Tampereen yliopisto

Tuomas Tuunainen

KETTERÄT MENETELMÄT INTEGRAATIOKEHITYSTIIMEISSÄ

Tapaustutkimus integraatiotiimien ketterien menetelmien käytöstä

Tietojohtaminen
Diplomityö
Maaliskuu 2020

TIIVISTELMÄ

Tuomas Tuunainen: Ketterät menetelmät integraatiokehityksessä
Diplomityö
Tampereen yliopisto
Tietojohtaminen
Maaliskuu 2020

Toimittajayritys, joka myy integraatiotuotantoa asiakkailleen, haluaa pitää asiakastyytyväisyyden hyvänä, henkilöstön vaihtuvuuden pienenä ja käyttöasteen korkeana. Voidaan olettaa, että tätä ongelmaa voidaan osaltaan ratkaista tiimien toimintatapoja kehittämällä. Ketterät menetelmät on todettu hyväksi keinoksi ohjelmistokehitystiimien toiminnan kehittämiseksi, mutta integraatiotuotanto kuitenkin eroaa perinteisestä ohjelmistokehityksestä niin paljon, että näiden menetelmien käyttö sellaisenaan voi olla vaikeaa tai mahdotonta. Tutkimuksen päämotivaationa olikin selvittää tarkemmin, millä tavoin ketterät menetelmät voisivat auttaa näiden tavoitteiden saavuttamiseksi ja kehittää integraatiotiimien toimintamalleja.

Tämä diplomityö on tapaustutkimus, jonka tarkoituksena on selvittää, millä tavoin kohdeorganisaation integraatiotiimeissä hyödynnetään ketteriä menetelmiä, millaisia haasteita niihin liittyen on havaittu sekä miten ketterien menetelmien avulla voitaisiin vastata näihin haasteisiin. Toiveena on, että tutkimuksen tuloksia voidaan hyödyntää jatkossa integraatiotiimien toimintatapojen kehittämisessä ja ihmisten työn mielekkyyden ja viihtyvyyden parantamisessa.

Tutkimus koostuu kahdesta osasta. Ensimmäinen osa on teoreettinen kirjallisuuskatsaus, jossa tutustutaan yleisimpiin ketteriin menetelmiin, integraatiotuotantoon ja sen erityispiirteisiin sekä haasteisiin. Tutkimuksen empiirinen aineisto kerättiin kahdella menetelmällä. Aluksi kerättiin tietoa case-organisaatiosta havainnoimalla, jonka avulla saatiin tietoa organisaation tilasta ja haasteista sekä tavoitetilasta. Näiden tulosten avulla pystyttiin muotoilemaan tarkempi haastattelurunko, jonka avulla haastateltiin asiantuntijoita monipuolisista taustoista. Haastatteluilla selvitettiin case-organisaation työntekijöiden kokemuksia ja näkemyksiä ketterien menetelmien haasteista, käytöstä sekä mielipiteitä niiden soveltuvuudesta integraatiotuotantoon. Kun lopuksi vastauksia analysoitiin ja yhdistettiin teoriaan, saatiin lopputuloksena joukko suosituksia, jotka perustuvat sekä haastateltavien näkemyksiin että kirjallisuussuosituksiin.

Tutkimuksen tulokset antavat ymmärtää, että yksikään tutkituista ketteristä menetelmistä ei sellaisenaan toimi integraatiotiimien toiminnassa kovinkaan hyvin. Sen sijaan jokaisesta menetelmästä voidaan poimia tekniikoita, joita on mahdollista hyödyntää integraatiotiimien arjessa. Voidaan todeta, että näiden tulosten avulla on mahdollista saavuttaa ketterien menetelmien etuja, jotka voivat parantaa integraatiotiimien toimintaa ja tuoda hyötyjä kaikille osapuolille.

Avainsanat: integraatiotuotanto, ketterät menetelmät, toimintatapojen kehitys

ABSTRACT

Tuomas Tuunainen: Development of integration team practices
Master of Science Thesis
Tampere University
Information and knowledge management
March 2020

A company that sells integration development to their customers aims for high customer satisfaction, low staff turnover and high utilization rates. It can be assumed that this challenge can be partly addressed by improving the ways of working in integration teams. Agile methodologies are proven to be effective in improving software development teams' ways of working, but integration development differs from traditional software development so much, that using these methodologies as-is can prove to be difficult or impossible. The main motivation for this thesis is to find out how agile methodologies can help a company with these challenges by improving the ways of working in integration teams.

This study is a case-study, that aims to find out how integration teams in the case organization utilize agile methodologies, what challenges have been perceived when using them and how these agile methodologies could be used to overcome these challenges. It is desirable that results of this study could be used in the future to improve the ways of working in integrations teams and increase the job meaningfulness and job satisfaction.

The study consists of two parts. First part is a theoretical literature review, where the reader is introduced into the most common agile methodologies, integration development and its special features and challenges. Empirical material was gathered using two methods. In the beginning information from the case-organization was gathered via observation, which helped in understanding the state of the case organization, its challenges and target state. Using the results from observation, the interview structure was created that was used to interview specialists from wide backgrounds. The interviews were used to gather experiences and views of the employees in the case organization about agile methodologies, for example challenges, usage and opinions about how they could be utilized in integration development. Afterwards the answers were analyzed in comparison with the literature, and a group of recommendations were created that are based on the interviewees' views and recommendations from the literature.

Results of this study suggest that none of the studied agile methodologies can be used as-is in the context of integration teams. Instead, individual techniques from each methodology can be picked that can be utilized in the work of the integration teams. We can conclude that with these results it is possible to reach benefits that can improve the ways of working in integration teams and benefit all parties whom it concerns.

Keywords: integration development, agile methodologies, improving ways of working

ALKUSANAT

Tätä työtä aloittaessani olin ollut jo työelämässä hyvän aikaa, ja ainoastaan lopputyö oli jäljellä valmistumisen tiellä. Halusin diplomityöni tukevan jollain tavalla työtehtäviäni ja ammatillisia mielenkiinnonkohteitani, jonka vuoksi olinkin tyytyväinen, kun työnantajani kanssa käytyjen keskustelujen tuloksena aiheeksi muodostui sellainen, joka mielestäni täytti nämä kriteerit. Haluan kiittää työnantajaani siitä, että tällainen mahdollisuus tarjoutui ja sain toteuttaa diplomityön itseäni kiinnostavasta ja lähellä olevasta aiheesta.

Erityisesti haluan kiittää diplomityöni ohjaajaa Samuli Pekkolaa työn ohjaamisesta ja avusta, joka lopulta teki sen toteuttamisen mahdolliseksi. Sain matkan varrella häneltä paljon hyviä neuvoja ja vinkkejä, joka ohjasi työtä lopulliseen muotoonsa. Haluan kiittää myös kaikkia tutkimukseeni osallistuneita työtovereita antoisista haastatteluista sekä yleisistä keskusteluista, joista olen saanut paljon hyviä ideoita työhöni. Lisäksi haluan kiittää ystäviäni, joilta olen saanut hyvää vertaistukea ja kiritysapua matkan varrella.

Tampereella, 7.3.2020

Tuomas Tuunainen

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

API	Application Programming Interface, ohjelmointirajapinta
EAI	Enterprise Application Integration, järjestelmäintegraatio
PO	Product Owner, rooli Scrumissa
SM	Scrum Master, rooli Scrumissa
XP	eXtreme Programming, Agile-viitekehys
TDD	Test Driven Development, ohjelmistokehitysmenetelmä
IP	Intellectual Property
WIP	Work in Progress

1. JOHDANTO

Monet ohjelmistoalan toimittajayritykset myyvät ohjelmistokehitystä tuntityönä, jossa tärkeää on pystyä toimittamaan laadukasta koodia kustannustehokkaasti ja tehokkaasti. Asiantuntijapalveluihin ja projektiliiketoimintaan keskittyneessä toimittajayrityksessä liiketoimintaperiaatteena on myydä ihmisten osaamista, ja tärkein seurattava asia tämän tyyppisessä yrityksessä on kapasiteetin käyttöaste työntekijöiden ollessa tuotantoresursseja (Karvonen 2004). Integraatioratkaisujen tarjoaminen onkin hyvin pitkälti asiantuntijapalveluiden myyntiä, kuten konsultointia ja toteutusta (Solita 2019). Sen vuoksi on luonnollista, että tämän tyyppisessä toimittajayrityksessä halutaan pitää ihmisistä erityistä huolta, koska he ovat päätekijä millä yritys pystyy kilpailemaan markkinassa. Koska projektiorientoituneen liiketoimintamallin, kuten IT-konsultoinnin, pääasiallinen myynnin lähestymistapa on kumppanuusmalli ja ansaintalogiikka suoriteperusteinen (Karvonen 2004), on tärkeää, että henkilöstö on osaavaa ja motivoitunutta. Kun yritys miettii tehokkuutta ja kustannusten minimointia, sen rinnalla on otettava huomioon myös inhimillinen näkökulma, kuten työntekijöiden viihtyvyys ja työssä jaksaminen. Lisätään yhtälöön alalla vallitseva kilpailutilanne sekä asiakkaista että työntekijöistä, asettaa se haasteita älykkäille työnteon tavoille, joilla voidaan täyttää nämä vaatimukset.

Integraatiotuotantoa ostavan asiakasyrityksen intresseissä on saada riittävän laadukasta koodia mahdollisimman kustannustehokkaasti, jotta kehityksen ulkoistaminen toimittajalle on kannattavaa omien työntekijöiden palkkaamisen sijasta. Usein se tarkoittaa työn ostamista tuntityönä, jolloin ainoastaan tehdyistä työtunneista maksetaan toimittajalle. Tätä varten toimittajayrityksen on pystyttävä tarjoamaan tarpeen mukaan riittävästi resursseja.

Toimittajayritys joutuu hankalaan tilanteeseen siinä vaiheessa, kun asiakkaalla ei ole tarvetta niin suurelle määrälle työtä, että se työllistäisi kokonaisia (työntekijä)resursseja. Koska toimittaja kuitenkin haluaa tehdä liiketoimintaa kyseisen asiakkaan kanssa, ei yksittäiselle asiantuntijalle riitä kyseisessä asiakkuudessa täysiä työtunteja. Toimittajayrityksen intresseissä on kuitenkin pitää kaikki työntekijänsä niin lähellä täyttä allokaatiota kuin mahdollista, jotta liiketoiminta olisi mahdollisimman kannattavaa. Tällöin joudutaan jakamaan yksittäisten työntekijöiden työaikaa useiden asiakkaiden kesken, josta seuraa ilmeisiä ongelmia. Jos työntekijän työaika olisi jaettu esimerkiksi kahden asiakkaan kesken, mutta jostain syystä molemmilla työkuorma kasvaa yhtä aikaa, tai kääntäen laskee yhtä aikaa, tarkoittaa se ongelmia joko yli- tai aliresursoinnin näkökulmasta. Huomioon

tulee ottaa myös esimerkiksi työntekijöiden sairastumiset ja lomat, joihin olisi hyvä varautua etukäteen. Toimittaja ei myöskään halua työntekijöidensä lähtevän yrityksestä esimerkiksi liian stressaavan työn takia, koska uuden työntekijän rekrytointi ja koulutus on kallista ja aikaa vievää puuhaa.

Työntekijän intresseissä taas olisi pystyä keskittymään mahdollisimman rauhassa siihen, mistä hän pitää ja mihin hänet on palkattu, eli tässä tapauksessa integraatioiden kehittämiseen. Tällöin työntekijän harteilta pitäisi poistaa stressi esimerkiksi asiakkuuksien välillä priorisoinnista ja kontekstinvaihdosta, jonka lisäksi työpaikan tulisi olla viihtyisä ja työ ei saisi olla muutenkaan liian kuormittavaa, jos hänet halutaan pitää tyytyväisenä ja tuottavina.

Ratkaistava haaste voidaan ajatella kolmiona, joiden kärjissä on asiakas, työnantaja ja työntekijä (tässä tapauksessa integraatioasiantuntija). Jokaisella näistä osapuolista on omat intressinsä, joita he tyypillisesti pyrkivät optimoimaan. Näiden kolmen osapuolen intressien ristiriidan optimointi ketterän ohjelmistotuotannon menetelmien avulla on päämotivaatio tämän tutkimuksen toteutukselle. Todellisuudessa edellä mainittuihin seikkoihin vaikuttaa moni muukin asia, kuten yleinen markkinatilanne, kilpailu, myynti, palkkaus ja monet muut asiat, joihin tässä työssä ei kuitenkaan oteta kantaa.

Ketterät menetelmät on kehitetty taklaamaan useita näistä ongelmista. Ongelmana on kuitenkin se, että usein ne keskittyvät puhtaaseen ohjelmistotuotantoon ja/tai tuotekehitykseen, eivätkä ratkaise integraatiotuotannossa ilmeneviä ongelmia. Vaikka integraatiotuotannossa painitaan usein samojen ongelmien kanssa, integraatioiden tuottamisessa on sellaisia erityispiirteitä, jotka asettavat työn tekemiseen, ohjaamiseen, suunnitteluun ja resursointiin omia haasteitaan. Tässä työssä esitellään yleisimmät ohjelmistotuotannon ketterät menetelmät, sekä millä tavoin niitä kannattaa hyödyntää integraatiokehityksessä ja millä tavoin taas ei.

Lisämotivaationa tutkimukselle toimii se, että kirjallisuudesta ei tällä hetkellä löydy integraatiotuotantoa varten kehitettyä viitekehystä. Useimmat ketteristä menetelmistä olettavat, että kehitystiimillä on täysi auktoriteetti ohjelmiston tai tuotteen kehitykseen, ja sidosryhmien määrä on kohtuullisen pieni. Integraatiotuotannossa asia on kuitenkin täysin päinvastoin, sillä kehitystiimillä harvoin on valta tehdä yksin integraatiota alusta loppuun, vaan lähes aina esiintyy kriittisiä riippuvuuksia useisiin eri sidosryhmiin ja osapuoliin. Tämä johtuu siitä, että integraatiolla yleensä halutaan saada kaksi tai useampia järjestelmiä keskustelemaan keskenään, joiden kanssa integraatiotiimillä ei kuitenkaan välttämättä ole mitään tekemistä.

Edellä mainittujen syiden vuoksi tässä työssä tutkitaan, miten yleisimpiä ketteriä menetelmiä hyödyntämällä voidaan kehittää integraatiotiimien toimintaa, sekä mitkä menetelmät eivät välttämättä integraatiotiimeissä toimi. Lopputuloksena on saavutettu teoreettinen malli, jota voidaan käyttää integraatiotiimien toiminnan kehittämisessä.

1.1 Tutkimusongelma ja -kysymykset

Tutkimusongelmana on selvittää, miten integraatiotiimien toimintamalleja olisi mahdollista kehittää ketterän ohjelmistokehityksen menetelmien avulla. Diplomityön pääkysymys on johdettu suoraan tutkimusongelmasta, eli miten ketterän kehityksen keinoja käyttämällä voitaisiin kehittää integraatiotiimien toimintaa. Pääkysymyksen lisäksi työssä on kolme alakysymystä:

1. Millaisia haasteita integraatiotekemisessä on
2. Mitkä ketterät menetelmät sopivat integraatiokehitykseen
3. Millä eri tavoin kukin osapuoli hyötyy integraatiotiimin ketteristä menetelmistä

Tutkimuksen tavoitteena on selvittää, millä tavoin integraatiotiimeissä käytetään ketteriä menetelmiä ja miten niitä kannattaisi käyttää, jotta integraatiotiimeissä työskentelevien henkilöiden arjesta saadaan entistä sujuvampaa ja mielekkäämpää sekä luoda lisäarvoa asiakkaille ja työnantajayritykselle.

1.2 Työn rakenne

Työ koostuu seitsemästä kappaleesta. Johdannossa taustoitetaan tutkimuksen syy, tutkimusongelma sekä tutkimuskysymykset. Toisessa kappaleessa on kirjallisuuden perusteella kerätty teoriaosuus, jonka tarkoituksena on antaa lukijalle riittävä ymmärrys sekä ketteristä menetelmistä että integraatiotyöstä, jotta tutkimuksen tuloksia voi arvioida riittävän kriittisesti. Kolmannessa kappaleessa esitellään käytetyt tutkimusmenetelmät, tutkimusaineisto sekä case-organisaatio. Neljännessä kappaleessa kuvataan tapaustutkimuksen taustana toimiva nykytilanne sekä havaintoja case-organisaatiosta suhteessa työn aiheeseen. Viides kappale sisältää haastattelujen tulokset. Kuudes kappale sisältää työn tulokset ja pohdintaa. Viimeisenä, seitsemäntenä kappaleena löytyy yhteenveto, jossa tiivistetään työn johtopäätökset sekä arvioidaan työn luotettavuutta ja jatkotutkimustarpeita.

2. KETTERÄT MENETELMÄT JA INTEGRAATIOTUOTANTO

Ketterä ohjelmistokehitys, eli Agile, on joukko menetelmiä ja metodologioita, mutta myös arvoja ja periaatteita, jotka auttavat ohjelmistokehitystiimiä työskentelemään tehokkaammin ja tekemään parempia päätöksiä. Kaikki menetelmät pohjaavat Agilen perusarvoihin, jotka arvostavat yksilöitä ja kanssakäymistä, toimivaa ohjelmistoa, asiakasyhteistyötä ja muutokseen vastaamista ohjelmistokehityksessä (Agile Manifesto 2001)

Perinteisesti ohjelmistokehitysprojekteissa on käytetty niin sanottua vesiputousmallia, jossa projekti etenee vaatimusmäärittelystä kehitykseen, testaukseen ja tuotantoon (Balaji 2012). Käytännössä on kuitenkin havaittu, että vaatimusten määrittely riittävän tarkasti etukäteen on mahdotonta. Määrittelijällä voi kyllä olla ajatus päässään siitä mitä halutaan tehdä, mutta sen kommunikointi täydellisesti kehitystiimille on todennäköisesti mahdotonta (Cockburn 2000). Cockburn ehdottaakin, että sen sijaan että vaatimusmäärittelyistä yritettäisiin heti alussa tehdä kaikille ymmärrettäviä, tulisi määrittelyn olla sillä hetkellä riittävällä tasolla sille tarkoitetulle yleisölle. Kun projekti etenee, saadaan lisää tietoa projektista ja sen haasteista, jolloin vaatimuksia voidaan jälleen tarkentaa kohdeyleisölle. Täten ollen projektin tehtävänä ei ole täydellinen kommunikointi, vaan epätäydellisen kommunikoinnin hallitseminen (Cockburn 2000). Ketterän ohjelmistokehityksen (Agile) ajatuksena onkin pilkkoa monimutkaiset ohjelmistoprojektit pienempiin, hallittavissa oleviin osasiin siten, että alussa ei jouduta sitoutumaan epätäydellisen tiedon varassa tehtyihin päätöksiin ja määrittelyihin (Cockburn 2000).

Ketterän kehityksen viitekehyksissä kehitystyötä tehdään yleensä iteroiden. Ajatuksena on, että iteraatioiden välissä pysähdytään arvioimaan aikaansaannoksia, ja tarvittaessa muutetaan määrittelyitä ja suuntaa. Tällä tavalla asiakas pääsee vaikuttamaan nopealla syklillä siihen, mitä tehdään, ja ohjaamaan tekemistä eniten arvoa tuottavaan suuntaan. Koska kehitystiimi ja asiakas ovat suorassa yhteydessä, kehitystiimi saa suoraa palautetta asiakkaalta, jonka ansiosta säästytään arvailulta ja turhalta työltä (Balaji & Murugaiyan 2012)

Jokainen ketterä menetelmä koostuu useista eri tekniikoista tai käytännöistä, joita noudattamalla viitekehyksiä voidaan ottaa käyttöön helposti ja yleensä kohtuullisen pienillä muutoksilla. Tällä tavoin toimien on tarkoitus välttyä muutosvastarinnan aiheuttamalta vastustukselta (Stellman et al. 2014). Agile on kuitenkin muutakin kuin kasa tekniikoita ja käytänteitä. Tätä osaa voidaan kuvailla Agile-mielenlaaduksi (Denning 2019). Jotta

ketteriä menetelmiä voidaan käyttää tehokkaasti, se vaatii tiimiltä oikeanlaisen asenteen ja yhteisen mielentilan suhteessa tekemiseen. Käytännössä Agile-mielentilalla tarkoitetaan sitä, että ihmiset ovat avoimia tiedon jakamiselle, yhdessä suunnittelulle ja tekemiselle ja prosessien jatkuvalle kehittämiselle (Denning 2019). Oikeanlaisessa mielentilassa ihmiset ohjautuvat itsenäisesti miettimään asiakasarvon luomista, arvostavat yhteistyötä ja verkostoitumista muiden kanssa sekä sitoutuvat ja ottavat vastuuta tiimin tuotoksista. Tällainen tiimi pystyy tutkitusti muokkautumaan nopeasti muuttuviin markkinavaatimuksiin ja toimimaan tehokkaasti muuttuvissa ympäristöissä (Denning 2019). Useimmiten haasteet ketterien menetelmien käyttöönotossa johtuvatkin vääränlaisesta mielenlaadusta, sillä vaikka ketteriä tekniikoita ja työkaluja otettaisiin käyttöön, ilman oikeanlaista mielenlaatua tiimin sisällä ei päästä toivottuihin hyötyihin ja tuloksiin. (Stellman et al. 2014)

Collabnetin (2019) tutkimuksen mukaan tänä päivänä ohjelmistokehityksessä käytetyimmät Agile-menetelmät ovat Scrum, Kanban, XP sekä Lean development. Näistä Scrumin ja Kanbanin on todettu olevan ylivoimaisesti tunnetuimmat ja käytetyimmät menetelmät (Lamelas 2018, Stellman et al. 2014). Jokaisella menetelmällä on omat vahvuutensa ja heikkoutensa, jonka vuoksi jokainen niistä soveltuu parhaiten hieman erilaisiin tilanteisiin. Yhteisenä tekijänä eri ketterillä menetelmillä on kuitenkin kyky vastata kesken projektin muuttuviin vaatimuksiin, sekä perimmäisenä ajatuksena kaikissa on jatkuva arvon tuoton maksimointi asiakkaalle (Lamelas 2018). On kuitenkin hyvä huomata, että ohjelmistokehityksessä ketteriä menetelmiä käytetään hyvin harvoin niiden puhtaassa muodossa. Sen sijaan eri menetelmistä poimitaan parhaita käytänteitä tai tekniikoita, joita sovelletaan tilanteen parhaaksi vaatimalla tavalla, jolloin puhutaan hybridimenetelmistä (Solinski & Petersen 2016).

Tässä kappaleessa esitellään näiden neljän yleisimmän menetelmän tekniikat ja niiden erityispiirteet, sekä kuinka niitä voidaan käytännössä soveltaa ohjelmistokehityksessä.

2.1 Scrum

Scrum on ylivoimaisesti tunnetuin ja käytetyin Agile-menetelmä ohjelmistokehityksessä. CollabNetin (2019) tutkimuksessa tutkittiin Agilea käyttävien organisaatioiden käyttämiä menetelmiä, ja tutkimuksen mukaan 54% vastaajista käyttää Scrumia ohjelmistokehitysprosesseissaan. Useat käyttävät myös jonkunlaista Scrumin hybridiversiota, jossa on yhdistelty Scrumin tekniikoita muiden metodologioiden kanssa. Yhteensä Scrumia ja Scrum-hybridimenetelmiä käyttää 78% vastaajista. (Collabnet 2019).

Scrumin arvoja ovat sitoutuminen, rohkeus, keskittyminen, avoimuus, kunnioitus sekä itseohjautuvuus (Stellman & Greene 2014, Schwaber & Sutherland 2017). Scrumin onnistuminen riippuu siitä, kuinka hyvin ihmiset pystyvät omaksumaan nämä arvot ja toimimaan arjessa niiden mukaan (Schwaber & Sutherland 2017). Scrumin arvojen omaksumisessa tärkeää on tiimin itseohjautuvuus. Scrumin teoria ja tekniikat on helppo oppia ja ottaa käyttöön, mutta oikeanlainen mielentila ja kyky sitoutua toimittamaan jotain konkreettista rajatussa ajassa vaatii tiimiltä kykyä toimia todellisesti yhtenä yksikkönä joukon yksilöitä sijaan. Tehokkaat Scrum-tiimit esimerkiksi jättävät päätökset viimeiseen mahdolliseen hetkeen, jolloin he voivat jättää vaihtoehdot mahdollisimman avoimiksi, joka tekee muutokseen sopeutumisen helpommaksi. On myös tärkeää, että tiimi on motivoitunut arvon toimittamiseen. (Stellman & Greene 2014)

Scrum-tiimissä työntekijät jaetaan kolmeen rooliin: Product Owner (PO), Scrum Master (SM) sekä kehitystiimi (Schwaber & Sutherland 2017). PO:n tehtävä on nimensä mukaisesti toimia ”tuotteen” omistajana, eli ylläpitää ja priorisoida tehtävälistaa (Product Backlog) vaatimuksista ja toiminnoista, joita tuotteeseen tulee kehittää. PO toimii linkkinä eri sidosryhmiin, joilla on intressejä tiimin rakentamaan tuotteeseen, ja edustaa heidän näkökantojaan tuotteen kehityksessä. PO:n tehtävä on varmistaa, että kehitystiimi pystyy tuottamaan maksimaalista arvoa, ja he tietävät mitä heiltä kulloinkin odotetaan (Jocham & McGreal 2018). Yksi PO:n tärkeimmistä ja vaikeimmista tehtävistä onkin pitää tiimi motivoituneena ja keskittyneenä arvon luontiin auttamalla heitä ymmärtämään kenelle ja miksi ohjelmistoa kehitetään (Stellman & Greene 2014). Hyvän PO:n löytäminen onkin yksi vaikeimmista asioista Scrumin käytössä. Tehtävään tarvitaan henkilö, jolla on auktoriteettia ja halua tehdä päätöksiä bisneksen edustajien puolesta. Hänen tulisi pystyä tekemään päätöksiä itsenäisesti ja kommunikoimaan ne tiimille. PO:n tehtävä on kerättävä paljon tietoa eri sidosryhmiltä ja keskusteltava tiimin jäsenten kanssa siitä, mitä tulee tehdä (Stellman & Greene 2014, Schwaber & Sutherland 2017).

Scrum Masterin tärkein tehtävä on pitää huoli siitä, että kaikki tiimin jäsenet pitäytyvät oikeaoppisesti Scrumissa. SM pyrkii poistamaan kehitystiimin tiellä olevat esteet, ja fasilitoi Scrum-palaverit ja muut tapaamiset (Stellman & Greene 2014). Kehittäjät puolestaan työskentelevät ”tuotteen” parissa ja toteuttavat ominaisuuksia ja toiminnallisuuksia, jotka PO on määritellyt yhdessä eri sidosryhmien kanssa. Scrum ei määrittele mitä rooleja tiimin kehittäjillä tulisi olla, kuten esimerkiksi testaaja, arkkitehti ja ohjelmoija, vaan ajatuksena on, että tiimi itseorganisoituu, miten parhaaksi näkee. (Stellman & Greene 2014, Schwaber & Sutherland 2017)

Ohjelmistoa rakennetaan aikarajoitetuissa iteraatioissa, joita kutsutaan sprinteiksi. Jokaisen sprintin alussa tiimi yhdessä suunnittelee sprintin sisällön (sprint planning), jossa

backlogilta valitaan toiminnallisuudet, jotka sprintin aikana kehitetään. Tätä listaa kutsutaan sprint backlogiksi, jonka parissa tiimi työskentelee seuraavan sprintin ajan ja sitoutuu kaikkien siinä olevien tehtävien valmiiksi saamiseen sprintin aikana. (Stellman & Greene 2014, Schwaber & Sutherland 2017)

Päivittäin pidetään lyhyt palaveri, Daily Scrum, jossa jokainen tiimin jäsen päivittää muulle tiimille kehityksen tilanteen ja näkemyksensä tulevista haasteista. Jokainen jäsen vastaa kolmeen kysymykseen: Mitä olen tehnyt viime Dailyn jälkeen? Mitä teen seuraavaan Dailyyn mennessä? Mitä esteitä minulla on tielläni? (Schwaber & Sutherland 2017)

Sprintin lopussa järjestetään Sprint Review, jossa toteutetut ominaisuudet esitellään PO:lle ja muille sidosryhmille. Tiimi järjestää myös retrospektiivin, jossa mietitään sprintin aikana opittuja oppeja, jotta he voivat parantaa toimintatapojaan ja esiin tulleita haasteita. (Stellman & Greene 2014, Schwaber & Sutherland 2017)

Scrumissa tarkoituksena on tuottaa joka sprintin päätteeksi toimivaa ohjelmistoa, joka itsenäisenä kokonaisuutena tuottaa asiakkaalle arvoa (Stellman & Greene 2014, Schwaber & Sutherland 2017).

2.2 Kanban

Kanban on alun perin lähtöisin autoteollisuudesta, jossa se syntyi 1940-luvulla ja on ollut yleisesti käytössä siitä lähtien. Sen tarkoituksena oli tehostaa Toyotan tehtaan tuotantoa tuomalla Lean- sekä just-in-time ajattelua tuotannon ohjaamiseen. Kanbanin nimi tulee kanban-lapuista, joita käytetään tuotannon imuohjauksessa. (Stellman & Greene 2014)

Ohjelmistokehityksessä Kanban on menetelmä jatkuvaan prosessien ja työtapojen kehittämiseen, joka perustuu Lean-arvoihin ja Lean-ajatteluun (Stellman & Greene 2014). Kanbania voidaan kuvailla strategiaksi, jolla optimoidaan prosessin arvovirtaa käyttämällä visuaalista, yhtäaikaisesti työn alla olevien tehtävien suhteen rajoitettua pull-järjestelmää (Vacanti & Yeret 2019). Kanbania voidaan käyttää apuna Lean-ajattelun tuomisessa organisaatioon. Kanbanin ajatuksena on auttaa tiimiä kehittämään tapoja ja prosesseja, joilla he tuottavat ohjelmistoja, poistamaan turhaa työtä ja kehittämään ajattelua arvoa tuottavampaan suuntaan. Kanbania käyttävillä tiimeillä on selvä kuva siitä, mitä toimintoja heidän ohjelmistotuotantoprosessissaan on, mitä riippuvuuksia heidän tekemisillään on muihin sidosryhmiin, missä he törmäävät hukkaan ja epätehokkuuteen sekä miten tekemistä voidaan kehittää ajan myötä puuttumalla näiden juurisyihin (Stellman & Greene 2014).

Kanbanin käyttöönotto on usein helppoa, eikä vaadi suuria muutoksia tiimin nykytilaan, joka voi olla hyvä asia kovin muutosvastarintaisessa tiimissä. Tämä johtuu siitä, että

Kanban ei ole menetelmä projektien hallitsemiseksi, vaan sen tarkoituksena on kehitystiimin prosessien tehostaminen niiden visualisoinnin, mittaamisen ja jatkuvan parantamisen keinoin. Stellmanin ja Greenen (2014) mukaan Kanbanin käyttöönotossa tulisi noudattaa seuraavia perusperiaatteita:

1. Aloita siitä mitä teet jo nyt
2. Sopikaa yhdessä, että tavoittelette jatkuvaa, evolutionääristä muutosta
3. Alussa pitäkää kiinni nykyisistä rooleista, vastuista ja titteleistä

Tämän jälkeen tulee ottaa käyttöön Kanbanin perustekniikat. Kaikkia ei kuitenkaan tarvitse ottaa käyttöön samalla kertaa, vaan muutosvastarinnan minimoimiseksi tekniikoita voidaan implementoida pikkuhiljaa (Stellman & Greene 2014). Yleensä ensimmäisenä tekniikkana otetaan käyttöön työnkulun visualisointi, joka usein toteutetaan kanban-taulun avulla. Visualisoinnin avulla tiimi kykenee näkemään koko ohjelmistotuotantoprosessin alusta loppuun, joka auttaa esimerkiksi hukan ja pullonkaulojen tunnistamisessa ja tukee jatkuvan kehityksen periaatetta (Stellman & Greene 2014).

Toinen Kanbanissa yleisesti käytetty tekniikka on WIP:in (Work in Progress) rajoittaminen. WIP-raja otetaan usein käyttöön samalla kertaa kanban-taulun kanssa (Stellman & Greene 2014). Rajan idea on siinä, että koska todellisuudessa työntekijä voi tehdä vain yhtä asiaa kerrallaan tehokkaasti, tulee yhtä aikaa työn alla olevien tehtävien määrää rajoittaa, jottei työn tehokkuus kärsi monen samanaikaisesti tehtävän välillä hyppimisen vuoksi. Tästä seuraa luontaisesti se, että myös tiimillä voi olla kullakin ajanhetkellä vain rajoitettu määrä asioita kerrallaan työn alla ennen kuin työn tehokkuus alkaa kärsimään kontekstinvaihdon seurauksena, joten on järkevää rajoittaa tällaisella säännöllä yhtäaikaisten työtehtävien määrää (Stellman & Greene 2014).

Kanbanin keskeinen käsite on virta ja sen hallitseminen. Virta on arvon liikkumista prosessin läpi, kuten esimerkiksi integraation valmistuminen määrittelystä toteutukseen, testaukseen ja tuotantoon. Kanban pyrkii optimoimaan tätä arvovirtaa parantamalla prosessin tehokkuutta, vaikuttavuutta ja ennustettavuutta (Vacanti & Yeret 2019). Virran mittaukseen käytetään neljää mittaria: Work in Progress (WIP), kiertoaika, tehtävän ikä ja suoritusteho. WIP tarkoittaa tehtävien määrää, jotka on aloitettu mutta jotka eivät ole vielä valmiita. Kiertoaika lasketaan siitä hetkestä, kun tehtävän tekeminen alkaa, siihen kun se on valmis. Tehtävän ikä on se aika, mikä tehtävän aloituksesta on kulunut tähän hetkeen, ja pätee vain työn alla oleviin tehtäviin. Suoritusteho taas kuvaa kuinka monta tehtävää saadaan valmiiksi tietyssä aikayksikössä. (Vacanti & Yeret 2019) Virran hallitsemisessa hyödynnetään matemaattista mallia, Littlen lakia, joka linjaa, että mitä enemmän jonopohjaisessa järjestelmässä on keskimäärin samanaikaisia tehtäviä työn alla

(WIP), sitä kauemmin jokaisen tehtävän suoritus keskimäärin kestää (Little 2011, Vacanti & Yeret 2019). Tämän vuoksi WIP:in rajoittaminen on keskeinen osa Kanbanin ajatusta (Vacanti & Yeret 2019).

Kanbanin periaatteiden mukaan prosessien sääntöjen tarkka kirjaus on tarpeellista, kuten esimerkiksi valmiin määritelmä ja prosessin kulun tarkka dokumentointi. Stellmanin ja Greenen (2014) mukaan jokaisella ohjelmistoja rakentavalla tiimillä on systeemi, jota se noudattaa. Vaikka se olisi miten kaoottinen ja muuttuva tahansa, aina on olemassa jonkunlainen yhteinen ymmärrys siitä, kuinka tiimin ohjelmistotuotantoprosessi toimii. Kun prosessi kirjoitetaan auki luettavaan muotoon, on sitä huomattavasti helpompi lähteä kehittämään ja parantamaan. Sen lisäksi prosessien dokumentointi auttaa yksittäistä kehittäjää ymmärtämään oman työnsä merkityksen osana laajempaa systeemiä, ja näin ollen tuo lisää merkitystä työhön. (Stellman & Greene 2014).

Kanbanista löytyvät myös tekniikat ”paranna kollaboroimalla, evolvoi kokeilemalla” sekä ”implementoi palautesilmukoita”, jotka ovat läheisesti yhteydessä toisiinsa. Ensinä mainittu tekniikka tulee suoraan Kanbanin tavoitteesta tehdä jatkuvia pieniä, parantavia muutoksia systeemiin, jotka evolvoivat systeemiä paremmaksi pala kerrallaan. Kuitenkin, jotta voidaan saada tietoa prosessin nykytilasta, tulee systeemiin implementoida palautesilmukoita, joiden avulla voidaan mitata systeemin tilaa ja näin ollen tehdä parantavia toimenpiteitä (Stellman & Greene 2014). Kanbanin jatkuvan parantamisen ideana on löytää jatkuvasti toistuvia ongelmia, päätellä mitä yhteistä niissä on ja valita oikeat työkalut niiden korjaamiseksi (Stellman & Greene 2014).

2.3 XP

XP:n (eXtreme Programming) tarkoituksena on auttaa tiimejä tuottamaan koodia, jota on helppo tarvittaessa muuttaa muuttuvien vaatimusten mukaan (Stellman & Greene 2014). Se on joukko tekniikoita, joiden avulla tiimi voi tuottaa korkeampilaatuista koodia tehokkaammin. Koska yksi Agilen perusperiaatteista on muutokseen vastaaminen, on tärkeää, että tuotettu koodi ei ”lukitse” itseään, vaan sitä on teknisesti mahdollisimman yksinkertainen muuttaa vastaamaan uusia vaatimuksia, tai kesken projektin tulleet hyvät ideat on mahdollista lisätä koodiin ilman, että muutoksella riskeerataan koko projekti yllättävien ongelmien ilmetessä sen seurauksena (Beck 2004). Menetelmän tarkoituksena on siis mahdollistaa muutosten tekeminen mahdollisimman nopeasti ja ongelmavapaasti, joka osaltaan parantaa laatua (Stellman & Greene 2014). XP:n perusajatuksena on keskittyä sen kuvaamiin tekniikoihin mahdollisimman tiukasti, ”äärimmäisyyksiin asti”. Tästä myös juontuu sen nimi, extreme programming (Beck 2004).

XP koostuu 13:sta tekniikasta, jotka ovat TDD, pariohjelmointi, 10-minute build, Continuous Integration, Weekly Cycle, Quarterly Cycle, Stories, Slack, Sit Together, Informative Workspace, Whole Team, Energized Work ja Inceremental Design. Näistä tekniikoista osa on suunnattu suoraan kehittäjille, kun taas osa kohdistuu tiimidynamiikkaan ja yhteisiin käytänteisiin.

TDD eli Test Driven Development (testi ensin -kehitys) on kehitystyyli, jonka tarkoituksena on aina aluksi kirjoittaa testit, jotka valmiin sovelluskoodin tulisi läpäistä. Kun testit ensimmäisen kerran ajetaan, testit luonnollisesti epäonnistuvat, koska sovelluskoodia ei ole vielä tehty. Tämän jälkeen on tarkoituksena kehittää koodia, ja ajaa testi uudestaan. Jos testit eivät vieläkään mene läpi, jatketaan kehitystä niin kauan, että testit läpäistään ja voidaan todeta, että koodi on kelvollista. Tämän tyylinen kehitysjärjestys lyhentää palautesykliä, jota kehittäjä saa koodistaan, jonka osaltaan on tarkoituksena vähentää bugien määrää ja kokonaisuudessaan parantaa koodin laatua. (Stellman & Greene 2014, Beck 2004)

Pariohjelmoinnin ideana on se, että kaikki sovelluskoodi toteutetaan siten, että kaksi kehittäjää istuu saman koneen ääressä työskennellen saman koodin parissa. Ideana se, että tällä tavalla toimien koodin katselmointi on jatkuvaa, ja saadaan nopeaa palautetta ongelmiin verrattuna siihen, että kehittäjät tekisivät kukin itsenäisesti työtään ja tietyin väliajoin pyytäisivät katselmointia tai apua ongelmiinsa. Sovelluskoodin toteuttaminen ei kuitenkaan kestä tuplasti kauempaa kuten ehkä voisi olettaa, vaan ongelmien ilmetessä kaksi henkilöä pystyy selvittämään ne nopeammin kuin jos työtä tehtäisiin yksin. Kun ongelmien ratkomiseen ei tarvitse käyttää niin paljoa arvokasta aikaa, pystytään keskittymään tehtävään tehokkaammin, joka oikein tehtynä johtaa siihen, että samaan toiminnallisuuteen vaaditaan vähemmän koodia. Lisäksi on havaittu, että pariohjelmointia käyttävät tiimit tekevät laadukkaampaa koodia kuin tiimit, joissa pariohjelmointia ei käytetä. (Stellman & Greene 2014, Beck 2004)

10-minute build tarkoittaa sitä, että koko järjestelmä voidaan rakentaa kymmenessä minuutissa. Tänä aikana ajetaan kaikki testit, joita järjestelmälle on kirjoitettu. 10 minuutin aikaraja on sen vuoksi, että jos projektin rakentaminen kestää reilusti yli 10 minuuttia, kehittäjät luistavat sen ajamisesta, jonka vuoksi myöskään ohjelmistoa ei tule testattua niin usein kuin olisi hyödyllistä. Tämä tekniikka varmistaa sen, että kehittäjät saavat mahdollisimman usein ja nopeasti palautetta kirjoittamastaan koodista. Lisäksi tekniikka kannustaa tiimiä automatisoimaan rakennusprosessin, jolloin se todennäköisesti tehdään useammin, joka osaltaan auttaa huomaamaan virheet nopeammin. (Stellman & Greene 2014, Beck 2004)

Continuous Integrationin tarkoituksena on se, että koodi testataan automaattisesti heti, kun se lisätään koodipohjaan. Tällä tavalla esimerkiksi ongelmat integroinnissa olemassa olevan sovelluskoodin kanssa huomataan välittömästi. Jos ongelmia ilmenee aina kun uusi koodi ajetaan koodipohjaan, on se selvä merkki siitä, että se pitäisi tehdä useammin. Näin varmistetaan se, että esiin ei tule kerralla niin paljoa ongelmia, ja esiin tulevatkin on helpompi korjata. (Stellman & Greene 2014, Beck 2004)

Weekly Cycle on Iteroiva tapa tehdä töitä XP:ssä. Ideana on, että tiimi tapaa aina viikkosyklin ensimmäisenä päivänä. Tässä tapaamisessa käydään läpi saavutettu edistys, asiakas valitsee Storyt joita haluaisi tehtävän sillä viikolla ja tiimi päättää millä tavoin edetään näiden kanssa. Tarkoituksena on se, että viikon lopussa valitut toiminnallisuudet on toteutettu ja testattu. Iteraatio on rajoitettu viikon mittaiseksi, jotta asiakkaalle pystytään esittelemään edistystä nopealla syklillä ja asiakas pääsee vaikuttamaan kehityksen kulkuun. Tämän lisäksi tiimi saa arvokasta palautetta ja tietoa asiakkaalta joka viikko, jonka ansiosta voidaan olla varmoja siitä, että tehdään oikeita asioita, jotka luovat arvoa asiakkaalle. (Stellman & Greene 2014, Beck 2004)

Quarterly Cycle on päällekäinen sykli Weekly Cyclen kanssa. Syklin alussa asiakas kertoo tiimille ylätasolla, mitä seuraavan kvartaalin aikana halutaan saavuttaa ja tehdä. tekniikka auttaa myös asiakasta omien sidosryhmiensä kanssa toimimisessa, sillä näin toimimalla asiakas pystyy kertomaan omille sidosryhmilleen suurin piirtein koska toivotut ominaisuudet tulevat olemaan valmiina. Kvartaalisuunnitelmaa voidaan käydä läpi ja tarkentaa tarpeen mukaan, esimerkiksi viikkopalavereissa, kun suunnitelma ja Storyt tarkentuvat. Näin ollen tiimin ei tarvitse sitoutua epätäydellisen tiedon varassa tehtyihin päätöksiin koko kvartaaliksi, vaan sisältöä voidaan muuttaa. (Stellman & Greene 2014, Beck 2004)

Stories, eli Suomeksi käyttäjätarinat, on sama asia kuin Scrumin User Stories. Lyhyt kuvaus asiasta tai toiminnallisuudesta, jolla tavalla käyttäjä haluaa käyttää ohjelmistoa. Käyttäjätarinoita käytetään suunnittelussa, ja ne toimivat hyvinä muistuttajina tiimille, kun toiminnallisuuksia määritellään teknisellä tasolla. Käyttäjätarinoiden avulla voidaan varmistaa se, että tiimi toteuttaa ohjelmistoa vastaamaan käyttäjien tarpeita, joka usein on tärkeä tavoite, kun ohjelmistoja valmistetaan. (Stellman & Greene 2014, Beck 2004)

Slack-tekniikan ideana on se, että jokaiseen viikko- ja kvartaalisykliin lisätään matalan prioriteetin tehtäviä, joita voidaan tarpeen mukaan jättää siinä syklissä tekemättä, jos tiimi jää jälkeen tärkeämmistä tehtävistä. Tällä varmistetaan se, että ennusteiden epätarkkuus ei aiheuta tiimille mahdotonta tilannetta, ja aikatauluihin voidaan päästä, vaikka

jossain kohtaa ennusteet pettäisivät. Toisaalta tällä varmistetaan myös se, että matalankin prioriteetin tehtävät tulevat tehtyä jossain kohtaa, sillä usein ne jäävät helposti jatkuvasti tärkeämpien asioiden jalkoihin. (Stellman & Greene 2014, Beck 2004)

Sit Together-periaate perustuu siihen, että suurimman osan ihmisistä mielestä kasvokkain kommunikointi on tehokkainta. Tämän vuoksi XP:n ”Sit Together” tekniikan mukaisesti tiimin tulisi istua fyysisesti samassa tilassa ilman tilaa rikkovia fyysisiä esteitä, kuten seiniä tai sermejä. (Stellman & Greene 2014, Beck 2004)

Informative Workspace tarkoittaa sitä, että tiimin työtilan tulisi olla mahdollisimman hyvin tiedon jakamista tukeva. Yksi hyvä keino on lisätä tilaan tietonäyttöjä, joista saa helposti ja nopeasti tietoa esimerkiksi projektin ja tiimin työtilanteesta, palveluiden tilasta, erilaista statistiikkaa tai muuta hyödyllistä tietoa. Toinen informatiivisen työtilan rakentamiskeino on se, että pyritään lisäämään osmoottista kommunikaatiota. Tämä tarkoittaa sitä, että työasioista keskustellaan lähtökohtaisesti tiimitilassa, sen sijaan että mentäisiin esimerkiksi neuvotteluhuoneeseen keskustelemaan asioista. Tällä saavutetaan se etu, että myös keskustelun ulkopuoliset ihmiset kuulevat asioita ”sivukorvalla” ja voivat puuttua keskusteluun, jos kokevat sen relevantiksi, josta on todettu olevan hyötyä tiedon leviämisessä tiimin sisällä. (Stellman & Greene 2014, Beck 2004)

Whole Team-periaatteen mukaan Tiimin tulisi koostua henkilöistä, jotka yhdessä pystyvät toteuttamaan tarvittavan projektin. Käytännössä se tarkoittaa sitä, että henkilöt, joilla on tarve tuotokselle sekä henkilöt, jotka työskentelevät sen rakentamisen parissa, tulisi työskennellä yhdessä päivittäin tiiminä. (Stellman & Greene 2014, Beck 2004)

Energized Work varmistaa, että ihmiset ovat fyysisesti ja henkisesti kykeneviä työskentelemään keskittyneinä ja virkeinä. Käytännössä tarkoittaa sitä, että ei ylityöllistetä työntekijöitä, pidetään huolta omasta hyvinvoinnista ja arvostetaan sekä pidetään huolta myös tiimikavereista. Oleellisena osana tekniikkaan kuuluu se, että työskennellään maksimissaan 40 tunnin työviikkoja työuupumuksen välttämiseksi. (Stellman & Greene 2014, Beck 2004)

Incremental Design-tekniikan tarkoituksena on varmistaa, että kehittäjät ymmärtävät ennen varsinaisen kehitystyön aloittamista koko työn laajuuden. Vasta kun on saatu riittävä perspektiivi tehtävään, voidaan syventyä tarkempiin teknisiin yksityiskohtiin. Tällä tavoin pystytään pienentämään mahdollisten muutosten ”hintaa”, kun jo suunnitteluvaiheessa ymmärretään kokonaiskuva rakennettavasta ohjelmistosta, ja voidaan ottaa huomioon mahdollisesti eteen tulevat muutokset. Tähän tekniikkaan kuuluu myös koodin refaktorointi, jonka tarkoituksena on siivota sovelluskoodia siten, että pidetään toteutus mahdollisimman yksinkertaisena ja luettavana. (Stellman & Greene 2014, Beck 2004)

2.4 Lean Software Development

Lean eroaa Scrumin, Kanbanin ja XP:n tapaisista viitekehyksistä siten, että se ei sinällään sisällä tekniikoita, vaan on ennemminkin kokoelma arvoja ja periaatteita. Lean Software Developmentia voisikin kuvata mielenlaaduksi, joka toimii erinomaisesti yhdessä muiden enemmän käytännönläheisten menetelmien ja prosessien kanssa. Leanin arvoja ja periaatteita kutsutaan ”ajattelun työkaluiksi” (thinking tools), ja Leanin mukaista mielenlaatua Lean-ajatteluksi. Lean-ajattelu auttaa ymmärtämään, mitä tiimi tekee päivittäin ja viikoittain ohjelmiston tuottamiseksi, ja auttaa tiimiä keskittymään arvon tuottoon (Stellman & Greene 2014)

Leanin seitsemän arvoa ovat Eliminate waste, Amplify learning, Tee päätökset mahdollisimman myöhään, Deliver as fast as possible, Empower the team, Build integrity in ja Näe kokonaisuus.

Eliminate waste perustuu siihen, että Lean-ajattelu pyrkii poistamaan hukkaa (eng. Waste) työnteon prosesseista. Hukkaa on kaikki sellaiset työvaiheet tai toiminnot prosessissa, jotka eivät auta luomaan asiakasarvoa. Perinteisesti Lean-ajattelussa hukkaa voidaan löytää kahdeksasta eri toiminnosta, jotka ovat: kuljetus, varastointi, turha liike, odottelu, ylituotanto, yliprosessointi, virheet ja taitojen hyväksi käyttämättä jättäminen (Giarrizzo 2018, Bicheno & Holweg 2000). Vaikka hukan määritelmä onkin kehitetty teollisuuden prosessien tehostamiseksi, pätevät tismalleen samat säännöt myös ohjelmistotuotannossa. Leanin mukaan tiimien ja työntekijöiden tulisikin pyrkiä löytämään omista prosesseistaan vaiheet, joista aiheutuu hukkaa, ja poistamaan ne omista prosesseistaan. (Stellman & Greene 2014)

Amplify learning-arvon perusajatuksena on se, että tiimin tulisi kerätä mahdollisimman paljon palautetta prosesseistaan, työtavoistaan ja toiminnoistaan, jotta on mahdollista parantaa kyseisiä tapoja ja tehdä parempia ohjelmistoja. Käytännön työkaluja arvon toteuttamiseen olisi esimerkiksi palaute ja iteraatiot, joihin myös muissa ketterissä menetelmissä otetaan kantaa. Esimerkiksi Scrumissa palautetta kerätään Daily Scrumin avulla päivittäin, ja Retrospektiivin avulla sprintin päätteeksi. XP:ssä tätä arvoa toteutetaan esimerkiksi pariohjelmoinnin ja jatkuvan integroinnin tekniikoiden avulla. (Stellman & Greene 2014)

Tee päätökset mahdollisimman myöhään on periaate, jonka mukaan kaikki tärkeät päätökset projektissa tulisi tehdä vasta silloin, kun päätöksestä ja sen seurauksista on saatavilla mahdollisimman paljon tietoa. Tämä tarkoittaa sitä, että päätökset tulee tehdä viimeisellä mahdollisella hetkellä. Esimerkiksi Scrumissa käytännön työkaluna tämän varmistamiseksi olisi Sprint planning – kun päätökset toteutettavista ominaisuuksista

tehdään viimeisellä mahdollisella hetkellä, eli juuri ennen sprintin käynnistymistä, voidaan varmistua siitä, että se mitä sprintin aikana tehdään, on arvokasta, eikä turhaa työtä. (Stellman & Greene 2014)

Deliver as fast as possible ei tarkoita sitä, että tiimin tulisi kaikin keinoin kiirehtiä sovelluskoodia valmiiksi esimerkiksi laadun ja työntekijöiden hyvinvoinnin kustannuksella. Leanin ajattelun mukaan kehitystyön tulisi olla kestävällä pohjalla, eli kehitystä tulisi tehdä sellaisella tahdilla, jota voitaisiin teoriassa ylläpitää loputtomiin. Laadusta ja muista periaatteista tinkimällä voidaan hetkellisesti nopeuttaa kehitystyötä, mutta missään nimessä se ei pidemmän päälle voi toimia ja kokonaisuuden kannalta hidastaa keskimääräistä kehitystahtia. ”Toimita mahdollisimman nopeasti” tarkoittaakin sitä, että tiimin tulee ymmärtää viivästyksen hinta, ja pyrkiä minimoimaan aika jonka tehtävät viettävät prosessissa. Tämä voidaan toteuttaa esimerkiksi jonojen ja pull-järjestelmien avulla, jossa ideana on, että tehtävät toteutetaan valmiiksi järjestyksessä, jossa ne on otettu työn alle, ja uutta tehtävää ei aloiteta ennen kuin edellinen on saatu valmiiksi. (Stellman & Greene 2014)

Empower the team on periaate, jonka tarkoituksena on antaa valta tiimille, ja rakentaa keskittynyt ja toimiva työtila. (Stellman & Greene 2014) On todettu, että kun ihmisille annetaan valta ja vapaus päättää omasta työnteostaan, on sillä positiivinen vaikutus tuottavuuteen, laatuun ja innovaatioiden määrään. Tämän lisäksi työntekijät ovat motivoituneempia ja tyytyväisempiä työhönsä, joka luonnollisesti johtaa parempaan asiakasarvoon luontiin (Tessem 2014). Työtyytyväisyydellä on myös todettu olevan henkilöstön vaihtuvuutta pienentävä vaikutus (Mobley 1982), joka jo itsessään tekee arvosta tavoittelemisen arvoista (Tessem 2014).

Build integrity in tarkoittaa sitä, että ohjelmistojen rakentamisessa ei tulisi tehdä kompromisseja esimerkiksi laadun kustannuksella, vaan tiimin tulisi ymmärtää käyttäjiä, heidän tarpeitaan ja pitää aina mielessä millä tavoin rakennettava ohjelmisto tuottaa arvoa. Järjestelmän keskeisten konseptien tulisi toimia keskenään siten, että ne toimivat saumattomasti muodostaen toimivan kokonaisuuden. Tällä tavoin ajatellessa tiimi tuottaa yhtenäistä sovelluskoodia, joka tukee kokonaistavoitetta ja kokonaisuus täyttää sille asetetut vaatimukset. (Stellman & Greene 2014)

Näe kokonaisuus – viimeinen Leanin arvo. Tarkoittaa sitä, että kehitystiimi ei toimi tyhjiössä, vaan erilaisilla organisaatiorakenteilla, säännöillä ja käytänteillä voi olla iso vaikutus kokonaisprosessin tehokkuuteen. Tämän takia olisi tärkeää ymmärtää työ, jota prosessissa tehdään, sekä kokonaiskuva tiimiä ympäröivästä järjestelmästä. Vasta tällöin

on mahdollista ymmärtää, toimiiko tiimi tehokkaasti ja tuloksellisesti, vai tehdäänkö tiimissä kokonaisuuden kannalta turhaa työtä. Kun työ, jota tiimissä tehdään, on ymmärretty kokonaisuuden kannalta, voidaan sitä mitata ja muiden Leanin arvojen mukaisesti korjata syitä, jotka johtavat epätehokkuuteen ja hukkaan. (Stellman & Greene 2014)

Kuten helposti voidaan havaita, Lean jakaa useita ominaisuuksiaan muiden ketterien menetelmien kanssa. Täten sitä voidaan helposti käyttää tukemaan muita metodologioita, ja useaan ketterään viitekehykseen Lean-ajattelu onkin sisällytetty jo sen kehitysvaiheessa. (Stellman & Greene 2014)

2.5 Integraatiotuotannon erityishaasteet

Ymmärtääksemme integraatiotuotannon erityishaasteita suhteessa edellä esitettyihin ketteriin menetelmiin, tulee ymmärtää, mitä yritystason järjestelmäintegraatioilla tarkoitetaan ja mitä integraatioiden tekijöiltä odotetaan.

Terminä integraatio on epämääräinen ja riippuu termin käyttäjästä mitä sillä tarkalleen tarkoitetaan. Se johtuu muun muassa siitä, että integraatioita tehdään yleisesti monessa eri kontekstissa ja alalla tietojärjestelmien integroinnin lisäksi (Kähkönen 2017). Tässä yhteydessä kuitenkin integraatiosta puhuttaessa tarkoitetaan nimenomaan tietojärjestelmien välistä integraatiota. Linthicum (2004) määrittelee järjestelmäintegraation ”strategiseksi lähestymistavaksi, jolla sidotaan monta tietojärjestelmää yhteen sekä palvelu- että informaatiotasolla, joka tukee niiden kykyä vaihtaa tietoa toistensa välillä oikeassa maailmassa” (Linthicum 2004). Lisäksi hän erittelee organisaation sisäisten, Enterprise Application Integration (EAI) ja ulkoisten järjestelmien, B2B Application integration, integraation omiksi kokonaisuuksikseen (Kähkönen 2017). Lamin (2005) mukaan monilla organisaatioilla onkin monimuotoinen portfolio erilaisista tietojärjestelmistä, joita pitää kehittää ja ylläpitää yrityksen laajuisten liiketoimintaprosessien ylläpitämiseksi. Hän kuvailee järjestelmäintegraatioita (EAI) tavaksi yhdistää organisaation järjestelmät yhteen, esimerkiksi middleware-ohjelmiston avulla (Lam 2005).

Vaikka ERP-ohjelmistot kehittyvät jatkuvasti kyvykkäämmiksi ja laajemmiksi, eivät organisaatiot nykypäivänä halua ostaa yhtä tuotetta kaikkien organisaation toimintojen pyörittämiseen (vertikaaliohjelmisto), vaan mieluummin ostetaan jokaiseen osatoimintoon paremmin soveltuvia horisontaalijärjestelmiä, joiden välisiä toimintoja tulee EAI:n keinoin integroida (Manninen 2015). Osatoimintoja yrityksessä voivat olla esimerkiksi myynti, talous, ja HR, joista kukin tarvitsee erityyppisiä toimintoja käyttämältään tietojärjestelmältä. Horisontaaliohjelmistot on räätälöity mahdollisimman hyvin kyseiselle toiminnalliselle alueelle, mutta kääntöpuolena tulee varsinkin suurissa yrityksissä se, että toimintojen

määrän kasvaessa myös järjestelmien määrä kasvaa vähintään samassa suhteessa (Manninen 2015). Koska järjestelmissä ei välttämättä ole natiiveja tapoja yhdistää toisiinsa, niiden tehokkaan käytön ja yrityksen kokonaisarkkitehtuurin vuoksi joudutaan rakentamaan räätälöityjä integraatioratkaisuja, joiden avulla muun muassa tiedon siiloutumiselta ja turhalta manuaaliselta työltä vältytään (Lam 2005).

Järjestelmäintegraatiot voidaan luokitella eri haastavuuskategorioihin, jolloin yksinkertaisimmillaan integraatiotyö voi olla hyvinkin helppoa ja onnistua natiivisti järjestelmien kesken. Integraation haastavuus riippuu muun muassa siitä, minkä tyyppisiä järjestelmiä ollaan integroimassa ja minkälaisia integraatioskenaarioita siihen liittyy, millaisia integraatiotekniikoita käytetään, millaisia rajoituksia järjestelmällä on ja kuinka useita osapuolia integraatiolla on (Prencipe et al., 2003).

Kähkösen (2017) mukaan integraatiotyö voidaan jakaa kolmeen eri tasoon, jotka kaikki rakentuvat toistensa päälle. Nämä ovat tekninen näkökulma, liiketoimintanäkökulma ja sosio-organisationaalinen näkökulma (Kähkönen 2017). Jokainen näkökulma tarjoaa oman näkemyksensä integraatioiden luonteeseen ja tuo omat haasteensa.

Teknisestä näkökulmasta integraatio on kahden tai useamman järjestelmän välistä tiedonsiirtoa (Kähkönen 2017). Tiedonsiirron toteuttamiseksi järjestelmien tulee ymmärtää dataformaatti ja sanomien rakenne. Yksinkertaisimmillaan toteutus on datan siirtämistä paikasta A paikkaan B API:en (Application Programming Interface) kautta (Kähkönen 2017). Monimutkaisemmissa integraatioskenaarioissa vaaditaan esimerkiksi transaktionhallintaa, asynkronista kommunikaatiota, rikastusta, reititystä ja muita kehittyneempiä integraatiotekniikoita (Kähkönen 2017). Esimerkkinä teknisestä haasteesta Kähkönen (2017) nostaa ERP-integraatioprojektissa monimutkaiset tietorakenteet, joiden mappaaminen ja muuntaminen toisen järjestelmän käyttämään muotoon voi vaatia suuria ponnisteluita, joka johtaa pahimmillaan aikataulujen myöhästymiseen ja koko projektin viivästymiseen (Kähkönen 2017).

Liiketoimintanäkökulmasta integraatiohaasteita voi tulla esimerkiksi heterogeenisten operointiympäristöjen vuoksi. Kähkösen (2017) mukaan integrointi on usein pakollista, kun halutaan automatisoida liiketoimintaprosesseja, ja yhden prosessin automatisointi voi vaatia useita järjestelmäintegraatioskenaarioita (Kähkönen 2017). Eräänä haasteena hän nostaa tilanteen, jossa prosessiin liittyy useita erilaisia ja eri tavalla toimivia järjestelmiä, jotka kuitenkin ovat kaikki osa liiketoimintaprosessia. Esimerkiksi verkkokaupan yleistyminen on ajanut yritykset integroitumaan liiketoimintaprosessien tasolla, joka käytännössä tarkoittaa sitä, että tietyn liiketoimintaprosessin suorittamiseen tarvitaan useita

järjestelmiä, jotka eivät natiivisti keskustele keskenään (Themistocleus 2004). Automatisointi vaatii usein muutoksia olemassaoleviin järjestelmiin, ja vasta kun koko end-to-end prosessi on implementoitu, voidaan ratkaisua testaamaan. Mitä enemmän muutoksia järjestelmiin tarvitaan, sitä kalliimmaksi tyypillisesti integraatioiden rakentaminen tulee, koska esimerkiksi virheenkäsittelyyn tulee kiinnittää enemmän huomiota (Linthicum 2004, Kähkönen 2017).

Sosio-organisationaalisesta näkökulmasta Kähkönen (2017) nostaa suurimmaksi haasteeksi yhteistyön eri sidosryhmien välillä ja eri ryhmien intressit. Barki ja Pinsonneault (2005) argumentoivat, että suurimmat organisatoriset integraation onnistumisen esteet ovat erikoistuminen ja politikoituminen. Esimerkiksi jotkut sidosryhmät tavoittelevat mieluummin paikallisia tavoitteita ja vasta sitten koko organisaation laajuisia, jolloin integraatioprojektiin osallistuvilla ryhmillä on omia erillisiä päämääriään (Barki & Pinsonneault 2005). Eräänä sosio-organisationaalisena haasteena Kähkönen (2017) nostaa organisaation kokemattomuuden integraatioprojekteista ja organisaation matalan maturiteetin, joka voi aiheuttaa suuria ongelmia integraation määrittelyvaiheessa, joka heijastuu projektin läpivientiin (Kähkönen 2017). Kähkönen nostaa lisäksi esimerkkinä tilanteen, joka ilmeni, kun ERP-järjestelmää yritettiin asentaa yrityksen toisessa maassa sijaitsevaan liikepaikkaan. Erilaisten liiketoimintaprosessien ja käytäntöjen vuoksi yritys pakotettiin harkitsemaan kokonaan erillisen ERP-instanssin asentamista kyseiseen maahan, joka olisi tarkoittanut sitä, että se tulisi integroida täysin yrityksen nykyiseen ERP-järjestelmään (Kähkönen 2017). Vaikka lopulta koko projekti peruttiin, kuvaa tapaus hyvin integraatioiden monitasoista luonnetta, eivätkä kaikki ongelmat suinkaan ole teknisiä.

Integraatioita tuotetaan yleensä tarkoitusta varten kehitetyillä työkaluilla. Tänä päivänä on huima määrä eri teknologioita, joista valita. Vaihtoehtona on esimerkiksi iPaas, erilaiset hybridi-integraatioalustat sekä On-Premise ratkaisut, joista voidaan räätälöidä kullekin asiakkaalle toimivimmat kokonaisuudet. IPaaS-mallissa voidaan hyödyntää esimerkiksi julkista pilveä, yksityistä pilveä tai näiden sekoitusta. (Liu et al. 2015)

2.6 Integraatiotoiminnan kehittäminen

Kuten yltä havaitaan, pieni osa integraation tekemisestä on puhdasta ohjelmointityötä. Tämän vuoksi ketterän ohjelmistokehityksen menetelmien hyödyntäminen ei auta läheskään kaikissa integraatiotuotantoon liittyvissä ongelmissa. Suurin osa integraation toteutukseen käytetystä ajasta koostuu suunnittelusta, määrittelystä, kommunikoinnista, testaamisesta ja koordinoinnista eri osapuolten välillä, kuten Kähkönen (2017) tuo esiin. Jos asiaa mietitään toimittajan kannalta, tärkeää olisi ymmärtää kokonaisvaltaisesti asiakkaan liiketoimintaprosesseja ja järjestelmiä, joita lähdetään integroimaan, ja muuttuviin

vaatimuksiin on pystyttävä reagoimaan nopeasti ja ketterästi kun end-to-end testausta aletaan tekemään.

Kähkönen (2017) ehdottaa integraatioiden haasteiden ylittämiseksi viittä keinoa:

- Integraatioita tulisi käsitellä systemaattisena ja hyvin suunniteltuina toimintoina, joka käsittää useita järjestelmiä ja sidosryhmiä. Erilliset ohjelmat tai projektit tulee aina perustaa suuria integraatiohankkeita varten.
- Omistautunutta asiantuntijuutta tarvitaan. Osan sidosryhmistä tulisi olla kokopäiväisesti vastuussa integraatioista, ja koordinaatio sekä kommunikaatio sidosryhmien välillä on kriittistä.
- Integraatioprojekteja tulee hallita eri tasoilla. Ylimmän johdon tuen lisäksi projekti- ja laatupäälliköitä ja muutostenhallintaa tarvitaan.
- Integraatioiden monimutkaisen luonteen takia on tärkeää ylläpitää arkkitehtuurikuvauksia integroitavista järjestelmistä, jotta voidaan tunnistaa integraatiotarpeita ja määrittelyitä.
- Yhtiötason integraatiostrategioita tarvitaan, jotta voidaan varmistua integraatioiden olevan linjassa organisaation tavoitteiden kanssa. (Kähkönen 2017)

Kähkösen (2017) tuomien kehitysehdotusten implementointi käytäntöön vaatii selvästi sitoutumista asiakasyritykseltä, ja toimittaja yksin tuskin pystyy kaikkia kohtia toteuttamaan. Kuitenkin esimerkiksi integraatiotyön organisointiin, omistautumiseen ja arkkitehtuurikuvausten ylläpito on sellaista, jota toimittajan asemassa on mahdollista tehdä, ainakin yhteistyössä asiakkaan organisaation kanssa.

3. TUTKIMUKSEN TAUSTA JA KÄYTETYT MENETELMÄT

Tässä luvussa kerrotaan yleisesti tutkimuksessa käytetyistä tutkimusmenetelmistä, tutkimuksen taustoista, millä tavalla aineistoa on kerätty ja analysoitu sekä miten tutkimus on toteutettu ja rajattu, sekä kerrotaan hieman tutkimuksen case-organisaatiosta.

3.1 Tapaustutkimus, validiteetti ja reliabiliteetti

Tapaustutkimus on strategia, jossa tarkoituksena on tutkia rajattua tutkimusyksikköä (casea), jotta voidaan paremmin ymmärtää sen monimutkaisuutta tai paikkaa laajemmassa kontekstissa (Bailey 2010). Tutkittava case voi olla esimerkiksi henkilö, ajanjakso, tapahtuma, ryhmä, prosessi tai vastaava konseptuaalinen kategoria, jota on mielekästä tutkia ja jonka avulla voidaan saada tietoa tutkittavasta ilmiöstä. Tämän vuoksi tapaustutkimuksessa on tärkeä määritellä casen rajaukset tarkasti (Bailey 2010). Tapaustutkimus hyödyntää kvantitatiivista ja/tai kvalitatiivista dataa ja useita datalähteitä selittämään, tutkimaan tai kuvailemaan ilmiöitä. Usein tapaustutkimuksen tutkimuskohteena on yksittäinen case, mutta myös monitapauksinen tutkimus on mahdollinen (Bailey 2010).

Tapaustutkimuksen validiteetin ja reliabiliteetin määrittely voi olla haastavaa, eikä kaikille tapaustutkimuksille ole yhtä tapaa mitata näitä ominaisuuksia (Riege 2003). Yleisesti tutkimuksen validiteetti ilmaisee sitä, miten hyvin tutkimuksessa käytetty mittaus- tai tutkimusmenetelmä mittaa juuri sitä ominaisuutta mitä on tarkoituskin mitata, eli mittaako tutkimus sitä mitä sen avulla on tarkoitus selvittää (Hiltunen 2009). Reliabiliteetti taas kuvaa sitä, että miten luotettavasti ja toistettavasti käytetty mittaus- tai tutkimusmenetelmä mittaa haluttua ilmiötä. Reliaabelin tutkimuksen tulokset pystytään siis alkuperäisen tutkimuksen edellytyksin riippumattomasti toistamaan eli saadaan samat tulokset (Hiltunen 2009). Tapaustutkimuksen validiteetin ja reliabiliteetin määrittelyyn on olemassa testejä, jotka on kuvattu Taulukko 1.

Taulukko 1: Tapaustutkimuksen validiteetti ja reliabiliteetti (Yin 2003)

Testi	Kuvaus	Tapaustutkimuksen taktiikka	Tutkimuksen vaihe
Rakenna validiteetti	Määritellään oikeat mittarit tutkittavaan aiheeseen liittyen	Käytä useita aineistolähteitä	Datan keruu Datan keruu

		Rakenna todistus-aineistoketju	
Sisäinen validiteetti	Määritellään kausaaliset suhteet ja erotellaan ne vääristä	Tee kognitiivinen kartta Tee seliterakennus	Datan analysointi Datan analysointi
Ulkoinen validiteetti	Määritellään alue, jolle tutkimustuloksia voidaan yleistää	Käytä replikaatiologiikkaa monitapaustutkimuksissa	Tutkimussuunnitelma
Reliabiliteetti	Näytetään että tutkimus voidaan uudelleenteoteuttaa samoilla tuloksilla	Käytä tapaustutkimuksen protokollaa Luo tapaustutkimuksen tietokanta	Datan keruu Datan keruu

3.2 Tutkimuksen taustat

Tämä tutkimus on eksploratiivinen tapaustutkimus, joka vastaa siihen, miten integraatiotiimien toimintaa voidaan kehittää ketterien menetelmien avulla sekä miten hyvin yleisimmät ketterät menetelmän sopivat integraatiokehitykseen. Tutkimuksen sisäinen validiteetti varmistetaan sillä, että aineistoa on kerätty useasta lähteestä ja haastatteluihin on valittu riittävän eri taustaisia henkilöitä haastattelurungon ollessa sama kaikille haastatelluille. Ulkoinen validiteetti varmistetaan siten, että tutkimuksen rajaukset ja toteutus kuvataan tarkasti tutkimussuunnitelman yhteydessä. Reliabiliteetti varmistetaan sillä, että tutkimusstrategia on valittu tutkimuksen tavoitteiden mukaiseksi ja tutkimusstrategiaa noudatetaan koko tutkimuksen läpi. Strategia valikoitui tapaustutkimukseksi sen perusteella, että sen avulla saadaan tietoa yksittäisestä tapauksesta syvällisellä tasolla ja voidaan tutkia toiminnan dynamiikkaa, prosesseja ja mekanismeja niin, että tuloksista saadaan yleistettäviä.

3.3 Aineiston kerääminen ja analysointi

Tässä tutkimuksessa case-aineistoa on kerätty kahdella eri menetelmällä, jotka ovat nykytilan havainnointi ja haastattelut. Näiden empiiristen aineistojen tueksi kirjallisuudesta on hankittu aineistoa, joka tukee empiirisen aineiston keräysprosessissa sekä auttaa sen validiteetin ja reliabiliteetin analysoinnissa. Tutkimusaineisto on pääosin laadullista, vaikka myös määrällistä aineistoa kirjallisuudesta on hyödynnetty.

Havainnointiaineisto kerättiin noin vuoden aikana, jonka perusteella muodostui kuva organisaation nykytilasta. Havainnointi suoritettiin yrityksen sisäisiä dokumentaatioita lukemalla sekä ns. kahvipöytäkeskusteluissa, jolla saatiin kerättyä tietoa nykytilanteesta, ongelmakohdista ja kehitystarpeista. Havainnointi auttoi myös valitsemaan sopivat haastateltavat tutkimuksen haastatteluosuuteen.

Kirjallisuusaineisto kerättiin havainnoinnin yhteydessä, mutta ennen haastattelujen suorittamista. Kirjallisuudesta kerätyn teoriapohjan haluttiin olevan riittävän vahva, jotta haastattelurunkoon saadaan riittävän monipuolisesti ja eri näkökulmista teoriaa, jota vasten arvioida haastateltavien näkemyksiä case-organisaation nykytilasta ja mahdollisuuksista. Kirjallisuusaineiston keräämiseen käytettiin pääasiassa Andor-hakupalvelua, jonka kautta saadaan pääsy suureen määrään tieteellistä kirjallisuutta. Osa kirjallisuuden materiaalista on etsitty myös muiden hakukoneiden kautta, kuten Googlen kautta.

Haastatteluaineisto kerättiin havainnoinnin jälkeen, kun teoriapohjan koettiin olevan riittävän vahva haastattelurungon toteuttamiseen. Haastateltavia henkilöitä oli viisi kappaletta, joilla kaikilla on riittävän erilainen tausta ja kokemukset, jotta voidaan olettaa haastattelujen tulosten kuvaavan eri näkökulmia ja todellisuutta riittävän kattavasti. Tällä myös varmistetaan tulosten validiteettia ja reliabiliteettia, kun saadaan monipuolinen kuva case-organisaation prosesseista ja käytännöistä eri projektien ja asiakkuuksien välillä. Haastattelun tukena käytettiin haastattelurunkoa, joka oli sama kaikille haastatelluilla. Haastattelukysymykset löytyvät liitteestä A. Haastattelut kestivät noin 1,5 tuntia per haastateltava. Haastattelut suoritettiin syys-lokakuussa 2019. Haastattelujen tulokset kerättiin excel-taulukoihin, joista sen vertailu ja analysointi oli nopeaa.

Kahden haastateltavan roolina oli ”Integration Specialist” (Integraatioasiantuntija A & B). Integraatioasiantuntija A:lla on työkokemusta IT-alalta neljä vuotta kahdesta eri yrityksestä, ja integraatiokehityksestä noin 2 vuoden ajalta. Integraatioasiantuntija B:llä on IT-alan työkokemusta 10 vuotta kolmesta eri yrityksesä, ja integraatiotyöstä 5 vuoden ajalta.

Yksi haastateltava oli titteliltään ”Senior Integration Specialist” (Senior-integraatioasiantuntija A). Senior-integraatioasiantuntija A:n työkokemus IT-alalla on noin 21 vuotta neljästä eri yrityksestä, ja integraatiotuotannosta yli 10 vuotta.

Haastateltavista yhden rooli oli ”Project Manager” (Projektipäällikkö A). Projektipäällikkö A on työskennellyt IT-alalla kahdessa eri yrityksessä noin 15 vuotta, joista integraatioiden parissa 9 vuotta.

Viidennen haastateltavan roolina oli ”Team Manager” (Tiimipäällikkö A). Tiimipäällikkö A:n työkokemus IT-alalta on noin 12 vuotta kolmesta eri yrityksestä, joista noin 10 vuotta integraatioiden parissa.

Integraatioasiantuntija A ja Senior-integraatioasiantuntija A työskentelivät haastattelun hetkellä samassa asiakasprojektissa, mutta heillä oli myös toisistaan eriäviä asiakkuuksia. Myös integraatioasiantuntija B:llä ja projektipäällikkö A:lla oli haastattelun aikaan yhteinen projekti, jossa molemmat työskentelivät sillä hetkellä täyspäiväisesti.

Haastattelujen perusteella saatu aineisto analysoitiin suhteessa kirjallisuudesta saatuun teoriaan, josta yhteenveto löytyy kappaleesta 5. Lopuksi kirjallisuuden, havaintojen ja haastatteluaineiston perusteella tehtiin analyysia ja pohdintaa, joka vastaa tutkimuskysymyksiin ja tutkimusongelmaan. Tämä pohdinta löytyy kappaleesta 6.

3.4 Case-organisaatio

Työssä käsiteltävä case-organisaatio on Solita OY:n integraatioyksikkö. Solita on suomalainen teknologia-, strategia- ja designyritys, joka perustettiin vuonna 1996. Yrityksellä on yli 900 työntekijää, jotka työskentelevät muun muassa strategisen konsultoinnin, palvelumuotoilun, digitaalisten palvelujen, AI- ja analytiikkaratkaisujen sekä pilvipalvelujen parissa (Solita 2019).

Solitan integraatioyksikön tarkoitus on muun muassa tuottaa integraatiopalveluita, toteuttaa integraatioprojekteja ja konsultoida Solitan asiakkaita integraatio- ja API-aihealueilla. Kehitystiimien koko vaihtelee tyypillisesti noin kahden ja kahdeksan henkilön välillä, ja useimmat työntekijät tekevät töitä joko kahdelle tai useammalle asiakkaalle samanaikaisesti. Yhteensä yksikössä työskentelee noin 50 integraatioasiantuntijaa, -arkkitehtia ja projektipäällikköä. Integraatioyksikön asiakkaiden lukumäärä, eli sellaiset asiakkaat, joille tällä hetkellä tuotetaan joko jatkuvia integraatiopalveluita tai -projekteja, lukumäärä on noin kaksikymmentä. Yksikössä tehty työ on lähes yksinomaan asiakastyötä, ja esimerkiksi sisäistä tuotekehitystä ei tutkimuksen kohteena olevissa tiimeissä tehdä. Asiakastyö koostuu muun muassa integraatioratkaisujen suunnittelusta ja toteutuksesta, ylläpidosta ja häiriönhallinnasta sekä konsultoinnista. Töitä tehdään sekä projektiluonteisesti että pienkehityksenä jatkuvien palvelujen piirissä.

3.5 Tutkimuksen rajaukset

Tutkimus on rajattu koskemaan toimittajayrityksen integraatiotiimejä. Tutkimuksessa otetaan kantaa ainoastaan asioihin, joita voidaan ratkaista ketterien menetelmien avulla. Tutkimuksen tuloksia voidaan yleistää koskemaan integraatiotiimejä, jotka toteuttavat järjestelmäintegraatioita yhteistyössä muiden osapuolien kanssa toimittajan roolissa.

4. HAVAINNOT ORGANISAATIOSTA

Tässä luvussa selvitetään organisaation nykytilaa, eli jo käytössä olevia menetelmiä, ongelmakohtia ja yleisiä käytänteitä, joita tiimeillä on. Näin saadaan yleiskuva toimintaympäristössä, jossa haastateltavat toimivat, sekä johon työn tuloksia olisi tarkoitus tulevaisuudessa soveltaa. Tämän luvun aineisto on kerätty havainnoimalla organisaatiota sisältä päin.

4.1 Nyky- ja Tavoitetila

Havainnoimalla case-organisaatiota voidaan havaita, että lähes kaikki integraatiotiimit toimivat sekä ylläpito- että kehitystehtävissä samanaikaisesti. Case-organisaatiossa integraatiopalveluita tuotetaan tyypillisesti jatkuvina palveluina, jossa samat kehittäjäresurssit hoitavat sekä uuden kehitystä että vanhan ylläpitoa.

Keskusteluissa yksikön johdon kanssa on tullut ilmi, että organisaation tavoitteena on parantaa jatkuvasti kehittäjän työtä ja arkea mielekkäämpään suuntaan. Tilannetta, jossa integraatiokehittäjän tulisi jatkuvasti miettiä priorisointia ja tulevaa työtä, tulisi välttää, sillä se on aina pois tuottavasta työnteosta ja aiheuttaa kehittäjille turhaa kuormaa. Johdon mukaan tulisi välttää tilannetta, jossa kehittäjää revitään asiakkuudesta toiseen, sillä on havaittu, että se vaikeuttaa keskittymistä ja aiheuttaa helposti ongelmia, jos joudutaan vaikkapa priorisoimaan kahden asiakkaan töitä keskenään saman kehittäjän välillä. Tämän vuoksi olisi hyödyllistä, että joku muu kuin itse kehittäjä miettisi ja pitäisi huolen siitä, että kehittäjällä on työrauha ja he voivat keskittyä tärkeimpään tehtäväänsä eli integraatioiden kehittämiseen.

Johdon mukaan juurisyynä ongelmille voidaan pitää sitä, että toimittaja-asiakassuhteessa luonnollisesti toimittaja haluaa maksimoida käyttöasteen ja tuoton, kun taas asiakas haluaisi minimoida kustannukset. Jos asiakas ei ole halukas ostamaan kokonaisia tiimejä käyttöönsä, joutuu toimittaja myymään resursseja ja kehitystyötä tuntityönä. Tällainen resurssien optimointi asiakkaiden ja projektien välillä siten, että sekä asiakkaat, työnantaja että kehittäjä ovat tyytyväisiä, on suuri haaste, jonka ratkaisuun pyritään

4.2 Organisaation ja tiimien kuvaus

Organisaatiokaaviosta voidaan havaita, että integraatiotiimit ovat tyypillisesti noin kahden ja kuuden kehittäjän kokonaisuuksia. Useissa tapauksissa kukaan yksittäinen kehittäjä ei ole sidottu tiettyyn asiakkuuteen 100%:in allokaatiolla, vaan useimmilla on kaksi

tai useampi asiakkuutta, joille he tekevät töitä vaihtelevasti. Jokaisella asiakkuudella on pääsääntöisesti projektipäällikkö, joka vastaa asiakkuuden ja töiden ohjaamisesta. Myös projektipäälliköillä on usein useampi kuin yksi asiakkuus, joita he hoitavat eri allokaatioilla.

Samasta organisaatiokaaviosta nähdään, että yksiköllä on työntekijöitä tutkimuksen kirjoituksen hetkellä neljässä eri kaupungissa. Monet asiakkuustiimeistä onkin koostettu siten, että kaikki saman asiakkuuden työntekijät eivät istu fyysisesti samalla toimistolla, jolloin puhutaan virtuaalitiimeistä. Yleisten keskustelujen perusteella on tullut ilmi, että tämä saattaa joissain tapauksissa tuoda lisähaasteita, varsinkin kommunikoinnin osalta. Työntekijät saavat pitää vapaasti etäpäiviä, ja tätä oikeutta myös ahkerasti harjoitetaan. Täten työtapojen tulee joka tapauksessa luonnollisesti tukea hajautetun tiimin vaatimuksia.

4.3 Kommunikointikäytänteet

Omien havaintojen perusteella suurin osa tiimeistä pyrkii kommunikoimaan mahdollisimman paljon kasvotusten, ja tätä tuetaan yrityksen tasolta esimerkiksi siten, että eri kaupungeissa sijaitsevia tiimiläisiä pyritään saattamaan aika-ajoin yhteen. Tiimit istuvat avokonttorissa, jossa kommunikointi on helppoa, lisäksi neuvotteluhuoneita on työpisteiden yhteydessä, johon voi mennä keskustelemaan asioista joita ei halua muiden korviin. Kasvotusten kommunikointi tuntuu olevan havaittu sekä nopeimmaksi että toimivimmaksi tavaksi kommunikoida asioista. Kasvotusten kommunikointi työpisteen äärellä auttaa vähentämään väärinkäsitysten määrää ja esimerkiksi työkaverin auttaminen saman koneen äärellä on ylivoimaisesti helpompaa, kun auttaja istuu saman koneen ääressä.

Organisaatiossa yleisin digitaalinen viestintäkanava on Slack-pikaviestinohjelma. Slackilla pystytään kommunikoimaan sekä kirjallisesti, että (video)puheluiden avulla, ja sen kautta pystytään muun muassa jakamaan tiedostoja. Yleisesti jokaisella tiimillä on Slackissa oma kanava, jossa kommunikoidaan tiimiä ja projektia koskevat. Tämän lisäksi on myös esimerkiksi yksiköiden välisiä kanavia ja muita vastaavia, joissa kommunikoidaan suuremmalle joukolle kuuluvia asioita.

Havaintojen perusteella suurin osa Slack-kommunikaatiosta hoituu yksityisviestien kautta. Yksityisviestejä käytetään silloin, kun koetaan, että asialla ei haluta häiritä koko tiimiä tai keskustelun sisällöstä ei olisi hyötyä yleisellä kanavalla. Slackin neuvottelupuheluita käytetään säännöllisesti, kun halutaan järjestää palaveri useamman ihmisen kesken, mutta myös muita videokonferenssipuheluita tukevia ohjelmistoja käytetään. Slackin huonona puolena voidaan pitää sitä, että verrattuna kasvotusten keskusteluun

se nostaa kynnystä kysyä pienimmistä asioista, ja esimerkiksi yksityisviestein asioiden kommunikointi ja selvittäminen saattaa olla hitaampaa ja vaikeampaa.

Sähköpostia tunnutaan käytettävän lähinnä virallisemmassa kommunikoinnissa ja usein esimerkiksi asiakkaan kanssa kommunikoinnissa.

Organisaatiossa käytetään Jira-tehtävienhallintaohjelmistoa työtehtävien tiketöintiin, ja siellä käydään keskustelua sekä asiakkaiden että työkavereiden kanssa kehitystaskeihin liittyvissä asioissa. Jiraa käytetään usein myös tehtävien priorisointiin ja visualisointiin. Tämän lisäksi Confluence-wikiohjelmisto on käytössä dokumentointiin ja tiedon säilömiseen.

4.4 Työn ohjaus ja suunnittelu

Havainnoimalla voidaan huomata, että työn ohjauksessa käytetään yleisesti Jira-ohjelmistoa Jiran kanban-tauluja. Joillain tiimeillä on ollut yhteisissä tiloissa käytössä fyysinen taulu, jossa post-it lappuja siirrellään käsin statuksesta toiseen. Tästä on kuitenkin luovuttu, ja tällä hetkellä kaikki tiimit ovat siirtyneet digitaaliseen muotoon. Joissain tapauksissa visualisointi hoidetaan tiimitilassa olevan television kautta, jokainen toki pääsee myös oman tietokoneensa kautta käsiksi tauluihin.

Havaintojen mukaan varsinkin isommissa tiimeissä työtä suunnitellaan päivittäin. Avokonttorissa voidaan huomata, että monilla tiimeillä on aamuisin daily-palaveri, jossa jokainen kertoo vuorotellen mitä teki eilen, mitä aikoo tehdä tänään ja onko esteitä tai ongelmia. Tällä tavalla jokainen tiimin jäsen tietää mitkä ovat päivän prioriteetit, sekä mitä muut tekevät. Tällä tavoin myös esteisiin saadaan puututtua ajoissa, ennen kuin niistä muodostuu isompia ongelmia. Pienemmissä tiimeissä tai järjestelmissä, joissa ei välttämättä ole usein tekemistä, työtä ohjataan ja suunnitellaan tarpeen mukaan ja päivittäistä seurantaa ei ole käytössä.

Vapaamuotoisten keskustelujen perusteella pidempiaikaista suunnittelua asiakkuuksissa tehdään projektipäällikköjen ja asiakkuusvastaavien toimesta yhdessä asiakkaan kanssa. Käytänteet hieman vaihtelevat asiakkaasta ja työn määrästä riippuen. Yleisin tapa on se, että projektipäällikkö käy asiakkaan kanssa läpi kahden kesken keskustelemalla lähitulevaisuudessa alkavat projektit ja työt esimerkiksi viikoittain. Pidemmän aikavälin suunnitelmat taas hieman harvemmin, ja näissä keskusteluissa usein on mukana myös muita henkilöitä sekä asiakkaan että toimittajan puolelta. Kehitystiimin kanssa asioita käydään läpi sekä dailyissa että viikkopalavereissa, käytänteet ja frekvenssit toki vaihtelevat hieman tiimien ja asiakkuuksien välillä.

4.5 Tietämyksen ja osaamisen jakaminen

Havaintojen perusteella organisaatiossa ei ole tiettyä sovittua tapaa järjestää tietämyksen ja osaamisen jakamista, vaan vastuu on ollut tiimeillä itsellään. Joissain tiimeissä keinona on järjestää koodikatselmointeja, joissa tiimi istuu yhdessä alas tietyin väliajoin, esimerkiksi viikon välein, ja jokainen saa esittää viikon aikana tekemiään tuotoksia. Katselmoinnin tarkoituksena on levittää tietämystä tiimin sisällä siitä, mitä tiimikaverit tekevät, sekä levittää tietoa parhaista käytänteistä ja kohdatuista ongelmista.

Yleisenä haasteena useilta henkilöiltä on tullut esiin, että usein aikataulu- ja työmääräpaineiden vuoksi samat henkilöt päätyvät usein tekemään samantyyppisiä asioita. Tähän vaikuttaa myös eri osaamisprofiilit, sillä tiimissä on erityyppistä osaamista eri teknologioista, joiden välillä siirtyminen ei ole millään tavalla triviaalia, vaan vaatii täysin erityyppistä osaamista.

4.6 Havaitut haasteet

Tässä luvussa kuvatut haasteet ovat sekä omien havaintojen, yksikön johdon kanssa käytyjen keskustelujen, että vapaiden keskustelujen perusteella tehtyjä johtopäätöksiä organisaation haasteista.

Moniasiakkuustiimeissä haasteita tuottaa aika-ajoin töiden priorisointi eri asiakkaiden välillä. Jos yhdellä kehittäjällä on useampia asiakkuuksia, joudutaan aika-ajoin tehtäviä priorisoimaan näiden välillä, joka voi johtaa hankaliin tilanteisiin. Tähän ajaudutaan varsinkin silloin, kun ylläpidossa on yksittäisiä pieniä järjestelmiä ja pieniä asiakkuuksia, joissa tulee epäsäännöllisin väliajoin epäsäännöllisen kokoisia pyyntöjä. Tällöin ei ole mielekästä kasata kokonaista tiimiä asiakkuuden tai järjestelmän ympärille, mutta myöskin yhdessä tekijässä on riskejä kuten asiakkaiden välinen priorisointi, sairastumisriski, lomat ja esimerkiksi työntekijän lähtö yrityksestä. Haaste on yksikön johdon tiedossa ja siihen pyritään jatkuvasti keksimään parempia tapoja.

Jatkuviin palveluihin kuuluva häiriönhallinta voi vaikuttaa kehityksen aikatauluihin, sillä häiriöitä on vaikea ennustaa etukäteen. Jos samat kehittäjäresurssit on varattu sekä uuden kehitykseen että vanhan ylläpitoon, aiheutuu ylläpidosta tulevista häiriönhallintatöistä aikatauluhaasteita kehitystyöhön. Reagointinopeus häiriönhallintaan halutaan pitää hyvänä, joten kehittäjien on tunnettava järjestelmä ja sen erityispiirteet, joka tarkoittaa sitä, että käytännössä henkilöt, jotka kehittävät sinne uutta tekevät on luonnollisinta hoitaa myös häiriönhallintaa.

Integraatioprojektit ovat usein kahden tai useamman osapuolen välissä toimimista, jonka seurauksena on haastavaa ennustaa tulevia esteitä tai ongelmia. Osapuolten määrän

kasvaessa ennustettavuus heikkenee entisestään, sillä lähes missä tahansa tapahtuva odottamaton ongelma vaikuttaa lähes poikkeuksetta myös integraatioiden aikatauluun, jonka seurauksena integraation toimittaja joutuu suunnittelemaan tekemisiään uusiksi. Usein vaaditaan huomattavaa joustoa ja priorisointi voi muuttua päivänkin aikana useasti, jos muualla tapahtuvat ongelmat vaikuttavat integraatioprojektiin.

Yksittäisen tekijän ylikuormitus on todennäköinen haaste, jota on todennäköisesti mahdoton kokonaan poistaa, mutta sitä varten voidaan varautua. Tekijöitä tälle on muun muassa pienet asiakkuudet, joissa ei ole riittävästi tekemistä isolle porukalle, joten osaaminen jää usein yhden tai kahden henkilön varaan. Jos sitten tällaisessa pienessä asiakkuudessa tulee yllättäen normaalia kovempi työkuorma, voi yksittäisen tekijän työkuorma lisääntyä, jos esimerkiksi muissakin asiakkuuksissa lisääntyy työmäärä samaan aikaan, tai esimerkiksi tiimissä esiintyy samaan aikaan sairaslomia. Toki yksi keino varautua tätä vastaan olisi lisätä tiimin kapasiteettia, mutta se on ongelmallista siinä kohtaa, jos työtä ei ole riittävästi. Tällaisiin kuormapiikkeihin vastaaminen siten, että sekä asiakas, työntekijä että työnantaja ovat tyytyväisiä, on yksi keskeisimmistä ongelmista, joihin tässä diplomityössä pyritään vastaamaan.

Tiedon, varsinkin hiljaisen tiedon siiloutuminen tietyille tekijöille on usein ongelma. Se aiheuttaa ongelmia sekä tekijöille, joille tieto on siiloutunut, että asiakkaalle ja tiimille. Tekijälle siiloutunut tieto on siinä vaiheessa ongelma, kun puhutaan ylikuormittumisesta ja työssä jaksamisesta. Jos henkilö on ainoa kenellä on riittävä tieto jostain asiasta mitä tarvitaan, ja hänellä olisi kuitenkin muita tehtäviä, sillä hetkellä tehtävänä, se voi johtaa ylikuormitukseen ja asioiden priorisointiin. Asiakkaalle tiedon siiloutuminen taas näkyy ongelmana siinä kohtaa, jos esimerkiksi kyseistä tekijää ei saada juuri tiettyä asiaa tekemään. Syinä voi olla vaikkapa muualle priorisoidut työtehtävät, sairasloma tai henkilö voi olla lähtenyt yrityksestä. Näissä tapauksissa, varsinkin jos dokumentaatio ei ole riittävällä tasolla, asian selvittämiseen voi mennä turhauttavan kauan aikaa ja resursseja. Kuten edellä mainittiin, eräs keino taistella tätä vastaan on hyvä dokumentaatio. On kuitenkin käytännössä havaittu, että pelkästään se ei ole riittävä tapa yllättävien tilanteiden sattuessa eteen, joissa tulisi omaksua iso määrä tietoa pelkästä dokumentaatiosta. Iso osa varsinkin integraatiotyössä tarvittavista tiedoista on sellaista tietoa, jota on vaikea tai mahdoton dokumentoida, vaan se leviää parhaiten tekemällä ja kommunikoimalla.

4.7 Hyvät puolet

Tässä kappaleessa kuvatut hyvät puolet perustuvat omiin kokemuksiin ja asiantuntemukseen, sekä keskusteluihin monien eri ihmisten kanssa. Havainnot perustuvat sekä omiin että muiden tiimien työskentelyyn.

Työrauhaan ja yhteen tehtävään kerrallaan keskittymiseen on päästy keskittymällä kulttuurin muutokseen, sekä asiakkaalla että talon sisällä. Sen sijaan, että työpyynnöt tulisivat suoraan kehittäjälle esimerkiksi puhelimella tai sähköpostilla, on opetettu kaikki osapuolet tiketöintijärjestelmän käyttöön. On havaittu, että työpyyntöjen tiketöinti rauhoittaa kehittäjiä ja poistaa stressiä. Kun tulevat tehtävät ovat näkyvissä taululla, niin ne eivät rasita kognitiivisesti mielen perukoilla, vaan kehittäjä voi aina luottaa, että seuraava tehtävä löytyy taululta prioriteettijärjestyksessä, eikä siitä tarvitse huolehtia ennen kuin edellinen on tehty.

Kehittäjille on annettu nykymallissa vapaus päättää asioista, joilla tuodaan merkityksellisyyden tunnetta työhön. Käytännössä tämä tarkoittaa sitä, että kun kehittäjä ottaa itselleen tehtävän taululta, hän saa itse päättää kuinka sitä lähtee toteuttamaan ja millä tavalla, toki realismin rajoissa. Kommunikoinnissa pyritään mahdollisimman välittömään yhteistyöhön asiakkaan, eli työn tilaajan kanssa, jotta vältytään turhalta välikerrokselta ja ”rikkinäiseltä puhelimelta”. Vapauden ja vastuun antaminen tuottaa parhaassa tapauksessa itseohjautuvan ja työssä viihtyvän tiimin, joka kykenee itsenäisesti suunnittelemaan, koordinoimaan ja toteuttamaan monimutkaisia usean sidosryhmän välisiä integraatioprojekteja. Toki tietynlaista ohjausta ja tukea tulee antaa, jotta kehittäjät eivät ylikuormitu liiasta epävarmuudesta ja eri puolilta tulevista pyynnöistä.

Yhdessä toimiminen ja yhteinen vastuu toimii tällä hetkellä melko hyvin, sillä kehitystiimin jäsenet pystyvät ratkomaan pulmia ja keskustelemaan ratkaisuista itsenäisesti ilman kolmannen osapuolen koordinointia. Tämä on osa tiimin DNA:ta, joka ei suoranaisesti johdu mistään yksittäisistä tekijöistä tai menetelmistä, vaan on tietyllä tapaa sanattomasti kaikkien kesken jaettu osa toimintakulttuuria.

5. HAASTATTELUJEN TULOKSET

Tässä luvussa esitellään haastattelujen tulokset, sekä avattu tarkemmin, jos haastateltavilla oli eriäviä kommentteja haastattelukysymyksiin. Kaikki vastaukset on poimittu suoraan haastatteluista.

5.1 Yleistä

Haastattelujen perusteella muodostui melko yksimielinen kuva siitä, mitä ketterien menetelmien tekniikoita integraatiokehityksessä on mahdollista ja järkevää käyttää ja missä tilanteessa. Tämä antoi lisävahvistusta oletukselle, että haastateltavien henkilöiden määrä ja näkökantojen monipuolisuus on riittävä antamaan tutkimukselle validiteettia ja reliabiliteettia.

Kukaan haastateltavista ei kokenut, että yhtäkään yksittäistä menetelmää voitaisiin menetelmän kuvaamalla tavalla suoraan hyödyntää integraatiokehityksessä. Kaikkien mielestä on järkevämpää ottaa käyttöön parhaita käytänteitä ja tekniikoita eri menetelmistä, ja soveltaa niitä projektin ja tiimin mukaan. Tästä poikkeuksena kuitenkin on Lean Development, sillä sen esittelemät arvot ja periaatteet ovat sen verran yleisluontoisia, että niitä voidaan nähdä käytettävän lähes joka tiimissä jollain tavalla sovellettuna. Kyseessä kun ei ole tiukka viitekehys, vaan ennemminkin joukko ajattelun malleja ja periaatteita.

5.2 Haastateltavilla käytössä olevat menetelmät

Kaikilla haastateltavilla oli nykyisissä tiimeissään Kanban-taulu käytössä työnohjauksessa, sekä joitain Scrumin tekniikoita. Yksikään haastatelluista ei seurannut työssään tiukasti mitään tiettyä ketterää menetelmää, vaan näistä on otettu eri tekniikoita käyttöön sekalaisesti.

Scrumin tekniikoista ylivoimaisesti käytetyin oli daily, josta haastateltavat kokivat olevan hyötyä integraatiotyössä sen nopeasti muuttuvan luonteen takia. Dailyssa on mahdollista käydä päivittäin läpi tehtyjä ja työn alla olevia asioita, ja siinä on helppo priorisoida asioita tiimin sisällä. Integraatioasiantuntija A kommentoi dailyn hyödyksi nimenomaan sen, että tilaisuudessa voidaan käydä läpi päivän tehtävät, jolloin kehittäjän kannalta on selkeää, mihin keskittyä sen päivän osalta. Projektipäällikkö A ja integraatioasiantuntija B totesivat käyttävänsä dailyn lisäksi retroja ja backlog grooming -sessioita tiimin toiminnan kehittämiseen ja uusien töiden määrittelemiseen.

Integraatiokehittäjä B ja projektipäällikkö A kommentoivat suoraan, että heidän tiimissään on kokeiltu erilaisia menetelmiä, mutta lopulta valittu eri menetelmistä parhaiten sopivia tekniikoita siten, mitkä sattuvat sopimaan kyseiseen projektiin ja tiimiin. Näin on saatu aikaiseksi systeemi, joka toimii, mutta sitä ei ole sen tarkemmin dokumentoitu tai määritelty.

5.3 Yleiset kokemukset ketteristä menetelmistä

Senior-integraatioasiantuntija A:n mukaan integraatioiden luonteeseen kuuluu iteroiva työ ja palautteen perusteella korjaus, joka tarkoittaa, että harvoin päästään tilanteeseen, jossa tehdään valmiin speksin perusteella toteutus alusta loppuun. Tämän vuoksi esimerkiksi Scrumin ei koeta soveltuvan integraatiotyöhön erityisen hyvin, kun taas Kanban ja Lean koetaan paljon sopivammaksi. Jokainen haastateltava oli tämän suhteen samoilla linjoilla, tosin tiimipäällikkö A totesi, että jossain projekteissa voidaan saada Scrum toimimaan, mutta itse ei ole ollut sellaisessa mukana. Kaikki haastateltavat nostivat erityispiirteeksi sen, että integraatiotyössä joudutaan odottamaan kolmansia osapuolia ja tekemään heidän kanssaan läheistä yhteistyötä, joka asettaa rajoitteensa käytettäville menetelmille. Projektipäällikkö A kommentoi, että tämän vuoksi kuitenkaan esimerkiksi vesiputousmalli ei sovellu käytettäväksi integraatiotyöskentelyyn, vaan on jopa luonnollista tehdä integraatioita ketterästi.

Senior-integraatioasiantuntija A kommentoi, että hän ei ole ollut projektissa, jossa tehtäisiin tiukasti jonkun tietyn menetelmän mukaan, mutta ei sulje pois mahdollisuutta, että sellaisesta toimintamallista voisi jossain projekteissa ollakin hyötyä. Hänen kokemuksensa mukaan monta kertaa on alettu tekemään jollain mallilla, mutta ajan myötä se on sitten muokkaantunut sellaiseen muotoon, mikä sopii siihen tiimiin ja projektiin.

Yleisesti hyvän ketterän menetelmän ominaisuutena pidetään sitä, että se ruokkii jatkuvaa oppimista ja tiimissä eri henkilöt kirjoittavat samankaltaista koodia tietyllä laatutasolla. Projektipäällikkö A kommentoi, että kun tiimille kertyy kokemusta, pystytään arvioimaan, kuinka nopeasti töitä tehdään ja arvioimaan tulevia töitä tarkemmin, jonka myötä taas asiakas saa parempaa vastinetta rahoilleen. Tiimipäällikkö A:lla otti aiheeseen liittyen esiin asiakasnäkökulman, eli jos toimittajalta on iso tiimi tekemässä asiakasprojektia, niin asiakas haluaa todennäköisesti tietää kuinka paljon työtä on mahdollista tehdä tietyssä ajassa. Projektipäällikkö A totesi, että on varmasti henkilöstä kiinni, kuinka mielekkääksi kokee ketterän toimintamallin, jossa keskitytään erityisesti laatunäkökulmaan. Hänen mukaansa on hyvä, että kaikki ymmärtävät yleisimpien ketterien menetelmien perusperiaatteet, jotta voidaan valita sopivimmat menetelmät kulloiseenkin tilanteeseen. Usein yhteistyö asiakkaan kanssa määrittelee pitkälti sen, miten sitä kannattaa tehdä.

5.4 Ketteristä menetelmistä koetut hyödyt integraatiotyössä

Kaikkien integraatioasiantuntijoiden yhteinen mielipide oli se, että tärkeimpänä ketterien menetelmien hyötynä heidän kannaltaan on työn priorisointi ja selkeä työjono. Sen avulla kehittäjä voi keskittyä kussakin tilanteessa oleelliseen ja tärkeimpään tehtävään, ilman että hänen tarvitsee käyttää aikaa ja energiaa asian erikseen huolehtimiseen. Projektipäällikön A ja tiimipäällikön A näkökulmasta taas tärkeimmäksi näkökulmaksi nousi se, että ketterien menetelmien myötä saadaan parempi näkyvyys tekemiseen sekä tiimille että asiakkaalle, arvonluonti läpinäkyvämmäksi sekä asiakkaan että tiimin suuntaan ja sitoutettua asiakas yhteisiin tekemisen tapoihin.

Senior-integraatioasiantuntija A:lla oli positiivisia kokemuksia Scrumista. Tässä tapauksessa asiantuntija oli ollut osana sovelluskehitystiimiä tekemässä yksin integraatioita, kun lopputiimi teki muita sovellukseen liittyviä tehtäviä. Avaintekijänä asiantuntija mainitsi hyvän Scrum Masterin ja sitoutuneen Product Ownerin, joiden avulla sprintteihin saatiin tavoite ja saatiin kehitys menemään eteenpäin. Tärkeää oli se, että integraation toteutuksen jälkeen se ei jäänyt roikkumaan odottamaan testausta, vaan koska koko tiimi oli sitoutunut toimittamaan kokonaisuuksia, myös integraatiot tulivat testattua sprintin sisällä. Sama asiantuntija totesi, että toisessa asiakkuudessa taas Kanban on toiminut huomattavasti paremmin eikä tiukkaa Scrumia voisi kuvitellakaan toimivan, koska kyseisessä asiakkuudessa pitäisi varata suuri osa ajasta suunnittelemattomalle työlle, joka pullahtaa testauksesta takaisin.

Integraatioasiantuntijat A ja B nostivat päivittäisen statuksen tärkeimmäksi ketteräksi tekniikaksi keinona työn ohjaamisessa. Vaikka ei noudatettaisikaan puhdasta Scrumin Dailya, niin päivittäisen statuksen avulla voidaan löytää nopeasti ongelmakohtia ja edetä oikeaan suuntaan, jos ongelmia ilmenee. Näin ainakin tiimin sisällä ongelmiin reagointi pysyy ketteränä. Varsinkin projektin kriittisissä vaiheissa päivittäisen seurannan hyöty korostuu, koska näin yksittäiset henkilöt eivät jää jumiin ongelmakohtiin vaan dailyn jälkeen esimerkiksi tiimikaveri voi heti tulla viereen katsomaan ongelmaa ja auttamaan.

Tiimipäällikkö A nosti suureksi hyödyksi tehtävien priorisoinnin, joka hyödyttää sekä asiakasta että kehittäjää. Asiakkaan puolelta tuoteomistaja saa tärkeää näkyvyyttä siihen mitä tehdään, ja missä tilanteessa eri tehtävät menevät. Asiakkaan osallistuminen on hänen mukaansa oleellista, jotta saadaan tuotettua arvoa ja tehtyä oikeita asioita. Käytännön tekniikkana priorisointi varmistaa sen, että pystytään keskittymään jatkuvasti kulloinkin tärkeimpään asiaan ja asiakkaalle tuotetaan maksimaalista arvoa. Toisaalta

myös kehittäjälle on tärkeää, että on selkeä tehtävälista, koska usein tulee monia asioita päällekkäin ja on oleellista, että ne pystytään laittamaan jonoon ja tärkeysjärjestykseen.

Integraatioasiantuntija B nosti esiin Kanban-taulun toimivuuden työn ohjauksessa ja priorisoinnissa, kun kaikki tehtävät taululla olivat samasta projektista ja kaikki henkilöt tiimistä tekivät samaa projektia. Hänen mielestään on tärkeää, että samalle taululle ei sotketa tehtäviä useammasta projektista tai asiakkuudesta, koska se sekoittaa priorisoinnin ja resursoinnin. Integraatioasiantuntija A:lla oli kanban-taulusta tosin kommentti, että taulusta häviää silloin hyöty, kun tiketit seisovat pitkään sarakkeissa koska tiimi ei pysty edistämään niitä ilman kolmannen osapuolen apua.

Projektipäällikkö A nosti ketteristä tekniikoista erityisesti Definition of Donen esiin. Kun se on selkeä, mahdollisimman kattava ja kaikkien allekirjoittama, yhtenäisyys kehityksessä ja tuotannon pyörittämisessä pysyy hyvänä ja esimerkiksi koodi pysyy luettavana ja yhtenäisenä. Tavoitteena on se, että tiimillä on yhteinen ymmärrys siitä, millä tavalla tiimissä tehdään työtä ja kun työtavat ja tuotokset ovat yhtenäisiä, on kehittäjän nopeampi lukea toisen koodia ja ymmärtää mitä siinä on tehty.

5.5 Ketterien menetelmien haasteet integraatiotyössä

Senior-integraatioasiantuntija A:n ja tiimipäällikkö A:n mielestä yksi suurimmista haasteista, kuten myös kirjallisuudessa on havaittavissa, on sopivan ja sitoutuneen PO:n löytäminen tiimille. PO voi periaatteessa tulla joko asiakkaan tai toimittajan puolelta, mutta haastattelujen perusteella luontevina olisi, jos PO löytyisi asiakkaan puolelta. Senior-integraatioasiantuntija A totesi, että integraatiot eivät monesti ole asiakkaan puolelta kenenkään päätyötä, vaan sivujuttu mitä pitää tehdä. Tämä asettaa omat haasteensa esimerkiksi roadmap-suunnittelussa, ja korostaa entistä enemmän sopivan PO:n löytämisen tärkeyttä.

Kaikki haastateltavat nostivat jossain muodossa haasteena esiin kolmannet osapuolet ja heidän aikataulujen sovittaminen julkaisurytmiin tai tiimin aikatauluihin. Integraatioasiantuntija A totesi, että usein asiakkaalle tehtävissä integraatiotehtävissä vaaditaan panosta myös asiakkaan asiakkaalta tai jopa heidän toimittajiltaan, jolloin eri osapuolten määrä pienissäkin kehitystehtävissä saattaa nousta suureksi. Tämän vuoksi työajassa yksinkertaiset tehtävät voivat viedä kalenteriajassa paljon aikaa. Senior-integraatioasiantuntija A taas kommentoi, että integraatiokehityksen luonteeseen kuuluu se, että niiden hyväksyntätestauksen voi usein tehdä vain integroitavien järjestelmien loppukäyttäjät ja loppupeleissä integraatioiden rakentaminen on yhteistyötä. Tämän vuoksi kehityksen

loppuun saattaminen ei usein ole kehitystiimin hallinnassa, vaan vaatii kolmansia osapuolia. Hänen mukaansa siitä johtuen tehtäviä voidaan joutua roikottamaan listoilla pitkiäkin aikoja kalenteriajassa, koska välttämättä ei ole tiimin hallinnassa, koska testiympäristöön vietyä integraatiota testataan. Integraatioasiantuntija A kommentoi aiheeseen liittyen, että testausvaiheessa usein paljastuu asioita, joita ei ole huomattu tai ymmärretty pyytää alkuperäisissä vaatimuksissa, tai siinä kohtaa sitä ei ole koettu oleelliseksi, mutta testauksen yhteydessä todetaan, että joku tietty toiminnallisuus olisikin hyvä olla integraatiossa. Sen seurauksena integraatio palautuu testauksesta takaisin kehitettäväksi, ja jatkokehityksellä voi olla nopeakin aikataulu, sillä projekti on jo testausvaiheessa ja aikataulut on jo lyöty lukkoon muualla. Tämä näyttäytyy integraatiotiimille satunnaisena ja nopealla aikataululla tulevana kehitystyönä, jota on erittäin vaikea ennustaa ja esimerkiksi priorisoida etukäteen. Esimerkiksi Scrum-menetelmään tämän tyyppiset kehitykset eivät myöskään istu lähes ollenkaan, koska harvoin projektilla on aikaa odottaa seuraavaan sprinttiin, jotta kehitys saadaan testattavaksi.

Projektipäällikkö A:n mielestä eräänä haasteena integraatiotyössä on eri osapuolten koordinointi, ja sitä ongelmaa ratkaisemalla voidaan saada sujuvammaksi koko julkaisuprosessi. Koordinoivan osapuolen löytäminen tosin ei aina ole aivan helppoa, sillä integraatiotyössä saattaa olla useita osapuolia, joista kukaan ei selkeästi ole koko ratkaisun omistaja.

Integraatioasiantuntija B:n kommenteissa nousi esiin se, että tehtävät tulisi pyrkiä osittamaan riittävän pieniin osiin, jotta niitä voidaan edistää irrallisina muista kokonaisuuksista. Näin tehtävät pystyttäisiin merkitsemään omalta osalta valmiiksi, vaikka jokin vaihe odottaisi vielä kolmatta osapuolta. Haittana tässä on se, että tehtävien määrä esimerkiksi Kanban-taululla nousee nopeasti niin suureksi, että kokonaisuuden hallinta muuttuu vaikeaksi ja työaikaa kuluu erillisten taskien luomiseen paljon suhteessa itse tekemiseen.

Määrittelyjen ja esityön merkitys nousi esiin useammalta haastateltavalta. Periaatteena monissa ketterissä menetelmissä on se, että ei pitäisi aloittaa tehtävää ennen kuin on mahdollista saada se kerralla valmiiksi. Senior-integraatioasantuntija A:n mukaan integraation toteutuksesta ei ole välttämättä saatavilla tietoa ennen kuin sitä aletaan tekemään, ja asioita paljastuu vasta kun niihin törmätään toteutusvaiheessa. Tästä seuraa, että usein määrittelyjä voidaankin tehdä vasta siinä vaiheessa, kun toteutusta on jo tehty jonkun verran ja on mahdollista keskustella integroitavien järjestelmien kanssa. Tiimipäällikkö A:n kokemus oli se, että usein tehtävät on aikataulutettu ja määritelty, mutta kun niitä aletaan tekemään, niin huomataankin että asia ei olekaan niin kuin on määritelty ja joudutaan määrittelemään tehtävä uusiksi, joka puolestaan näkyy aikatauluissa.

Integraatioasiantuntija A nosti isoksi ongelmaksi sen, että tekeminen on usein katkonaista, ja tarkennuksia määrittelyihin joudutaan kysymään siinä kohtaa, kun tekeminen on jo aloitettu, josta seuraa helposti kontekstinvaihtoa. Hänen mukaansa esimerkiksi Scrumissa tekniikkana oleva taskien etukäteismäärittely on haastavaa, koska usein integraatiotehtävää ei voida määritellä loppuun asti ennen kuin sen tekeminen on jo aloitettu, josta aiheutuu looginen ongelma tekniikan käytön suhteen.

Eräs haaste liittyy integraatioiden toimitusmalliin. Senior-integraatioasiantuntija kommentoi, että integraatioita tuotetaan lähes aina jatkuvina palveluina, jossa kehitys sekä ylläpito hoidetaan saman tiimin toimesta. Jatkuvissa palveluissa vaatimukset ovat erilaisia verrattuna projekteihin, sillä projekteissa on yleensä selkeä aikataulu, vaatimukset ja tavoitteet, kun jatkuvat palvelut ovat ennemminkin jatkuvaa priorisointia, pienkehitystä ja häiriönhallintaa. Kun kaikkia tarpeita ei tiedetä etukäteen, se tuottaa suuria haasteita siirryttäessä ketteriin menetelmiin, jotka on usein kehitetty projekteja ja tuotekehitystä silmällä pitäen. Tiimipäällikkö A oli myös sitä mieltä, että jatkuvien palveluiden mallissa on vaikea löytää Scrumin roolituksille paikkaa, ja vaatisi ison projektin, jotta esimerkiksi SM:lle olisi oikeasti tarvetta ja mahdollisuus työskennellä.

5.6 Asiakkaiden suhtautuminen ketteriin menetelmiin

Haastateltavien kokemukset asiakkaiden suhtautumisesta ketteriin menetelmiin vaihtelivat. Osa asiakkaista jopa kannustaa ketterien menetelmien käyttöön, kun taas osa asiakkaista ei ole valmiita ketterään tekemiseen, vaikka toimittajan puolelta siihen olisi halua. Täten haastattelujen perusteella ei siten voida tehdä yleistyksiä siitä, miten asiakkaat tällä hetkellä kokevat ketterät menetelmät. Yleiskuva kuitenkin oli se, että asiakkaat tuntevat tänä päivänä ketterät menetelmät paremmin kuin ennen, ja suhtautuvat niihin positiivisemmin verrattuna aikaisempaan.

Asiakkaan rooli ketterien menetelmien käytössä koetaan tärkeäksi. Senior-integraatioasiantuntija A kommentoi, että jos halutaan oikeasti tehdä integraatioita ketterästi, tulisi asiakkaan puolelta nimetä henkilö, joka edustaa liiketoimintaa ja jolla on riittävästi auktoritettia tehdä liiketoiminnan puolesta päätöksiä, keskustella heidän kanssaan ja joka saa riittävän ajoissa tietoa uusista asioista. Tasan yksi nimetty henkilö on tärkeää siksi, että jos mukana on liikaa sidosryhmiä, tekeminen alkaa helposti rönsyilemään ja esimerkiksi priorisointi koetaan haastavaksi. Sama henkilö totesi myös, että asiakkaan suhtautuminen ja mukanaolo ketterässä kehityksessä liittyy suoraan siihen, miten paljon asiakas kokee saavansa arvoa kyseisistä toimintamalleista. Jos asiakas ei näe paljoa arvoa siinä, niin varsinkin PO:n rooli jää helposti nimelliseksi, jolloin toteutustiimi voi toki yrittää

pyörittää omaa menetelmäänsä, mutta osa ketterien menetelmien hyödyistä jää realisoitumatta. Hänellä on kertomansa mukaan ollut asiakkuuksia, jossa asiakas on ollut mukana tiiviisti ja sitoutunut Scrumiin, ja joka on toiminut hyvin siten että asiakkaan PO on toiminut oikeasti rajapintana asiakkaan liiketoiminnan ja kehitystiimin välissä.

Tiimipäällikkö A oli huomannut sen, että kun asiakkaalle on oma-aloitteisesti tuotu ja opetettu uusia tapoja tehdä asioita ketterästi, ovat he yleensä antaneet siitä positiivista palautetta. Varsinkin silloin, kun asiakkaalle on tarjottu työkaluja, joilla he ovat saaneet parempaa näkyvyyttä tiimin tekemiseen ja päässeet itse vaikuttamaan esimerkiksi priorisointiin, on ollut suuri vaikutus asiakkaan suhtautumisessa ja sitoutumisessa ketterään tekemiseen. Tämä on luonut tyytyväisyyttä molemmissa osapuolissa ja haastatellut ovat kokeneet sen hyödyllisenä.

5.7 Scrum integraatiotyössä

Scrumin roolijako toimii haastattelujen perusteella integraatiotyössä melko hyvin. Kommentit roolijaon suhteen olivat melko yksimielisiä: normaaleissa integraatioprojekteissa ja jatkuvissa palveluissa PO:n tulisi tulla asiakkaan puolelta, kun taas SM ja kehitystiimi toimittajan puolella. Projektipäällikkö A tosin totesi, että on myös mahdollista, että PO tulisi toimittajan puolelta, joka keskustelisi eri sidosryhmien kanssa, mutta asiakkaan PO on luontaisinta. Senior-integraatioasiantuntija A:n mukaan asiakkaan PO:n tulisi olla yksi henkilö, eli roolia ei mielellään saisi jakaa useamman henkilön kesken. Tärkeimmät tehtävät tälle roolille on pystyä priorisoimaan tekemiset, tietää mitä liiketoiminta ja eri sidosryhmät haluavat ja kommunikoida näiden välillä. Useampi PO aiheuttaa herkästi hankaluuksia, kun pitäisi priorisoida tekemistä, mutta näillä on eri prioriteetit.

Asiakkaan PO:n tulisi tuntea integraatiotyö yleisesti sen verran hyvin, että pystyy ennakoimaan tulevia tilanteita ja suunnittelemaan työtä. Tiimipäällikkö A totesi, että talon sisältä tuleva PO vaatisi kohtuullisen suuren projektin, mutta tällaisesta ei haastateltavilla ollut kokemusta ennestään. Senior-integraatioasiantuntija A:n mukaan malli voisi toimia esimerkiksi projektissa, jossa integraatiot ovat yksi osa toimitusprojektia, joka on kokonaisuudessaan saman toimittajan hallussa. Hänen mukaansa luontaisinta kuitenkin perinteisessä integraatiotyössä on, että PO on asiakkaan edustaja ja loput tiimistä toimittajan puolelta. Kaikissa tilanteissa Scrumin roolijako ei kuitenkaan ole optimaalinen. Senior-integraatioasiantuntija A:n mukaan voi olla hyödyllistä, että esimerkiksi tiimissä on erikseen integraatioarkkitehti, joka keskustelee asiakkaan teknisen arkkitehdin kanssa, joka on eri kuin PO. Scrum masterin hyödyllisyyskin riippuu täysin siitä, millaisen viitekehyksen mukaan töitä tehdään – Scrumin kanssa luonnollisesti lähes pakollinen, mutta muiden viitekehysten kanssa roolin hyödyllisyydestä voidaan olla montaa mieltä, ja kuten

tiimipäällikkö A totesi niin mahdollisesti perinteisempi projekti/palvelupäällikön rooli istuu näissä tapauksissa paremmin integraatiotekemiseen.

Rajatun mittaisesta ja laajuisesta sprintistä haastateltavien kommentit olivat pääosin negatiivisia. Tämä johtuu siitä, että Scrumissa pitäisi sprintille valitut tehtävät saada sen sprintin aikana valmiiksi. Integraatiotyössä kuitenkaan harvoin tähän on mahdollisuutta. Integraatioasiantuntija A piti suurimpana ongelmana sitä, että integraation toteuttaminen ei useinkaan työmäärällisesti vie paljoa aikaa, mutta kalenteriaikaa sen valmiiksi saamiseen voi mennä moninkertaisesti. Se johtuu siitä, että integraatiotyössä ollaan perinteiseen sovelluskehitykseen verrattuna huomattavan paljon riippuvaisempia kolmansista osapuolista, heidän määrittelyistään ja aikatauluistaan. Lisäksi asiakkaalta tulee paljon tekemistä ”ohituskaistaa pitkin”, jonka seurauksena alkuperäiset aikataulut eivät pidä. Senior-integraatioasiantuntija A totesi että vaikka oletetaan, että integraation määrittelyt olisivat täydelliset, kaikki esivalmistelut kuten tietokantataulut, palomuuriavaukset ja rajapinnat ovat valmiina ja integraatio saadaan kehitettyä sovitusti aikataulussa, voi sen valmiiksi saaminen silti venyä kalenteriajassa esimerkiksi sen takia että kolmannella osapuolella ei ole aikaa testata integraatiota tai heillä prioriteetit muuttuvat jostain syystä, eikä integraation valmiiksi saaminen olekaan niin tärkeää sillä hetkellä. Voi myös olla, että kun integraatiota päästään testaamaan, huomataan että jonkun asian tulisikin toimia eri tavalla kuin alun perin oli suunniteltu, ja se pitää korjata pikaisesti. Tämä voi tapahtua useita viikkoja sen jälkeen, kun sprintti, jolle integraation kehitys on suunniteltu, on jo valmistunut. Tämän tyyppiset yllätykset vaikuttavat sprintin aikatauluun ja laajuuteen, ja näitä integraatioasiantuntija A:n ja senior-integraatioasiantuntija A:n kokemuksen mukaan tapahtuu usein. Vaikka oletetaankin, että kaikki esivalmistelut ja määrittelyt on tehty täydellisesti, niin tosielämässä myös näiden kanssa tulee aika-ajoin ylimääräistä säätämistä loppuvaiheessa. Senior-integraatioasiantuntija A:n mukaan tämä on toki yleisesti sovelluskehityksen ongelma, eikä pelkästään integraatiokehityksen. Kuitenkin muun muassa näistä syistä johtuen integraatiotiimin on hyvin vaikea sitoutua kalenteriajassa ja laajuudessa rajattuihin sprintteihin. Jotta tämän tyyppinen toimintamalli voisi käytännössä toimia, hänen mukaansa tulisi kaikkien integraatioihin liittyvien sidosryhmien sitoutua samoihin aikatauluihin ja sprintin laajuuteen, joka muuttuu sitä vaikeammaksi mitä monimutkaisempi ja useaan osapuoleen integroitu ympäristö on kyseessä.

Eri Scrum-palaverit kuten **Daily Scrum**, **Sprint Planning**, **Sprint Review** ja **Retrospective** saivat haastatelluilta positivista kommenttia. Varsinkin Dailyt koetaan jopa pakolliseksi, kun puhutaan toimivasta integraatiotiimistä. Integraatioasiantuntija A nosti sen seikan esiin, että dailyn avulla saadaan mahdolliset ongelmat ja esteet nopeasti, oikein

käytettynä alle päivän reaktioajalla selville ja niistä päästään keskustelemaan tiimin kesken. Myös integraatioasiantuntija B oli samoilla linjoilla. Hänen mukaansa Iílman dailya kehittäjä voisi jäädä jumiin ongelmiensa kanssa, jos jostain syystä apua ei ole helposti saatavilla eikä sitä oma-aloitteisesti ymmärtäisi siinä tilanteessa kysyä. Hänen mukaansa dailyn avulla tällaiset tilanteet tulee jopa ”pakosti” ilmi, ja niihin voidaan reagoida välittömästi. Integraatioasiantuntija B kuitenkin totesi, että daily voi jossain tilanteissa kuitenkin olla hyödytön, jos tiimin jäsenet tekevät keskenään hyvin erilaista työtä, niin he eivät välttämättä ymmärrä mistä toinen puhuu omalla vuorollaan, joten puuttumismahdollisuudet ovat hyvin rajalliset. Sprint Planningista kommentit olivat samansuuntaisia kaikilla, eli pakollinen osa jos halutaan pyörittää Scrumia, mutta muuten sen hyödyllisyydestä voidaan olla montaa mieltä. Jos ei haluta toimia täysin Scrumin mukaan, niin sama hyöty saadaan esimerkiksi viikottaisilla palavereilla, kuten projektipäällikkö A toimii projektissaan. Hän on sitä mieltä, että jonkunlainen näkyvyys ja suunnittelu tuleviin töihin on kuitenkin hyvä olla, olkoon se sitten Sprint Planningin tai jonkun muun nimen alla. Tämän avulla saadaan tiimille motivaatiota ja sitoutumista, ja ihmiset ymmärtävät paremmin kokonaiskuvaa sekä mitä seuraavaksi on tulossa. Integraatioasiantuntijan mukaan toimiva malli vaatii sitoutumista ja ”ohituskaistan” pitäisi olla mahdollisimman pieni. Sprint Planningiin liittyy Backlog Refinement, joka olisi senior-integraatioasiantuntija A:n mukaan hyödyllinen käydä läpi aika-ajoin erityisesti asiakkaan kanssa. Näin tulisi keskusteltua tehtävät läpi etukäteen, ja paljastuisi jo etukäteen mistä määrittelyistä puuttuu tarpeellinen tieto eikä tietoja tarvitsisi alkaa kaivelemaan vasta silloin kun työn tekeminen aloitetaan. Retrot koetaan hyödyllisiksi, mutta ehkä kahden viikon välein niitä ei ole tarpeellista järjestää. Kuitenkin selkeästi useammin kuin kerran vuodessa, integraatioasiantuntija B:n kokemuksen mukaan niin pitkällä aikajänteellä ei ehditä enää puuttumaan päivän polttaviin ongelmiin. Niissä on hyvä käydä aika-ajoin kevyesti läpi, onko tiimillä parannettavaa esimerkiksi toimintatavoissa tai kommunikaatiossa. Review on hyödyllinen silloin, jos on jotain näytettävää, mutta harvoin integraatioissa on mitään sellaista, jota voitaisiin demota tai näyttää porukalle. Projektipäällikkö A totesi että integraatioita on vaikea demota, toteutusta voidaan toki käydä läpi, mutta se on ennemminkin tiimin sisäinen asia, ja siitä ei todennäköisesti ole asiakkaan kaikille sidosryhmille erityistä hyötyä.

Storyja ja **Story Pointeja** vierastetaan senior-integraatioasiantuntija A:n mukaan integraatiomaailmassa, koska tekeminen on ennemminkin laitteiden välistä integraatiota, jolloin ei ole ”user”ia joka voisi kirjoittaa Storyn. Toisaalta hän kokee, että integraation määrittelyssä olisi hyvä olla taustatietomielessä tarina, joka ei ole pelkkä tekninen selostus, vaan jossa kerrotaan mitä integraatio käytännössä tekee eli mistä tieto tulee, mitä sille tehdään ja minkä takia. Integraatioasiantuntija A kommentoi, että Story Pointien arviointi

on melko vaikeaa integraatiotekemisessä, koska tekeminen harvoin on määrämittaista ja itseään toistavaa, jolloin pystyttäisiin vertailemaan tekemistä helpommin toisiinsa. Senior-integraatioasiantuntija A nosti esiin seikan, että pisteiden arviointi tiimin kesken olisi siinä mielessä hyödyllistä, että jos arviot suuresti eroavat toisistaan, niin tällaisista keskusteluista voi selvitä paljonkin siitä mitä pitää tehdä. Tiimipäällikkö A löytäisi hyötyä Storyjen käytöstä sitä kautta, että asiakkaan edustajat voisivat huomata määritellä tarkemmin bisnestarpeen ja tarpeen, eli miksi asioita tehdään, joka on kuitenkin tärkeää integraatiotyön kannalta. Tämä voisi tuoda lisää merkitystä tekijöille, ja toisaalta samalla voidaan huomata, että on olemassa vaihtoehtoinenkin tapa tehdä sama asia järkevämmin.

Sprint ja **Product Backlogeista** haastateltavat olivat yksimielisesti samaa mieltä, eli ne ovat hyödyllisiä ja tärkeitä olla olemassa. Integraatioasiantuntija B:n mukaan kehittäjää tämän kaltainen tehtävälistan visualisointi auttaa siinä mielessä, että hänelle saadaan näkyvyys tulevaan tekemiseen ja selkeä priorisoitu työjono, ja näin ollen rauhoitettua mieli käsillä olevaan tehtävään. Hänen mukaansa usein jonossa on useita tehtäviä yhtä aikaa, ja jos näitä ei ole selkeästi visualisoitu ja priorisoitu, saattaa osa niistä mennä päällekkäin tai unohtua, ja seurauksena jäädä tekemättä. Tiimipäällikkö A kommentoikin, että backlogista on myös suuri hyöty asiakkaalle, sillä se on käytännössä tärkein keino vaikuttaa siihen mitä tiimi tekee, ja priorisoinnin avulla varmistaa, että tiimin työ tuottaa asiakkaalle maksimaalista arvoa. Priorisointi ei integraatioasiantuntija B:n mukaan kuitenkaan saisi vaihdella liian usein, varsinkaan saman päivän sisällä, jotta saadaan tekijöille työrauha ja varmistettua tehtävien tehokas läpivienti. Projektipäällikkö A nosti esiin ajatuksen, että Product Backlogin sijaan voidaan myös käyttää roadmap-tyylistä suunnittelua, mutta se riippuu projektista ja sen tyylistä.

Definition of Done ja **Definition of Ready** koetaan tärkeiksi määritellä, jotta saadaan kehittäjälle muistilista esimerkiksi dokumentaatioista, asennuksista ja muista huomioon otettavista asioista integraation toteutuksessa kuten senior-integraatioasiantuntija A totesi. Hänen mukaansa on myös tärkeää, että kaikilla on sama käsitys siitä, mitä taustatietoja tarvitaan integraation toteutukseen ja missä muodossa, jotta kesken kehityksen ei tarvitse pysähtyä kyselemään näitä tietoja. Integraatioasiantuntija A kommentoi, että varsinkin uusille tiimin jäsenille hyödyllinen, kun tieto on dokumentoidussa muodossa eikä pelkästään hiljaisena tietona, näin työn laatu ja parhaat käytänteet pysyvät tiimissä korkealla tasolla, vaikka tekijät vaihtuvat. Tämä on oleellisen tärkeää, jotta asiakkaalle saadaan tuotettua arvoa ja tehtyä oikeita asioita ilman heittelyitä palvelutasossa. Integ-

raatioasiantuntija A:n mukaan on myös hyvä käydä ne läpi asiakkaan kanssa, jotta kaikilla on sama käsitys siitä, missä kohtaa tehtävä tai projekti on valmis, eikä jää turhia häntiä.

Erilaiset seurantatekniikat, kuten **Velocity ja Burndown Chart** eivät haastattelujen perusteella ole toimivia tapoja mitata integraatiokehitystä. Integraatioasiantuntija A kommentoi, että tehtävien valmistuminen ei useinkaan ole millään tavalla yhteydessä siihen, milloin suurin osa kehitystyöstä tehdään. Määrittelyt myös usein muuttuvat ja tarkentuvat kun tehtävää aletaan tekemään, ja kun tällaista Scope Creepiä tulee paljon sprintin sisällä, seuraa siitä jatkuvaa ylivuotoa seuraavalle sprintille. Tämä on senior-integraatioasiantuntija A:n mukaan iso syy miksi kaavioita ei haluta käyttää integraatioprojekteissa. Vaikka käytännössä integraation kehitys olisikin saatu ajoissa valmiiksi, ei sitä välttämättä voida laskea tehdyksi johtuen syistä, jotka eivät ole tiimin hallinnassa. Tästä syystä mainituista mittareista saadaan helposti väärä käsitys tekemisen tilasta, jonka seurauksena mittarit muuttuvat melko hyödyttömäksi.

5.8 Kanban integraatiotyössä

Kanban oli tuttu menetelmä kaikille haastateltaville, ja se olikin kaikilla jossain muodossa jo tällä hetkellä käytössä. Haastateltavilla tuntui olevan konsensus, että Kanban on paljon sopivampi integraatiotyöhön puhtaaseen Scrumiin verrattuna.

Visualisointi on iso osa kanbania, ja usein se on toteutettu kanban-tauluna jossa tehtävät liikkuvat eri vaiheiden läpi taululla. Kanbanin avulla nähdään mitä tehtäviä kullakin ajanhetkellä on työn alla, ja mitä on seuraavaksi tulossa. Senior-integraatioasiantuntija A kommentoi, että tehokkaasti toimiva kanban vaatii henkilön, joka pitää backlogin kasassa, ja backlogin kärjessä tulisi olla priorisoituna tehtävät, jotka hyödyttävät eniten asiakasta ja tuovat eniten arvoa. Hänen kokemuksensa mukaan taulun avulla nähdään myös, että työt etenevät tai jos ne eivät etene, sekä kenellä on paljon töitä ja kenellä ei. Tämä toimii lisäksi projektipäällikölle hyvänä työkaluna. Tiimipäällikkö A toi esiin sen, että taulun avulla asiakas näkee myös mitä työtä on valmistumassa ja mitä on valmistunut, joka tuo osaltaan läpinäkyvyyttä tiimin toimintaan. Tiimipäällikkö A:n mukaan on tärkeää, että taulussa on riittävästi eri statuksia tiketeille, ja että siitä tulee ilmi selkeästi kenellä, milloinkin on vastuu mistäkin tehtävästä. Projektipäällikkö A totesi, että tämän tyyppinen taulu on melkeinpä pakollinen nykyään. Integraatioasiantuntija A nosti ongelmana integraatiokontekstissa sen, että tiketit seisovat usein tietyissä sarakkeissa, kuten testaussarake, koska testaus vaatisi jonkun kolmannen osapuolen työtä ja tiimi ei voi siihen vaikuttaa. Hänen mukaansa tämä voi johtaa siihen, että hallitsemattomana taulu

täyttyy tiketeistä ja muuttuu vaikeaksi lukea. Kaikilla haastatelluilla oli käytössä kanbantaulu, johon on pääsy sekä tiimillä että asiakkaan edustajilla.

WIP:n rajoittaminen taas ei saanut suurta kannatusta haastateltavien keskuudessa. Kuten integraatioasiantuntija B kommentoi sen toteuttaminen vaatii ylimääräistä työtä ja jatkuvaa statuksien välillä siirtelyä. Hänen mukaansa integraatiotyö saattaa usein olla sitä, että tiettyä tehtävää tehdään ja välillä siihen joudutaan kysymään tarkennusta, ja työn edistäminen on sen aikaa seis, kunnes asiaan saadaan vastaus. Tällöin on usein järkevää odottelun sijasta ottaa toinen homma työn alle, mutta jos WIP on tarkasti rajoitettu, ei se ole mahdollista jollei edellistä tehtävää siirretä pois ”työn alla” olevista tehtävistä. Integraatioasiantuntija A kommentoi, että eräs mahdollisuus on myös, että johonkin aiemmin toteutettuun integraatioon pitää testauksen perusteella muuttaa jokin pikkuhomma, mutta tiukasti toteutettuna WIP ei salli tällaisen tekemistä ennen kuin aiemmin työn alla ollut tehtävä on valmis. Käytännössä integraatioiden järkevä toteuttaminen vaatii sitä, että tehtäviä edistetään rinnakkain, ja kun yhteen asiaan odotetaan vastausta esimerkiksi asiakkaalta niin sillä välin edistetään toista asiaa. Tätä voidaan toki taklata sillä, että nostetaan WIP tarpeeksi korkealle kuten integraatioasiantuntija B totesi, joka voi auttaa tilanteissa, joissa esiintyy lukkotilanteita missä mikään ei oikein edisty aivan liian suuren yhtäaikaisen tehtävämäärän takia. Periaatteena se siis hyvä, ja jos esimerkiksi havaitaan, että tietyllä henkilöllä on useita tehtäviä yhtä aikaa työn alla, sitä voidaan pitää oireena jostain ongelmasta, johon olisi hyvä puuttua. Käytännössä kuitenkaan WIP:iä ei todennäköisesti saada toimimaan integraatiotiimissä, tai ainakaan sen hyöty suhteessa vaivaan ei ole kovin hyvä.

Virran hallitsemiseksi haastateltavat ehdottavat keskustelua ihmisten kanssa. Tiimipäällikkö A totesi, että jos WIP olisi käytössä, niin sen avulla voitaisiin tietyllä tavalla hallita virtaa, mutta ihmisten kanssa keskustelu koetaan mukavammaksi tavaksi hallita tehtävävirtaa. Senior-integraatioasiantuntija A:n mukaan paras olisi luoda kaikille yhteinen ajattelutapa siitä, että tehtäviä valitaan ennemmin loppupäästä kanban-flow’ta, ja pyritään saamaan jo aiemmin aloitetut hommat valmiiksi, kuin aloittaa uutta. Pullonkaulojen havaitseminen on tärkeä osa sekä toimittajan että asiakkaan projektipäällikön työtä, johon tulee puuttua kulloinkin tilanteen vaatimalla tavalla. Niiden havaitsemiseksi visualisointi on tärkeä työkalu.

Prosessien käytäntöjen kirjaaminen täsmällisesti koetaan hyvinkin tärkeäksi, varsinkin siinä tilanteessa, kun uusi henkilö tulee tiimiin. Integraatioasiantuntija B:n kokemuksen muikaan tässä tapauksessa oppiminen ja vauhtiin pääseminen on huomattavasti nopeampaa, kuin jos asioita ei olisi dokumentoitu. Hänen mukaansa myös esimerkiksi tikettien statuksien siirtymät on hyvä määritellä täsmälliseksi ja mahdollisesti rajoittaa

sitä, mistä statuksesta voidaan milloinkin siirtyä mihin statukseen. Tämä helpottaa kehittäjän taakkaa muistaa täsmällistä prosessia, kun prosessi ei mahdollista väärinkäytöstä. Integraatioasiantuntija B:n mukaan on tärkeää, että tehdyistä asioista jää audit-jäljet ja että kaikki tiimissä tuntevat prosessin, ja että prosessi ei mahdollista sen väärin tekemistä. Alussa on toki oma vaivansa pistää sellainen pystyyn. Sillä kuitenkin luodaan selkeyttä ja se mahdollistaa tehokkaan yhteistoiminnan. Integraatioasiantuntija A oli sitä mieltä, että kuitenkin rajatapauksia tulee, joissa prosessit eivät aina toimi, joten niihin ei pidä suhtautua liian sitovasti. Kuitenkin kaikkiin toistuviin tehtäviin on hyvä olla dokumentoidut prosessit, jotta käytännössä kuka vaan voisi ne tehdä niiden avulla, jotta vältytään henkilöriippuvaisuuksilta ja ongelmilta.

Feedback-loopeista haastateltavat olivat sitä mieltä, että mitä lyhyempiä niin sen parempi. Senior-integraatioasiantuntija A kommentoi, että varsinkin silloin, jos henkilö ei ole niin tuttu asiakkaan tai teknologian kanssa, olisi hyvä saada mahdollisimman paljon feedbackia mahdollisimman usein. Integraatiotyövälineet ovat sellaisia, että feedback-loopin tulisi olla mahdollisimman lyhyt, koska jälkeenpäin toteutuksen muuttaminen on käytännössä sitä, että koko toteutus joudutaan tekemään uudelleen. Integraatioasiantuntija A nosti esimerkkejä toimivista feedback-loopeista parikoodauksen, jossa palautetta saadaan välittömästi, ja daily-käytännön, jossa palaute saadaan päivän sykleissä. Kaikilla haastatelluilla mielipide oli, että integraatiotyössä palaute tulisi saada nopeammin kuin kerran kahdessa viikossa, koska integraatiotekeminen itsessään on melko pienten osuuksien tekemistä ja yksittäiset komponentit valmistuvat melko nopeasti. Palaute on tärkeää myös prosessien ja käytänteiden kehittämisessä, koska ilman palautetta asiat eivät kehity parempaan suuntaan, kuten tiimipäällikkö A totesi.

5.9 XP integraatiotyössä

Yleinen mielipide oli se, että XP tiukasti noudatettuna ei sovellu integraatiotyöhön. Siinä on kuitenkin hyviä tekniikoita, jotka haastateltavien mielestä olisivat hyödyllisiä integraatiotyössä.

TDD ei haastateltavien kokemuksen mukaan sovellu integraatioihin kovinkaan hyvin. Senior-integraatioasiantuntija A totesi, että varsinkin vanhemmilla työkaluilla tällä tavoin toteutus olisi mahdotonta, koska toteutusta pitäisi pystyä testaamaan hyvin kattavasti ja refaktoroinnin tulisi olla helppoa. Hänen mukaansa monista kehitysvälineistä puuttuu esimerkiksi kunnon yksikkötestaustyökalut, joten kehityksen aikainen testaus muuttuu nopeasti hyvin hankalaksi. Lisäksi, kun integraatio on kerran tehty, sen muuttaminen on vaikeaa. Projektipäällikkö A:n mukaan käytännössä on havaittu, että integraatio valmis-

tuu nopeammin käytännössä samalla laadulla, kun ensiksi tehdään toteutus ja sen jälkeen testit. Integraatioasiantuntija B totsei, että uusimmilla työkaluilla TDD:n käyttö voisikin olla mahdollista, mutta TDD luultavasti tekee kehityksestä turhan hankalaa ja hidasta, eikä sen avulla saavuteta merkittävää hyötyä.

Pariohjelmointi herätti kaikissa haastatelluissa positiivisia kommentteja. Integraatioasiantuntija A ja tiimipäällikkö A kommentoivat molemmat, että pariohjelmointi on erityisen hyödyllistä silloin, kun tiimiin tulee uusi kokemattomampi kehittäjä, ja tarkoituksena on saada työntekijä tehokkaasti perehdytettyä projektiin ja tiimin tapoihin. Projektipäällikkö A:n mukaan pariohjelmointi varmistaa myös sen, että tieto liikkuu tehokkaasti tekijöiden kesken ja esimerkiksi käytänteet pysyvät samanlaisina kehittäjästä riippumatta. Integraatioasiantuntija A:n mukaan välttämättä kaikkein yksinkertaisimmissa integraatioissa pariohjelmoinnista ei ole hyötyä, mutta projekteissa, joissa asiat riippuvat toisistaan ja useat kehittäjät toteuttavat samanaikaisesti integraatioita samaan kokonaisuuteen liittyen, tekniikka koetaan erittäin hyödylliseksi. Integraatioasiantuntija B arvioi, että pariohjelmointi olisi tehokkainta silloin, kun pariohjelmointia ei orjallisesti tehdä jatkuvasti, vaan tarpeen mukaan esimerkiksi edellä mainituissa tapauksissa.

10-minute build on tärkeä asia, mutta toteutuu nykyisillä kehitysvälineillä lähes itsestään. Senior-integraatioasiantuntija A on sitä mieltä, että koska jokainen integraatio buildataan erikseen, ja käytännössä on mahdotonta saada yksittäisen integraation build kestämään yli kymmentä minuuttia, tämä ei ole erityisen olennainen ongelma integraatioista puhuttaessa.

Continuous Integration on integraatiokehityksen kannalta tärkeä keino varmistua siitä, että kun tehdään muutoksia olemassaoleviin integraatioihin, ei vahingossa rikota vanhaa toiminnallisuutta. Kaikkien haastateltujen mielipide oli se, että CI olisi ehdottoman hyödyllinen laadun varmistamiseksi, ja projektipäällikkö A totesi jopa, että ilman kattavaa CI:tä ei uskallettaisi tehdä tuotantoon muutoksia. Senior-ntegraatioasiantuntija A:n mukaan haasteita tulee siinä, että kaikki työkalut eivät välttämättä tue tällaista testaamista, joten paljon riippuu käytössä olevasta teknologiasta. Hän kommentoi myös sitä, että vaikka regressiotestit olisivatkin hyödyllisiä, integraatiokehityksessä testin saaminen aikaa vie usein paljon enemmän aikaa kuin käsin testaaminen, joten tulee miettiä tapauskohtaisesti, onko CI ajankäytön ja hyödyn kannalta järkevää.

Weekly Cycle ja **Quarterly Cycle** jakoivat mielipiteitä. Osa haastatelluista oli sitä mieltä, että hyvä idea ja tuo motivaatiota tiimille, kun tuodaan näkyvyys sekä pidemmän että lyhyen aikavälin suunnitteluun koko tiimille. Osa taas oli sitä mieltä, että kvartaalisuun-

nittelu on liian epämääräistä ja vaikea sitoutua niin pitkäksi aikaa. Senior-integraatioasiantuntija A kuitenkin pitää sykliajattelua positiivisena, hänen mukaansa tämän tyyppinen aikataulutus tuo laajempaa ymmärrystä siitä, miksi mitäkin tehdään, ja miksi esimerkiksi joitain tehtäviä nostetaan korkeammalle prioriteetille kuin toisia. Kun tekijöillä on ymmärrys isommasta kontekstista, kvartaalin syklistä, voidaan helpommin huomata miten viikottain tehtävä työ liittyy isompaan kokonaisuuteen, joka osaltaan jäsentelee ja tuo merkitystä työlle mitä tehdään, ja auttaa tekemään päätöksiä riippuen siitä mitä tulevaisuudessa on tulossa eteen. Integraatioasiantuntija A ja B kuitenkin näkevät priorisoinnin isona haasteena, tämän tyyppinen aikataulutus vaatii sitoutumista asiakkaalta ja pitkää näkyvyyttä tulevaan tekemiseen, joka integraatiotyössä saattaa olla haastavaa. Osaltaan viikkosyklissä tulee vastaan sama ongelma kuin Scrumin sprinteissä, eli harvoin tehtävää saadaan tehtyä kerralla alusta loppuun, vaan kun kehitys on valmista, se luovutetaan testattavaksi, jonka jälkeen se saattaa tulla milloin tahansa takaisin tiimille esimerkiksi testauksen tukemisen tai jatkokehityksen tarpeen takia. Tämän vuoksi aikataulut alkavat helposti luistamaan, ja tehtäviä joudutaan siirtämään viikosta toiseen.

Stories on käytännössä sama asia kuin Scrumin ajatus Storyistä, joten tähän kommentit olivat samat kuin Scrumin Storyjen kohdalla.

Slack sai kaikilta positiivisia kommentteja, ja ainakin teoriassa oli kaikkien haastateltujen mielestä hyvä idea. Senior-integraatioasiantuntija A:n mukaan monesti pienemmän prioriteetin perusparannukset jäävät jatkuvasti tärkeämpien asioiden jalkoihin, joten olisi tärkeää tietoisesti ottaa näitä mukaan, kun suunnitellaan lyhyen aikavälin tekemistä, jotta ne voidaan tehdä silloin kun aikaa on eikä vasta pakon edessä. Integraatioasiantuntija A kommentoi siten, että vaatii sen tyyppistä tekemistä, joka ei kärsi siitä, että konteksti vaihtuu, eli jos kyseessä on isompi jatkuva projekti niin välttämättä jokaiseen sykliin ei kannata ottaa Slack-taskeja mukaan. Hänen mukaansa kuitenkin on tärkeää, että tekijöillä on backlogilla aina jotain tehtävää, koska taukoja kehityksessä tulee pakostikin aika-ajoin.

Sit together koetaan tiimin ryhmädynamiikan ja tekemisen tasoa parantavana. Senior-integraatioasiantuntija A:n kokemusten mukaan tieto liikkuu paljon paremmin ja nopeammin kuin jos pitäisi esimerkiksi chattailla muualla istuvan kollegan kanssa. Hänen kokemuksiensa mukaan työt kyllä hoituvat etätiimienkin kesken, eli samassa tilassa istuminen ei ole välttämätöntä, mutta tiedostetaan että kommunikointi ja tiedonvaihto on kuitenkin helpompaa ja väärinkäsityksiltä vältytään, kun istutaan fyysisesti samassa tilassa. Saman ajatuksen jakaa myös integraatioasiantuntija B. Integraatioasiantuntija A:n mukaan etätiimeissä esimerkiksi kaikki pieni keskustelu ja ajatustenvaihto vieruskaverin kanssa jää helposti välistä, josta voisi kuitenkin viritä hyvinkin hedelmällistä keskustelua.

Projektipäällikkö A:n mielipide oli se, että projektin suunnitteluvaiheessa on kriittistä, että henkilöt istuvat samassa tilassa, sillä avoimia ja uusia asioita tulee niin paljon, että niiden kommunikointi etänä istuvien henkilöiden välillä on liian hidasta ja virheherkkää. Kun vihdoin päästään kehitysvaiheesta ylläpitomoodiin ja pienkehitykseen, projektipäällikkö A koki että samassa tilassa istumisen merkitys pienenee, mutta tiimin tulee kuitenkin olla vähintään samalla aikavyöhykkeellä ja jonkunlainen pikaviestin oltava käytössä.

Informative workspace vaatii haastateltujen kokemusten mukaan sen, että tiimi istuu samassa tilassa, vaikkakin se on mahdollista toteuttaa myös virtuaalisesti. Integraatioasiantuntija B koki varsinkin ylläpitotyössä radiaattorinäytöt tiimitilassa hyödylliseksi, sillä näistä voidaan esimerkiksi esittää erilaista statistiikkaa ja palveluiden tilaa. Myös se, että keskustelut pidetään pitkälti tiimitilassa, voi tuoda paljon uusia ratkaisuja ongelmiin, kun niistä keskustellaan avoimesti ja ihmisillä on mahdollisuus liittyä keskusteluun. Senior-integraatioasiantuntija A toi haasteena esiin sen, että avokonttorissa samassa tilassa saattaa olla useita tiimejä, jolloin keskustelu aiheuttaa hälinää ja häiriötä muualle, joka tulee ratkaista jollain keinoin. Tiimipäällikkö A toi esiin sen, että virtuaalitiloissa, kuten pikaviestimissä, on myös mahdollista saman tyyppinen avoin keskustelu. Integraatioasiantuntija B kuitenkin kokee, että tästä seuraa helposti tarpeetonta hälinää ja ”spämmiä” kanaville, joka on mahdollisesti enemmän häiritsevää kuin hyödyllistä. Pikaviestimissä tekstimuodossa käytyä keskustelua ei siis välttämättä koeta yhtä tehokkaaksi tähän tarkoitukseen.

Whole team ei integraatiotyössä toteudu, sillä usein koko integraation valmiiksi saattaminen vaatii kolmansien osapuolien työtä, joka varsinkin pienkehityksessä ja jatkuvassa ylläpidossa on ongelma konseptin kannalta, kuten tiimipäällikkö A toi esiin. Yhtenä vaihtoehtona tälle voisi olla adhoc-työtilat, kuten integraatioasiantuntija B esitti, joihin voidaan kerätä tarpeellinen porukka kasaan tilapäistä tarvetta varten, sen sijaan että tiimi koostuu jatkuvasti kaikista mahdollisista sidosryhmien jäsenistä. Riippuen sidosryhmästä, esimerkiksi ovatko he saman yrityksen sisäisiä vai ulkoisia toimijoita, ja sijaitsevatko he samalla konttorilla kuin päätiimi, riippuu missä muodossa adhoc-työtiloja kannattaa organisoida. Projektipäällikkö A kommentoi, että joissain tapauksissa on joka tapauksessa tehokkainta kerätä ihmiset fyysisesti kasaan, joissain tapauksissa taas riittää esimerkiksi yhteinen kanava Slackiin.

Energized work -periaatteesta kaikki olivat täysin yksimielisiä, ja tämä periaate koetaan ehdottoman tärkeäksi periaatteeksi, jota tulisi aina pyrkiä noudattamaan mahdollisimman tarkasti. Yrityskulttuuri on isossa osassa, esimerkiksi siihen pitäisi pyrkiä, että ylipitkiä työviikkoja ei ihailla ja niiden tekijöistä ei tehdä ”sankareita”, kuten senior-integraatioasiantuntija A asian ilmaisi. Tiimipäällikkö A myöntää, että joskus tulee kiireitä, mutta

kaikilla pitäisi olla ajattelutapa, että yksittäisen tekijän ei tarvitse venyä vaan asiat pitää pystyä hoitamaan muuten. Projektipäällikkö A:n mukaan haaste on siinä, että esimiesten tulisi pystyä seuraamaan yksittäisten tekijöiden työkuormaa melko reaaliajassa, joka on haastavaa, kun monilla henkilöillä on useampia projekteja päällekkäin, eikä kaikissa ole välttämättä sama projektipäällikkö, joka pystyisi seuraamaan tekijän kokonaiskuormitusta.

Incremental design -tekniikkaa tulisi haastateltujen mukaan soveltaa tilanteen vaatimalla tavalla. Senior-integraatioasiantuntija A:n mukaan varsinkin isommissa projekteissa on hyvä ymmärtää laajempi konteksti mihin toteutettava integraatio liittyy, ja tämän tyyppisissä projekteissa olisi hyvä olla mahdollisimman aikaisessa vaiheessa mukana integraatioiden määrittelyssä. Projektipäällikkö A kommentoi, että usein toimittaja joutuu tekemään työtä kokonaiskuvan rakentamiseksi, koska se ei välttämättä ole tilaajallakaan täysin selvillä. Toisaalta yksittäisissä, yksinkertaisissa integraatioissa voi hyvinkin riittää pelkkä tekninen määrittely ilman sen kummempaa käsitystä laajemmasta liiketoimintatarpeesta. Integraatioasiantuntija A:n mukaan integraatioiden suunnittelussa mennään usein hyvin nopeasti detailitasolle, koska yksittäiset integraatiot usein ovat melko yksinkertaisia, ja hommia on mahdollista tehdä niinkin, että ei ole ymmärrystä laajemmasta kokonaisuudesta. Tiimipäällikkö A on kuitenkin sitä mieltä, että kaiken kaikkiaan laajemmasta ymmärryksestä on yleisesti hyötyä, sillä yksittäinen integraatio on usein niin sanotusti puu metsässä, ja on tärkeää nähdä myös metsä puilta. Liian tarkka suunnittelu ylätasolla on kuitenkin turhaa, koska lopulta yksityiskohdat ratkaisevat integraation toteutuksessa.

5.10 Lean Development integraatiotyössä

Eliminate waste koetaan tärkeäksi periaatteeksi ja haastateltavat pyrkivätkin tämän periaatteen mukaan tekemään töitä. Integraatioasiantuntija A kommentoi, että tämän periaatteen pitäisi olla integraatiotekemisessä ohjenuorana, sillä aika on tärkein resurssi, ja sen käytön optimointi tulisi olla lähtökohta kaikelle tekemiselle. Hukkaa integraatioprojekteissa kuitenkin väistämättä tulee, kuten senior-integraatioasiantuntija A kommentoi, esimerkiksi kehityksen jälkeen joudutaan lähes aina odottamaan, että integraatiota testataan, ja kun integraatio palautuu takaisin testauksesta kehittäjälle, pitää uudestaan keräillä takaisin mieleen, että mistä olikaan kyse. Tiimipäällikkö A pitää tärkeänä, että omissa arjen valinnoissa pitää hukan minimoinnin ja asiakkaan lisäarvon kirkkaana mielessä sekä pyritään välttämään turhaa työtä. Hänen mukaansa asiat tulisi pyrkiä teke-

mään riittävällä tasolla, mutta ei pyrkiä perfektionismiin. Projektipäällikkö A pitää varsinkin priorisoinnissa tätä periaate hyvin tärkeänä, sillä siinä tulisi pyrkiä valitsemaan isoa kuvaa ajatellen tärkeimmät tehtävät ja välttää turhaa työtä ja hukkaa.

Amplify learning sai haastateltuja samoja kommentteja kuin Kanbanin feedback-looppien implementointi, joka onkin pitkälti samaa asiaa. Integraatioasiantuntija A nosti esimerkiksi parikoodaamisen ja avoimen keskustelun ilmapiirin, jossa on mahdollista pienellä kynnyksellä sparrata toteutustapoja kaverin kanssa, tärkeinä keinoina, joilla saadaan oppimista tehostettua. Projektipäällikkö A nosti eräänä keinona parhaiden käytänteiden listaaminen, joita voidaan referoida tarpeen vaatiessa. Tämä tarkoittaa sitä, että käytännössä kuka tahansa voisi tehdä integraatioita samojen käytänteiden mukaan kuin aiemmatkin on toteutettu, ilman että tarvitsee jatkuvasti varmistaa muilta, onko toteutustapa oikea. Avoin ilmapiiri ja keskusteluyhteys tiimikavereiden kanssa, joka luo jatkuvan oppimisen ja positiivisen palautekulttuurin ilmapiirin, on integraatioasiantuntija A:n mukaan paras keino rikkoa sisäisiä siiloja ja kannustaa jatkuvaan oppimiseen.

Tee päätökset mahdollisimman myöhään on kaikkien mielestä järkevä periaate, mutta kaikkia asioita ei kuitenkaan ole hyvä jättää viime tinkaan. Projektipäällikkö A:n mukaan esimerkiksi päätökset siitä, tehdäänkö koko integraatiota tai tietyt sitoutumispäätökset joudutaan tekemään aikaisessa vaiheessa, mutta tarkempi toteutustapa ja yksityiskohdat on hyvä jättää myöhäisempään vaiheeseen. Kontekstinvaihdot ovat tekijöille pahasta, joten on hyödyllisempää, että toteutukseen liittyviä päätöksiä tehdään vasta siinä vaiheessa, kun itse toteutustakin on mahdollista tehdä, jottei tule tilannetta, jossa integraatiota suunnitellaan etukäteen mutta sitä ei ole mahdollista vielä siinä vaiheessa tehdä, ja myöhemmin joudutaan palaamaan suunnitelmiin ja palauttamaan konteksti mieleen. Asiat kuitenkin muuttuvat usein alkuperäisiin määrittelyihin nähden, joten turhan aikaisessa vaiheessa tehdyt päätökset voivat olla jopa turhaa työtä, ja pahimmassa tapauksessa voivat johtaa isoihinkin korjauksiin, jos toteutusta on jo ehditty tekemään näiden oletusten perusteella. Kuten integraatioasiantuntija A totesi, asioiden etukäteen tekeminen voi olla helposti turhaa työtä.

Deliver as fast as possible on periaatteena tärkeä, joka integraatiotyössä kuitenkaan ei niinkään kosketa integraation kehittäjää kuin sen tilaajaa eli asiakasta kuten integraatioasiantuntija A totesi. Integraation valmistuminen on käytännössä aina sidoksissa muihin sidosryhmiin ja asiakastestaukseen, joten integraation kehitystiimi ei tähän useinkaan voi vaikuttaa. Periaate on toki tärkeä pitää mielessä ja tiimipäällikkö A:n mukaan on tärkeä muistuttaa asiakastakin siitä, jos esimerkiksi on kertynyt paljon avoimia kehityksiä, joita olisi hyvä siirtää tuotantoon ennen kuin uutta aloitetaan, jotta vähennetään

kompleksisuutta ja saadaan selkeyttä tekemiseen. Isommissa projekteissa on hyvä ymmärtää kokonaiskuva ja miten pienen jutun viivästymisellä voi olla kallis kustannus lopussa kuten projektipäällikkö A totesi.

Empower the team on käytännössä sama periaate kuin XP:n informative workspace, whole team ja energized work yhdisteltynä, joten haastateltavat olivat vastauksissaan samoilla linjoilla. Kaikkien mielestä tämä on ehdottoman tärkeä periaate. Projektipäälliikkö A tiivisti asian olevan silloin hyvin, kun tiimillä on valta valita mitä tekee ja mikä itseä kiinnostaa. Tällöin tekemistä ei tarvitse työntää, vaan tiimin jäsenet osaavat itsenäisesti ottaa tekemistä itselleen, kun edellinen loppuu. Tähän liittyy olennaisesti se, että tiimillä on tietoa siitä, mikä tekeminen milloinkin on tärkeää, jotta tiimi osaa itseohjautua tekemään eniten arvoa tuottavaa työtä, eikä aikaa tuhlata matalan prioriteetin tehtäviin, jos tärkeämpää tekemistä on tarjolla. Integraatioasiantuntija A toi esiin, että tekijät kokevat tarvitsevansa vastuuta, ja tämä on erinomainen tapa antaa vastuuta ja itsenäisyyttä. Tiimipäällikkö A:n mukaan on työnantajan ja esimiesten vastuulla pitää huoli että tilat ovat tarkoituksenmukaiset ja ihmisillä mahdollisuus tehdä töitä tehokkaasti.

Näe kokonaisuus on projektipäällikkö A:n mukaan integraatioissa luonnollista, ja tässä vastaukset olivat samat kuin XP:n incremental design periaatteessa. Integraatioasiantuntija B kokee, että kokonaisuuden ymmärtäminen ja aika-ajoin siitä muistuttaminen on tärkeää, koska muuten kokonaiskuva saattaa hukkua tekemisen jalkoihin.

6. POHDINTA

Integraatiokehityksen haasteet eivät ole yksinään ratkaistavissa kehitystiimin työtapoja parantamalla. Kuten Kähkönen (2017) kuvaa, suuri osa ongelmista liittyy sosio-organisationaalisiin ja liiketoimintanäkökulmiin, ja vain pieni osa haasteista liittyy itse kehitystyöhön. Lisäksi haastattelujen perusteella on selvää, että läheskään kaikki tutkittujen ketterien menetelmien tekniikat eivät sellaisenaan sovellu integraatiokehitykseen, ja niistäkin, jotka sopivat, kaikkia ei voida soveltaa joka tilanteessa. Tämän vuoksi tässä tutkimuksessa käsitellyt ketterien menetelmien tekniikat on jaettu kolmeen ryhmään, eli yleisesti toimivat, tapauskohtaisesti toimivat ja integraatiokehitykseen sopimattomat tekniikat.

6.1 Yleisesti toimivat ketterien menetelmien tekniikat

Haastattelujen ja teorian perusteella voidaan koostaa 13 kohdan lista tekniikoista, jotka haastattelujen perusteella toimivat yleisesti integraatiotiimeissä ja eri tilanteissa ainakin kohdeorganisaation kontekstissa. Suositellut tekniikat on valittu analysoimalla haastattelujen tuloksia vertaamalla haastateltavien vastausten yhdenmukaisuutta ja perusteluja. Lisäksi analysoinnissa on arvioitu perustelujen pitävyyttä suhteessa kirjallisuudesta löytyviin integraatiotuotannon haasteisiin ja luvussa 4 käsiteltyihin tavoitteisiin nähden. Jos tavoitteena on kasata tehokas ja toimiva integraatiotiimi, ainakin nämä tekniikat ja periaatteet tulisi olla tiimillä käytössä.

Empower the team **-** Tiimin valtuuttaminen organisoimaan ja hoitamaan oman työnsä parhaaksi näkemällään tavalla on yksi tärkeimmistä ketterien menetelmien periaatteista, kun halutaan kasata tehokas ja toimiva integraatiotiimi. Ilman vastuuntuntoista, itseohjautuvaa, avointa ja oppimishaluista ilmapiiriä on lähes mahdoton yrittää ottaa suurinta osaa muista tekniikoista käyttöön. Tällainen tiimi on paitsi sitoutuneempi työhönsä, myös tehokkaampi. Zenger Folkmanin tutkimuksen mukaan, kun valtuutusaste oli matala, vain 4% työntekijöistä oli valmis näkemään ylimääräistä vaivaa työnsä eteen. Kun taas aste oli korkea, jopa 67% työntekijöistä oli valmiita venymään tarpeen vaatiessa (Folkman 2017). Tyytyväinen tiimi, jossa on mukava tehdä töitä ja joka on silloin tällöin valmis ponnistelemaan vähän enemmän, on kaiken lisäksi suorituskykyisempi ja tuottavampi (Schwaber & Sutherland 2017). On helppoa nähdä, että tällainen tilanne on kaikkien kannalta etu, ja johon tulisi pyrkiä kaikissa tapauksissa.

Energized work - Jotta integraatiotiimi pystyy työskentelemään laadukkaasti ja tehokkaasti ja noudattamaan muita ketterien menetelmien tekniikoita, tulee työntekijöiden hyvinvoinnista ja jaksamisesta pitää hyvää huolta. Integraatiotuotannossa arvoa tuottavan toiminnan lähtökohtana on integraatiokehittäjät, ja kuten esimerkiksi teollisuustuotannossa valmistuksessa käytettäviä koneita ylläpidetään ja huolletaan, tulee integraatiotiimissä yhtä lailla ylläpitää jaksamista ja pitää huolta työntekijöistä. Jos tästä vaiheesta luistetaan, ovat seuraukset vakavia ja kauaskantoisia, kuten työntekijöiden uupuminen ja vaihtuvuuden lisääntyminen (Stellman & Greene 2014). Haastattelujen perusteella kohdeorganisaatiossa ei tämän kanssa ole ongelmia, mutta periaate on silti nostettu listan kärkeen, koska se on kuitenkin yksi tärkeimmistä asioista listalla ja joka tulee pitää aina mielessä kaikessa tekemisessä.

Työn visualisointi - Visualisointi on nostettu listan kolmanneksi, mutta todellisuudessa sen tärkeyttä on vaikea erottaa muutamasta seuraavasta listan kohdasta, ja tärkeysjärjestys varmasti riippuu tilanteesta ja tiimistä. On kuitenkin perusteltua väittää, että visualisointi on jopa pakollinen osa integraatiotiimin toimintaa, varsinkin kun tiimin jäsenten ja työn määrä kasvaa. Ilman tehtävien visualisointia työn organisointi ja järjestely muuttuu tietyn rajan jälkeen jopa mahdottomaksi, ja visualisointi onkin edellytyksenä monen muun ketterien menetelmien tekniikan käytölle. Esimerkiksi tiimin itseohjautuvuus kärsisi suuresti tilanteessa, jossa työtä ei ole visualisoitu johonkin muotoon, vaan työtehtävät ovat ainoastaan yksittäisten työntekijöiden päässä, muistiinpanoissa tai sähköpostiketjuissa. Kuten Stellman & Greene (2014) muistuttavat, on esimerkiksi hukan tunnistaminen ja prosessin ongelmiin puuttuminen huomattavasti helpompaa, kun työ on visualisoitu ja kaikille näkyvissä. Myös Kähkönen (2017) nostaa yhdeksi onnistuneen integraatioprojektin avaintekijäksi projektin hallinnan monella eri tasolla ja eri sidosryhmien tekemisen hallitsemisen, jossa työn visualisointi on ensisijaisen tärkeää.

Product Owner - Kuten kirjallisuudessa todetaan, tärkein ja usein vaikein osa Scrum-tiimiä on löytää hyvä Product Owner (Stellman & Greene 2014). Kähkönen (2017) nostaa myös esiin sen, että onnistuneen integraatioprojektin vaatimuksena on täyspäiväiset ja sitoutuneet henkilöt, joille integraatiot eivät ole vain jokin sivutyö. Haastattelujen perusteella tämä pätee integraatiotuotannossa, vaikka Scrumia ei noudatettaisikaan. PO:n tulee olla sellainen henkilö, joka on sitoutunut, jolla on riittävästi auktoriteettia tehdä päätöksiä liiketoiminnankin puolesta, joka pystyy keskustelemaan eri sidosryhmien kanssa ja hankkimaan oleellista tietoa integraatiotiimille ja joka pystyy sanomaan, mikä milloinkin on tärkeintä. Ilman hyvää PO:ta ketteristä menetelmistä ei voida saada maksimaalista hyötyä irti. Periaatteessa PO voisi olla useampikin henkilö, mutta haastattelujen ja teorian perusteella rooli on tehokkaimmillaan silloin, kun sitä hoitaa yksi nimetty henkilö.

Feedback-loopit & Amplify learning - Erittäin tärkeitä periaatteita, jotka varmistavat sen, että laatu pysyy korkeana ja ongelmiin puututaan nopeasti. Vaatii kohtalaisen itseohjautuvan tiimin, joka pystyy itsenäisesti toimimaan näiden periaatteiden mukaisesti. Ilman näitä tekemisen tavat jumittuvat hyvin helposti toistamaan samaa kaavaa, eikä innovaatioita tai tehokkaampia työtapoja synny, eli nämä tekniikat varmistavat korkean laadun ja osaltaan kilpailukyvyn ylläpitämisen. Haastateltavien mukaan tärkeä periaate, jota tulee suosia tekniikoiden valinnassa, ja kuten Stellman & Greene (2014) tuovat esiin, ovat periaatteet tärkeitä, jos halutaan ylipäätään kerätä tietoa tiimin toiminnasta ja kehittää sitä.

Daily - Daily on tavallaan yksi Feedback-loop, mutta tämä on haluttu nostaa listalle erillisenä siksi, että kaikissa haastatteluissa tekniikka nousi esiin erityisen tärkeänä ja se koetaan lähes pakollisena osana integraatiotiimin toimintaa. Kähkönen (2017) tuo esiin erityisesti eri sidosryhmien välillä tapahtuvan kommunikoinnin tärkeyden, ja Daily on erinomainen väline uusimman tiedon kommunikoimiseksi kehitystiimille lyhyellä syklillä.

Review & Planning - Vaikka kyseiset Scrumin tekniikat on alun perin tarkoitettu sprintmuotoisen tekemisen ympärille, voidaan ne saada istumaan myös integraatiotuotantoon. Periaatteessa kaikki kolme tekniikkaa voidaan yhdistää yhdeksi, jossa koko tiimi käy sisällön läpi yhdessä PO:n (asiakkaan) kanssa. Tilaisuudessa on hyvä käydä backlogia läpi, kuten scrumissakin, ja valita sieltä tehtäviä, joita voidaan nostaa tiimin työlistalle. Ainoana erona Scrumiin on se, että niiden valmistumista ei sidota Sprinttiin, vaan voidaan käyttää jotain muuta frameworkia (esimerkiksi Weekly Cycle), aikatauluttaa, joka tehtävä erikseen tai jättää tässä tilanteessa aikatauluun sitoutuminen kokonaan pois. Samassa yhteydessä on hyvä järjestää (sprint) review, jossa käydään läpi viime kerran jälkeen valmistuneet tehtävät ja keskeneräisten tehtävien status. Kuten Kähkönen (2017) tuo esiin integraatioprojektin onnistumisen edellytyksiä esitellessään, vaaditaan monen tasoista hallintaa, jolloin nämä tekniikat ovat tärkeitä tavoitteiden viestimisessä kehitystiimille.

Story Refinement - Story refinement on ehdottoman hyödyllinen tekniikka integraatiotuotannossa. Sen tarkoituksena on määritellä tehtävät sellaiselle tasolle, jotta ne voidaan viedä läpi keskeytyksettä, kun tehtävää aletaan toteuttamaan. Jos Refinement halutaan pitää määrämuotoisena palaverina, tämä voidaan pitää samassa yhteydessä kuin Sprint planning & review. Jos palaverit uhkaavat venyä liikaa, voidaan se järjestää myös erillisenä tilaisuutena. Pakko ei kuitenkaan ole järjestää erillistä tilaisuutta Refinementia varten.

Eliminate waste - Hieman abstraktimpi käsite kuin monet muista listan kohdista, joten suhteellisen tärkeyden määrittäminen on hankalaa. Hukan (Waste) minimointi on kuitenkin äärimmäisen oleellinen osa integraatiotyötä, ja varsinkin toimittajan asemassa on elintärkeää sekä kilpailukyvyn että asiakastyytyväisyyden näkökulmasta optimoida käytetty aika asiakasarvon suhteen. Hukan minimointi pitäisi olla kaiken tekemisen lähtökohta ja yksi tiimin perusarvoista, kuten Stellman & Greene (2014) esittävät.

(Product) backlog - Backlog, olkoon se sitten nimetty product backlogiksi tai joksikin muuksi, on oleellinen osa integraatiotyötä. Käytännössä ei ole väliä onko kyseessä integraatioprojekti, jatkuvat palvelut tai näiden sekoitus, ilman jonkinlaista backlogia integraatiotiimin on vaikea toimia suunnitelmallisesti ja tehokkaasti. Kähkönen (2017) tuo esiin faktan, että onnistuneessa integraatioprojektissa vaaditaan tehokasta kommunikointia eri osapuolten välillä, ja kehitystiimin suuntaan toiveiden kommunikointi onnistuu oleellisen helposti backlogin kautta.

Virran hallitseminen - Tehtävävirran hallitseminen on oleellinen osa integraatiotiimin tehokasta työskentelyä, ja sen merkitys vain korostuu sitä mukaa kun tekemisen ja tiiminjäsenten määrä kasvaa. Koska virta on arvon liikkumista prosessin läpi, voidaan sen hallinnalla parantaa prosessin tehokkuutta, vaikuttavuutta ja ennustettavuutta (Vacanti & Yeret 2019).

Tee päätökset mahdollisimman myöhään - Jotta integraatioita voidaan tuottaa tehokkaasti, ei alkuvaiheessa kehitystä tule sitoutua sellaisiin päätöksiin, jotka voivat vielä muuttua. Tämä on totta myös ohjelmistokehityksessä, mutta erityisen tärkeää integraatioista puhuttaessa, jossa toteutustapa ja yksityiskohdat voivat muuttua niin pitkään, kunnes kaikki osapuolet ovat saaneet omat järjestelmänsä valmiiksi, ja joskus määrittelyt voivat muuttua vielä senkin jälkeen. Tämän vuoksi on tärkeää, että ei tehdä liikaa työtä etukäteen, koska on suuri riski, että niihin tulee muutoksia vielä hankkeen edetessä, kuten monet haastateltavat kokivat usein tapahtuvan. Lisäksi, kun päätökset tehdään myöhäisessä vaiheessa, ymmärretään paremmin eri päätösten hinta ja niiden vaikutus lopulliseen tuotteeseen (Beck 2004).

Näe kokonaisuus & Incremental design - Suunnitteluperiaatteita, jotka pätevät integraatioiden tekemisessä. Kun tehdään integraatioita kahden tai useamman eri järjestelmän välillä, kannattaa integraation suunnittelu aloittaa siitä, mikä on se kokonaisuus mihin integraatio liittyy ja mitä sillä yritetään saavuttaa. Kun ylätason prosessi on selvä, on paljon todennäköisempää, että integraation toteutus suunnitellaan vastaamaan todelliseen tarpeeseen. Täten lopputuloksesta saadaan asiakkaalle arvokkaampi, ja todennä-

köisesti laatu pysyy parempana, kun ymmärretään tehtävien asioiden sijainti kokonaiskuvassa (Stellman & Greene 2014, Beck 2004). Integraatioiden luonteen takia on myös tärkeää ylläpitää arkkitehtuurikuvauksia integroitavista järjestelmistä, jotta voidaan tunnistaa integraatiotarpeita ja määrittelyitä (Kähkönen 2017)

10-minute build - Lähestulkoon itsestäänselvyys nykyisillä integraatiotyökaluilla, mutta silti tärkeä periaate, josta tulee pitää kiinni, kuten haastatteluissa tuli ilmi. Ilman tätä tekniikkaa kehityksen laatu voisi kärsiä.

6.2 Tapauskohtaisesti toimivat ketterien menetelmien tekniikat

Osa tekniikoista on sellaisia, jotka toimivat tietyissä tilanteissa, tiimeissä tai projekteissa, mutta joita ei voida yleistää koskemaan kaikkia integraatiotiimejä ja tilanteita. Näin ollen seuraavien tekniikoiden käyttöä tulee arvioida tapauskohtaisesti, eli saavutetaanko sen käytöllä haettua hyötyä kulloisessakin tilanteessa.

Sit together - Sekä teorian että haastattelujen perusteella erittäin tärkeä periaate, joka tehostaa suuresti tiimin toimintaa ja muiden ketterien menetelmien tekniikoiden käyttöä. Valitettavasti aina yhdessä istuminen ei käytännön syistä ole mahdollista, josta syystä Sit together on listattu tapauskohtaisesti käytettävissä tekniikoissa. On kuitenkin perusteltua väittää, että tiimien koostamisessa tulisi aina pyrkiä siihen, että ihmiset istuisivat fyysisesti samassa paikassa. Tällä saadaan helposti todella suuria hyötyjä, ja se mahdollistaa myös muiden tekniikoiden tehokkaan käytön. Kuten Kähkönen (2017) toteaa, eri osapuolten kommunikointi on integraatioprojektissa hyvin tärkeää, ja kuten haastattelujen perusteella voidaan todeta, auttaa yhdessä istuminen siihen huomattavasti.

Pariohjelmointi - Pariohjelmoinnilla saadaan paitsi laatua paremmaksi (Stellman & Greene 2014), nopeuttaa se uuden työntekijän perehdyttämistä tiimiin ja näin ollen tehostaa perehdyttämisprosessia. Tämä on sekä toimittajan että asiakkaan kannalta etu, sillä pariohjelmointia hyödyntämällä uusi työntekijä pääsee nopeammin mukaan tekemään arvoa tuottavaa asiakastyötä, kun taas asiakkaan näkökulmasta laatu pysyy tasaisena henkilöstövaihdoksista huolimatta. Pariohjelmointia kannattaa hyödyntää myös projekteissa, joissa useampi integraatiokehittäjä tekee integraatioita samaan kokonaisuuteen liittyen ja on tärkeää, että kokonaisuus pysyy koherenttina. Ilman pariohjelmointia joudutaan todennäköisesti jälkeenpäin refaktoroimaan toteutuksia, josta todennäköisesti aiheutuu suurempi työkuorma kuin siitä, että toteutusta olisi tehty pariohjelmointina alusta lähtien. Tekniikka on valittu kuitenkin tapauskohtaisesti käytettäväksi, sillä kaikissa tilanteissa ei ole hyötyä pariohjelmoinnista ja silloin siitä voi aiheutua turhia kustannuksia.

Informative workspace – Haastattelujen perusteella ehdottoman hyödyllinen, mutta yleensä vaatii sen, että tiimi istuu fyysisesti samassa tilassa. Voi olla mahdollista rakentaa myös virtuaalinen tila, jossa simuloidaan vastaavia vaikutuksia, mutta ainakaan haastateltavat eivät uskoneet, että siitä saataisiin vastaavia hyötyjä kuin fyysisestä tilasta. Tämän vuoksi tekniikan käyttöä pitää arvioida tapauskohtaisesti, todennäköisesti käytetään silloin kun ”Sit together” tekniikan kanssa. Informatiivinen työtila on mahdollista toteuttaa kuitenkin myös ilman ”Sit together” tekniikan, kunhan riittävän suuri osa tiimistä istuu samassa tilassa, mutta luonnollisesti hyöty kasvaa sen mukaan mitä suurempi osa tiimistä on samassa lokaatiossa. Tiedon leviäminen tiimin jäsenten kesken on ehkä suurin etu joka tekniikalla saavutetaan (Beck 2004).

Prosessien kirjaaminen ja dokumentointi - Voidaan pärjätä ilmankin, mutta kuten haastatteluissa tuli ilmi, varsinkin henkilöstömuutoksien tapauksessa auttaa todella paljon, että prosessit ja työtavat on kirjattu ja dokumentoitu jonnekin ylös. Prosessien dokumentointi auttaa myös huomaamaan epäkohdat ja paikat, joita muuttamalla voidaan tehostaa toimintaa, joka on yksi Kanbanin arvoista (Stellman & Greene 2014).

Definition of Done & Definition of Ready - Tärkeää että tiimi on yhdessä määritellyt DoD ja DoR, sillä ilman eksplisiittistä määrittelyä eri henkilöillä voi olla eri oletukset siitä mitä vaaditaan. Varsinkin silloin, kun tiimissä on kokeneempia ja uudempia kehittäjiä, on hyvä käydä läpi oletukset ja sopia yhdessä mikä on taso mitä määrittelyissä vaaditaan ja mitä tarkoittaa, että työ on valmis. Haastattelujen perusteella on kuitenkin todettava, että ilmankin näiden määrittelyä varsinkin pienehkö tiimi pystyy toimimaan tehokkaasti, kunhan kehittäjillä on ennestään kokemusta integraatioiden toteuttamisesta. Tämän vuoksi näiden käsitteiden määrittely on tapauskohtaisesti hyödyllistä, mutta missään tapauksessa niiden määrittelystä ei ole haittaa. Koska Kähkönen (2017) listaa integraatioiden määrittelyn yhdeksi isoksi osaksi integraatioprojektien toteutusta, voidaan DoD:n ja DoR:n etukäteen määrittelemällä saavuttaa suuria hyötyjä suunnitteluvaiheessa jakamalla määrittelyjen vaatimukset eri osapuolille.

User Stories - User storyjen käyttö integraatiotuotannossa jakoi haastateltujen mielipiteitä. Koska integraatiot yleensä ovat järjestelmien välistä dataliikennettä, voi tietyissä tapauksissa olla vaikea löytää ”käyttäjää” jonka näkökulmasta kirjoittaa käyttäjätarina. Tämän vuoksi käyttäjätarinoiden määritelmää tulee hieman laajentaa, jotta niitä voidaan hyödyntää integraatioiden toteutuksessa. Vaikka integraation tapauksessa ei voida puhua käyttäjästä, tulisi user storyn kuvata integraation bisneslogiikkaa ja mikä sen reaalimaailman vaikutus on. User storyn määritelmä tulisikin määritellä yhdessä tiimin ja asiakkaan kanssa, ja sen määritelmä onkin läheisesti yhteydessä Definition of Readyn mää-

rittelyn kanssa. Jos integraatio jää kehitysvaiheen jälkeen ylläpitoon samalla kehittäjätiimille, on haastattelujen perusteella ehdottoman hyödyllistä, että on olemassa tarkempi kuvaus integraation sisällöstä ja tarkoituksesta. Kuitenkaan aina ei ole näin, jonka vuoksi tulee harkita tapauskohtaisesti, saadaanko user storyjen käytöstä haluttua hyötyä. Välttämättä kaikissa tapauksissa ei ole tarpeen ymmärtää laajempaa kontekstia, vaan voi riittää, että kehittäjälle annetaan ainoastaan riittävät tekniset tiedot, jonka perusteella integraatio voidaan toteuttaa. User storyja kannattaa hyödyntää silloin, kun on olemassa selkeä lopputuloksen käyttäjä, jolloin kehitystiimi voi keskittyä parhaan mahdollisen teknisen ratkaisun tuottamiseen suhteessa asiakkaan haluamaan arvoon (Stellman & Greene 2014).

Retro - Retrospektiivi on hyödyllistä, kun halutaan saada selville, onko tiimillä parannettavaa esimerkiksi toimintatavoissa tai kommunikaatiossa. Retron avulla saadaan käytyä läpi esimerkiksi syvemmät ongelmat prosesseissa, mutta myös muistuteltua mieleen mitä on tehty hyvin ja kuinka jatkossa tulisi toimia. Niitä olisi hyvä järjestää aika-ajoin, mutta sopiva aikaväli riippuu tiimistä ja tilanteesta. Haastattelujen perusteella kaksi viikkoa on todennäköisesti liian lyhyt väli ja vuosi on ehdottomasti liian pitkä aikaväli. Retro ei ole oikea työkalu päivän polttaviin ongelmiin, kriiseihin tai muihin tilanteisiin, joissa tarvitaan nopeaa reagointia – näitä tilanteita varten Dailyn tulisi olla käytössä jokaisessa tiimissä. Tämän vuoksi Retroa ei voida pitää pakollisena, mutta se on silti ehdottoman hyödyllinen tekniikka oikein käytettynä. Stellman & Greene (2014) muistuttavat, että Agilen arvojen perusteella on tärkeää löytää prosessista ongelmia ja ratkaista niitä, jonka vuoksi retroa voi suositella käytettäväksi.

Scrumin roolijako - Kuten haastattelujen perusteella voidaan todeta, integraatiotiimi voidaan hyvin roolittaa Scrumin roolien mukaiseen jaotteluun – Product Owner, Scrum Master ja kehitystiimi. Jaottelu on aivan toimiva, vaikka Scrumia ei oikeaoppisesti toteutettaisikaan, ja se toimii hyvin myös esimerkiksi Kanban-prosessia noudattaessa. Jaottelu ei kuitenkaan ole kaikissa tilanteissa optimaalinen, sillä kuten haastatteluissa tuli ilmi, Scrum Master ei ole aina välttämätön varsinkaan, jos ei noudateta suurinta osaa Scrumin tekniikoista. Tällaisessa tilanteessa roolijakoa tulee muokata tiimin ja projektin vaatimaan muotoon, ja esimerkiksi erillisen arkkitehtiroolin käyttöä voidaan harkita. Tärkeää on, että projektissa on omistautunutta asiantuntijuutta ja täyspäiväisiä henkilöitä, kuten Kähkönen (2017) listaa integraatioprojektin onnistumisen edellytyksissä.

Deliver as fast as possible - Tärkeä periaate, joka integraatiotyössä koskettaa kaikkia sidosryhmiä. Tiimi ei yksin pysty useinkaan vaikuttamaan tähän, joten kommunikaatio ja koordinointi eri sidosryhmien välillä on ehdotonta tavoitteen saavuttamiseksi kuten Kähkönen (2017) toteaa.

Continuous integration - Hyvä toimintamalli, jos vain mahdollista. Kaikki työkalut eivät tue CI:tä, joten tulee käyttää mahdollisuuksien mukaan. Toisaalta myöskään ei pidä pakottaa tiimiä käyttämään tiettyjä kehitystapoja, jos tiimillä on selkeä syy, miksei esimerkiksi CI:tä haluttaisi käyttää. Näin ollen tätä tekniikkaa tulee hyödyntää tarpeen mukaan. Joissain projekteissa erittäin tärkeää, kuten haastatteluissa tuli ilmi, ja onnistuessaan hyödyllinen ja iso vaikutus laatuun ja kehitysnopeuteen, vaikka vaatiikin ehkä enemmän ylläpitoa kuin kehitys ilman CI-putkia.

Weekly & Quarterly cycle - Vaihtoehtoinen tapa aikatauluttaa ja priorisoida työtä. Tuo näkyvyyttä tulevaisuuteen ja auttaa päätöksenteossa, kuten resursoinnissa. Riippuu kuitenkin asiasta mitä tehdään, että onko paras tapa hoitaa asioita. Tekniikka vaatii asiakkaalta ison panoksen, joten mahdoton käyttää, jos asiakasta ei saada sitoutumaan tekniikkaan ja priorisoimaan työtä. Kuitenkaan mitään negatiivista tähän tekniikkaan liittyen ei tullut haastatteluissa ilmi, joten jos on mahdollista, niin ehdottoman hyödyllistä vähintäänkin kokeilla toimisiko tämä tyyli integraatiotiimin työn ohjaamiseen. Tekniikka tukee Kähkösen (2017) ajatusta siitä, että integraatioita tulisi käsitellä systemaattisena ja hyvin suunniteltuina toimintoina, joka käsittää useita järjestelmiä ja sidosryhmiä – näiden välisesten toimintojen suunnittelussa pitkä- että lyhyempiaikainen suunnittelu toimii hyvänä työkaluna.

Slack - Haastattelujen perusteella toimiva tekniikka integraatiotyössä. Varmistaa sen, että myös matalan prioriteetin tehtävät pysyvät mielessä, eivätkä tule listoille vasta sitten kun ne on pakko tehdä. Kuitenkaan jos työtä ei suunnitella esimerkiksi syklimallilla, ei Slack tuo välttämättä matalan prioriteetin tehtäviä näkyväksi. Riippuu siis muista käytössä olevista tekniikoista, miten Slack-tehtäviä kannattaa ottaa mukaan ja onko se hyödyllistä. Kokonaisuuden kannalta tekniikan käyttäminen voisi kuitenkin olla hyödyllistä, sillä näin toimien ennusteiden epätarkkuus ei aiheuta tiimille mahdotonta tilannetta, ja aikatauluihin voidaan päästä, vaikka jossain kohtaa ennusteet pettäisivät (Stellman & Greene 2014, Beck 2004).

Whole team - Jos on mahdollista saada kaikki asian parissa työskentelevät henkilöt samaan tiimiin, saadaan siitä huomattavaa etua. On kuitenkin todennäköistä, että integraatiokehityksessä sidosryhmien määrä kasvaa niin suureksi, että millään järkevällä tavalla ei saada kaikkia samaan tiimiin. Toimii esimerkiksi sellaisissa projekteissa, joissa tarvitaan lyhyeksi aikaa monelta osapuolelta työtä, mutta jatkuvissa palveluissa vaikea toteuttaa. Integraatioprojektissa tekniikkaa tulisi toteuttaa mahdollisuuksien mukaan, koska Kähkösen (2017) mukaan on tärkeää että kommunikaatio on mahdollisimman mutkatonta ja projektin parissa työskentelee kokopäiväisesti henkilöitä.

6.3 Integraatiokehitykseen sopimattomat ketterien menetelmien tekniikat

Tietyistä tekniikoista ei koeta olevan integraatiotyössä hyötyä, tai ainakaan niiden käytöstä ei saada niin suurta hyötyä, että kannattaisi nähdä vaivaa niiden käyttämiseen.

Sprint & Sprint backlog - Scrumissa sprintin ajatuksena on se, että tiimi sitoutuu saamaan kullekin sprintille valitun sisällön sen aikana valmiiksi. Kuitenkin, kuten haastatteluissa tuli ilmi, tämä on integraatiotyössä jokseenkin mahdotonta. Sprint backlog ei täten myöskään ole hyödyllinen tekniikka. On kuitenkin huomioitava, että vaikka Sprint backlog ei tekniikkana toimi integraatiokehityksessä, backlog on erillisenä kohtana nimetty suositelluissa tekniikoissa.

Demotilaisuudet - Scrumissa demotilaisuuden tarkoituksena on demota sidosryhmille sprintin aikana toteutettuja ominaisuuksia. Tämä on helppoa, kun esimerkiksi rakennetaan käyttöliittymiä, mutta integraatioiden tapauksessa tilanne on täysin eri. Tekniselle henkilölle on teoriassa mahdollista esitellä integraation logiikkaa ja koodia, mutta kuten haastatteluissa tuli ilmi, esimerkiksi liiketoiminnan edustajalle sen esittely ei todennäköisesti anna mitään lisäarvoa, ja on kyseenalaista antaako edes asiakkaan tekniselle henkilölle esittely. Tämän vuoksi integraatioiden demoaminen ei ole hyödyllistä, lukuunottamatta harvoja tilanteita joissa integraatio on osa jotain suurempaa kokonaisuutta jota halutaan esitellä, mutta silloinkaan ei todellisuudessa demota itse integraatiota.

Story pointit - Story pointien tarkoituksena on arvioida tehtävän työmäärää suhteessa muihin backlogin tehtäviin (Stellman & Greene 2014). Integraatioiden tapauksessa tämä on haastavaa, sillä tekeminen on harvoin määrämittaista ja itseään toistavaa, sillä jokainen uusi integraatio on jollain tavalla uusi. Story pointien käyttöä voidaan perustella siinä tapauksessa, että halutaan abstrahoida tehtävän monimutkaisuuden arviointi työmäärän arvioinnista. On kuitenkin arvioitava, mitä hyötyä story pointien kautta arvioinnista on suhteessa työmäärien arviointiin käyttäen työtunteja, joka on tapa, jolla asiakkaat usein vaativat arvion. Tämän vuoksi Story Pointien käyttö ei todennäköisesti maksa vaivaa integraatioiden kehityksessä.

Velocity & Burndown chart - Kyseiset mittarit mittaavat sitä, kuinka paljon tehtäviä on saatu valmiiksi sprintin aikana sekä kuinka nopeasti tiimi saa keskimäärin työtä tehtyä. Ongelma tulee siinä, että mittari mittaa tehtävän valmiiksi vasta siinä vaiheessa, kun se on valmistunut, joka integraatioiden tapauksessa ei ole millään tavalla yhteydessä siihen, milloin suurin osa kehitystyöstä tehdään eikä tiimi useinkaan voi vaikuttaa siihen koska tehtävä valmistuu. Tämän vuoksi kyseiset mittarit eivät anna realistista kuvaa integraatiotiimin tehokkuudesta ja niitä ei tulisi käyttää, jotta vältytään väärinkäsityksiltä.

WIP:in rajoittaminen - Periaatteessa hyvä ajatus, ja ajatuksen tasolla voi toimia. Haastateltavat kuitenkin totesivat, että käytännössä kuitenkin integraatioiden kehitys on sellaista, että usein kehittäjä joutuu pakostikin pitämään useampaa kehitystehtävää samanaikaisesti aktiivisena, jotta vältytään ”Wastelta”. Integraatiokehitys vaatii monesti yhteistyötä muiden osapuolten kanssa, ja kalenteriajallisesti esimerkiksi määrittely ja kommunikointi vievät usein paljon kauemmin kuin tekninen toteutus, koska välissä tulee pakostikin esimerkiksi vastausten odottelua. Tämän vuoksi ei ole järkevää rajoittaa WIP:iä, ainakaan kovin matalaksi.

TDD – Kuten haastatteluissa tuli ilmi, kaikki integraatiotyökalut eivät tue TDD:tä ollenkaan, joten TDD on mahdotonta osalla teknologioista. Sen lisäksi haastateltavat ovat käytännössä havainneet, että integraatio valmistuu yleensä nopeammin käytännössä samalla laadulla, kun ensiksi tehdään toteutus ja sen jälkeen testit. Tämän vuoksi, vaikka TDD:tä olisikin teknisesti mahdollista tehdä, ei siitä saavuteta haluttua hyötyä integraatioiden tuotannossa.

6.4 Käytännön implementointi

Kuten esimerkiksi Stellman & Greene (2014) kirjoittavat, on ketterien menetelmien käytössä lähdettävä liikkeelle siitä, että saadaan ihmiset ymmärtämään Agilen arvot ja toimimaan niiden mukaan. Vasta sen jälkeen on kannattava alkaa implementoimaan erilaisia ketterien menetelmien tekniikoita. Integraatiokehityksen tapauksessa tämä tarkoittaa usein sitä, että kehitystiimin lisäksi tulee saada motivoitua sekä asiakas että muut sidosryhmät, ja saada heidät ymmärtämään syyt menetelmien käytön taustalla. Haastatteluissa tuli esiin, että tärkeintä on asiakkaan saaminen mukaan ja sitoutumaan ketteriin menetelmiin. Tämän jälkeen on todennäköisesti huomattavasti helpompaa ottaa käyttöön erilaisia ketterien menetelmien tekniikoita ja työkaluja. Jos muutosvastarinta on kovaa, se tuo oman haasteensa, ja voi jopa estää kokonaan ketterien menetelmien ja tekniikoiden käytön integraatioiden tuottamisessa.

Ketterien menetelmien ja niiden tekniikoiden käyttöönotto vaatii tiimistä ja tilanteesta riippuen erilaisia lähestymistapoja. Osa tiimeistä käyttää jo valmiiksi lähes kaikkia edellisessä kappaleessa esiteltyjä tekniikoita, kun taas joillain tiimeillä lähtökohta voi olla täysin eri. Tärkeää on pystyä perustelemaan kunkin menetelmän käyttö ja valita tapauskohtaisesti ne tekniikat ja työkalut, jotka aidosti tukevat arvon luomista ja tuovat hyötyä eri osapuolille.

6.5 Poikkeamat oletetuista tuloksista

Tutkimuksen tuloksia voidaan pitää luotettavana, koska pohdinnassa esille tulleet tulokset ovat saman suuntaisia kuin mitä teoriassa esitellyt integraatiotuotannon haasteet ja niiden ratkaisut antavat ymmärtää.

7. YHTEENVETO

7.1 Johtopäätökset

Yhteenvetona voidaan todeta, että yksikään tutkituista ketteristä menetelmistä ei suoraan sovi integraatiotuotannon käyttöön sellaisenaan. Kaikissa niissä on kuitenkin hyviä elementtejä, joita yhdistelemällä voidaan luoda kulloiseenkin tilanteeseen soveltuva hybridimenetelmä, jolla on mahdollista saavuttaa ketteryyden tuomia hyötyjä kuten kommunikaation paraneminen, työn merkityksellisyyden lisääntyminen, työntekijöiden parempi sitoutuminen sekä loppujen lopuksi tehokkaampi asiakasarvon tuotto.

7.2 Tutkimuskysymyksiin vastaaminen

Pääkysymys työssä on ”miten ketterän kehityksen keinoja käyttämällä voidaan parantamaa integraatiotiimien toimintatapoja”. Vastauksena voidaan tutkimuksen perusteella sanoa, että valitsemalla eri menetelmistä tiimiin ja projektiin sopivia tekniikoita ja käyttämällä niitä siten, että toimintatavat parantuvat. Käytännössä tulisi toimia siten, että otetaan käyttöön ketterien menetelmien tekniikoita, joilla tunnistetaan ongelmia tai haasteita prosesseissa ja tiimin toimintatavoissa. Tämän jälkeen valitaan käyttöön tekniikoita, joilla tunnistettuja ongelmia korjataan. Lopuksi implementoidaan sellaisia tekniikoita, joilla saadaan aikaan jatkuvia palautelooppeja ja jatkuvan kehityksen kierre, jolla toimintaa pyritään kehittämään jatkuvasti edellä esitetyn prosessin mukaisesti. Näitä tekniikoita on kuvattu tarkemmin kappaleessa 6.

Pääkysymyksen lisäksi työssä on kolme alakysymystä. Ensimmäiseen kysymykseen, ”millaisia haasteita integraatiotekemisessä on”, saadaan vastaus sekä kirjallisuudesta että havaintojen ja haastattelujen perusteella. Kähkönen (2017) on jakanut integraatiotekemisen kolmeen näkökulmaan: tekninen näkökulma, liiketoimintanäkökulma ja sosio-organisationaalinen näkökulma. Näistä jokaisessa on omat haasteensa, eikä integraatioiden kehitystiimin toimintatapoja kehittämällä voida vastata läheskään kaikkiin. Tekniset haasteet ovat esimerkiksi kahden eri tietomallin toisiinsa sopimattomuus, joiden yhteensovittaminen voi vaatia suuria ponnisteluja ja vaivannäköä, jotta se saadaan aikaan (Kähkönen 2017). Liiketoimintanäkökulman haasteet taas voivat olla esimerkiksi tilanteita, joissa liiketoimintaprosessin automatisoinniksi vaaditaan muutoksia useisiin järjestelmiin sekä lisäksi integraatioita näiden välille, mutta prosessia ei voida testata

ennen kuin kaikki osapuolet ovat saaneet kehitysversionsa maaliin (Kähkönen 2017). Sosio-organisationaalisena haasteena taas esimerkki voisi olla haasteet yhteistyössä eri sidosryhmien välillä ja eroavat tavoitteet. Jos esimerkiksi joku sidosryhmä tavoittelee organisaation laajuisessa projektissa ensisijaisesti omaa paikallista etuaan ja vasta toissijaisesti koko organisaation etua, voi se aiheuttaa haasteita sidosryhmien välillä (Kähkönen 2017).

Haastattelujen perusteella paljastui myös useita haasteita. Eräs haaste on sopivan ja sitoutuneen PO:n tai hankkeen omistajan löytyminen voi olla vaikeaa, ja jos tällaista ei löydy, johtaa se helposti priorisoinnin ja suunnittelun ongelmiin. Toisena haasteena nousi esiin suuri sidosryhmien ja kolmansien osapuolten määrä sekä näiden välillä koordinointi. Eri osapuolten eri aikataulut ja heiltä vaadittavat panostukset integraation loppuunviemiseksi on usein ongelma, sillä kehitystiimillä ei välttämättä ole sananvaltaa siihen, koska integraation hyväksymistestaus suoritetaan ja he voivat viedä tuotoksensa maaliin asti. Koordinointi voi olla vaikeaa, sillä monimutkaisessa integraatioprojektissa ei välttämättä ole yhtä selkeää omistajaa, joten koko tekemisen koordinoivan osapuolen löytyminen voi olla hankalaa. Kolmantena haasteena haastatteluissa nousi määrittelyjen ja esityön vaikeus, sillä integraatioiden määrittely etukäteen voi olla vaikeaa tai mahdotonta, ennen kuin integroitavista järjestelmistä on riittävästi tietoa ja kokemusta. Varsinkin, jos järjestelmiä ei vielä ole olemassa tai niitä vasta kehitetään, on muuttuvat vaatimukset ja vaillinaiset määrittelyt toistuva haaste. Neljäntenä haasteena nousi jatkuvien palveluiden toimitusmalli, jossa integraatiotiimi sekä ylläpitää integraatioita että toteuttaa uusia. Tällaisissa tapauksissa voi olla haastava saada näkyvyyttä tulevaisuuteen ja epävarmuuden kanssa on pystyttävä toimimaan, joka aiheuttaa haasteita resursoinnin näkökulmasta. Tämän myötä myös priorisointi eri tehtävien kesken voi olla haastavaa, sillä samoja resursseja joudutaan käyttämään eri projekteissa ja asiakkuuksissa, ja näiden välissä toimiminen tuo ajoittain haasteita.

Toinen alakysymys, ”Mitkä ketterät menetelmät sopivat integraatiokehitykseen”, tosiasiallisena vastauksena on, että yksikään tutkituista menetelmistä ei sovellu sellaisenaan integraatiokehitykseen. Kuitenkin, kuten luvussa 6 on kuvattu, monet näiden menetelmien tekniikoista soveltuvat hyvinkin integraatiokehitykseen. Integraatiotiimin toiminnan kehittämiseksi eri menetelmistä voidaan valita tekniikoita, joista kaikki sopivat yleisesti integraatioitiimien toimintaan. Scrumin integraatiotuotantoon yleisesti soveltuvat tekniikat ovat Daily, Review & Planning, Product Owner, Story refinement ja backlog. Kanbanista yleisesti toimivat tekniikat taas ovat työn visualisointi, virran hallitseminen XP:n yleisesti toimivat tekniikat ovat Energized work, Näe kokonaisuus, incremental design sekä

10-minute build. Leanin yleisesti sovellettavat tekniikat ovat Empower the team, Eliminate waste, Tee päätökset mahdollisimman myöhään, Feedback-loopit & Amplify learning

Näiden tekniikoiden lisäksi osa tekniikoista sopii tapauskohtaisesti integraatiotuotantoon, mutta eivät kaikissa tapauksissa. Nämä tekniikat ovat Scrumin osalta User Stories, Definition of Done & Definition of Ready, retrospektiivi sekä scrumin roolijako. Kanbanin tekniikoista tällainen on prosessien kirjaaminen ja dokumentointi. XP:n tekniikoista tähän kategoriaan listautuu Sit Together, pariohjelmointi, Informative Workspace, Continuous integration, Slack, Weekly & Monthly cycle sekä Whole team. Leanista periaate ”Deliver as fast as possible” on myös listattu tähän kategoriaan. Tarkemmat kuvaukset ja perustelut on luettavissa kappaleesta 6.

Kolmas kysymys, ”millä eri tavoin kukin osapuoli hyötyy integraatiotiimin ketteristä toimintamalleista”, voidaan perustella monin tavoin eri näkökulmista. Asiakkaan kannalta ketteristä toimintamalleista saadaan hyötyä sekä arvon luonnin että näkyvyyden parantumisella. Asiakas pystyy olemaan tiiviimmin mukana kehitystiimin arjessa ja halutessaan ohjaamaan kehitystä lyhyellä palautesyklillä toivomaansa suuntaan ja varmistamaan saamansa arvon maksimoinnin. Työnantajayrityksen näkökulmasta ketterät menetelmät mahdollistavat tiimien itsenäisemmän toiminnan, jolloin välijohdon tarve on pienempää, sekä kasvaneen laadun, toimitusnopeuden ja asiakastyytyväisyyden. Lisäksi ketterien menetelmien avulla voidaan parantaa työntekijöiden viihtyvyyttä ja pienentää vaihtuvuutta. Työntekijöiden kannalta ketterät menetelmät auttavat työn ennustettavuuden lisääntymisessä ja toiminnan selkeytymisessä. Kun prosessit ja toimintatavat ovat selkeitä ja tekemistä pyritään jatkuvasti kehittämään, auttaa se työssä viihtymisessä ja stressitason laskemisessa verrattuna tilanteeseen, jossa näihin asioihin ei kiinnitettäisi huomiota.

7.3 Tutkimuksen arviointi ja luotettavuus

Tutkimusta voidaan arvioida sen mukaan, miten hyvin se vastaa tutkimusongelmaansa tutkimuskysymysten kautta. Tutkimusongelman ollessa ”miten integraatiotiimien toimintamalleja olisi mahdollista kehittää ketterän ohjelmistokehityksen menetelmien avulla”, voidaan siihen mielestäni vastata tämän tutkimuksen kysymysten avulla, jotka käsittelevät sekä integraatiokehityksen haasteita, ketterien menetelmien arvioitua sopivuutta integraatiotekemiseen sekä eri osapuolten saamia hyötyjä menetelmistä.

Haastattelujen perusteella saadut näkökulmat ovat peräisin suurilta osin organisaation jäsenten haastatteluista, joihin valittiin viisi henkilöä noin viidenkymmenen henkilön

case-organisaatiosta. Haastateltavat pyrittiin valitsemaan siten, että he edustaisivat organisaatiota ja sen ominaisuuksia mahdollisimman monipuolisesti. Todennäköistä on, että tässä tavoitteessa onnistuttiin, sillä haastateltavien vastaukset olivat hyvin samansuuntaisia huolimatta eri taustoista ja nykyprojekteista. Jos taas vastaukset olisivat suuresti eronneet toisistaan, olisi voinut olettaa, että muita ihmisiä haastattelemalla voitaisiin saada vielä erilaisia tuloksia, ja haastateltavien määrää olisi tullut lisätä. On toki pieni mahdollisuus, että valitut haastateltavat edustavat pientä osajoukkoa organisaation sisällä, jotka sattumalta ajattelevat asioista samalla tavalla. Tässä tapauksessa lisähaastattelut olisivat voineet tuoda varmuutta tuloksille, mutta arvioisin todennäköisyyden tälle olevan pieni.

Tutkimuksen haasteena voidaan pitää sitä, että tutkittavat kohteet ovat samasta organisaatiosta. Eri organisaatioiden henkilöitä haastattelemalla olisi voinut olla mahdollista saada uusia näkökulmia esimerkiksi haasteisiin ja mahdollisuuksiin. Esimerkiksi asiakasnäkökulmaa ei tutkimuksessa selvitetty asiakkaan henkilöiltä, vaan luotettiin haastateltavien näkemyksiin aiheesta.

Voidaan myös ajatella, että syventymällä tarkemmin esimerkiksi yhden tiimin toimintatapoihin olisi tuloksista mahdollisesti saatu tarkempia kyseisen tiimin näkökulmasta. Nyt työssä otettiin eri näkökulmia organisaation eri tiimeistä, jolloin saatiin laajempi näkökulma, mutta ehkä menetettiin hieman mahdollisuutta syvempään ja tarkempaan asioiden tarkasteluun.

Tutkimuksen reliabiliteetti – eli voidaanko tutkimus toistaa ja saada samoja tuloksia – on mielestäni hyvä. Haastattelurunko on kuvattu liitteessä A, ja kappaleessa 5 avattu mahdollisimman läpinäkyvästi haastateltujen kommentteja aiheesta. Myös päättelyprosessi kappaleessa 6 on pyritty tekemään mahdollisimman johdonmukaiseksi ja avoimeksi, joilla tuloksiin on päästy.

7.4 Jatkotutkimustarpeet

Tässä tutkimuksessa tutkittiin neljää yleisintä ketterää menetelmää. Jatkotutkimuksena voisi olla hyödyllistä tutkia myös muita ketteriä menetelmiä, joissa saattaa olla muita sellaisia asioita, mistä integraatiotiimeille olisi hyötyä. Erilaisten menetelmien määrä lisääntyy jatkuvasti, ja todennäköisesti neljän yleisimmän menetelmän ulkopuoleltakin löytyy mielenkiintoisia tutkimuskohteita.

Toisena jatkotutkimusehdotuksena on syventyä tarkemmin asiakasnäkökulmaan ja asiakaskokemukseen ketteristä menetelmistä. Tässä työssä asiakasnäkökulmaa kartoitettiin

vain organisaation sisäisten kokemusten kautta, mutta jatkotutkimuksena olisi mielenkiintoista saada tietoa asiakkaiden kokemuksista ja toiveista toimittajan käyttämien menetelmien suhteen.

LÄHTEET

Agile Manifesto. 2001. Saatavilla: https://agilemanifesto.org/iso/fi/manifesto.html (Haettu 11.9.2019)

Bailey, L. E. (2010) Case Study Research. Encyclopedia of Curriculum Studies, Thousand Oaks, CA: SAGE Publications, Inc, 2010. p.103–105.

Balaji, S. & Murugaiyan, 2012. M. Waterfall vs V-model vs Agile: A Comparative Study On SDLC. International Journal of Information Technology and Business Management 29th June 2012. Vol.2 No. 1

Barki, H. & Pinsonneault, A. 2005. A Model of Organizational Integration, Implementation Effort, and Performance. OrganizationScience16 (2): 165–179.

Beck, K. 2004. Extreme programming explained: Embrace change. 2nd ed. Boston, MA: Addison-Wesley.

Bicheno, J., & Holweg, M. 2000. The lean toolbox (Vol. 4). Buckingham: PICSIE books.

Cockburn, A. Agile Software Development. 2000. Highshmith Series Editors.

CollabNet. 2019. 13th Annual State of Agile Report. Saatavilla: https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508 (Haettu: 4.10.2019)

Giarrizzo, M., J.R. (2016). Create Flow by Eliminating Waste. Automotive Body Repair Network, 55(6), 16-16,18,20.

Hiltunen, L. 2009. Validiteetti ja reliabiliteetti. Saatavilla: http://www.mit.jyu.fi/ope/kurssit/Graduryhma/PDFt/validius_ja_reliabiliteetti.pdf (Haettu 16.2.2020)

Hobday, M., Davies, A. & Prencipe, A. 2005. Systems integration: a core capability of the modern corporation, Industrial and Corporate Change, 14, no. 6 (2005), 1109-1143

Karvonen, M. 2004. Ohjelmistoyritysten liiketoimintamallit Kaakkois-Suomessa. Diplomityö, Lappeenrannan Teknillinen Yliopisto.

Kähkönen, T. 2017. Understanding and Managing Enterprise Systems Integration. Thesis for the degree of Doctor of Science (Technology), Lappeenranta University of Technology.

Linthicum, D. S. 2004. Next Generation Application Integration: From Simple Information to Web Services. Addison-Wesley Information Technology Series. Boston, MA: Addison-Wesley

Little, J. 2011. Little's Law as Viewed on Its 50th Anniversary. Operations Research, 59(3), 536-549.

McGreal, D. & Jocham, R. 2018. The Professional Product Owner: Leveraging Scrum as a Competitive Advantage. Addison-Wesley Professional; 1 edition

Mobley, W. H. 1982. Employee turnover: Causes, consequences, and control. Addison-Wesley.

Prencipe, A. 2003. Corporate strategy and systems integration capabilities: managing net- works in complex systems industries, in A. Prencipe, A. Davies and M. Hobday (eds), The Business of Systems Integration. Oxford University Press: Oxford.

Riege, A. 2003. Validity and reliability tests in case study research: a literature review with “hands-on” applications for each research phase, Qualitative Market Research, Vol. 6 No. 2, pp. 75-86.

Schwaber, K. & Sutherland, J. 2017. The Scrum Guide. Scrum, Inc. Saatavilla: https://www.scrum.org/resources/scrum-guide (Haettu 23.11.2019)

Solinski, A., & Petersen, K. 2016. Prioritizing agile benefits and limitations in relation to practice usage. Software Quality Journal, 24(2), 447-482.

Stellman, A. & Greene, J. 2014. Learning Agile. O’Reilly Media, Inc.

Tessem, B. 2014. Individual empowerment of agile and non-agile software developers in small teams. Information and Software Technology, Vol. 56, Issue 8, pp. 873-889

Themistocleuous, M. 2004. Justifying the decisions for EAI implementations: a validated proposition of influential factors. Journal of Enterprise Information Management, Vol. 17 No. 2, pp. 85-104.

Vacanti, D. & Yeret, Y. 2019. The Kanban Guide for Scrum Teams. Scrum, Inc. Saatavilla: https://www.scrum.org/resources/kanban-guide-scrum-teams (Haettu 25.11.2019)

Lam, W. 2005. AN ENTERPRISE APPLICATION INTEGRATION (EAI) CASE-STUDY: SEAMLESS MORTGAGE PROCESSING AT HARMOND BANK. The Journal of Computer Information Systems, 46(1), 35-43.

YIN, R. K. (2003). Case study research: design and methods. Thousand Oaks, Calif, Sage Publications.

EI-AKATEEMISET LÄHTEET

Denning, S. 2019. Understanding the Agile Mindset. Forbes Media LLC. Saatavilla: https://www.forbes.com/sites/stevedenning/2019/08/13/understanding-the-agile-mindset (Haettu 11.9.2019)

Folkman, J. 2017. The 6 Key Secrets to Increasing Empowerment in Your Team. Forbes Media LLC. Saatavilla: https://www.forbes.com/sites/joefolkman/2017/03/02/the-6-key-secrets-to-increasing-empowerment-in-your-team/ (Haettu 23.11.2019)

Lamelas, A. 2018. Top 5 main Agile methodologies: advantages and disadvantages. Saatavilla https://www.xpand-it.com/2018/10/11/top-5-agile-methodologies/. (Haettu: 31.7.2019)

Liu, L., Hays, T., Weixin, X. & Dunnigan, N. 2015. Integration platform as a service: The next generation of ESB. Saatavilla: https://developer.ibm.com/articles/cl-ipaas-next-generation-of-esb1/ (Haettu 15.12.2019)

Manninen, M. 2015. Vertikaali SaaS on tulikuuma aihe – 6 asiaa, jotka sinun tulee tietää siitä. Saatavilla: https://zure.com/fi/blogi/vertikaali-saas-on-tulikuuma-aihe-6-asiaa-jotka-sinun-tulee-tietaa-siita/ (Haettu 15.12.2019)

Solita OY. 2019. Integraatioratkaisut. Saatavilla: https://www.solita.fi/integraatioratkaisut/ (Haettu 16.5.2019)

LIITE A: HAASTATTELUKYSYMYKSET

Kysymys	
Mitä mieltä olet ketteristä menetelmistä integraatiotyössä	
Mitä ketteriä malleja käytät nykytekemisessä	
Mitä koet tärkeimmäksi hyödyksi mitä malline pitäisi antaa kehittäjälle	
Koetko ne hyödylliseksi tai tarpeelliseksi	
	Jos koet, niin mitkä?
	Mitä et koe hyödylliseksi, ja miksi?
Mitä ovat ketterien menetelmien käytön isoimmat haasteet kehittäjän näkökulmasta?	
Miten asiakkaat mielestäsi suhtautuvat ketteriin menetelmiin?	
Kokisitko että seuraavista yleisimmistä ketteristä menetelmistä on hyötyä, ja mitä hyötyä jos on? Onko jotain mistä ei ole hyötyä?	
Scrum	
	Roolit: SM, PO, Kehitystiimi
	Rajatun pituinen sprint, jossa rajattu scope mikä tehdään sprintin aikana valmiiksi
	Scrum-palaverit: Sprint planning, Sprint review, Retrospective, Daily Scrum
	Story refinement & arviointi
	Product & Sprint backlog
	Story pointit Definition of Done, Velocity, Agile Coach
Kanban	
	Visualisointi
	WIP:in rajoittaminen
	Virran hallitseminen
	Prosessien käytännöt täsmälliseksi
	Feedback-looppien implementointi
	Paranna kollaboroimalla, evolvoi kokeilemalla
XP	
	TDD
	Pariohjelmointi
	10-minute build (build saa kestää max 10min jotta se tehtäisiin mahd usein)
	Continuous integration (koodille ajetaan aina testit kun lisätään koodipohjaan)
	Weekly cycle
	Quarterly cycle
	Stories (arviointi, refinement)

	Slack (low-prio taskeja mukaan sykleihin)
	Sit together
	Informative workspace (information radiators, keskustellaan tiimitilassa, postereita)
	Whole team (tiimi täyttää kaikki roolit mitä tarvitaan)
	Energized work (max 40h viikko, arvostus itselle ja työkavereille)
	Incremental design (aloitetaan isosta ja siirrytään tarkempaan, refaktorointi)
	Roolit: asiakas, kehittäjä, "the tracker", valmentaja (ei tiukasti määrätty xp:ssä)
Lean	
	Eliminate waste
	Amplify learning (palaute ja iteraatiot)
	Tee päätökset mahd myöhään
	Deliver as fast as possible -> viivästyksen hinta, pull-järjestelmät ja jonot
	Empower the team (rakennetaan työtila joka toimii ja tiimissä energisiä ihmisiä)
	Build integrity in
	Näe kokonaisuus
Vapaita ajatuksia / kommentteja	