Tomi Makkonen

OHJELMISTOKEHITYKSEN MUUTOS JA SEN VAIKUTUS OHJELMISTOPROJEKTIEN JOHTAMISEEN

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2015

TIIVISTELMÄ

Makkonen, Tomi
Ohjelmistokehityksen muutos ja sen vaikutus ohjelmistoprojektien johtamiseen
Jyväskylä: Jyväskylän yliopisto, 2015, 25 s.
Tietojärjestelmätiede, kandidaatin tutkielma
Ohjaaja: Pirhonen, Maritta

Tutkielman tarkoituksena oli selvittää, miten ja miksi ohjelmistokehittäminen on muuttunut viimeisen 10 vuoden aikana ja etenkin, kuinka tämä muutos on vaikuttanut ohjelmistokehitysprojektien johtamiseen. Aihetta lähestyttiin perinteisen ohjelmistokehittämisen vesiputousmallin ja ketterän kehittämisen scrum-menetelmän kautta. Esimerkkinä toimivista menetelmistä käytiin läpi perusteet toimintatavoista sekä johtamisesta. Kun menetelmiin oli tutustuttu, siirryttiin haasteisiin, joita yrityksissä kohdattiin perinteisestä ketterään kehitykseen siirryttäessä. Tämän jälkeen syvennyttiin vielä hieman tarkemmin johtamiseen liittyviin haasteisiin ja ominaisuuksiin, joita hyvällä projektipäälliköllä ja scrummasterilla tulisi olla. Tutkimusmenetelmä oli kirjallisuuskatsaus ja lähteinä käytettiin alan kirjallisuutta ja tieteellisiä artikkeleja. Hakusanoina käytettiin sanoja: perinteinen ohjelmistokehittäminen, vesiputousmalli, ketterä kehittäminen, scrum, projektipäällikkö, scrummaster ja edellä mainittujen roolit sekä tehtävät myös englanniksi haettuna. Perinteisestä ohjelmistokehityksestä sekä ketterästä kehittämisestä oli paljon aineistoa sekä vertailua. Myös esimerkkimenetelmistä oli saatavilla runsaasti lähdemateriaalia. Haasteet tiedonkeruussa painottui scrum-menetelmän johtamisen tieteellisen lähdeaineiston vähäisyyteen aihepiirin ollessa vielä suhteellisen tuore.

Asiasanat: perinteinen ohjelmistokehittäminen, ketterä kehittäminen, projektipäällikkö, scrummaster, roolit, tehtävät

ABSTRACT

Makkonen, Tomi
Change in software development and how it effects to the project management
Jyväskylä: University of Jyväskylä, 2015, 25 p.
Information Systems, Bachelor's Thesis
Supervisor: Pirhonen, Maritta

The aim of the thesis was to explore how and why software development has changed during last ten years and, especially, how this change has affected to management of development projects. The subject was approached through traditional software development's waterfall model and agile development's scrum-method. These example methods were covered telling how those basically work and little detailed about management and manager's roles and tasks. This thesis was done using literature review and references were literature and scientific articles. Keywords that were used: traditional software development, agile software development, project manager, scrummaster, tasks and roles. There was a lot of information about traditional and agile software development and about waterfall-model and scrum. Some challenges were faced when searching scientific information about scrummaster because the title is quite immature.

Keywords: traditional software development, agile software development, project manager, scrummaster, tasks, roles

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Tietojärjestelmät ovat olleet osa elämäämme jo yli viidenkymmenen vuoden ajan ja niiden vaikutus arkielämään jatkaa kasvuaan. Arkielämässä ohjelmistoihin törmää, kun ottaa käteen puhelimen, avaa tietokoneen tai lähes minkä tahansa elektronisen laiteen. Tietojärjestelmiä on kehitetty koko niiden olemassaolon ajan ja tällä hetkellä vallalla on kaksi erityylistä kehittämisen ajattelutapaa. Toiset käyttävät niin sanottua perinteistä ohjelmistokehitystapaa, jossa tukeudutaan suunnitteluun. Toiset taas käyttävät kevyempää, ketterää ohjelmistokehittämistä, jonka käyttö on kasvanut viimeisen kymmenen vuoden aikana selkeästi. Ketteriin menetelmiin siirtyminen perustellaan useimmiten sillä, että markkinat tarvitsevat erittäin nopeasti muutokseen vastaavia toimittajia perinteisen kehittämisen kärsiessä kyvystä toimia nopeasti pitkine vaiheineen ja suunnitteluineen (Awad, 2005).

Molemmat ohjelmistokehitysmenetelmät tunnistavat kolme kehitystä rajoittavaa tekijää: kustannukset, aikataulun ja sisällön. Näihin tekijöihin kuitenkin suhtaudutaan eri tavoin. Perinteisessä ohjelmistokehityksessä vaatimukset lukitaan, jotta aikataulu ja kustannukset pystytään arvioimaan. Ketterässä kehityksessä puolestaan vaatimukset, eli sisältö muuttuu, joten myös aikataulu sekä kustannukset ovat muuttuvia.

Tässä tutkielmassa tarkastellaan ensin perinteistä ohjelmistokehitystä vesiputousmallin avulla ja sen jälkeen siirrytään ketterään ohjelmistokehitykseen scrum-menetelmään tutustumalla. Kyseiset menetelmät käydään läpi, jotta myöhemmässä vaiheessa tutkielmaa kyetään vertailemaan projektien johtamista ja sitä, mitä haasteita ohjelmistokehittämisen muutos aiheuttaa organisaatioissa. Tutkimuskysymykset ovat:

1. Miten ohjelmistokehittäminen on muuttunut?
2. Mitä haasteita ohjelmistokehittämisen muutoksessa kohdataan ja miten muutos vaikuttaa projektien johtamiseen?

Tutkielman ensimmäisessä luvussa perehdytään perinteiseen kehittämiseen ja alaluvuissa syvennytään ensin vesiputousmalliin ja sitten projektipäällikkyyteen. Seuraavassa luvussa käsitellään ketterää kehittämistä ja alaluvuissa syvennytään scrum-menetelmään ja scrum-prosessin toimintaan sekä scrum-masterina toimimiseen. Kolmannessa sisältöluvussa käsitellään perinteisestä

kehittämisestä ketterään kehittämiseen siirtymisestä aiheutuvia muutoksia. Alaluvuissa syvennytään erilaisiin haasteisiin, mitä kohdataan siirryttäessä perinteisestä kehittämisestä ketterään. Lopuksi käydään läpi, mitä ominaisuuksia hyvältä projektipäälliköltä ja scrummasterilta odotetaan ja kuinka samankaltaisia nämä odotukset ovat.

Lähdekirjallisuutta tutkiessani törmäsin ongelmaan, että osa tutkielman kannalta oleellisista sanoista on suomentamatta, ja tutkielman ollessa suomenkielinen, ratkaisin ongelman käyttämällä alan osaajien tuottamaa sanastoa aihepiiriin liittyen. Sanasto on scrumin perustajien Schwaberin ja Sutherlandin julkaisemasta Scrum Guidesta suomennettu. Suomenkielisen scrum-sanaston päivitykseen osallistuivat tammikuussa 2014 suomalaiset scrumin ammattilaiset. Sanasto on haettavissa osoitteesta: http://lekman.fi/scrumguide/ (Eskelinen, 2014).

2 PERINTEINEN OHJELMISTOKEHITTÄMINEN

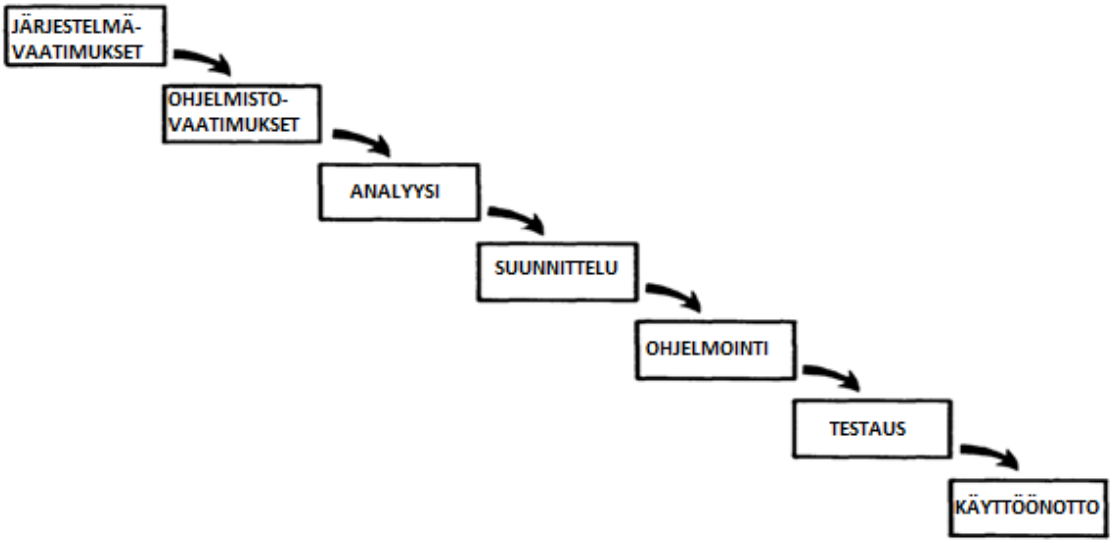
Tämä luku keskittyy perinteiseen ohjelmistokehitykseen ja projektien johtamiseen ohjelmistokehityksessä. Ensimmäisessä alaluvussa käsitellään mitä perinteinen ohjelmistokehitys on ja esimerkkimenetelmänä käytetään vesiputousmallia. Vesiputousmallista käsitellään yleiset toimintaperiaatteet sekä vahvuudet ja heikkoudet. Toisessa alaluvussa syvennytään projektipäällikön toimintaan ja tehtäviin.

Perinteiselle ohjelmistokehitykselle tyypillistä on perinpohjainen suunnittelu ja hallinta. Projektin jokainen vaihe on helposti eroteltava ja tunnistettava, ja tehtävät suoritetaan aina järjestyksessä siten, että uusi tehtävä aloitetaan vasta kun edellinen on valmis. Perinteinen ohjelmistokehittäminen vaatii paljon suunnittelua ennen toteutuksen aloitusta ja sitä voi verrata esimerkiksi rakennusprojektiin, jossa määritellään ensin vaatimukset ja niiden pohjalta suunnitellaan ja muotoillaan koko rakennus ja vasta sitten aloitetaan itse rakentaminen (Hass, 2007).

Perinteisessä kehittämisessä on neljä tunnuksenomaista vaihetta: Ensimmäinen vaihe on *vaatimusten määrittäminen* sekä jokaisen vaiheen vaatiman ajan suunnitteleminen, ennustaen kaikki projektiin liittyvät mahdolliset ongelmat. Määrittämisen jälkeen siirrytään *arkkitehtuurisuunnitteluun*, joka toteutetaan mallintamalla suunnitelmat erilaisia kaavioita hyödyntäen. Kaavioiden tarkoituksena on selventää kehittäjille, mitä ohjelmistolta odotetaan. Suunnitteluosion ollessa tyydyttävällä tasolla siirrytään kehitysvaiheeseen eli *ohjelmointiin*. Ohjelmointia suoritetaan, kunnes määrätyt tavoitteet saavutetaan. Kehitystyö jaetaan usein pienempiin kokonaisuuksiin siten, että eri tiimit vastaavat omien osaamisalueidensa tekemisestä. *Testaaminen* aloitetaan usein kehitystyön ollessa vielä kesken, sillä tämä helpottaa mahdollisten ongelmakohtien osoittamista. Kun projekti on lähes valmis, asiakas otetaan mukaan testaukseen ja palautteen perusteella tuote viimeistellään halutunlaiseksi (Beng et al., 2012).

2.1 Vesiputousmalli

Vesiputousmalli on perinteisen ohjelmistokehityksen käytetyimpien mallien perusta ja se muodostui 1970-luvulla, kun Royce jaotteli ohjelmistoprojektin analysointi- ja ohjelmointiosioon, ja vielä tarkemmin kuvassa esiintyvään seitsemään eri vaiheeseen: järjestelmävaatimukset, ohjelmistovaatimukset, analyysi, suunnittelu, ohjelmointi, testaus ja käyttöönotto (Royce, 1970).



KUVIO 1 Vesiputousmalli (Royce, 1970)

Vesiputousmallin ajatuksena on, että kehitys menee vaihe kerrallaan läpi kaikki eri vaiheet. Toimintatavan oletuksena on, että projektiin vaikuttavat tekijät ovat ennustettavissa, ja että työkalut ja toimenpiteet tunnetaan. Toimintatapa vaatii suuria määriä etukäteissuunnittelua, sillä vaiheen valmistuttua oletetaan, ettei siihen enää palata. Vahvuutena vesiputousmallia käytettäessä on selkeä jako kehityksen eri vaiheisiin ja vaatimusten tärkeyden korostus. Rajoittavia tekijöitä on se, että projekti etenee harvoin peräkkäisenä vaiheiden virtana ja se, että asiakkaiden on haastavaa muodostaa kaikki vaatimukset kehityksen alussa (Hass, 2007).

Vesiputousmallin hyviksi puoliksi mainitaan vaatimusten selvyys ja sitä kautta selkeys ennen kehitystyötä, jokaisen vaiheen suorittaminen määritetyssä ajassa ennen siirtymistä seuraavaan sekä mallin suoraviivaisuus ja sen helppo toteutettavuus. Haastaviksi piirteiksi mainitaan muun muassa se, että vaiheen valmistuminen täysin vaiheelle varatun ajan aikana ei onnistu. Siirryttäessä jo seuraavaan vaiheeseen, edellisen vaiheen ongelmat seuraavat vielä siirtymisen jälkeenkin. Haasteita tulee myös jos asiakas haluaa muuttaa vaatimuksia kesken kehitysprosessin, sillä jos ohjelmistoa on kehitytetty jo pitkälle aiempiin vaiheisiin siirtyminen ja niiden muokkaaminen on työlästä ja vaikeaa (Balaji & Murugaiyan, 2012).

Ajatus ohjelmistoprojektista, jossa vaatimukset voidaan määritellä täysin ja sitten suorittaa analyysivaihe valmiiksi ja tämän jälkeen suunnitella ohjelmisto kokonaan on käytännössä mahdoton. Näin toimittaessa ei huomioida, että koko kehityksen aikana opitaan paljon oleellista tietoa, minkä perusteella tulee tehdä muokkauksia jo olemassa olevaan ohjelmistoon. Malli ei enää sellaisenaan olekaan vallalla, mutta siihen pohjautuvat suunnittelua ja vaiheistusta painottavat toimintatavat ovat, joten on hyvä ymmärtää mikä on kaiken perustana (Aitken & Ilango, 2013).

2.2 Projektipäällikön rooli ja tehtävät

Projekti on tietyn ajan kestävä yritys toteuttaa tai luoda ainutlaatuinen tuote, palvelu tai tulos (Guide, 2001). Määräaikaisuus tarkoittaa sitä että projektilla on selkeä alku ja loppu. Lopetuksen aika on silloin, kun asetetut tavoitteet saavutetaan tai kun todetaan, ettei niitä ole enää mahdollista saavuttaa. Projektinhallinta taas on tiedon, taidon ja työkalujen sekä tekniikoiden hyödyntämistä tavoitteiden saavuttamiseksi. Projektinhallinta jakautuu viiteen prosessiryhmään ja nämä prosessit ovat aloitus, suunnittelu, toteutus, hallinta ja lopetus. Tyypillisesti ohjelmistoprojektin hallitsemiseen kuuluu vaatimusten määrittäminen, sidosryhmien tarpeiden tunnistaminen ja tasapainoilu rajoittavien tekijöiden kuten sisällön, laadun, aikataulun, budjetin ja riskien välillä (Guide, 2001).

Projektipäällikön rooli määritellään seuraavasti: ”Yksilö, joka on vastuussa projektin hallitsemisesta” (Guide, 2001). Hallinnalla tarkoitetaan johdonmukaista työskentelyä sidosryhmien odotuksien mukaisesti. Muita päällikön rooleja ovat delegoija, joka jakaa toteutusvastuuta projektitiimille ja vastuuhenkilöille sekä integroija, joka yhdistää projektin tehtävät ja vaiheet ja pitää sidosryhmät ajan tasalla. Edellä mainittujen lisäksi rooleja ovat myös ihmisten ja asioiden johtaja, mentori, esimies, asiantuntija sekä neuvottelija.

Projektipäällikön tehtävät muodostuvat luonnollisesti prosessien pohjalta ja niihin kuuluvat määrittely, suunnittelu, johtaminen, hallitseminen ja valmiiksi saaminen. Määrittely tarkoittaa projektin tavoitteiden selvittämistä. Suunnittelu koostuu visioista kuinka projektipäällikkö ja tiimi tyydyttävät asiakkaan tarpeet rajoittavien tekijöiden kuten aikataulun, budjetin ja työkyvyn puitteissa. Johtaminen tarkoittaa johtajan antamaa opastusta, jonka tavoitteena on tehokas ja tehtävän vaatima työskentely. Hallitseminen tarkoittaa projektin tilan hahmottamista suhteessa suunniteltuun ja reagoimista tarvittavin muutoksin. Valmiiksi saaminen on huolehtimista siitä, että valmis tuote vastaa sitä, mitä pitikin (Rosenau, 2005).

3 KETTERÄ KEHITTÄMINEN

Tässä luvussa käsitellään ketterää kehittämistä, sen perus periaatteita sekä esimerkinomaisesti scrum-menetelmää ja sen toimintaa. Ensimmäinen alaluku keskittyy ketterään kehittämiseen, sen syntymiseen ja ydinajatuksiin. Toinen alaluku esittelee scrum-menetelmän ja siinä käytettävän scrum-prosessin perusteet. Kolmannessa alaluvussa paneudutaan scrumin käytänteisiin, rooleihin sekä johtamiseen.

Kolme suurinta haastetta, joita perinteinen ohjelmistokehittäminen kohtaa tämän päivän kehityksessä on kehityksen kiihtyvä nopeus, kyky muutokseen ja tuotteiden kasvava monimutkaisuus. Nämä haasteet tarvitsevat yhä kevyempiä ja nopeampia ohjelmistokehitysprosesseja. Vastaukseksi on kehitetty monia uusia ohjelmistojen kehittämismenetelmiä. Kaikkia uusia menetelmiä yhdistää ketteryys, joka on määritelty seuraavasti: valmiutta liikkeeseen, aktiivisuuta ja kätevyyttä liikkeessä ollessa (Awad, 2005). Miten nämä määritelmät yhdistetään ohjelmistokehitykseen?

Ensimmäiset ideat ketteristä menetelmistä jäljitetään 1986 ilmestyneeseen Nonakan ja Takeuchin julkaisuun, mutta vasta 1995 järjestetyssä konferenssissa mainitaan ohjelmistokehityksen ketterät menetelmät ensi kerran. Ketterien menetelmien varsinaisena syntyhetkenä voidaan kuitenkin pitää vuotta 2001, jolloin julkaistiin The Agile Manifesto, Ketterän ohjelmistokehityksen julistus. Tässä julistuksessa muutettiin ohjelmistokehityksen painotukset päälaelleen lupaamalla, että arvostetaan:

- Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
- Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
- Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
- Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Näiden ajatuksien pohjalta on muotoutunut useita ketteriä kehitysmenetelmiä, ja tässä tutkielmassa syvennytään niistä scrum-menetelmään ja sen käytänteisiin (Beck et al., 2001).

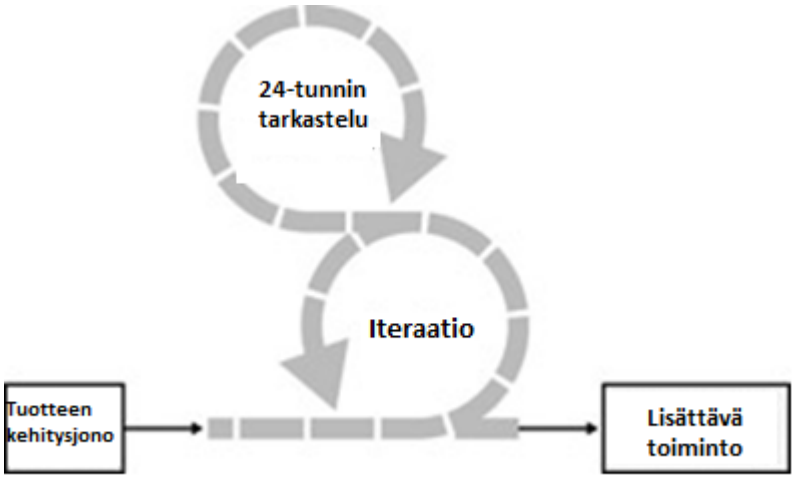
Ketterien menetelmien ydin on helpompi ymmärtää yhdeksän tunnuksenomaisen piirteen kautta. Näitä piirteitä ovat ihmiskeskeisyys, mukautuvuus, tuloksellisuus, tasapainoilu joustavuuden ja suunnittelun välillä, prosessin empiirisyys, hajautettu lähestymistapa, yksinkertaisuus, yhteistyö ja pienet itseohjautuvat tiimit. Ihmiskeskeisyys tarkoittaa sitä, että asiakkaat, kehittäjät, sidosryhmä sekä loppukäyttäjät ovat tärkein tekijä ohjelmistokehityksessä. Mukautuvuus ilmenee siten, ettei muutosta pelätä ja muutoksia voi tulla missä tahansa projektin vaiheessa. Tuloksellisuus on sitä, että onnistumista verrataan todellisiin tuloksiin eikä yksityiskohtaisiin suunnitelmiin. Tasapainoilu joustavuuden ja suunnittelun välillä tarkoittaa sitä, että suunnitelmat ovat tärkeitä, mutta haasteena on se, että ohjelmistoprojekteja ei voida ennustaa pitkälle tulevaisuuteen. Toimivammaksi ratkaisuksi koetaan suunnittelu, jossa tehdään yksityiskohtaiset suunnitelmat seuraaviksi viikoiksi ja karkeita arvioita muutamaksi kuukaudeksi eteenpäin. Tasapainoilua tapahtuu siis jo tehtyjen suunnitelmien ja muutoksiin reagointien välillä. Empiirinen prosessi tarkoittaa sitä, että prosessin lopputulos ei ole aina sama vaan olosuhteet muuttuvat koko ajan ja nämä vaikuttavat lopputulokseen. Hajautettu lähestymistapa ilmenee tiimin omana päätöksentekona, ilman ylempää auktoriteettia. Yksinkertaisuus on käsillä olevaan työvaiheeseen keskittymistä, tulevia ongelmia ei pohdita etukäteen. Yhteistyössä merkittävässä roolissa on asiakkaalta saatava jatkuva palaute ja asiakkaan läsnäolo kehitystyössä. Myös tiimin sisäinen johdonmukainen vuorovaikutus on olennaisessa osassa työskentelyä. Tiimi organisoituu ja jakaa vastuut itse. Olennaista ketterässä kehittämisessä on myös tiimin sisäinen keskustelu ja kommunikaatio. Vuorovaikutuksen helpottamiseksi tiimit ovat useimmiten varsin pieniä, noin yhdeksän hengen tiimejä (Awad, 2005).

3.1 Scrum-prosessi

Tässä alaluvussa käsitellään, mikä scrum on, miten sitä käytetään ja keitä kuuluu scrumtiimiin. Scrum on tietynlainen ohjelmistokehitystä varten luotu viitekehys, jonka sisällä voidaan käyttää erilaisia prosesseja ja tekniikoita. Scrum perustuu Schwaberin ja Sutherlandin (2013) mukaan empiiriseen prosessinhallintateoriaan ja empirismiin. Siinä tieto perustuu kokemukseen ja päätöksentekoon tiedettyjen asioiden perusteella. Menetelmänä scrum koostuu tiimeistä rooleineen, prosessista, tapahtumista ja tuotoksista. Käsitteenä scrum tulee rugby-lajista, jossa scrum on tapa käynnistää peli uudelleen jonkin keskeytyksen jälkeen. Ohjelmistokehityksessä tämä tarkoittaa sitä, että tiimit kehittävät tuotetta iteratiivisesti eli pienissä osissa, prosessia toistaen ja keskeytyksen jälkeen uudelleen aloittaen, inkrementaalisti eli koko ajan loppua kohti kasvaen.

Scrum-prosessissa on viisi päätoimintoa: aloitus, sprintin suunnittelu, sprintti, päiväpalaveri ja sprintin katselmointi. Aloitustapaamisessa tiimi määrittelee karkeasti kehitettävän tuotteen koko projektia ja projektin suurimpia

tavoitteita silmällä pitäen. Sprintin suunnittelutapaaminen koostuu kahdesta osasta. Ensimmäisessä osassa tiimi määrittelee tuotteen kehitysjonon eli toisin sanoen projektin vaatimukset, ja sen jälkeen määritellään yksittäisen sprintin tavoitteet. Toisessa osassa keskitytään sprintin kehitysjonon laatimiseen. Seuraavassa kuvassa esiintyy scrumin prosessirunko, johon koko menetelmä pohjautuu.



KUVIO 2 Scrumin runko (Schwaber, 2004)

Alempi iteraatiota esittävä ympyrä edustaa kehitystoimintoja jotka esiintyvät yksi toisensa jälkeen. Näitä lyhyitä kehityksen vaiheita kutsutaan scrumissa myös sprinteiksi. Sprintit ovat enintään kuukauden mittaisia ajanjaksoja, jotka koostuvat suunnittelupalavereista, päiväpalavereista, kehitystyöstä, sprintin katselmoinnista ja yleiskatsauksesta. Jokaisessa sprintissä valmistunut osa lisätään varsinaiseen ohjelmistoon ja näin tuote kasvaa koko ajan valmistumiseen asti. Ylempi ympyrä taas edustaa päivittäistä tarkastelua sprintin sisällä. Jokapäiväisessä palaverissa käydään läpi, mitä kukin tiimin jäsen on tehnyt edellisen tapaamisen jälkeen, onko esteitä ilmennyt ja, mitä tulee tehdä ennen seuraavaa tapaamista. Sprintin katselmointi on ajankohtainen jokaisen sprintin lopussa. Tässä tapaamisessa esitetään tuotteen omistajalle sprintin aikana luotu toiminnallisuus (Cervone, 2011).

3.2 Scrumin käytänteet

Scrumtiimiin kuuluu tuoteomistaja, scrummaster ja kehitystiimi. Kaikki hallinnolliset vastuut projektin sisällä jakautuvat näiden kolmen roolin kesken. Tuoteomistaja päättää, mitä tehdään, kehitystiimi päättää, miten ja scrummaster tukee ja mahdollistaa muiden tekemisen (Yi, 2011). Tuoteomistaja on vastuussa tuotteen arvon maksimoinnista ja tuotteen kehitysjonon hallinnasta. Kehitysjonon hallinta tarkoittaa tuotteen tunnistamista ja vaatimuksien määrittelyä asiakkaan kanssa sekä vaatimusten priorisoimista valmiiksi kehitystiimille. Kehitystiimi rakentaa tuotteen tuoteomistajan priorisoidun vaatimuslistan perus-

teella. Kehitystiimi koostuu monialaisista osaajista sisältäen kaiken tarvittavan osaamisen tuotteen kehittämiseen. Tiimi johtaa itse itseään ja saa annetuissa rajoissa päättää, mitä kukakin tekee, missä ajassa ja miten (Deemer et al., 2012).

Scrummaster ei ole tiimin päällikkö, vaan palveleva johtaja. Tämä tarkoittaa sitä, että scrummaster palvelee niin tuotteenomistajaa, kehitystiimiä kuin koko organisaatiotakin (Schwaber, 2004). Scrummaster on siis palvelija ja sen lisäksi auttaja, valmentaja ja ongelmien ratkaisija. Scrummaster siis auttaa poistamaan esteitä, suojelee projektin ulkopuolelta tulevilta häiriöiltä sekä opastaa sopeutumaan uusiin kehityskäytänteisiin. Scrum-menetelmän kouluttamisen, valmentamisen ja opastamisen piiriin kuuluu kehitystiimin lisäksi tuoteomistaja sekä loput organisaatiosta (Deemer et al., 2012).

Scrummaster täyttää usein paikan, jota on hoitanut projektipäällikkö. Tämä perinteinen projektipäällikkö on ollut vastuussa määrittelystä ja työn hallinnasta, kun taas scrummaster on vastuussa scrum-menetelmän hallinnasta. Käytännössä scrummaster vastaa siis scrum-prosessin käytöstä ja siitä mitä siihen kuuluu ja mitä ei. Tämä scrum-menetelmän hallinta sisältää käytänteiden, tapaamisten ja termien määrittämisen. Scrummasterin tulee tietää nämä määritelmät sekä auttaa niiden käytäntöön viemisessä. Scrumin kehittäjä Schwaber kuvailee scrummasteria lammaspaimeneksi, joka pitää yksilöistä koostuvan hajoilevan lauman yhdessä sekä sudet poissa, toisin sanoen siis tiimin kasassa ja yhteistyökykyisenä samalla häiriötekijät poistaen (Sutherland & Schwaber, 2013).

Scrummaster on siis valmentaja, opettaja sekä paimen. Valmentaja tuntee käytetyt menetelmät, kannustaa tiimiläisiä kehittymään, on kiinnostunut projektin etenemisestä sekä pyrkii motivoimaan tiimiä. Opettaja vastaa scrum-menetelmän jalkauttamisesta organisaatioon ja huolehtii, että kaikki projektin osalliset ymmärtävät toimintakäytänteet ja noudattavat niitä. Paimen huolehtii, että kehitystiimi saa työskennellä omassa rauhassa, ja ettei työskentelyä häiritä ulkopuolelta. Scrummasterin tehtävänä ei ole kertoa, mitä tehdä, tai määrätä vastuualueita tiimiläisille vaan helpottaa prosessia tukien tiimiä organisoitumaan ja johtamaan itse itseään.

4 SIIRTYMINEN PERINTEISESTÄ KEHITTÄMISESTÄ KETTERÄÄN KEHITTÄMISEEN

Neljännessä luvussa perehdytään kehittämismenetelmien eroihin ja niistä aiheutuviin muutoksiin. Ensimmäisessä alaluvussa käsitellään yleisemmällä tasolla perinteisestä ketterään kehittämiseen siirtymistä, minkä jälkeen seuraavassa alaluvussa tutustutaan haasteisiin, mitä kohdataan vaihdoksessa. Kolmannessa alaluvussa syvennytään muutoksiin, mitä tapahtuu johtajien tasolla. Neljäs alaluku esittelee hyvän projektipäällikön ja scrummasterin kompetensseja ja pohtii niiden yhtäläisyyksiä ja eroja.

4.1 Kehittämistapojen erot ja niistä aiheutuvat muutokset

Perinteistä ohjelmistokehittämistä verrataan talon rakentamiseen, koska se vaatii huolellista suunnittelua ja muotoilua ennen kuin itse rakentaminen alkaa. Perinteiselle ohjelmistokehittämiselle tyypillistä on myös se, että asiakas on mukana suunnittelemassa tuotetta ja sitten se valmistetaan ja asiakas saa valmiin tuotteen - toisin sanoen talon ostaja suunnittelee talon ja muuttaa siihen vasta, kun se on valmis. Ketterää kehittämistä voidaan lähestyä myös samalla talonrakennus esimerkillä. Ketterän kehittämisen tapauksessa talo kuitenkin rakennetaan huone kerrallaan lisäten vedet ja sähköt ensin ensimmäiseen huoneeseen sitten seuraavaan vähän kerrallaan rakentaen. Asiakas voi muuttaa taloon heti kun kokee, että se on riittävän valmis ja lisäyksiä voi tehdä jatkuvasti koko kehityksen ajan asiakkaan tahdon mukaisesti (Schwaber, 2004).

Kehityttämistavan vaihtaminen vaikuttaa ihmisiin, prosessiin ja projektiin. Ihmisissä muutos vaikuttaa kehittäjiin, testaajiin, projektin johtajiin sekä asiakkaisiin. Vaikutus ilmenee kehittäjien kohdalla siten, että heiltä vaaditaan paljon. Heidän pitää olla joustavia, taitavia, asiantuntevia ja heidän tulee kommunikoida paljon. Heidän tulee myös pyrkiä toimimaan tiiminä, sillä siten ongelmien ratkaiseminen ja muutokseen reagoiminen on tehokkaampaa. Testaajiin muutos vaikuttaa siten, että testaamista tehdään koko projektin läpi, kehityksen tapahtuessa lyhyemmissä sykleissä. Projektin johtajat jaetaan useimmiten pro-

jektipäälliköihin ja tiimin vetäjiin. Molemmissa kohdataan omat haasteet siirryttäessä ketterään kehittämiseen. Tiimin vetäjien kohdalla muutos ilmenee siten, että kehitystiimin ollessa ammattitaitoa täynnä, johtaminen on lähempänä valmentamista ja mentorointia. Johtajuus tulee vuorovaikuttamisen ja muiden kehittämisen kautta. Organisaatiotason muutos päätösten tekemisessä ja päätöksenteon jakaminen ovat koko muutoksen merkittävimpiä seurauksia. Projektipäälliköiden sopeutuminen on vielä tiimin vetäjiä haastavampaa, sillä totutut aikataulutus- ja suunnitteluosiot eivät ole ketterässä kehityksessä yhtä oleellisia kuin ennen. Painotus on selkeästi muutokseen vastaamisessa tehdyn suunnitelman noudattamisen sijaan. Asiakkaalta vaaditaan osallistumista paljon enemmän kuin aiemmin. Osallistumisen lisäys tarkoittaa myös vaikutusvallan lisääntymistä, sillä yhteydenpidon ollessa säännöllisesti toistuvaa, kehitykseen pystyy vaikuttamaan myös jatkuvasti. Toisaalta asiakkaan vaikutusvallan lisääntyessä myös tiedollinen vastuu kasvaa, asiantuntemus loppukäyttäjistä sekä sitoutuminen koko projektiin on oltava korkealla tasolla (Coram & Bohner, 2005).

Prosessissa muutokset perustuvat kehitystapojen erilaisuuteen. Ketterälle kehitykselle tyypillisesti aikaa käytetään vähemmän formaaliin suunnitteluun, mutta koska koko projektin suunnittelu jaetaan pieniin tehtäviin, suunnitteluun käytetty kokonaisaika saattaa olla jopa suurempi. Itse kehitysprosessissa keskitytään ohjelmistokoodin paranteluun ja jatkuvaan yhtenäisyyteen. Ketterä kehitys soveltuu joihinkin projektityyppeihin ja liiketoimintamalleihin paremmin kuin toisiin. Esimerkkinä voi käyttää sopimuksellisia velvoitteita. Jos vaatimukset kirjataan sopimukseen ja niitä ei voi muuttaa kesken kehityksen, niin ketterä kehittäminen ei ole välttämättä soveltuvin kehitystapa (Coram & Bohner, 2005).

4.2 Haasteet siirryttäessä ketterään kehittämisen

Ketterien menetelmien ajattelutavan yhdistäminen perinteisen kehittämisen ympäristöön on vaikeaa. Organisaatio kohtaa monia muutoksia siirtyessä ketteriin menetelmiin. Organisaatiokulttuurissa säännöt ja niiden noudattaminen jätetään ja tilalle tulee kehityksen vapaus ja tiimin itseohjautuminen ja samalla hylätään yksilöidyt roolit ja työskennellään tiiminä. Kehityksessä tulee myös siirtyä tuotekeskeisyydestä kohti asiakaskeskeistä toimintatapaa. Johtamiseen liittyvät muutokset ilmenevät käskevästä ja hallitsevasta toimintatavasta siirryttäessä kohti vuorovaikutuksessa johtamista sekä päätöksenteon jakautumista. Itse kehitysprosessissa muutos tapahtuu raskaasta prosessikeskeisestä menetelmästä siirryttäessä kevyempään, ihmiskeskeiseen kehittämiseen (Subhas et al., 2006).

Miksi yritykset haluavat tehdä muutoksen, vaikka sen haastavuus on todettua? Syitä muutokseen on monia ja luonnollisesti toiminnan kehittäminen muun kehityksen mukana sekä aiemmat epäonnistuneet projektit vaativat muutosta, mutta ne eivät yksinään riitä. Muutosta perustellaan muun muassa seuraavilla syillä:

- Toimiva ohjelmisto on parempi edistymisen mittari kuin ansaittu arvo
- Huonot toiminnot pystytään tunnistamaan aiemmin ja siksi niihin ei käytetä turhaan aikaa
- Lyhyet syklit auttavat kiinnittämään huomion olennaisiin toimintoihin ja epäolennaisuudet huomataan aiemmin
- Ketterät menetelmät soveltuvat paremmin muuttuviin vaatimuksiin

Syyt muutokseen on oleellista huomioida, sillä haasteita kohdataan koko organisaation laajuudella. Ensimmäinen ja selkein haaste on kehitysprosessien ristiriitaisuus. Kun pyritään sulauttamaan ketterän kehittämisen ja perinteisen kehittämisen prosessit, on todennäköistä, että joko ketteryys tai määrittelytyö kärsii. Kehitykseen liittyviä haasteita kohdataan myös, kun tuotteen kehitystehtävät jaetaan, sekä kun tehtävien kestoa arvioidaan. Oma haasteensa on myös epävarmuuden sietäminen aika-arvioita tehdessä, sillä ketterä kehittäminenhän perustuu muutosten sietämiseen, kun taas perinteisessä kehityksessä on totuttu etenemään tarkasti suunnitelman mukaan.

Kehittämiseen liittyvien haasteiden lisäksi kohdataan liiketoimintaan liittyviä ristiriitoja sekä ihmisiin liittyviä konflikteja. Liiketoimintaan liittyvistä haasteista suurin on varmuuden ja epävarmuuden välinen tasapainoilu. Epävarmuus korostuu pitkän tähtäimen aika-arvioinneissa kun kehitys tapahtuu iteratiivisesti, pienissä osissa. Ihmisiin liittyvät haasteet ovat merkittäviä, sillä ellei ketteriä menetelmiä ja käytänteitä saada sopeutettua henkilöstön käytössä oleviin prosesseihin, ei muutos voi onnistua (Boehm & Turner 2005).

Eräässä tutkimuksessa yrityksen päälliköiltä kysyttiin, mikä oli suurin haaste kun scrum otettiin käyttöön. Kaksi päälliköistä kertoi, että vaikeinta oli vakuuttaa epäilevät ja muutosta vastustavat henkilöt. Kolmella muulla oli ollut haasteita pitää tuotteen kehitysjono ajan tasalla. Yksi päällikkö koki, ettei hän pysynyt mukana tiimin kehityksen vauhdissa. Kolmas ja muissakin yhteyksissä esiintynyt haaste oli uudet roolit ja niiden vastuualueet (Friis, Ostergaard & Sutherland, 2011).

4.3 Muutoksen vaikutus projektin johtamiseen

(Yi, 2011) toteaa, että projektipäällikkö ei voi toimia scrummasterina, sillä yleisesti ajatellaan, että käskevä ja hallitseva projektipäällikkö ei voi johtaa ja valmentaa niin kuin scrummasterin odotetaan tekevän. Roolien erilaisuutta perustellaan juurikin päätösvallasta luopumisen sekä itseohjautuvan tiimin kautta. Tässä alaluvussa tutkitaan, mitä yhtäläisyyksiä ja eroja johtajien ominaisuuksissa on ja mihin haasteisiin törmätään eroista johtuen.

Kun ohjelmistokehitysprojektit siirtyvät ketterään kehitykseen, projektipäälliköt korvataan usein uusilla henkilöillä. Syynä tähän on se, että perinteiset projektipäälliköt menevät sekaisin uusista rooleista ja vastuista ja yksin päätöksiä tekeville päälliköille ei ole enää tarvetta tiimien siirtyessä itseään johtaviksi.

Projektipäällikölle suurin muutos on palvelujohtamisen oppiminen ja sen käytäntöön paneminen. Se ilmenee muun muassa tiimin tarpeiden priorisoinnissa, yksilöiden kehityksessä sekä kontrollista luopumisessa (Sliger, 2006).

Yi (2011) tutki käytännössä, kuinka siirtyminen scrumiin tapahtui eräässä yrityksessä ja hän totesi, että suurin haaste oli saada projektipäälliköt ymmärtämään roolien muutos käskevästä ja hallitsevasta johtavaan ja valmentavaan rooliin. Muita muutoksia oli keskitettyjen tiimien uudelleen muodostaminen siten, että osaaminen jakautui monipuolisesti eri tiimeihin. Scrummaster oli täysin uusi rooli ja sen täyttämiseksi ei ollut ennestään työntekijöitä. Itseohjautuvuuden edesauttamiseksi työntekijät kasattiin samaan tilaan ja he saivat muodostaa itse tiiminsä. Samasta syystä scrummasterin valinta olisi voitu suorittaa tiimin kesken, mutta koska scrum oli menetelmänä vielä suurimmalle osalle täysin tuntematon, valinta suoritettiin organisaation toimesta. Muutos ei sujunut haasteitta tässäkään yrityksessä.

4.4 Suositeltavia projektipäällikön ja scrummasterin ominaisuuksia

Projektipäällikkö ja scrummaster vaikuttavat varsin samanlaisilta työnimikkeiltä, joten on oleellista vertailla kuinka paljon samoja ominaisuuksia menestyvillä projektipäälliköillä ja scrummastereilla on. Tätä kautta on mielestäni helpompi hahmottaa, miksi roolista toiseen siirtyminen voisi olla myös mahdollista haasteista huolimatta. Hyvän projektipäällikön ja scrummasterin vertailun mahdollistamiseksi seuraavaksi käsitellään molempien roolien tunnuksenomaisia piirteitä ja menestyksen mahdollistavia kompetensseja. Kompetenssit ovat (Liikamaa, 2006) mukaan yksilön ominaisuuksia, jotka kuvaavat työntekijän tapaa toimia erilaisissa työtilanteissa. Seuraavassa taulukossa on listattu kompetenssit, joiden omaaminen erottaa erinomaiset projektipäälliköt hyvistä (Cooke-Davies, Crawford & Lechler, 2009). Scrummasterin kompetenssit on koottu scrumin luojien, Schwaberin ja Sutherlandin (2013) oppaan pohjalta.

Taulukko 1 Projektipäällikön ja scrummasterin kompetenssit (Cooke-Davies, Crawford & Lechler, 2009),(Schwaber & Sutherland, 2013)

Projektipäällikön kompetenssit	**Scrummasterin kompetenssit**
Projektin päätarpeiden tunnistaminen	Scrum-prosessin hallitseminen
Projektipäällikön roolin ymmärtäminen	Palvelevan roolin ymmärtäminen
Lopputuotteen määrittäminen	Koko tiimin osallistuminen suunnitteluun
Onnistumisen suunnittelu	Tehokas muiden työn tekemisen helpottaminen
Toimintojen järjestäminen	Selkeän suunnan näyttäminen

Budjetoiminen	
Esiintymistaidot	
Tiimin kokoaminen	Itseohjautuvan tiimin perustaminen
Tiimin kehittäminen	Tiimin jäsenten kehittäminen
Projektinjohtamistaidot	
Tiimin motivoiminen	Yksilöiden motivoiminen
Viestintätaidot	Ulkoinen ja sisäinen viestintä
Tehokas raportoiminen	Ajantasainen raportoiminen
Suorituksen hallitseminen	Projektin tilan hallitseminen
Konflikteihin vaikuttaminen niiden ja ratkaiseminen	
Neuvottelutaidot	
Ajanhallinta	Projektin aikataulun hallinta
Tapaamisten järjestäminen	
Ongelmien ratkaiseminen	Esteiden poistaminen
Henkilökohtaiseen kehittymiseen sitoutuminen	Jatkuva oppimisen arvioiminen

Molemmat roolit edellyttävät varsin paljon samoja tai samantyyppisiä kompetensseja menestyksekkään johtamisen yhteydessä. Kahdestakymmenestä kompetenssista puolet, eli kymmenen kompetenssia on käytännössä samantyyppisiä. Etenkin tiimin ja yksilöiden oppimiseen ja kehittämiseen liittyvät taidot vastaavat toisiaan. Suurin ero tulee suunnitteluun ja päätöksentekoon liittyvissä ominaisuuksissa.

Projektityöskentelyssä ensiarvoisen tärkeää on että tieto kulkee tiimin jäsenten ja eri sidosryhmien välillä tehokkaasti, joten viestintätaidot ja raportointi näyttelee oleellista osaa projektin onnistumisessa. Projektin päätarpeiden tunnistaminen tarkoittaa sitä, että projektipäällikkö pyrkii tunnistamaan asiakkaan tarpeet projektin alussa ja niiden avulla määrittämään lopputuotteen. Suunnittelun jälkeen on myös tärkeää kirkastaa tavoitteita tiimille läpi koko projektin, jotta haluttu lopputulos saavutetaan. Hänen täytyy myös ymmärtää oma roolinsa, vastuunsa ja tehtävänsä, jotta myös tiimi tietää kuka vastaa mistäkin osiosta. Onnistumisen suunnittelulla tarkoitetaan läpiviennin suunnittelua, jota toteuttamalla projektin eteneminen ja lopputulos ovat toivottuja. Toimintojen järjestäminen on osa projektipäällikön käytännöllisempiä tehtäviä, johon kuuluu tiimin käytännön toiminnan vaatimat järjestelyt. Käytännön tehtäviin ja ominaisuuksiin kuuluvat luonnollisesti myös projektia rajoittavista tekijöistä huolehtiminen kuten budjetointi ja aikataulutus. Käytännön järjestelyt vaativat projektipäälliköltä taitoja kuten tapaamisten järjestäminen sekä niissä esiintyminen ja neuvotteleminen. Projektityöskentely vaatii omanlaista johtamista ja tärkeässä roolissa ovat tiimin kokoamisen, tiimin kehittämisen ja tiimin motivoimisen taidot sekä tiimin jäsenten henkilökohtaisen kehittämisen avustaminen. Näiden lisäksi hyödyllisiä taitoja ovat ongelmien ratkaisukyky sekä konflikteihin vaikuttaminen ja niiden ratkaisu (Cooke-Davies, Crawford & Lechler, 2009).

Scrummasterin päätehtävänä on hallita scrum-prosessin käytänteitä ja opastaa projektin osapuolille työskentelytavan perusteet. Käytänteisiin kuuluvat esimerkiksi koko tiimin osallistaminen ja tiimin itseohjautuvuus. Scrummasterin tehtävänä on näyttää suuntaa ja tarkkailla työskentelyä sekä varmistaa viestinnän ja raportoinnin kautta, että asiakas ja tiimi ovat ymmärtäneet tavoitteet ja tarpeet samalla lailla. Hänen tulee ymmärtää myös oma roolinsa ja siitä seuraavat tehtävät palvelevana johtajana. Roolista aiheutuvia tehtäviä ovat esimerkiksi työskentelyyn liittyvien esteiden poistaminen ja muiden työn tekemisen helpottaminen. Häneltä odotetaan myös tiimin jäsenten kehittämistä sekä tämän kehityksen ja oppimisen jatkuva arviointia, minkä avulla virheistä opitaan eikä toisteta samoja ratkaisuja uudelleen.

Perinteisessä kehittämisessä projektipäällikkö on vastuussa tuotteen määrittämisestä ja suunnittelusta pääosin itse, scrumissa taas koko tiimi on jakamassa vastuuta. Selkeitä eroja on myös projektin johdon rooleissa. Projektipäällikön tulee ymmärtää oman projektinsa vastuut tarkasti, kun taas scrummaster keskittyy palvelemaan muita kehityksen osapuolia. Viimeinen ja kenties merkittävin ero on projektin johtajien päävastuualueessa, sillä projektipäällikkö pyrkii tunnistamaan projektin päätarpeet ja vaatimukset, kun taas scrummaster pyrkii hallitsemaan scrum-prosessin mahdollisimman hyvin. Projektipäällikön roolin ja ajattelutavan muuttaminen ei ole helppoa, mutta lopulta muuntautuminen scrummasteriksi vaatii samoja asioita kuin mikä tahansa uuden työtehtävän omaksuminen: itseopiskelua, kokemusta, harjoittelemista ja ulkopuolista ohjausta sekä aikaa tehdä tämä kaikki (Sliger, 2006).

5 JOHTOPÄÄTÖKSET

Ensimmäinen tutkimuskysymys herättelee aiheeseen ja kysyy miten ohjelmistokehittäminen on muuttunut? Aihetta lähestytään ohjelmistokehityksen muutoksella perinteisestä kohti ketterää kehittämistä. Liiketoiminnassa ja sitä kautta myös ohjelmistokehityksessä on lisääntynyt selkeästi tarve nopeampaan muutokseen reagointiin. Tästä seuraa ketterien kehitysmenetelmien käytön selkeä kasvu suhteessa perinteiseen ohjelmistokehittämiseen. Perinteistä ohjelmistokehittämistä tarkastellaan vesiputousmallin avulla ja ketteristä menetelmistä tutustutaan scrumiin. On tärkeää kuitenkin huomioida että syvennyttäessä vesiputousmalliin ja scrum-prosessiin, käytännön ohjelmistokehityksessä yritykset muovaavat menetelmät heidän liiketoimintaansa sopivaksi. Toisin sanoen täysin teoreettisesti puhtaassa muodossa olevaa scrumia tai vesiputousmallia ei ole käytössä, mutta niiden avulla on mahdollista selvittää muutoksen pääpiirteitä ja haasteita.

Ohjelmistokehityksen muutos näkyy selkeästi muutokseen vastaamisessa tehdyn suunnitelman noudattamisen sijaan. Myöskin asiakaskeskeisyys on oleellisempaa ja asiakkaalta vaaditaan osallistumista paljon enemmän kuin perinteisessä kehittämisessä. Käytännön tasolla nämä muutokset näkyvät lyhyempänä tarkasteluvälinä eli 2-4 viikon mittaisina sprintteinä ja asiakaskeskeisyys ilmenee myöskin palavereissa, joita pyritään pitämään useammin. Osa asiakaskeskeisyyttä on myös mahdollisuus muutoksiin kesken projektin, koska ohjelmisto rakentuu vähän kerrallaan ja tarvittavia muutoksia pystytään toteuttamaan vaivattomammin.

Toisen tutkimuskysymyksen kautta pohditaan, mitä haasteita ohjelmistokehittämisen muutoksessa kohdataan ja miten muutos vaikuttaa projektien johtamiseen? Yksi yleisimmistä haasteista on yrityksen valmius muutokseen. Usein siirtymisessä jää huomioimatta se, että yritystä ja organisaatiota ei ole valmisteltu muutokseen ja ketterän kehityksen ja scrumin kannalta oleelliset periaatteet ovat vielä omaksumatta. Jos organisaatiotasolla siirtymiseen ollaan valmiita, seuraavat haasteet kohdataan työntekijätasolla. Perinteisessä kehittämisessä päälliköt muodostavat tiimit ja määräävät työtehtävät, kun taas ketterässä kehittämisessä tiimit muodostuvat usein itse ja toimivat itseohjautuvasti. Perinteisen projektipäällikön tehtävät jakautuvat scrumissa kaikille osapuolille,

joten tarvetta vastaavalle työnimikkeelle ei ole. Scrummaster keskittyy työssään palvelemaan muita, helpottamaan heidän työskentelyä ja auttamaan heitä kehittymään. Eli projektipäällikön roolia ei samanlaisilla vastuilla tai tehtävillä ole scrumissa, mutta toisaalta projektipäällikön osaamisella pystyy työskentelemään myös ketterän kehittämisen parissa. Roolin puuttumisen perusteella on ymmärrettävää, että monet tutkijat eivät suosittele projektipäälliköiden siirtymistä ketterään kehittämiseen tai ainakaan scrummasterin tehtäviin. Toisaalta rooleissa on myös hyvin paljon samaa, ja kynnyskysymykseksi muodostuukin se, pystyykö päätösvaltaan tottunut projektipäällikkö luopumaan auktoriteetistaan ja onko yrityksen johto, kehitystiimi sekä organisaatio ylipäätään sisäistänyt ketteryyden perusperiaatteet. Näiden tekijöiden ollessa kunnossa, uskon että projektipäälliköstä voi tulla menestyvä scrummaster.

6 YHTEENVETO

Tässä tutkielmassa on selvitetty kirjallisuuskatsauksen avulla, miten perinteinen ohjelmistokehittäminen ja ketterä kehittäminen eroavat toisistaan ja esimerkkimenetelmien vesiputousmallin ja scrumin avulla tarkastelua on syvennetty myös johtamisen tasolle. Tutkielmassa on selvitetty mitä erottavia tekijöitä on perinteisen ohjelmistokehittämisen projektipäälliköillä ja ketterän kehittämisen scrummastereilla sekä millaisiin haasteisiin on törmätty organisaatioiden siirtyessä perinteisistä menetelmistä ketteriin menetelmiin. Selkeimpänä erona kehittämistavoissa on havaittu ajan käyttäminen ja tuotteen prosessointi. Perinteisessä menetelmässä edetään suunnitelupainotteisesti ja vaihe kerrallaan, kun taas ketterässä kehittämisessä suunnitellaan vaihe kerrallaan, ja aina vaiheen jälkeen tarkastellaan tilannetta ja tehdään tarvittavat suunnanmuutokset.

Ohjelmistokehityksesmenetelmän vaihdoksesta aiheutuvat haasteet ilmenevät organisaation, johtamisen ja työntekijän yksilöllisellä tasolla. Organisaatiotasolla ketteryyden perusperiaatteiden ymmärryksen puute tekee kehitystavan vaihdosta vaikeaa. Vaikeuksia aiheutuu myös jos yksilötasolla ketterien menetelmien käytänteitä ei saada sopeutettua henkilöstön jokapäiväisiin rutiineihin. Johtamisen tasolla suurimmat haasteet tulevat roolien ja vastuiden sekoittumisesta. Käytänössä tämä ilmenee päätöksenteossa, sillä perinteisen kehittämisen projektipäällikön päätösvalta jakautuu ketterässä kehittämisessä tiimin kesken. Tästä erosta tulee haaste organisaation siirtyessä perinteisestä ketterään kehittämiseen, sillä projektipäällikön rooli ikään kuin jakautuu tiimin eri roolien kesken.

LÄHTEET

Aitken, A., & Ilango, V. (2013, January). A comparative analysis of traditional software engineering and agile software development. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on* (pp. 4751-4760). IEEE.

Awad, M. A. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*.

Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), 26-30.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development.

Leau, Y. B., Loo, W. K., Tham, W. Y., & Tan, S. F. (2012). Software development life cycle AGILE vs traditional approaches. In *International Conference on Information and Network Technology* (Vol. 37, No. 1, pp. 162-167).

Boehm, B., & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. *Software, ieee*, *22*(5), 30-39.

Cervone, H. F. (2011). Understanding agile project management methods using Scrum. *OCLC Systems & Services: International digital library perspectives*,*27*(1), 18-22.

Cooke-Davies, T. J., Crawford, L. H., & Lechler, T. G. (2009). Project management systems: Moving project management from an operational to a strategic discipline. *Project Management Journal*, *40*(1), 110-123.

Coram, M., & Bohner, S. (2005, April). The impact of agile methods on software project management. In *Engineering of Computer-Based Systems, 2005. ECBS'05. 12th IEEE International Conference and Workshops on the* (pp. 363-370). IEEE.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2012). The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum. *Retrieved August*, *3*, 2014.

Eskelinen, A. (2014) Suomenkielinen scrum-sanasto (LIITE). Haettu 28.9.2015 osoitteesta http://lekman.fi/scrumguide/

Friis, D., Ostergaard, J., & Sutherland, J. (2011, January). Virtual Reality Meets Scrum: How a Senior Team Moved from Management to Leadership. In*System Sciences (HICSS), 2011 44th Hawaii International Conference on* (pp. 1-7). IEEE.

Guide, A. (2001). Project Management Body of Knowledge (PMBOK® GUIDE). In *Project Management Institute*.

Hass, K. B. (2007). The blending of traditional and agile project management.*PM world today*, *9*(5), 1-8.

Liikamaa, K. (2006). *Piilevä tieto ja projektipäällikön kompetenssit*. Tampereen teknillinen yliopisto.

Rosenau, M. D., & Githens, G. D. (2011). *Successful project management: a step-by-step approach with practical examples*. John Wiley & Sons.

Royce, W. W. (1970, August). Managing the development of large software systems. In *proceedings of IEEE WESCON* (26(8), 328-388).

Sliger, M. (2006). A project manager's survival guide to going agile.

Misra, S. C., Kumar, U., Kumar, V., & Grant, G. (2006, December). The organizational changes required and the challenges involved in adopting agile methodologies in traditional software development organizations. In *Digital Information Management, 2006 1st International Conference on* (pp. 25-28). IEEE.

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft Press.

Schwaber, K., & Sutherland, J. (2011). The scrum guide. *Scrum Alliance*.

Takeuchi, H., & Nonaka, I. (1986). The new new product development game.*Harvard business review*, *64*(1), 137-146.

Yi, L. (2011, August). Manager as Scrum Master. In *Agile Conference (AGILE), 2011* (pp. 151-153). IEEE.