oikein. Kehitystiimin jäsenet ovat ammattilaisia, jotka kehittävät tuotetta. He valitsevat suoritettavat kayttäjätarinat ja jakavat ne tehtäviksi sprintin ajaksi. Tiimi on yhdessä vastuussa tuotteesta.

Scrum sisältää neljä standardipalaveria: *sprintin suunnittelu* (engl. Sprint Planning), *päiväpalaveri* (engl. Daily Scrum), *sprintin katselmointi* (engl. Sprint Review) ja *sprintin retrospektiivi* (engl. Sprint Retrospective). Sprintin suunnittelupalaveri on kaksiosainen. Ensimmäisessä osapalaverissa tuoteomistaja, joka on yleensä asiakas, kertoo, millaisen kokonaisuuden hän haluaa seuraavaksi toteutettavan esittelemällä *tuotteen kehitysjonosta* (engl. Product Backlog) korkeimman prioriteetin *käyttäjätarinoita* (engl. User Story). Toisesssa osapalaverissa kehitystiimi valitsee tarinat ja suunnittelee, miten ne toteutetaan. Tiimi jakaa käyttäjätarinat yksityiskohtaisiksi *tehtäviksi* (engl. Task), joiden kesto on muutamasta tunnista pariin työpäivään. Kehitystiimi ylläpitää tehtäviä *sprintin kehitysjonossa* (engl. Sprint Backlog). Sprintin suunnittelupalaverin aikaraja (engl. Time-Box) on kuukauden sprinteissä kahdeksan tuntia ja lyhyemmissä sprinteissä neljä tuntia. *Sprintin tavoite* (engl. Sprint Goal) on yhteen lauseeseen kiteytetty uusi *tuoteversio* (engl. Increment) kaikkien *sidosryhmien* (engl. Stakeholder) tiedoksi.

Päiväpalaveri on kehitystiimin sisäinen palaveri, johon avoimuuden nimissä kaikki sidosryhmät ovat tervetulleita kuuntelemaan. Tehokas palaveri pidetään joka päivä samaan aikaan. Varttitunnin aikana jokainen tiimin jäsen kertoo, mitä on saanut

aikaan edellisen tapaamisen jälkeen, mitä aikoo tehdä seuraavaksi ja mitä esteitä mahdollisesti on etenemisessä.

Sprintin lopussa tiimi esittelee asiakkaalle ohjelmiston *valmiit* (engl. Done) osat sprintin katselmoinnissa ja miettii omaa suoriutumistaan retrospektiivissä. Tuoteomistaja hyväksyy laadulliset kriteerit täyttävät ominaisuudet ja palauttaa kelpaamattomat osat takaisin tuotteen *kehitysjonoon* (engl. Backlog Item) uutta iteraatiota varten. Sprintin katselmoinnin suositusaika on neljä tuntia. Retrospektiivi on tiimin työprosessin kehittämisen paikka. Tiimi päättää, mitä asioita prosessissa pitää kehittää, mitä voidaan jättää seuraavassa sprintissä pois ja mitä hyviä käytänteitä sovelletaan sellaisenaan edelleen.

Scrumin dokumentaatio sisältyy kolmeen kaikille avoimeen *tuotokseen* (engl. Artifact): tuotteen kehitysjono, sprintin kehitysjono ja toimiva sovellus. Tuotteen kehitysjono on tuoteomistajan priorisoima jono kehitettävistä mielekkäistä toiminnallisista kokonaisuuksista. Tuoteomistaja *työstää tuotteen kehitysjonoa* (engl. Product Backlog Refining) jatkuvasti: tarkentaa, lisää, poistaa ja priorisoi käyttäjätarinoita. Sprintin kehitysjono on kulloinkin sprintissä työn alla oleva ohjelmisto. Siinä näkyy jokainen tehtävä, tekijä, tila ja jäljellä oleva työmääräarvio. Toimiva sovellus on kaikkien valmistuneiden sprinttien summa - inkrementti. Inkrementti on toimiva kokonaisuus tuotteesta ja viimeisen sprintin jälkeen valmis tuote.

## 2.4 Kanban-käytänteitä

Kanban-käytännössä (jälj. Kanban) [8] on viisi keskeistä käytännettä: visualisoi työnkulku, rajoita käynnissä olevaa työtä, hallitse jatkuvaa työn virtausta, tee prosessikäytänteistä yksiselitteisiä ja paranna yhteistyötä. Kanban toimii imuohjauksella: uutta tehtävää imetään prosessissa eteenpäin sitä mukaa, kun resursseja vapautuu [5, 8]. Al-Baikin ja Millerin [5] tutkimuksen mukaan Kanbanilla on kyky sopeutua vaatimusten muutoksiin, visualisoida projektin prosessit sekä lisätä tiimin yhteistyötä ja viestintää.

Kanbanin kehitysprosessin- ja projektinhallinnan keskeiset työkalut ovat työnkulun visualisointityökalu *Kanban-taulu* (engl. Kanban Board) ja käyttäjätarinoiden hallinnan *kehitysjono* (engl. Backlog) [5]. Kehitysjono sisältää käsittelemättömät työt priorisoituna tärkeyden, kiireellisyyden tai arvon tuottamisen perusteella. *Sisäänpääsykriteerit* (engl. Inclusion Criteria) takaavat sen, että jokainen käyttäjätarina tuo lisäarvoa asiakkaalle. Kanban-taululle on priorisoitu *käyttäjätarinat* (engl. User Story), jotka *tehtäviksi* (engl. Task) pilkottuina liikutetaan tiimin määrittelemistä tilasarakkeista toiseen, kunnes tehtävä on valmis. *Visuaalinen hahmo* (engl. Avatar) on tiimin jäsenen virtuaalipersoona, joka on kiinnitetty tehtävään. Tehtävää kuljetetaan taululla imuperiaatteella sarakkeesta toiseen sitä mukaa, kun tehtävä on täyttänyt sarakkeen toiminnon. Jokaisella sarakkeella on oma *tehtävien maksimilukumäärä* (engl. Work In Progress, WIP). WIP:eillä hallitaan suoritettavaa ko-

konaistyön määrää. *Peruutettu tehtävä* (engl. Reverse Item) on syystä tai toisesta taaksepäin siirretty tehtävä, jonka valmistamista ei voida jatkaa. *Pullonkaula* (engl. Bottleneck) voi syntyä, jos töitä ei saada valmiiksi. Tällöin on saavutettu sarakkeen tehtävien maksimilukumäärä. Töitä voidaan laittaa myös *pinoon* (engl. Stack) tai *varastoon* (engl. Buffer), jos kukaan kehitystiimistä ei voi viedä niitä eteenpäin.

Prosessin tehokkuutta mitataan jatkuvasti *läpimenoajoilla* (engl. Cycle Time, Lead Time) ja *edistymiskäyrillä* (engl. Burn Down Chart). Mittaaminen on prosessin suorituskyvyn motivaattori, mitataanpa sitten yhden tehtävän keskimääräistä suoritusaikaa tai uuden ominaisuuden rakentamisaikaa. Koko projektin suorituskykyä mitataan päivittäisten edistymiskäyrien tai kumulatiivisten tehtävien lukumäärä- ja läpimenoaikadiagrammien avulla.

Tiimin yhteistyön perusta on yhteisesti sovitut *käytänteet* (engl. Policy) ja työstä oppiminen. Käytänteet ovat tiimin itsensä päättämät yksiselitteiset ja visualisoidut säännöt, joiden mukaan työtä tehdään: mitä työkaluja käytetään viestintään, miten päätöksiä vaativat asiat tuodaan esille, miten pullonkaulat ratkaistaan, miten uudet työt priorisoidaan jne. *Perusteltu oppiminen* (engl. Validated Learning) tapahtuu kokeilujen kautta: ota vastaan asiakkaan palaute, haasta nykyiset tavat ja reagoi nopeasti muutoksiin. *Palautesilmukka* (engl. Feedback Loop) on evoluutioprosessi, jossa verrataan odotettetuja tuloksia saatuihin tuloksiin ja tehdään tarvittavia muutoksia.

Kanban noudattaa Lean-ajattelumallia, mikä tarkoittaa ohjelmistokehityksessä sitä, että työskennellään kestävällä tahdilla, eliminoidaan *hukkaa* (engl. Waste), tuotetaan usein arvoa ja edistetään jatkuvaa prosessin parantamisen kulttuuria [8]. Hukkaa ovat turhat prosessit, turhat ominaisuudet, virheet ja kesken jäänyt työ. *Päiväpalaveri* (engl. Stand Up Meeting) jakaa asiantuntijat kahtia: onko se hukkaa vai onko tarpeen käydä läpi projektin tilanne kasvotusten? *Suunnittelupalaverin* (engl. Planning Meeting) tilanne on sama: onko se hukkaa vai tärkeä palautteen antamisen ja saamisen hetki tiimille?

## 2.5 Ketterän ohjelmistokehityksen skaalautuminen

Ketterä ohjelmistokehitys on kehitetty paikallisen ja pienehkön tiimin (3-9 henkilöä) tarpeisiin [2], mutta ajan myötä on tullut tarve soveltaa sitä kymmenien - jopa satojen - ihmisten tiimeihin ja organisaatioihin. Version One on kehittänyt ketterän kehityksen hallintatyökaluja vuodesta 2002 ja tuottanut lähes vuosittain käyttäjäraportin. Viimeisin raportti - 11th Annual State of Agile ™ Report - huhtikuussa vuonna 2017 osoitti, että *SAFe*® (engl. Scale Agile Framework) on yleisin ketterän ohjelmistokehityksen skaalausmenetelmä [3]. SAFe® on vuonna 2007 Dean Leffingwellin kehittämä menetelmä, jonka uusin versio (4.5) on julkaistu lokakuussa vuonna 2017 [16]. Tässä tutkielmassa SAFe®:a on kuvattu version 4.5 mukaan.

SAFe® on vastaus ketterän kehityksen tarpeeseen skaalautua henkilömäärältään suurempiin projekteihin ja useamman tuotteen tai tuoteperheen yhtäaikaiseen kehittämiseen tai ylläpitämiseen. Lisäksi sitä voi soveltaa muihin liiketoimintoihin kuin ohjelmistokehitykseen. SAFe®:ssa on neljä hierarkiatasoa ylhäältä alas: *salkku-* (engl. Portfolio), *arvovirta-* (engl. Large Solution), *hanke-* (engl. Program) ja *tiimitaso* (engl. Team). Jokaisella tasolla on oma kaikille avoin töiden kehitysjono. Yrityksen strategia levitetään kehitysjonojen tehtävien kautta alemmille tasoille. Aikaisemmin tässä luvussa käsitellyt Kanban- ja Scrum-käytännöt ovat SAFe®:n tiimitason kivijalka, jonka päälle muu hierarkia rakentuu.

SAFe®:n hierarkin ylimmällä tasolla hallitaan yrityksen strategiaa ja liiketoimintasuunnittelua kolmella Scrumiin nähden uudella roolilla: *salkunhaltijat* (engl. Lean Portfolio Management, LPM), *kehitysaihion omistajat* (engl. Epic Owner) ja *yritysarkkitehti* (engl. Enterprise Architect). Salkunhaltijat vastaa ryhmänä ensisijaisesti kolmesta alueesta: Agile-ohjelman ohjauksesta, Lean-johtamisesta sekä strategiasta ja investointirahoituksesta. Salkunhaltijat on ryhmä henkilöitä, joilla on korkein päätöksenteko- ja talousvastuu. *Kehitysaihiot* (engl. Epic) ovat uusia liiketoimintamahdollisuuksia, jotka vaativat suunnittelua, hyväksyntää, investointeja ja vaikuttavat pitkälle tulevaisuuteen. Kehitysaihion omistajat ottavat vastuun kehitysaihioista ja määrittelevät liiketoimintakuvauksen ja budjetin. *Salkun kehitysjo* (engl. Portfolio Backlog) sisältää hyväksyttyjä liiketoiminnan kehitys- ja mahdollistaja-aihioita. *Mahdollistajat* ovat arkkitehtonisia ja -teknisiä aloitteita, jotka ovat välttämättömiä uusien liiketoimitaideoiden toteuttamiseksi. Kehitysaihioiden virran visualisoimiseksi ja hallitsemiseksi käytetään *salkun valmisteluseinää* (engl. Portfolio Kanban). Yritysarkkitehti ohjaa kokonaisvaltaista teknologian toteuttamista eri hankkeiden välillä.

Hanketasolla monta tiimiä synkronoidaan yhteen samalla syklillä eteneväksi toimitusjunaksi, joka on ketterien kehitystiimien muodostama tiimien tiimi, jossa työskentelee yleensä 50-125 henkilöä. Hanketasolla käytetään ylemmiltä tasoilta tulevaa priorisointia tiimien kehityksen ohjaukseen. Toimitusjunalla on yhteinen rytmi tiimien suunnittelulle, kehitykselle ja retrospektiiville. Jokaisella junalla on omat resurssinsa. Tyypillisesti sprintin kesto on kaksi viikkoa ja uusi tuoteversio julkaistaan kymmenen viikon välein. SAFe® suosittelee kaikkien tuotteiden synkronointia samaan kymmenen viikon aikatauluun.

*Toimitusjunan päällikkö* (engl. Release Train Engineer) päällikkö on laajemman mittaluokan scrummaster ja *tuotehallinta* (engl. Product Management) on laajemman mittaluokan tuoteomistaja. Toimitusjunan päällikkö arvioi toimitusjunan prosesseja ja toteutusta. Hän poistaa esteitä, arvioi riskejä, huolehtii arvon toimittamisesta ja toiminnan jatkuvasta kehittämisestä. Toimitusjunan päällikkö on suuremman mittakaavan scrummaster. Tuotehallinta on vastuussa asiakkaan tarpeiden tunnistamisesta. Se omistaa toimitusjunan vision, etenemissuunnitelman, hinnoittelun, lisensoinnin, investoinnin tuoton sekä hanketason kehitysjonon. Tuotehallinta huolehtii inkrementtien tavoitteista. Tuotehallinta julkaisee sisältöä priorisoitujen toiminnalli-

suuksien ja hyväksymiskriteerien muodossa sekä hyväksyy inkrementin järjestelmän pääkehityshaaraan.

Tässä esitellään lyhyesti kaksi muuta skaalautumiskehystä SAFe®:n lisäksi ketterälle ohjelmistotuotannolle: *LeSS* (engl. Large-Scale) [15] ja *DAD* (engl. Discipline Agile Development) [7]. Craig Larman ja Bas Vodden ovat kehittäneet LeSS:in. LeSS soveltaa Scrumin periaatteita, sääntöjä, elementtejä ja käytänteitä laajalla skaalalla ja niin yksinkertaisesti kuin mahdollista. LeSS on alkujaan suunniteltu yhdelle tuotteelle, jota kehittää monta scrumtiimiä. Mark Ambler ja Scott Lines ovat kehittäneet DAD:in vuosina 2009-2012 IBM:llä työskennellessään nimenomaan sovelluskehitykseen. Uusin versio on julkaistu vuonna 2016. DAD:illa on hybridilähestymistapa skaalautuvaan sovelluskehitykseen: se hyödyntääää kaikkia ketteriä käytänteitä ja strategioita ja antaa ohjeet siitä, milloin ja miten niitä sovelletaan.

## 2.6 Kehitysehdotuksia ketterään kehitykseen

Ketterää ohjelmistokehitystä on sovellettu kohta kaksi vuosikymmentä, joten käytännöistä on noussut esille uusia mahdollisia rooleja. Af Orns ym. [4] tutkivat hajautetun GSD-projektin johtamisen parhaita käytänteitä ketterän kehityksen näkökulmasta. He löysivät tarpeen kahdelle lisäroolille projektissa: *kulttuurilähettiläs* (engl. Cultural Liaison) ja *asiakkaan edustaja* (engl. Proxy Customer, Customer Representative). Kulttuurilähettiläs on *ulkomailla sijaitsevan* (engl. Off-shore) tiimin jäsen, joka siirtyy fyysisesti *kotimaassa sijaitsevan* (engl. On-shore) tiimin jäseneksi. Hän pehmentää kulttuurieroja, madaltaa kielimuuria, välittää tietoa projektin tilanteesta ja toimii suorana kontaktina ulkomaan tiimiin päin. Roolissa on hyvä olla henkilö, joka tuntee molempien maiden perinteet ja hänellä on jo kokemusta työskentelystä kansainvälisessä ryhmässä. Asiakkaan edustaja on henkilö, joka on suorassa yhteydessä asiakkaaseen ja toimii kuten asiakas kehitystiimiin nähden. Hänen pitäisi pystyä vastaamaan tiimin kysymyksiin koskien liiketoiminta-aluetta ja vaatimusmäärityksiä.

Ketterään ohjelmistokehitykseen on esitetty myös uusia käytänteitä: *tiedonsiirto* (engl. Knowledge Transfer), *jäljitettävä hierarkia* (engl. Traceable Hierarchy) ja *suora viestintä* (engl. Direct Communication) [4]. Käytänteet ovat linjassa ketterän ohjelmistokehityksen julistuksen periaatteiden kanssa. Af Ornsin ym. [4] mukaan tiedonsiirto on äärimmäisen tärkeää toimintaa, joka olisi toteutettava ensin perusteellisesti hankkeen alkuvaiheessa ja sen jälkeen jatkuvana toimintona kevyemmin jokaisessa iteraatiossa. Projektin ensimmäinen tiedonsiirto on sellainen, jossa kaikki tiimit ovat samassa tilassa kasvokkain sekä saamassa teoreettista taustatietoa liiketoiminnasta, projektin arkkitehtuurista ja käytänteistä että suorittamassa todellisia, mutta pieniä kehitystehtäviä toivotulla tavalla oikeassa ympäristössä. Projektiryhmälle pitää antaa mahdollisuus tutustua toisiinsa, rakentaa luottamusta ja suhteita. Tämä mahdollistaa aktiivisemman viestinnän jäsenten kesken senkin jälkeen, kun tiimit on hajautettu. Lisäksi päiväpalaverin voisi toteuttaa niin, että kukin tiimi ko-

koontuisi ensin omassa lokaatiossaan kasvokkain ja sen jälkeen olisi kaikkien tiimien yhteinen palaveri sähköisesti. Roolit ja tehtäväjako pitäisi olla julkisesti esillä tiimin tiedotuskanavissa, jotta voi ottaa nopeasti yhteyttä pätevimpään henkilöön. Suora viestintä maantieteellisten rajojen yli vähentää väärinkäsityksiä ja jatkuva viestintä tiimien välillä rakentaa yhteishenkeä.

Globaalisti hajautettu ohjelmistokehitys vaatii enemmän organisointia kuin paikallinen ohjelmistokehitys. Tehtäväjako on ketterissä kehityksessä ollut perinteisesti niin, että jokainen sovelluskehittäjä valitsee itse tehtävänsä. Af Orns ym. [4] kuitenkin ehdottavat, että tiiminvetäjä kohdistaisi tehtävät sovelluskehittäjille, koska hän tietää parhaiten kulloinkin käytettävissä olevat resurssit.

# 3 Globaalin ohjelmistokehityksen haasteita ketterässä kehityksessä

Edellisessä luvussa on esitelty tämän tutkielman konteksti: GSD ja ketterä ohjelmistokehitys. Tässä luvussa perehdytään tapaustutkimuksiin ja kirjallisuuskatsauksiin sanallisesti ja havainnollistavien kaavioiden kautta. Samalla kerätään tutkimusmateriaalista haasteita yhteenvetotaulukkoon Taulukko A.1 Liitteessä A. Haasteet luokitellaan kahteen kategoriaan: yhteistyö ja viestintä. Yhteenvetotaulukkoon hyväksytty haaste on **vahvistettu**, kun se ilmenee ensimämistä kertaa.

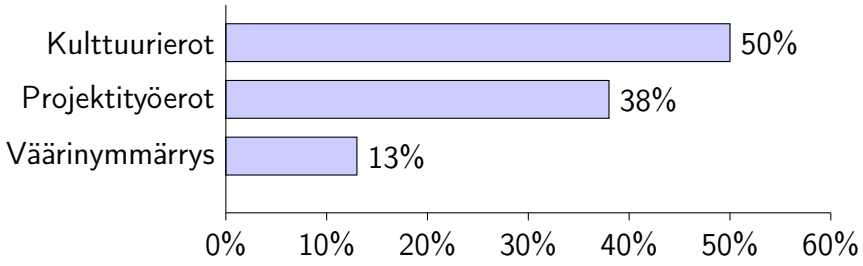
## 3.1 Yhteistyön haasteita

Tässä on esimerkki aidosta globaalista tiimistä. Tannerin ja Dauanen [20] tutkimuksen Kanban-käytäntöä soveltava ohjelmistokehitys sijaitsi kahdessa kaupungissa: Lontoossa (Englannissa) ja Kapkaupungissa (Etelä-Afrikassa). Lontoon tiimi vastasi itsenäisesti vaatimusmäärittelyjen keräämisestä ja Lontoon tiimiin osallistui myös asiakkaan edustajia. Kapkaupungin tiimi vastasi vaatimusten suunnittelusta, kaikkien ohjelmistokehityshankkeiden hallinnoinnista ja asiakkaan nykyisten sovellusten ylläpidosta. Molemmat tiimit tekivät kehitystyötä. Tiimin jäsenillä oli yhteinen kieli englanti, mutta henkilöt puhuivat eri murteita. Kapkaupungin työntekjät aloittivat työpäivänsä kaksi tuntia ennen Lontoon työntekijöitä. Vaatimusmäärittelyt kirjoitettiin Lontoossa, mutta kaikkien tuli ymmärtää ne yksiselitteisesti.

Lontoo-Kapkaupunki -tapaustutkimuksessa ilmeni seuraavat yhteistyön haasteet: **huono johtaminen**, epätasainen työnjako, epäyhtenäiset työskentelytavat, **läpinäkymättömyys** ja **projektin viivästykset**. Ketterä kehitys antaa mahdollisuuden jatkuvaan muutokseen, mutta silti asiakkaan muuttuvat **vaatimukset** aiheuttivat ongelmia. Vaatimusten tekniset epäselvyydet aiheuttivat ajanhukkaa, jolloin itsenäinen työn jatkaminen ei olllut mahdollista [20]. Uusia, kiireellisiä vaatimuksia ilmeni jopa kesken iteraation. Tutkimuksessaan Tanner ja Dauane [20] huomasivat, että projektin eteneminen hidastui, jos henkilö työskenteli monessa projektissa yhtä aikaa, eikä läpinäkymättömyyden vuoksi pystynyt vaihtamaan sujuvasti projektista toiseen.

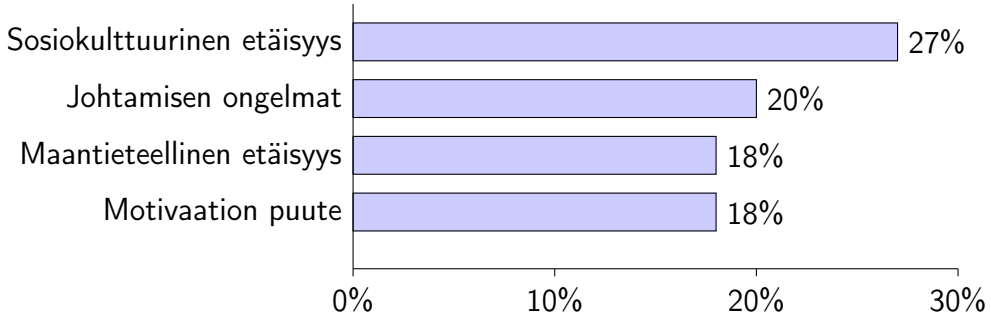
Yhteistyön ongelmia ovat **kulttuurierot**, **projektityöerot** ja vaatimusten heikko taso [14]. Kamaruddinin ym. [14] kirjallisuuskatsaus tunnisti 13 erilaista ongelmaa, joista tässä tutkielmassa luokitellaan kolme yhteistyön ja kymmenen viestinnän otsikon alle. Kirjallisuuskatsauksen tiedot on kerätty sekä aikaisemmista tutkimuksista että kansainvälisiltä asiantuntijoilta siten, että vastauksia on saatu kaikkiaan 16 osallistujalta. Puolet vastaajista kertoi erilaisten kulttuuritaustojen aiheuttavan ideologioiden ja uskomusten konflikteja kuten Kuvassa 3.1 on havainnollistettu. Ongelmanratkaisu hankaloituu, kun negatiivisia asioita ei voi tuoda yhteiseen tietoon kasvojaan menettämättä. Lisäksi kulttuurisidonnaiset tai kansalliset juhla- ja vapaapäivät aiheuttavat ikäviä yllätyksiä. Kun tiimin jäsenellä on erilainen projektityö-

tausta, haasteena on ketterän kehityksen omaksuminen. Tiimin jäsenen on vaikea tiedostaa, mikä on hyödyllistä tiedottaa omasta työstään muille tiimin jäsenille ja mitä ei. Tutkimuksessa tuli esille myös vaatimusten väärinymmärrys, mikä lisää projektin epäonnistumisriskiä.




Kuva 3.1: Kamaruddinin ym. [14] tutkimustulos yhteistyön haasteista.

Sosiokulttuurinen sopeutuminen tarkoittaa sitä, että henkilöt ovat riittävästi keskenään tekemisissä, jotta he oppivat toisiltaan tavat ja käytännöt ja siten sopeutuvat uuteen (työ)kulttuuriin niin hyvin, että osaavat toimia arkipäivän tilanteissa oikein [21]. Ghafoorin ym. [12] kirjallisuustutkimus vuodelta 2017 sisältää 51 eri paperia, joiden perusteella he etsivät vastausta kysymykseen: "Mitä ongelmia on kuvattu kirjallisuudessa ketterän kehityksen soveltamisessa globaalissa ohjelmistokehityksessä?". Tähän tutkielmaan on otettu mukaan kymmenen yleisintä ongelmaa, joista neljä luokitellaan yhteistyö ongelmiin ja kuusi viestinnän ongelmiin. Yhteistyön ongelmat ovat: sosiokulttuurinen etäisyys (14/51), johtamisen ongelma (10/51), **maantieteellinen etäisyys** (9/51) ja **motivaation puute** (9/51). Luku 9/51 tarkoittaa, että 9 vastaajaa 51:stä on tätä mieltä. Ongelmien esiintyminen on esitetty prosentteina Kuvassa 3.2.




Kuva 3.2: Ghafoorin ym. [12] tutkimustulos yhteistyön haasteista.

Hajautetun ohjelmistokehityksen riski on hallinnointikustannusten ylittyminen aikataulun pettäessä. Asiakkaan puutteellinen läsnäolo on yhtä suuri riskitekijä (62%)

kuin riittämätön dokumentointi [18]. Asiakkaan vähäinen osallistuminen pidentää ohjelmistotuotteen valmistumisaikaa ja lisää projektin epäonnistumisen todennäköisyyttä. Pitkän aikavälin suunnittelun puute on myös merkittävä riskitekijä, jonka tunnisti 48% Rashidin ja Khanin [18] tutkimuksen 42 ketterästä tiimistä. Tutkimus painottaa ajanhallinnan ja työtekijäreservin hallinnan merkitystä.

Yhteenvetotaulukon pohjaksi Taulukkoon 3.1 viedään yhteistyön haasteiksi johtamistekijät (sisältää epätasaisen työnjaon ja huonon johtamisen), läpinäkymättömyys, projektin viivästys, vaatimus (sisältää vaatimusten heikon tason ja tekniset väärinymmärrykset), kulttuuriero, projektityöero (sisältää epäyhtenäiset työtavat ja sosiokulttuurisen etäisyyden), maantieteellinen etäisyys ja tiimin motivaation puute (sisältää sitoutumisen ja luottamuksen puutteen).

Taulukko 3.1: Yhteistyön haasteita

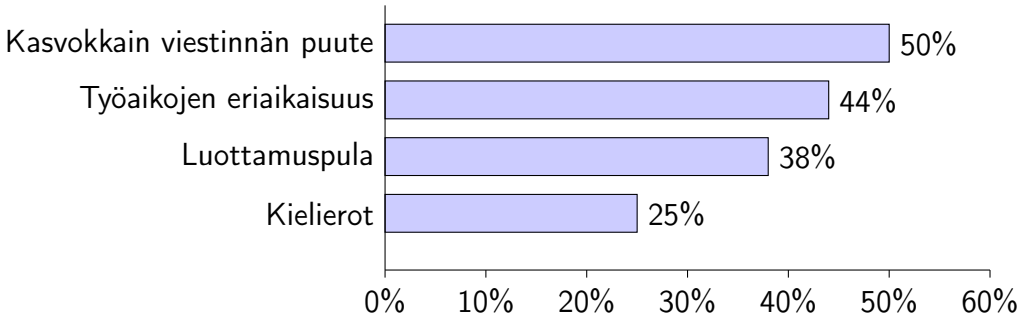
| **Viite** | **Haaste** |
|---|---|
| Y1 | Johtamistekijät [6, 12, 20] |
| Y2 | Läpinäkymättömyys [6, 20] |
| Y3 | Projektin viivästys [6, 20] |
| Y4 | Vaatimus [14, 20] |
| Y5 | Kulttuuriero [6, 14] |
| Y6 | Projektityöero [6, 14] |
| Y7 | Maantieteellinen etäisyys [6, 12] |
| Y8 | Motivaation puute [6, 12] |

## 3.2 Viestinnän haasteita

Kapkaupungin ja Lontoon tiimien tapaustutkimus toi esille viestinnän suurimpina haasteina **aika- ja kielieron** sekä **kasvokkain kohtaamisten vähyyden**. Aikaeron vuoksi henkilöitä ei tavoita tai yhteisen ajan löytämiseksi osa joutuu jäämään ylitöihin. Kielen paikalliset murre-erot aiheuttavat vaatimusten tulkintaepäselvyyksiä, vaikka tiimeillä on yhteinen kieli englanti [20].

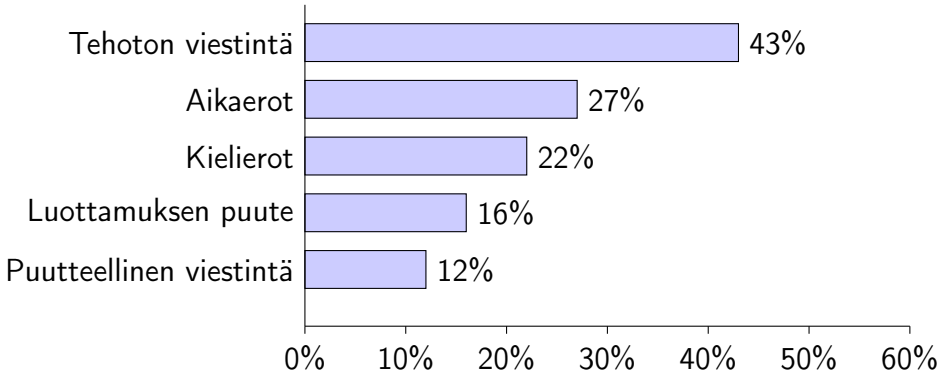
Erilaiset työajat ja kasvokkain viestinnän puute ovat syynä siihen, että tiimin jäsenet eivät ole kiinnostuneita toistensa projektityötaustoista, kulttuurista tai kielestä, mikä johtaa tiimin sisäiseen luottamuspulaan [14]. Asiakkaan läsnäolo koetaan vähäiseksi osittain etäisyyden vuoksi ja osittain siksi, että vastoin ketterän kehityksen periaatetta keskityttiin enemmän prosessiin kuin ihmisten väliseen viestintään. Ongelmia aiheuttavat myös viestinnän heikko tekniikka, suuret kustannukset ja viestintätyökalujen puutteellinen käyttöosaaminen. Kamaruddin ym. [14] tutkimuksen 13 haasteesta luokitellaan tässä tutkielmassa kymmenen seuraavaa haastetta viestintään liittyväksi: kasvokkain viestinnän puute (8/16), työaikojen eriaikaisuus (7/16),

luottamuspula (6/16), kielierot (4/16), **asiakkaan vähäinen läsnäolo** (3/16), sitoutumisen puute (2/16), viestintäyhteyksien heikko tekninen laatu (3/16), korkea viestintäkustannus (3/16), puutteellinen viestintätyökalun käyttö (4/16) ja heikko viestinnän infrastruktuuri (2/16). Neljä yleisintä haastetta on havainnollistettu Kuvassa 3.3.




Kuva 3.3: Kamaruddinin ym. [14] tutkimustulos viestinnän haasteista.

Aikaero vaikeuttaa yhteisten palaverien ajankohdan löytämistä ja ylipäätään viestinnän ajoittamista. Ghafoorin ym. [12] kirjallisuustutkimuksessa havaittu suurin ongelma on tehoton viestintä (22/51) Kuvassa 3.4. Muita huomattavia viestinnän ongelmia ovat aikaerot (12/51), kielierot 11/51), luottamuksen puute tiimeissä (8/51) ja puutteellinen viestintä (6/51).



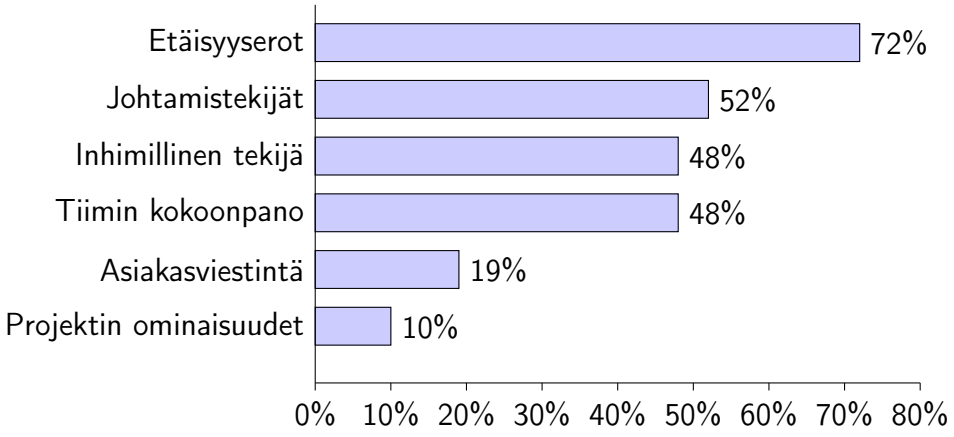

Kuva 3.4: Ghafoorin ym. [12] tutkimustulos viestinnän haasteista.

Ajallinen ja maantieteellinen etäisyys on ylivoimaisesti suurin (72%) viestinnän haaste Azoubin ym. [6] kirjallisuuskatsauksen mukaan. He tutkivat maantieteellisesti hajautettujen ketterän kehityksen tiimien viestinnän haasteita ja ratkaisuja. Tutkimus sisälsi 21 empiirisen tutkimuksen artikkelia, joissa yli puolessa oli käytäntönä Scrum tai Scrumin hybridi. Tutkimus jakaa haasteet kuuteen kategoriaan: etäisyyserot, johtamistekijät, inhimillinen tekijä, tiimin kokoonpano, asiakasviestintä ja projektin ominaisuudet. Kuva 3.5 havainnollistaa tutkimustuloksen. Yleisin kategoria on etäisyyserot, jossa haasteina ovat vähentyneet viestintämahdollisuudet,

epävirallisen viestinnän puute, kasvokkain viestinnän puute, viestintäviipeet, pidentyneet palaverit, palaverien järjestämisen vaikeus työaikojen puitteissa, uskonnollisten juhlapäivien erot, kehittäjien saavutettavuusvaikeudet, lisääntynyt integraatio, lisääntyneet koordinaatio- ja viestintäkulut, tehtävien ymmärtämisen heikko taso ja epäluottamus. Toiseksi yleisin kategoria on tiimin kokoonpano. Haasteina ovat varhaiset viestintäongelmat, tiimin jäsenten roolitus, haluttomuus viestiä muiden tiimin jäsenten kanssa, vähäinen ymmärrrys tiimityöstä, viestinnän hitaus ja pariohjelmoinnin hidastuminen, kun pariohjelmointia suoritetaan hajautetussa tiimissä.

Asiakasviestinnän haasteina ovat harva yhteydenpito asiakkaaseen, huonot suhteet asiakkaaseen, asiakkaan sitoutumisen puute ja tietojen pimittäminen asiakkaalta. Kun asiakaskontakti on heikko, alkavat sovelluskehittäjät tehdä vaatimuksiin liittyviä päätöksiä oman kokemuksensa tai arvaustensa pohjalta [6].

Projektitason haasteina ovat väärinkäsitykset, tarpeeton viestintä, eri lokaatien prosessien vastaavuuksien puute ja viestintämahdollisuuksien puute. Yli puolet, 52%, Azoubin ym. [6] tutkimuksen tapauksista havaitsi johtamistekijät haasteellisiksi. Johtamisen haasteita aiheuttvat tiimin laskenut tehokkuus, projektin vähentynyt laatu ja arvontuotto, rajoitteiden ja riskien lisääntyminen, useat erilaiset organisatoriset mallit, semanttinen yhteensopimattomuus, monet erilaiset vara- ja elpymissuunnitelmat, useat viestintäkanavat, työkalujen opettamisesta aiheutuvat ylimääräiset kustannukset, lokaatioiden tekninen ja kulttuurinen yhteensopimattomuus ja luottamuksen puute. Kuudes kategoria on inhimillinen tekijä. Ongelmia aiheuttavat väärinkäsitykset, eriaikaiset lomat, kielieroista johtuvat viestintäongelmat, aikaavievä tiedon jakaminen, tulkkausongelmat ja hiljaiset osallistujat.




Kuva 3.5: Azoubin ym. [6] tutkimustulos yhteistyön ja viestinnän haasteista.

Aikaeron vaikutukset ovat monimutkaisempia kuin maantieteellisen etäisyyden vaikutukset. Kahden eri tiimin välisen lineaarisen työn synkronointiin vaikuttavat tehtäväpyynnön aika, tehtävän suoritusaika, aikaero, päällekkäisen työn aika ja vuorovaikutusten määrä, jotka tapahtuvat päällekkäisen työajan aikana [11]. Työn lop-

puunsaattamisen kannalta on merkityksellistä, tekeekö tiimin jäsen toisen tiimin jäsenelle tehtäväpyynnön päällekkäisten työtuntien aikana vai ei, suorittaako tiimin jäsen tehtävän päällekkäisten tuntien aikana vai ei ja tapahtuuko päällekkäisyys työpäivän alussa vai lopussa.

Yhteenvetotaulukon pohjaksi Taulukkoon 3.2 viedään viestinnän haasteiksi aikaero, kasvokkain kohtaamisten vähyys, kieliero, asiakkaan vähäinen läsnäolo ja **viestintätekniikka** (sisältää osaamisen, laatu- ja hintatekijät, työkalut, viestintäviipeet ja kokoustekniikan). Näistä syistä johtuvat seuraukset kuten sitoutumisen puute tai epäluottamus ovat osa yhteistyön haasteetta motivaation puute (Haaste: Y8). Sekä yhteistyön että viestinnän haasteet yhdessä ovat syynä tehottomaan tai puutteelliseen viestintään.

Taulukko 3.2: Viestinnän haasteita

| **Viite** | **Haaste** |
|---|---|
| V1 | Aikaero [6, 12, 14, 20] |
| V2 | Kasvokkain kohtaamisen puute [12, 14, 20] |
| V3 | Kieliero [12, 14, 20] |
| V4 | Asiakkaan läsnäolon puute [6, 14, 20] |
| V5 | Viestintätekniikka [6, 14] |

# 4 Ketterän ohjelmistokehityksen ratkaisuja

Edellisessä luvussa löydettiin ja luokiteltiin yhteistyön ja viestinnän haasteita. Tässä luvussa etsitään ja yhdistetään ketterän ohjelmistokehityksen empiirisiä ratkaisuja haasteisiin. Haasteet on merkitty sulkuihin lauseen loppuun, jos lause sisältää ratkaisuelementtejä. Tässä luvussa luodaan lopullinen yhteenvetotaulukko (Taulukko A.1, Liite A), joka on tarkistustyökalu haasteiden havaitsemiseen ja ratkaisujen soveltamiseen.

## 4.1 Yhteistyön haasteiden ratkaisuja

Ketterän kehityksen käytetyimmät käytänteet ovat sprintin suunnittelupalaveri (90%), päiväpalaveri (88%), retrospektiivi (83%), sprintin katselmus (81%) ja lyhyt iteraatio (71%) [3]. Ketterien käytäntöjen toteuttaminen hajautetussa projektissa asettaa haasteita projektin johtajille, jotka joutuvat räätälöimään käytänteitä ja kompensoimaan yhteistyön ja kasvotusten kohtaamisen vuorovaikutuksen puutetta viestintätyökalujen ja innovatiivisen tietotekniikan avulla [6].

Ajantasalla oleva Kanban-taulu lieventää sellaisia yhteistyön haasteita kuten epätasainen työnjako, projektin viivästykset, huono johtaminen ja läpinäkymättömyys [20] (Haasteet: Y1, Y2, Y3). Kun jokainen sovelluskehittäjä päivittää oman tehtävänsä tilan säännöllisesti visuaalisen hahmon avulla, niin kaikki tietävät, mitä kukin on parhaillaan tekemässä, miten kauan siihen on mennyt aikaa ja miten kauan se vielä vaatii aikaa. Kuluneen ja vielä tarvittavan ajan kirjaaminen edistää tasaisempaa työnjakoa kunkin työtahti optimoiden (Haaste: Y1). Taulun visuaalisuus ja läpinäkyvyys edistää myös johtamista ja resurssien hallintaa ja siten pullonkaulojen purkamista [5] (Haasteet: Y1, Y3).

Johtamisen haasteisiin vastaus on luoda yhteisymmärrystä tiimin ja sidosrhymien välille hankkeen vaatimuksista, edistää koordinointia yleisen strategian ja paikallisten prosessien kesken sekä kannustaa tiimien jäseniä osallistumaan ristikkäin toisten tiimien päiväpalavereihin [6] (Haaste: Y1). Azoubi ym. [6] ehdottavat lokaatiokohtaisia tiimejä, joista jokaiseen palkataan oma scrummaster ja tuoteomistaja (Haasteet: Y1, Y4, Y7). Tutkimus ehdottaa myös tiimien yhdistettyjä kuukausittaisia retrospektiivejä, päiväpalavereita ja sprintin katselmuksia (Haaste: Y1, Y5). Tiimit voisivat kierrättää eri roolien edustajia säännöllisesti lokaatiosta toiseen. Tutkimus ehdottaa arkkitehdin käyttämistä, jotta korkean tason järjestelmävaatimuksista saavutetaan yhteisymmärrys mukaan lukien uudelleenkäytettävyys laatutekijänä (Haaste: Y4). Kirjoittajat ehdottavat, että kehittäjille pitäisi tarjota kokonaisvaltaisen kehittämisen strategia ja mukauttaa tiimin toiminta vastaamaan strategiaa (Haaste: Y2). Strategian levittäminen on hyvä tehdä lokaatiokohtaisesti ja siinä voidaan käyttää sosiaalisen median työkaluja.

Edellisessä kappaleessa esille tulleet ratkaisut ovat hyvin samankaltaisia, mitä edellisessä luvussa kerrottiin SAFe®-skaalautumiskehyksestä. Voidaan päätellä, että henkilömäärältään laajempi sovelluskehitys hyötyy järjestelmällisestä uudelleenorganisoinnista siten, että tiimit jaetaan homogeenisiin ja paikallisiin yksiköihin, jotka sitten synkronoidaan tuotekohtaiseksi toimitusjunaksi. Edellisessä kappaleessa kuvattu strategian "valuttaminen"hierarkiassa alaspäin ja yritystason arkkitehti löytyvät SAFe®:sta.

Kun haasteina ovat erilaiset työskentelytavat, ratkaisuina ovat Kanban-käytänteet [20] (Haaste: Y6). Käytänteet pitävät sisällään yhteisesti sovitut ja dokumentoidut ohjeet, joita noudattamalla työtavat yhtenäistyvät. Retrospektiivi on Scrum-kehittäjien palautefoorumi, jossa kukin voi tuoda esille omia havaintojaan myös työtavoista (Haasteet: Y5, Y6). Tiimi päättää, mitä käytänteitä kehitetään edelleen, mitä jatketaan ja mitä ei jatketa [19].

Scrumissa ei saa kesken sprintin puuttua kehitystiimin työhön, ellei ole kysymys erittäin painavasta syystä: kehitettävää ominaisuutta ei tarvita ollenkaan, jolloin sen kehitys keskeytetään milloin tahansa työhukan estämiseksi [19] (Haaste: Y4). Muissa tilanteissa tuoteomistaja priorisoi keskeneräiset ja uudet tehtävät tuotteen kehitysjonoon odottamaan seuraavaa sprinttiä (Haaste: Y4). Kun asiakkaan vaatimukset muuttuvat tai uusia vaatimuksia ilmenee kesken iteraation, voidaan tilanne hoitaa käyttäen Kanbanin käytänteistä sisäänpääsykriteeriä tai takaisinvetoa [20] (Haaste: Y4). Sisäänpääsykriteerit sijoittavat uuden vaatimuksen Kanban-taululle priorisoituun sijaintiin odottamaan vuoroaan tai joku muu tehtävä vedetään takaisin ja allokoidaan näin vapautunut resurssi heti käyttöön.

Vaatimusten tarkentamiseen tarvitaan asiakasta. Azoubi ym. [6] esittävät säännöllisten palaverien järjestämistä asiakkaan kanssa (Haaste: V4). Tutkimus ehdottaa sprintin suunnittelupalavereita varten erillistä kokoushuonetta, joka on varusteltu digitaalisilla viestintävälineillä ja videoneuvottelulaitteilla. Lisäksi voisi edistää asiakkaan edustaja -roolin käyttöä (Haaste: Y4, V4). Vaatimusten teknisiin epäselvyyksiin liittyviä haasteita lieventää Kanbanissa palautesilmukka-käytänteen käyttö, sillä jatkuva avoin vuorovaikutus tiimin jäsenten kesken kehittää yksilön taitoja [20] (Haaste: Y4).

Sosiokulttuurisesta etäisyydestä, läpinäkymättömyydestä ja nopeasti muuttuvista vaatimuksista aiheutuneet ongelmat ratkaistiin sillä, että lyhennettiin sprintit neljästä viikosta kahteen viikkoon ja synkronoitiin kaikkien tiimien sprintit samanaikaisiksi [13] (Haasteet: Y2, Y4, Y5). Hossain ym. [13] tutkivat, miten Scrumin käytänteitä sovellettiin neljässä eri GSD-projektissa. Lyhyemmät sprintit toivat lisää näkyvyyttä, nopeampia toimituksia ja lisäksi kehittäjien osaaminen ketterästä kehityksestä lisääntyi nopeasti (Haaste: Y6). Projektipäällikkö jakoi sprinttikohtaisen työn erillisiin moduuleihin ja kohdisti moduulit tiimeille (Haaste: Y1). Sen jälkeen jokainen tiimi piti oman sprintin suunnittelupalaverinsa.

Tiimit voivat osallistua toistensa päiväpalavereihin, mikä helpttaa luottamuspulaa. Varsinkin vierailut sprintin loppuvaiheessa vähentävät väärinkäsityksiä ja muiden osaamisen kyseenalaistamista (Haaste: Y5, Y6, Y8). Tuotteen ja sprintin kehitysjonot ja edistymiskäyrät ylläpidetään jaetussa työkalussa, mikä lisää luottamusta (Haaste: Y8). Projektien tiedottajat (neljästä eri projektista) totesivat, että tiimeiltä vaaditaan suurta itseorganisoitumiskykyä, mikä nostaa motivaatiota ja tiimityöskentelyn tasoa [13] (Haaste: Y8).

## 4.2 Viestinnän haasteiden ratkaisuja

Scrum tarjoaa mahdollisuuden parantaa hajautetun kehityksen näkyvyyttä, mutta on kriittistä räätälöidä käytänteitä projektin ominaispiirteiden mukaan sekä löytää sopivat yhteistyön työkalut ja periaatteet, jotta hyödyt toteutuisivat [13]. Tietyt Kanbanin elementit (sisäänpääsykriteerit, peruutettu tehtävä, Kanban-taulu, käytänteet, hahmo ja tuotteen kehitysjono) oikein käytettyinä voivat lieventää yhteistyön ja viestinnän haasteita globaalissa ohjelmistokehitysprojekteissa [20]. Scrumban, joka on Scrumin ja Kanbanin parhaiden käytänteiden hybridi, voi lieventää jotain GSD:n haasteista, mutta tarvitaan lisätutkimusta, jotta hajautetun tiimin käyttö olisi järkevämpää kuin paikallisen tiimin [9]. Sähköiset palaverit, tietokonepohjaiset työkalut, joustava työaika ja lokaatiovierailut ovat lisääntyvässä määrin käytössä operatiivisina tukitoimintoina globalisaatiohankkeissa ja organisaatioiden välisessä yhteistyössä [13] (Haaste: V1, V2).

Visuaalinen Kanban-taulu on hyödyllinen viestintäkeino silloin, kun tiimin jäsenet kokevat kasvokkain kohtaamisia olevan liian vähän tai rasitteena on aikaero [20] (Haasteet: V1, V2). Taulusta näkee nopeasti projektin kokonaistilanteen, resurssien käytön ja työnkulun. Lisäksi hyväksytty käytäntö pikaviestityökalujen käytöstä vähentää tarvetta fyysiseen kasvokkain viestintään (Haaste: V2). Pikaviestimiin kirjoitettu keskustelu jää talteen myöhempää tarvetta varten (Haaste: V5).

Projektipäälliköt käyttävät aikaerosta johtuen monenlaisia yhteistyövälineitä ja toimintatapoja sprintin suunnittelupalaverin järjestämiseksi [13]. Hossainin ym. [13] tutkimuksen esimerkkiprojekteissa tiimin jäsenet käyttivät käytännettä lokaatiovierailu osallistuakseen sprintin suunnittelupalaveriin ja sprintin katselmukseen kasvokkain (Haaste: V2). Jokainen tiimi voi pitää erillisen sprintin katselmuksen ja kirjata tulokset projektin jaettuun tietokantaan (Haaste: V1, V2). Jos tiimissä on paljon jäseniä, voidaan retrospektiivi ja sprintin katselmus tehdä vain joka toinen iteraatio.

Lähtökohtaisesti kannattaa tiimit hajauttaa eri aikavyöhykkeille niin vähän kuin mahdollista: mieluummin Pohjois-Etelä -suunnassa läheisille aikavyöhykkeille kuin Itä-Lämsi -suunnassa ajallisesti kauaksi toisistaan. Jos kahdella tiimillä on kahden tunnin aikaero ja molempien tiimien jäsenet käyttävät lounastaukoon neljännen tuntinsa, niin yhteistä työaikaa jää kolme tuntia. Jos molempien tiimien jäsenet jousta-

vat työajoissaan tunnin siten, että toiset tulevat tunnin aikaisemmin ja toiset tunnin myöhempään, saadaan aikaero poistettua kokonaan.

Scrumin kaikille avoin sprintin kehitysjono lieventää kasvokkain viestinnän tarvetta, koska se kertoo tilanteen reaaliajassa: kuka on tehnyt mitäkin, kuka on tekemässä mitäkin ja miten paljon menee vielä aikaa (Haaste: V1, V2). Tämä edellyttää, että jokainen ylläpitää osaltaan sprintin kehitysjonoa aktiivisesti. Päiväpalaverin voi pitää eri tiimien välilllä virtuaalityökalujen avulla ja jatkaa tarvittaessa keskustelua kahden kesken pikaviestimillä tai puhelimitse (Haaste: V1, V5).

Tiimien väliset riippuvuudet minimoidaan siten, että komponettien omistajuus annetaan paikalliselle tiimille [6] (Haaste: Y8). Azoubin ym. [6] tutkimus kannustaa joustavien työaikoihin, säännöllisiin vierailuihin muissa lokaatioissa ja henkilökohtaiseen tiiviiseen viestintään sekä kehittäjien että projektinhallintaryhmän kanssa. Jokaisen tulisi sitoutua palaveriaikatauluihin ja päätöksiin. Sosiaalisten taitojen kehittäminen edistää koordinointia. Tiimin kannattaa vaihtaa itselleen sopiviin työkaluihin; käyttää sekä synkronisia että asynkronisia työkaluja ja käyttää turvallisia jaettuja tietovarastoja (Haaste: V1, V5). Tuoteomistaja on tietämyskeskus, jota pitäisi hyödyntää mahdollisimman paljon: synkronoi viestintä paikallisen tuoteomistajan ja hajautettujen scrummastereiden kanssa (Haaste: V4).

Tiimin viestintää voidaan Azoubin ym. [6] mukaan parantaa rohkaisemalla tiimin jäseniä kasvokkain tapaamisiin jokaisen projektin alussa, kannustamalla osallistumaan sprintin katselmukseen, aikatauluttamalla säännölliset palaverit, käyttämällä selkeitä rooleja ja vastuualueita ja lisäämällä muodollista viestintää. Tutkimus kehottaa välttämään hajautettua pariohjelmointia ja hajautettuja Scrum-tiimejä. Tutkimus kannustaa pienikokoisiin julkaisuihin ja lyhyisiin iteraatioihin.

Kielen murre-eroista johtuvia haasteita voidaan vähentää, kun edistetään Kanban-käytännettä, jossa kysymykset ja ongelmat kirjoitetaan auki Kanban-taululle kaikkien nähtäväksi ja keskusteltavaksi yhteisen konsensuksen saavuttamiseksi [20] (Haaste: V3). Azoubi ym. [6] suosittelevat tutkimuksensa pohjalta kielieroihin seuraavia tekniikoita: rohkaise hajautetun tiimin jäseniä osallistumaan paikallisen tiimin päiväpalaveriin, puhu hitaasti ja selkeästi, käsittele kielikysymystä avoimesti mahdollisimman varhaisessa vaiheessa, käytä vähemmän yksityiskohtaista viestintää, pidä yhteiset tiedot ajan tasalla ja varmista, että palaverin aiheet ovat varmasti kaikkien tiedossa etukäteen (Haaste: V3). Paikallinen scrummaster, joka on kielitaitoinen, vastaa kysymyksiin tuoteomistajan sijaan (Haaste: V4). Viestintätekniikkaa auttaa, kun asynkroniset palaverit tallennetaan, jolloin niihin voi palata jälkikäteen [6, 13] (Haaste: V4, V5). Tutkimus suosittelee pitämään kiinni Scrum-käytännöstä, vaikka on kieliongelmia. Sprintin suunnitteluun voidaan lisätä esi- ja jälkipalavereita, joissa tuodaan yhteiset tavoitteet esiin ja lopussa validoidaan kunkin sprintin sisältö [13] (Haaste: Y7).

# 5 Yhteenveto

Ketterän kehityksen käytänteillä on mahdollista lieventää yhteistyön ja viestinnän haasteita ja edistää projektin onnistumista. Tässä tutkielmassa on selitetty globaalin ohjelmistokehityksen etuja ja haasteita ketterässä kehityksessä. Parhaimmillaan globaali ohjelmistokehitys on työvoiman osalta kustannustehokkaasti hajautettu projekti, joka tuottaa asiakkaalle jatkuvaa lisäarvoa vuorokauden ympäri tapahtuvalla kehitystyöllä.

Tässä tutkielmassa on otettava rajoituksina huomioon, että jokainen ohjelmistokehitystiimi on uniikki ja sen toimintaympäristö ja kehitettävä tuote ovat ainutlaatuisia, joten haasteet ja ratkaisut ovat ainutlaatuisia. Globaalin ja ketterän ohjelmistokehityksen tunnistettuja ristiriitoja ovat myös muutosvetoinen kehitys vs. standardoitu kehitys ja itseohjautuva kehitystiimi vs. ohjattu kehitystiimi, mutta tässä tutkielmassa huomioitiin vain yhteistyön ja viestinnän aiheuttamat ristiriidat. Tutkimusaineistossa on tuloksia 73 tapaustutkimuksesta joko suoraan tai systemaattisten kirjallisuuskatsausten kautta. Kaikki tutkimukset ovat halutussa kontekstissa: globaali ja ketterä ohjelmistokehitys.

Taulukko A.1 on vastaus tutkimuskysymykseen: Miten ketterän kehityksen käytänteet vastaavat GSD-kontekstista nouseviin yhteistyön ja viestinnän haasteisiin? Tämän tutkielman kolme osatavoitetta saavutettiin: kerättiin ja kategorisoitiin tutkimusmateriaalista haasteita, kerättiin ja kohdistettiin haasteisiin empiirisiä ratkaisuja ja lopulta luotiin yhteenvetotaulukko, joka on haluttu tarkistustyökalu ( Liite A). Mikä tahansa ohjelmistoprojekti voi edistää projektin onnistumista käyttämällä työkalua haasteiden havaitsemiseen ja ratkaisemiseen. Ratkaisuja suositellaan sovellettavaksi tarpeen mukaan. Ketterän ohjelmistokehityksen elementit kuten sprintin pituus, retrospektiivien määrä/ajoitus/laajuus, palaverien ajoitus/hajautus/keskitys, tehtävien hajautus/keskitys, roolit ja tehtäväjako sekä työn alla olevien tehtävien lukumäärä ovat kaikki räätälöitävissa jokaisen projektin erityistarpeisiin. Lisäksi käytänteiden synkronointi, tiimin jäsenten vierailut toistensa luona ja joustava työaika tukevat omalta osaltaan tiimityötä. Taulukkoa suositellaan säilytettäväksi ja ylläpidettäväksi jaetussa tietämyskannassa yhtenä projektin dokumenttina.

Ohjelmistokehitys on suunnannäyttäjä monimutkaisuuden hallinnassa. Ketterän kehityksen tulevaisuus näyttää valoisalta, sillä sitä omaksutaan lisääntyvässä määrin sovelluskehityksen ulkopuolelle yrityksen kaikkiin toimintoihin: liiketoiminnan suunnitteluun, markkinointiin, mainontaan, myyntiin ja viestintään. Ketterällä kehityksellä on kasvupotentiaalia myös muille toimialoille kuin ohjelmistokehitykseen. Rahoituspalvelut, konsulttipalvelut, vakuutus ja terveydenhuolto ovat ohjelmistokehityksen jälkeen eniten ketterää kehitystä käyttävät toimialat [3]. Ohjelmistokehitys vakuuttaa suunnannäyttäjänä myös viestinnässä. Koska tehoton viestintä on tiedostettu haaste globaalissa ympäristössä, sen ratkaisemiseksi on jo kehitetty visuaalisia työkaluja ja uusia käytänteitä. Tehokas viestintä korreloi onnistuneen projektin kanssa [22].

# Lähteet


1 "Agile manifesto," http://agilemanifesto.org/iso/fi/manifesto.html, [Viitattu 1.10.2017].

2 "The state of scrum report 2017," https://www.scrumalliance.org/why-scrum/state-of-scrum-report/2017-state-of-scrum, 2017, [Viitattu 3.11.2017].

3 "Versionone releases 11th annual state of agile report," http://www.agile247.pl/wp-content/uploads/2017/04/versionone-11th-annual-state-of-agile-report.pdf, Apr, 6 2017, [Viitattu 1.11.2017].

4 M. P. N. H. af Orns, P. Hynninen, and C. L. T. N. A. Piri, "Practical guide to managing distributed software development projects."

5 O. Al-Baik and J. Miller, "The kanban approach, between agility and leanness: a systematic review," *Empirical Software Engineering*, vol. 20, no. 6, pp. 1861–1897, 2015.

6 Y. I. Alzoubi, A. Q. Gill, and A. Al-Ani, "Empirical studies of geographically distributed agile development communication challenges: A systematic review," *Information & Management*, vol. 53, no. 1, pp. 22 – 37, 2016.

7 S. W. Ambler and M. Lines, "The disciplined agile process decision framework," in *International Conference on Software Quality*. Springer, 2016, pp. 3–14.

8 D. J. Anderson and A. Roock, "An agile evolution: why kanban is catching on in germany and around the world," *Cutter IT Journal*, vol. 24, no. 3, p. 6, 2011.

9 A. Banijamali, R. Dawadi, M. O. Ahmad, J. Similä, M. Oivo, and K. Liukkunen, *Empirical Investigation of Scrumban in Global Software Development*. International Conference on Model-Driven Engineering and Software Development, 09 2017.

10 D. Battiston, J. B. i Vidal, and T. Kirchmaier, "Face-to-face communication in organisations."

11 J. A. Espinosa and E. Carmel, "The effect of time separation on coordination costs in global software teams: A dyad model," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*. IEEE, 2004, p. 10 pp.

12 F. Ghafoor, I. A. Shah, and N. Rashid, "Issues in adopting agile methodologies in global and local software development: A systematic literature review protocol with preliminary results," *International Journal of Computer Applications*, vol. 160, no. 7, 2017.

13 E. Hossain, P. L. Bannerman, and R. Jeffery, "Towards an understanding of tailoring scrum in global software development: a multi-case study," in *Proceedings of the 2011 International Conference on Software and Systems Process*. ACM, 2011, pp. 110–119.

14 N. K. Kamaruddin, N. H. Arshad, and A. Mohamed, "Chaos issues on communication in agile global software development," in *Business Engineering and Industrial Applications Colloquium (BEIAC), 2012 IEEE*. IEEE, 2012, pp. 394–398.

15 C. Larman and B. Vodde, *Large-scale scrum: More with LeSS*. Addison-Wesley Professional, 2016.

16 D. Leffingwell, R. Knaster, and I. Oren, *SAFe® 4.5 Introduction Overview of the Scaled Agile Framework® for Lean Enterprises*, second edition. ed. Sebastopol: O'Reilly, Oct 3, 2017. [Online]. Available: http://replace-me/ebraryid=11263316

17 G. Purna Sudhakar, A. Farooq, and S. Patnaik, "Soft factors affecting the performance of software development teams," *Team Performance Management: An International Journal*, vol. 17, no. 3/4, pp. 187–205, 2011.

18 N. Rashid and S. U. Khan, "Developing green and sustainable software using agile methods in global software development: Risk factors for vendors," in *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*. SCITEPRESS-Science and Technology Publications, Lda, 2016, pp. 247–253.

19 K. Schwaber and J. Sutherland, "The scrum guide," https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf, Jul 2016, [Viitattu 15.10.2017].

20 M. Tanner and M. Dauane, "The use of kanban to alleviate collaboration and communication chanllenges of global software development." *Issues in Informing Science Information Technology*, vol. 14, 2017.

21 C. Ward and A. Kennedy, "The measurement of sociocultural adaptation," *International journal of intercultural relations*, vol. 23, no. 4, pp. 659–677, 1999.

22 M. Wasim Bhatti and A. Ahsan, "Effective communication among globally distributed software development teams: Development of an "effective communication" scale," *Journal of Global Information Management*, vol. 25, pp. 40–62, 07 2017.

23 P. J. Ågerfalk, B. Fitzgerald, H. H. Olsson, and E. Conchir, "Benefits of global software development: the known and unknown," in *International Conference on Software Process*. Springer, 2008, pp. 1–9.

# Liite A. Tarkistustyökalu: taulukko haasteista ja ratkaisuista

Taulukko A.1: Yhteenveto yhteistyön ja viestinnän haasteiden ratkaisuista.

| Viite | Haaste | Ratkaisuja |
|---|---|---|
| Y1 | Johtamistekijät | • tuotteen kehitysjono • sprintin kehitysjono • lokaatiovierailut palavereihin • lokaatiokohtainen tiimi/scrummaster/tuoteomistaja/asiakkaan edustaja • tiimien yhdistetyt kuukausittaiset retrospektiivit ja sprintin katselmus • hahmo (kulunut aika/tarvittava aika) • skaalaus: SAFe®:n hierarkia ja SAFe®:n hanketason toimitusjuna • selkeät roolit ja vastuut • projektipäällikkö modulorisoi sprintin tiimien kesken • Kanban-taulu • peruutettu tehtävä • projektipäällikkö resursoi pullonkaulat |
| Y2 | Läpinäkymättömyys | • tuotteen kehitysjono • sprintin kehitysjono • lyhyet sprintit ja tiimien sprinttien sykronointi • sprintin suunnittelupalaverin tallennus • hahmo • strategiatietoisuuden lisääminen • Kanban-taulu |
| Y3 | Projektin viivästys | • tuotteen kehitysjono • sprintin kehitysjono • Kanban-taulu • maksimityötehtävien säätäminen • resurssien siirtäminen pullonkauloihin |
| Y4 | Vaatimus | • tuotteen kehitysjonon priorisointi • sprintin keskeytys • lyhyet sprintit • projektikohtainen arkkitehti • lokaatiokohtainen tuoteomistaja/asiakkaan edustaja • sisäänpääsykriteerit • palautesilmukka • peruutettu tehtävä |
| Y5 | Kulttuuriero | • retrospektiivi • lokaatiovierailut • kuukausittaiset katselmukset • tiimien sprinttien synkronointi |
| Y6 | Projektityöero | • lyhyet sprintit • lokaatiovierailut palavereihin • Kanban-käytänteissä työohjeet • retrospektiivi |
| Y7 | Maantieteellinen etäisyys | • lokaatiokohtainen scrummaster/tuoteomistaja • etu- ja jälkipalaverit • palaveritallenteet |
| Y8 | Motivaation puute | • tiimin autonomia • projektin jaettu tietämyskanta • lokaatiovierailut • komponettien osaomistajuus |
| V1 | Aikaero | • tuotteen kehitysjono • sprintin kehitysjono • jaettu tietämyskanta • palaveritallenteet • Kanban-taulu |
| V2 | Kasvokkain kohtaamisen puute | • tuotteen kehitysjono • sprintin kehitysjono • lokaatiovierailut päiväpalavereihin • tiimien yhdistetyt kuukausittaiset retrospektiivit ja sprintin katselmus • jaettu tietämyskanta • nauhoitetut palaverit • Kanban-käytänteet • sähköinen ja jaettu Kanban-taulu |
| V3 | Kieliero | • tallenteet • palaverin aiheet etukäteen • Kanban-käytänteet • Kanban-taulu • lokaatiovierailu palavereissa |
| V4 | Asiakkaan läsnäolon puute | • asiakkaan edustaja -rooli • ennakoidut palaverit • palaveritallenteet |
| V5 | Viestintätekniikka | synkroniset: • pikaviestimet • puhelin • videoneuvottelu • palaverit<br>asynkroniset: • palaveritallenteet • jaettu tietämyskanta • palaverin aiheet etukäteen |