jamk.fi

Tuotekehityksen trendit WIMMA Lab -kontekstissa

Case: WIMMA Lab 2018

Minttu Mäkäläinen

Opinnäytetyö
Syyskuu 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), Tieto- ja viestintätekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Mäkäläinen, Minttu	Opinnäytetyö, AMK	Syyskuu 2018
	Sivumäärä 58	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Tuotekehityksen trendit WIMMA Lab -kontekstissa
Case: WIMMA Lab 2018

Tutkinto-ohjelma
Tieto- ja viestintätekniikan tutkinto-ohjelma, Ohjelmistotekniikka

Työn ohjaaja(t)
Kari Niemi

Toimeksiantaja(t)
Jyväskylän ammattikorkeakoulu, Marko Rintamäki

Tiivistelmä

Opinnäytetyö toteutettiin Jyväskylän ammattikorkeakoulun IT-instituutin tilauksesta, WIMMA Lab -opintokokonaisuuden kehittämiseksi. Tavoitteena oli tutkia nykypäivän tuotekehityksen trendejä ja selvittää, onko WIMMA Labin konsepti ajan tasalla. Lisäksi tuli ottaa selvää niistä uusista metodeista, joita WIMMA Labissa voisi tulevaisuudessa käyttää.

Tietoperustassa perehdytään erilaisiin ketteriin menetelmiin, joita tuotekehityksessä nykypäivänä käytetään. Lisäksi tutustutaan tiimin toiminnan kehittämiseen, koska ketterät menetelmät pohjautuvat hyvin vahvasti tehokkaan tiimin itsenäiseen toimintaan. Käytännön osuudessa käydään läpi WIMMA Lab -opintokokonaisuutta hakijoiden valinnasta aina WIMMA Labin päättymiseen saakka. Analyysissä katsotaan, mitä ketteriä menetelmiä ja niiden mukanaan tuomia työkaluja WIMMA Labissa jo nykyisellään käytetään.

Opinnäytetyössä löydettiin vastaukset esitettyihin kysymyksiin ja tuloksena on, että WIMMA Labissa käytetään onnistuneesti useita erilaisia ketteriä menetelmiä ja niiden työkaluja. Ketteryyden todellinen merkitys jää kuitenkin opintokokonaisuudelle osallistuvilta opiskelijoilta sisäistämättä, joten siihen tulisi panostaa.

Tulevissa WIMMA Labin toteutuksissa kannattaisi nostaa enemmän esille Mob Programmingia sekä DevOps -tuotekehitysmallin tyyppistä työskentelytapaa. Lisäksi tiimin kommunikointia asiakkaan kanssa tulisi vahvistaa, sillä käyttäjien tyytyväisyys on nykypäivänä yksi tärkeimmistä menestyksen mittareista.

Avainsanat (asiasanat)
Agile, ketterät menetelmät, tuotekehitys, WIMMA Lab, Lean, Kanban, Scrum, Six Sigma, Extreme Programming, DevOps

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Mäkäläinen, Minttu	Bachelor's thesis	September 2018
	Number of pages 58	Language of publication: Finnish
		Permission for web publication: x

Title of publication

Trends of Product Development in WIMMA Lab Context

Case: WIMMA Lab 2018

Degree programme

Information and Communications Technology, Software Engineering

Supervisor(s)

Niemi Kari

Assigned by

JAMK University of Applied Sciences, Rintamäki Marko

Description

The thesis was carried out at the Jyväskylä University of Applied Sciences, and it was assigned by IT Institute to further develop the WIMMA Lab study program. The aim was to investigate the trends in today's product development and to find out whether the concept of WIMMA LAB is up to date. In addition, the new methods WIMMA Lab could use in the future implementations were also to be explored.

The information base familiarizes the reader with the various agile methods used in product development today. In addition, the thesis acquaints the reader with team development, since agile methods are based on the team's ability to work independently and effectively. In the practical part of the thesis, WIMMA Lab is examined from the selection of applicants to the end of WIMMA Lab. The analysis section inspects what agile methods and tools are already used in WIMMA Lab.

The thesis found answers to the questions presented. As a result it can be said, that WIMMA Lab successfully uses a variety of agile methods and tools. However, for the participating students the true meaning of agility remains obscure. Hence, in the future it should be invested in more.

Future WIMMA LAB implementations could use the techniques used in Mob Programming and DevOps product development model. In addition, team communication with the customer should be strengthened, as user satisfaction is one of the most important indicators of success today.

Keywords (subjects)

Agile, product development, WIMMA Lab, Lean, Kanban, Scrum, Six Sigma, Extreme Programming, DevOps

Miscellanous

Sisältö

Kuviot

Käytetyt termit ja lyhenteet

Asiakas

Voi tarkoittaa tuotteen tilaajaa, loppukäyttäjää tai seuraavaa prosessia.

DoD

Definition of Done, valmiin määritelmä.

Haavoittuvuus

Aukko tietoturvassa.

Hukka

Kaikki se, mikä ei lisää arvoa tuotteeseen asiakkaan näkökulmasta.

JIT

Just-in-time, juuri oikeaan aikaan, tuotetta valmistetaan vain todellisen tarpeen mukaan.

Loppukäyttäjä

Se henkilö, joka lopulta käyttää tuotetta tai palvelua.

Projektitiimi

Noin 5-10 hengen työryhmä.

Pullonkaula

Prosessin se vaihe, joka rajoittaa tekemisen määrää.

Sprintti

1-4 viikon mittainen kehitysjakso**.**

Tietoturvatestit

Testejä, joilla testataan sovelluksen turvallisuutta.

Vaatimusmäärittely

Dokumentti, jossa kuvataan projektin tavoitteet ja vaatimukset.

Vesiputousmalli

Ohjelmistokehityksen vanhempi malli, jossa edetään järjestelmällisesti yhdestä vaiheesta seuraavaan eikä aikaisempaan vaiheeseen enää palata.

1 Työn lähtökohdat

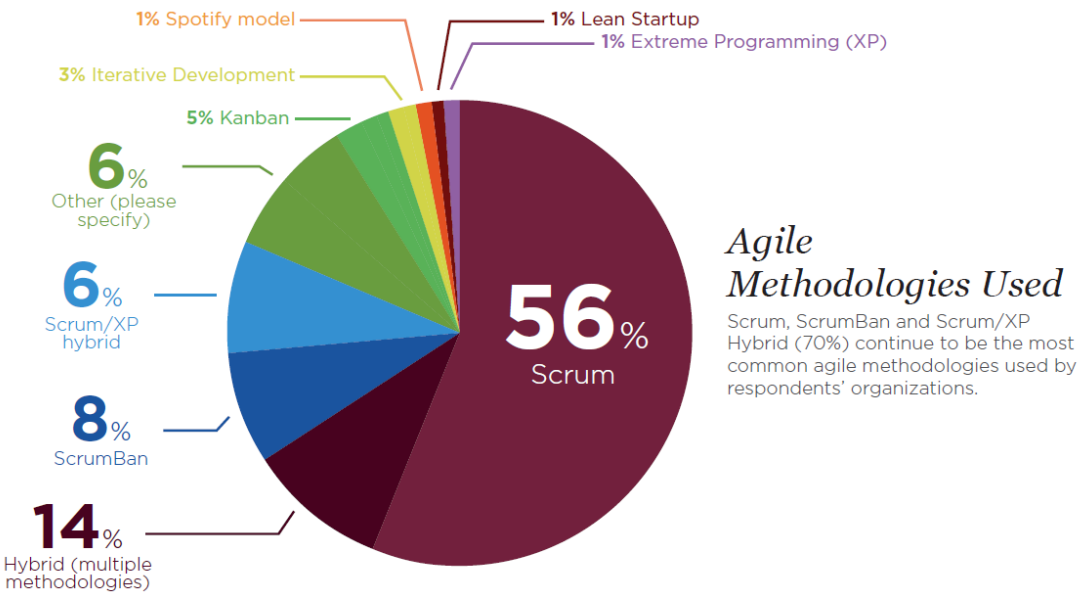
1.1 Tausta

Miltei kolmenkymmenen vuoden aikana ketterät menetelmät ovat vallanneet alaa perinteisemmiltä ohjelmistotuotannon tavoilta, kuten esimerkiksi vesiputousmallilta. Nykyaikana ohjelmistojen valmistumista ei voida odottaa pitkiä aikoja, sillä markkinat ja tarpeet muuttuvat vikkelästi. Hämäläisen (2016) mukaan tulevina vuosina ketterien tiimien ohjaukseen soveltuvien mallien käyttöönotto tulee jatkumaan hyvin vahvana ja ohjelmistot tulevat kiinteästi kaikille toimialoille vahvistamaan yritysten markkina-asemia. Yritykset alkavat enenevissä määrin kehittämään sisäistä ohjelmistokehityksen osaamistaan ja dokumentaation painoarvo tulee nousemaan paremman jäljitettävyyden saavuttamiseksi. Ketteryyden todellinen merkitys ja periaatteet tulee ymmärtää, jotta tiimit voivat kutsua itseään ketteriksi.

Shanholtzerin (2018) mukaan suurimpia ketteryyden trendejä vuonna 2018 ovat mm.

- Jatkuva testaus, eli CI-ketjut, virtuaaliset testiympäristöt ja koodin laadun parantaminen.
- Tiimin toimiminen yhdessä tuottaakseen asiakkaalle arvoa.
- Tuotteen kehittäminen oikeaan suuntaan nopeasti saatavan palautteen perusteella.
- Mobbaus (Mob Programming), eli ryhmä työskentelee yhdessä, rooleista riippumatta saavuttaakseen korkeamman sitoutumisen tuotteeseen.

12. vuosittaisen ketteryystutkimuksen mukaan ketterien menetelmien käyttöönotto kasvaa koko ajan ja yksi tärkeimmistä menestyksen mittareista on käyttäjien tyytyväisyys. Kun sovelluksen pitäisi olla myös laadukas ja valmistua nopeasti, erityisesti DevOps tulee jatkossa yleistymään. Scrum ja sen sovellukset ovat nykypäivänä käytetyimpiä ketteriä menetelmiä (ks. kuvio 1). (The 12th Annual State of Agile Report 2018.)



Kuvio 1. Käytetyimpiä ketteriä menetelmiä vuonna 2018 (The 12th Annual State of Agile Report 2018)

Suurimpia haasteita ketterien menetelmien käyttöönotossa ovat organisaation arvojen ristiriitaisuus ketterien arvojen kanssa, yleinen muutosvastaisuus sekä riittämätön johdon tuki. Kaikkein hyödyllisimmiksi taas on koettu valmentajat, johdonmukaiset käytännöt ja prosessit sekä yhteiset työkalut. (The 12th Annual State of Agile Report 2018.)

Ketteristä ohjelmistotuotannon menetelmistä on useita erilaisia näkemyksiä, eikä kirjallisuudessa ole tarkkaan määritetty, milloin organisaatio voi sanoa olevansa ketterä (Laanti, Similä & Abrahamsson 2013, luku 6). Puhuessaan ketteryydestä, ihmiset usein tarkoittavat eri asioita ja ketterien menetelmien käyttöönotto onkin erittäin vaikeaa (Laanti ym. 2013, luku 7).

Opinnäytetyön aihe valikoitui Jyväskylän ammattikorkeakoulussa (JAMK) järjestettävän WIMMA Labin kautta. WIMMA Lab on monialainen projektioppimisympäristö, joka on pyörinyt kirjoitushetkellä menestyksekkäästi jo kahdeksana peräkkäisenä kesänä ja kehittänyt vuosittain toimintaansa. Mäkäläinen sai mahdollisuuden päästä vetämään WIMMA Labia valmentajan roolissa, joka tarjosi erinomaisen mahdollisuuden kokeilla erilaisia metodeja ja työkaluja käytännössä.

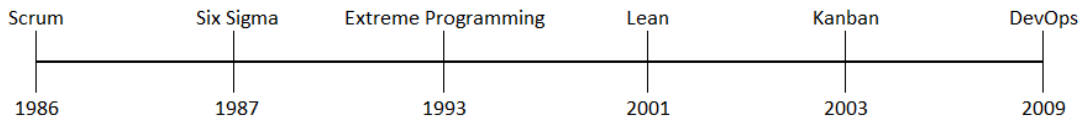
JAMK:ssa on useita erilaisia projektimallisia kursseja käynnissä, mutta mikään niistä ei ole vastaavanlainen, kuin WIMMA Lab.

Kesän 2018 kokemuksien perusteella Mäkäläinen kirjoitti WIMMA Lab Black Book 1.0:n. Opaskirjaan on koottu hyväksi todettuja harjoituksia ja vinkkejä, joita on testattu oikeissa olosuhteissa. WIMMA Labin avulla muovataan tulevaisuuden tekijöitä ohjelmistoalalle ja opetetaan jo alusta alkaen ketterien menetelmien käyttöä. (Mäkäläinen 2018, 6-7.) Opaskirja auttaa tulevien vuosien WIMMA Labin toteuttamisessa ja konseptin kehittämisessä.

Myös muistakin korkeakouluista on osoitettu kiinnostusta WIMMA Labin toimintaan. Opiskelijat hyötyvät laajemmista, työelämän toimintatapoja jäljittelevistä projekteista tavattoman paljon.

Ketteriin ohjelmistotuotannon menetelmiin liittyviä tutkimuksia on kirjoitettu lukuisia, mutta ne on kohdistettu yleensä eri yrityksiin tai organisaatioihin. Niitä ei voi sellaisenaan soveltaa WIMMA Labiin, sillä tämän kaltaiset tutkimukset ovat aina tapauskohtaisia.

Ketteristä menetelmistä löytyy kirjallisuutta runsain määrin ja nykyään on saatavilla satoja erilaisia työkaluja eri käyttötarkoituksiin. Kaikkia on mahdoton käydä läpi opinnäytetyön puitteissa, joten tarkastelun kohteeksi valittiin Lean, Kanban, Scrum, Six Sigma, Extreme Programming ja DevOps. Nämä termit yleistyivät kuvion 2 esittämän aikajanan mukaisesti (Six Sigman kehitysvaiheet n.d; Agile Practices Timeline n.d.).



Kuvio 2. Valitut tarkastelukohteet aikajanalla

1.2 Toimeksiantaja

Opinnäytetyön tilaajana toimi Jyväskylän ammattikorkeakoulun IT-instituutti. Aineiston kerääminen toteutettiin WIMMA Lab -opintokokonaisuuden puitteissa kesällä 2018.

WIMMA Lab on 15 opintopisteen kokoinen opintokokonaisuus, jossa opiskelija osallistuu järjestelmäprojektiin ja työskentelee projektiryhmän jäsenenä. Eri projektiryhmät ratkaisevat yritysten tai yhteisöjen tuomia toimeksiantoja ja oppivat näin käytännön kautta nykyaikaisen tuotekehityksen menetelmiä ja toimintatapoja. (Opintojakson kuvaus 2016.) Opintokokonaisuus sai alkunsa vuonna 2011 nimellä Summer Factory, tämän jälkeen nimi muuttui Challenge Factoryksi vuonna 2014. Vuodesta 2017 lähtien nimi on ollut WIMMA Lab. Marko Rintamäki oli yksi kurssia suunnitelleista päätekijöistä ja hän toimii edelleen kurssin vetäjänä.

1.3 Tehtävä ja tavoitteet

Opinnäytetyön tavoitteena oli kartoittaa nykypäivän ohjelmistokehityksen trendejä ja vertailla niitä. Tarkoituksena oli löytää vastaukset seuraaviin kysymyksiin:

1. Miten hyvin WIMMA Lab seuraa aikaansa?
2. Millaisia ovat ohjelmistokehityksen trendit nyt ja millaisia uusia metodeja on tullut tarjolle?
3. Mitä näistä olisi mahdollista kokeilla tulevissa WIMMA Lab -toteutuksissa?

1.4 Työssä käytetyt menetelmät

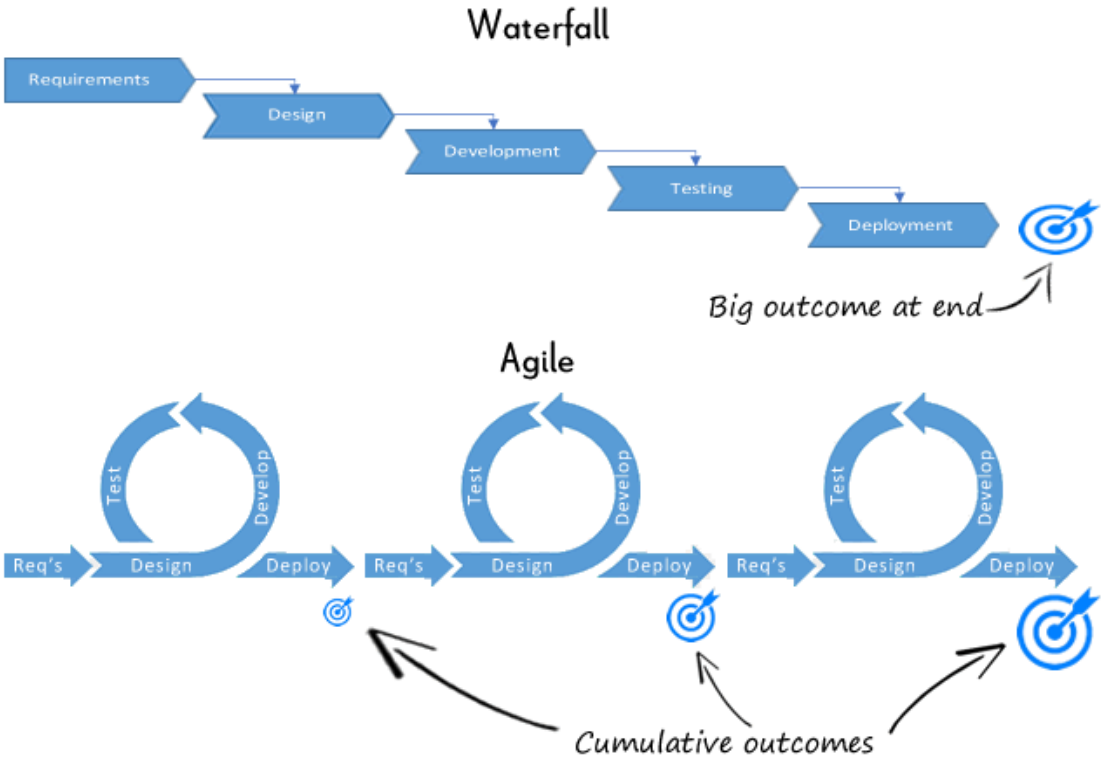
Mäkäläinen toimi WIMMA Labin valmentajana kahden ja puolen kuukauden ajan kesällä 2018. Tapaustutkimus on toteutettu keräämällä aineistoa jokapäiväisen toiminnan myötä ja erilaisien harjoituksien kautta, joita WIMMA Labissa pystyttiin suorittamaan.

Teoriaosuuteen aineisto hankittiin tutustumalla laajasti ohjelmistotuotannon eri tapoihin niin kirjallisuuden, verkkosivujen kuin erilaisten ketterien työkalujenkin kautta.

2 Ketterä ohjelmistokehitys

2.1 Mitä on ketterä ohjelmistokehitys?

Ketterien menetelmien tavoitteena on parantaa tuotteiden laatua sekä nopeuttaa tuotteiden julkaisemista verrattuna perinteisempiin ohjelmistokehityksen menetelmiin, kuten esimerkiksi vesiputousmenetelmään (ks. kuvio 3). Ketterät menetelmät mukautuvat joustavasti muutoksiin ja ongelmiin niiden ilmentyessä, mutta projektin lopputulos on kuitenkin aina vähän epävarmempi. (Altman 2017a, Chapter 1: What is agile project management?)



Kuvio 3. Ketterä kehitys vs. perinteinen vesiputousmalli (Schaeffer n.d.)

Ketterät menetelmät panostavat tiimin yhteistyöhön sekä sen itseohjautuvuuteen. Kaikilla tiimin jäsenillä on yhtä suuri vastuu työn onnistumisesta. Säännöllisillä tiimin tapaamisilla varmistetaan, että kaikki pysyvät mukana työn tilanteesta sekä pidetään silmällä, että kaikki työskentelevät tehokkaasti. Tiimin jäsenet jakavat työt itsenäisesti tiimiläisten taitojen perusteella, joten projektimanagerille ei ole tarvetta. Ideaalitilanteessa tiimin jäsenet työskentelevät samassa toimistossa, joka luo mahdollisuuden tiiviimmille ihmissuhteille. Tiimi tulisi alusta alkaen rakentaa ketterät menetelmät silmällä pitäen; tiimin tulisi olla energinen, innovatiivinen ja muutoksiin kykeneväinen. Jokainen tiimin jäsen pitäisi ottaa mukaan kaikkiin suunnittelun vaiheisiin, jotta varmistetaan aikataulussa pysyminen ja voidaan tunnistaa mahdolliset ongelmat. (Altman 2017a, Chapter 1: What is agile project management?)

Ketteriin menetelmiin kuuluu hyvin olennaisesti myös jatkuva kommunikointi asiakkaan kanssa, joka mahdollistaa paremman ymmärryksen valmistettavaan tuotteeseen (Altman 2017a, Chapter 1: What is agile project management?).

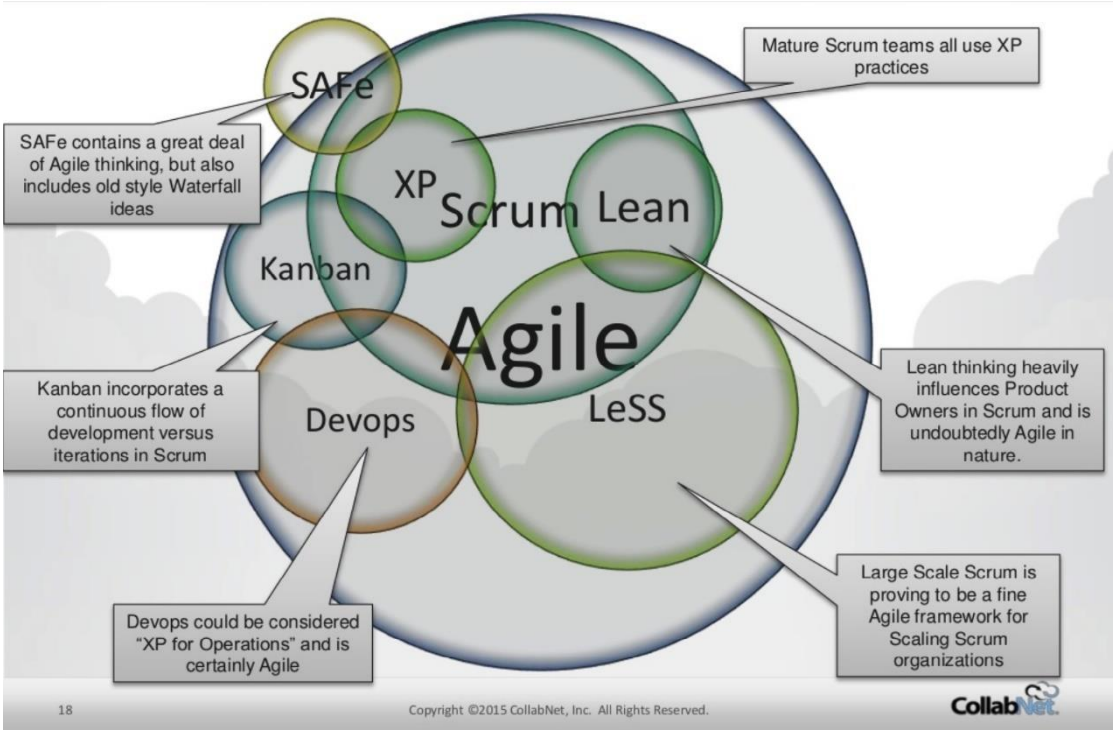
Projektitiimi työskentelee tyypillisesti iteratiivisissa sykleissä ja aikaansaatua tuotetta arvioidaan aina syklin lopussa. Arvioinnin perusteella tuotetta voidaan tarpeen mukaan muokata seuraavalle syklille, jotta se vastaisi paremmin asiakkaan tarpeisiin. (Project Management Guide n.d.) Lyhyet työskentelysyklit ovat 12. vuosittaisen ketteryystutkimuksen mukaan yksi tärkeimmistä ketteryyden tekniikoista (ks. kuvio 4) (The 12th Annual State of Agile Report 2018).

Kuvio 4. Tärkeimmät ketterät tekniikat (The 12th Annual State of Agile Report 2018)

Jokainen sykli käy läpi sovelluksen elinkaaren, joka sisältää tuotteen määrittelyn, suunnittelun, kehityksen, testauksen ja käyttöönoton (ks. kuvio 5). Määrittelyvaiheessa keskustellaan asiakkaan ja mahdollisesti jopa sovelluksen loppukäyttäjien kanssa, mitä tuotteelta oikeastaan halutaankaan. Nämä vaatimukset kirjataan ylös vaatimusmäärittelyyn, jota päivitetään pitkin projektia. Suunnitteluvaiheessa valitaan käytettävät tekniset ratkaisut, kuten esimerkiksi ohjelmointikieli ja sovelluksen arkkitehtuuri. Kehitysvaiheessa tapahtuu itse ohjelmointi. Testausvaiheessa aiemmin tuotettua koodia testataan useilla erityyppisillä testeillä, jotta sinne ei jää (liikaa) virheitä. Käyttöönottovaiheessa sovellus voidaan ottaa käyttöön, mikäli se vastaa asiakkaan tarpeita. Tarvittavat muutostoiveet siirretään seuraavan syklin tehtävälistaan. (Barrios n.d.) Jatkuva, jo aikaisessa vaiheessa saatu palaute on yksi tärkeimmistä ketterän kehityksen työkaluista.

Kuvio 5. Sovelluksen elinkaari (Barrios n.d.)

Erilaisia ketterän kehityksen malleja omine työkaluineen on nykyaikana lukuisia (ks. kuvio 6) ja oikean valitseminen voikin olla haastavaa. Mikään sääntö ei kerro, mitä menetelmää minkäkin yrityksen tai organisaation tulisi käyttää saavuttaakseen parhaimman hyödyn projektissaan. Menetelmän valitseminen perustuu pikemminkin henkilökohtaiseen kokemukseen ja menetelmän ideologian ymmärtämiseen, sekä siihen, miten hyvin menetelmä saadaan otettua käyttöön. (Altman 2017a, Chapter 2: How to execute agile project management?) Menetelmän valintaan vaikuttaa myös projektin luonne sekä siinä työskentelevät ihmiset.



Kuvio 6. Ketterä kehitys (Agile) sisältää monia erilaisia menetelmiä (Dawson 2015)

Mistä sitten voi tietää, onko yritys tai organisaatio oikeasti ketterä? Yksi helpommin havaittavista merkeistä on sprinttimallin mukainen työskentely. Saadun palautteen ja informaation pitäisi tavoittaa kaikki projektiin osallistuvat tahot, niin yrityksen sisällä kuin myös asiakkaan päädyssä. Muutoksia tulisi pystyä toteuttamaan dynaamisesti tuotekehityksen aikana. Toimiva tuote pitäisi saada julkaistua mahdollisimman aikaisessa vaiheessa, vaikka siinä ei vielä olisikaan kaikki suunnitellut ominaisuudet käytössä. Tarkkoja linjauksia on mahdoton vetää, sillä tutkiminen ja sopeutuminen on ketterän kehityksen ydin. (Altman 2017a, Chapter 8: Pointers to help you know if you are "agile" or not.)

2.2 Ketterän ohjelmistokehityksen julistus ja sen kaksitoista periaatetta

Ketterä ohjelmistokehitys määritetään tyypillisesti Ketterän ohjelmistokehityksen julistuksen kautta, joka muodostettiin vuonna 2001 (Laanti ym. 2013, luku 1).

Julistuksen takana ovat kaksitoista periaatetta:

> *Tärkein tavoitteemme on tyydyttää asiakas toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.*
>
> *Otamme vastaan muuttuvat vaatimukset myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.*
>
> *Toimitamme versioita toimivasta ohjelmistosta säännöllisesti, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.*
>
> *Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin koko projektin ajan.*
>
> *Rakennamme projektit motivoituneiden yksilöiden ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja luotamme siihen, että he saavat työn tehtyä.*
>
> *Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on kasvokkain käytävä keskustelu.*
>
> *Toimiva ohjelmisto on edistymisen ensisijainen mittari.*
>
> *Ketterät menetelmät kannustavat kestävään toimintatapaan. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.*
>
> *Teknisen laadun ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.*
>
> *Yksinkertaisuus - tekemättä jätettävän työn maksimointi - on oleellista.*
>
> *Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseorganisoituvissa tiimeissä.*
>
> *Tiimi tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan, ja mukauttaa toimintaansa sen mukaisesti.*
>
> *(Julistuksen takana olevat periaatteet 2011.)*

Ensimmäinen periaate korostaa asiakastyytyväisyyttä, arvoa sekä jatkuvia toimituksia, jotka alkavat aikaisessa vaiheessa. Toisen periaatteen painoarvot ovat sopeutumiskyvyssä, kilpailukyvyssä sekä asiakkaan saamassa hyödyssä. Kolmas periaate korostaa myös jatkuvia toimituksia, kuten ensimmäinenkin periaate. Neljäs periaate painottaa yhteistyön merkitystä. Viidennen periaatteen mukaisesti motivoituneet henkilöt, hyvä työympäristö, riittävä tuki sekä luottamus ovat tärkeitä.

Kuudes periaate painottaa kommunikaatiota sekä tehokkuutta. Seitsemäs periaate korostaa toimivan tuotteen aikaansaamista. Kahdeksannen periaatteen mukaisesti ihmiset ja ylläpidettävyys ovat tärkeitä. Yhdeksäs periaate korostaa teknistä osaamista sekä hyvää suunnittelua. Kymmenes periaate muistuttaa yksinkertaistamisesta ja työn optimoinnista. Yhdennentoista periaatteen mukaan itseohjautuvuus on tärkeää. Kahdestoista periaate painottaa sisäänrakennettua tehokkuuden ja käyttäytymisen kehittämistä. (Laanti ym. 2013, luku 2.)

2.3 Ketterän ohjelmistokehityksen menetelmiä ja malleja

2.3.1 Lean

Leanista ei voi puhua kertomatta ensin japanilaisesta Toyotan tuotantojärjestelmästä (Toyota Production System, TPS) sekä Toyotan tavasta (Toyota Way).

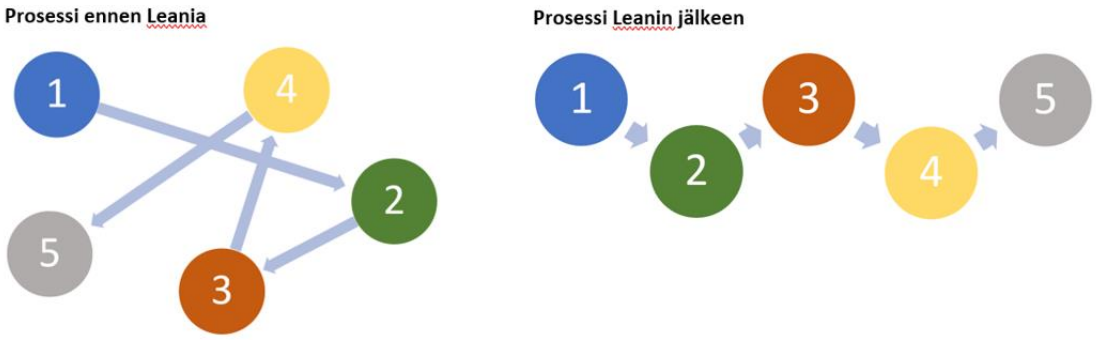
Toyota kehitti tuotantojärjestelmänsä jo toisen maailmansodan jälkeen autoteollisuuden käyttöön. Toyotan piti tuottaa monia erilaisia automalleja samalla kokoonpanolinjalla mahdollisimman nopeasti, joustavasti ja hintatehokkaasti. Toyota kehitti vuosikymmenien aikana filosofiaansa, Toyotan tapaa. Toyotan tapaan kuuluu erityisesti jatkuvan parantamisen kulttuuri, johon kaikki yrityksen työntekijät johtoportaasta aina tuotantotason työntekijöihin asti noudattavat. Lisäksi Toyotalla on käytössä useita erilaisia työkaluja ja menetelmiä, jotka auttavat eliminoimaan resurssien tuhlausta, parantamaan laatua sekä tasapainottamaan aikataulua. Toyotan tapaan pohjautuva Lean-ajattelu tuli yleisesti käyttöön vasta paljon myöhemmin. (Liker 2006, 7-15.)

Vaikka Lean pohjautuukin autoteollisuudessa käytettyihin menetelmiin, sitä voidaan soveltaa minkä tahansa organisaation käyttöön, tuotteesta riippumatta.

Lean perustuu virtauksen maksimointiin ja hukan (muda) poistamiseen. Tämä tarkoittaa sitä, että kaikista työvaiheista pitää saada mahdollisimman sulavia ja tarpeettomat vaiheet tulee eliminoida (ks. kuvio 7). Jotta tämä onnistuu, johdon ja esimiesten pitää ensin ymmärtää prosessi kokonaisuudessaan. Ymmärtääkseen paremmin, johtajan tulee itse mennä paikan päälle tarkkailemaan normaalia työn

tekoa ja oppimaan työntekijöiltä. Tätä kutsutaan Gemba-kävelyksi. Parannukset kehitetään yhdessä työntekijöiden kanssa, sillä he ovat oman työnsä asiantuntijoita. (Lopresti 2018.)

Yksi Leanin yleisimmin käytetyistä työkaluista on arvovirtakuvaus (Value Stream Map, VSM), jonka avulla tunnistetaan virtauksen esteitä. Prosessin kaikki toiminnot ja niiden väliset yhteydet kuvataan vaiheittain yhdelle paperille. Seuraavaksi pyritään tunnistamaan ongelmat ja hukan lähteet ja eliminoidaan ne. Tarkoitus on sujuvoittaa työnkulkua ja yhtenäistää toimintatapoja, jolloin kaikki työntekijät tekevät asiat samalla tavalla. (Väisänen 2013.)



Kuvio 7. Prosessi ennen Leania ja Leanin jälkeen

Toyota on tunnistanut seitsemän erilaista hukan muotoa, jotka ohjelmistotekniikan puolella ovat Mary ja Tom Poppendieckin (2006) mukaan seuraavia:

1. **Keskeneräinen työ**. Jos tehtävä aloitetaan, mutta sitä ei koskaan saada valmiiksi, se säilyy avoimena työjonossa. Keskeneräistä työtä voi olla myös esimerkiksi kommentoimaton koodi, jota joudutaan kommentoimaan myöhemmin tai liian tarkkaan kirjoitettu dokumentointi, jota joudutaan muokkaamaan jälkikäteen.

2. **Ylimääräiset ominaisuudet**, eli ylituotanto. Ominaisuuksia, joita ei tulla koskaan ottamaan käyttöön tai joista asiakas ei saa hyötyä, ei kannata tehdä. Pyritään ensisijaisesti saamaan aikaiseksi pienin mahdollinen julkaistava tuote (Minimum Viable Product, MVP).
3. **Uudelleen oppiminen**. Vanhaan koodiin palaaminen ja sen uudelleen oppiminen voi olla työlästä ilman kunnollista dokumentaatiota. Uudelleen oppimista tapahtuu myös tuotteen luovutuksen yhteydessä.
4. **Tuotteen luovutus.** Projektin siirtäminen tiimiltä tai henkilöltä toiselle lisää työn määrää. Kaikkien työhön tarvittavien henkilöiden pitäisi toimia saman tiimin sisällä.
5. **Tehtävien vaihdot**. Erityisesti ohjelmointitehtävissä hyppiminen projektista toiseen on haastavaa. Ohjelmointi vaatii pitkäjänteistä keskittymistä ja oikeaan mielentilaan pääseminen vaatii aikaa. Yksi työ tulisi viedä loppuun kerrallaan.
6. **Viivästykset**. Mikäli asiakas ei katselmoi tehtyä työtä mahdollisimman pian ja toimita palautetta eteenpäin, uudelleen oppimisen riski kasvaa. Viivästyksistä voi aiheutua myös muuta hukkaa, kuten tehtävien vaihtoa ja keskeneräistä työtä. Palautetta tulisi saada mahdollisimman usein.
7. **Viat**. Viallisen koodin korjaukseen voi kulua paljon aikaa. Testaukseen tulisi panostaa jo hyvissä ajoin.

(Stine 2010.)

Hukkaa pyritään poistamaan käyttämällä Kaizen-nimistä periaatetta.

Kaizen on japaninkielinen sana, joka tarkoittaa kirjaimellisesti hyvää muutosta. Kaizenin periaatteen mukaan muutoksia tehdään pienin askelin ja jokainen työntekijä nähdään mahdollisena ideoiden ja tiedon lähteenä. Muutos koetaan yleensä negatiivisena asiana. Joillekin ajatus muutoksesta voi aiheuttaa ahdistusta, toisille jopa pelkoa. Kaizenissa pyritään tekemään niin pieniä muutoksia, että ne on helppo toteuttaa ja eivät aiheuta negatiivisia reaktioita. Kehitysaskeleet voivat toisinaan tuntua jopa naurettavan pieniltä, mutta niillä voi kuitenkin olla yllättävän suuri vaikutus kokonaisuuden kannalta. Kaizenin avulla voidaan mm. parantaa ilmapiiriä ja laatua, hillitä kuluja sekä kehittää uusia tuotteita ja palveluita. (Maurer 2013, 7-20.)

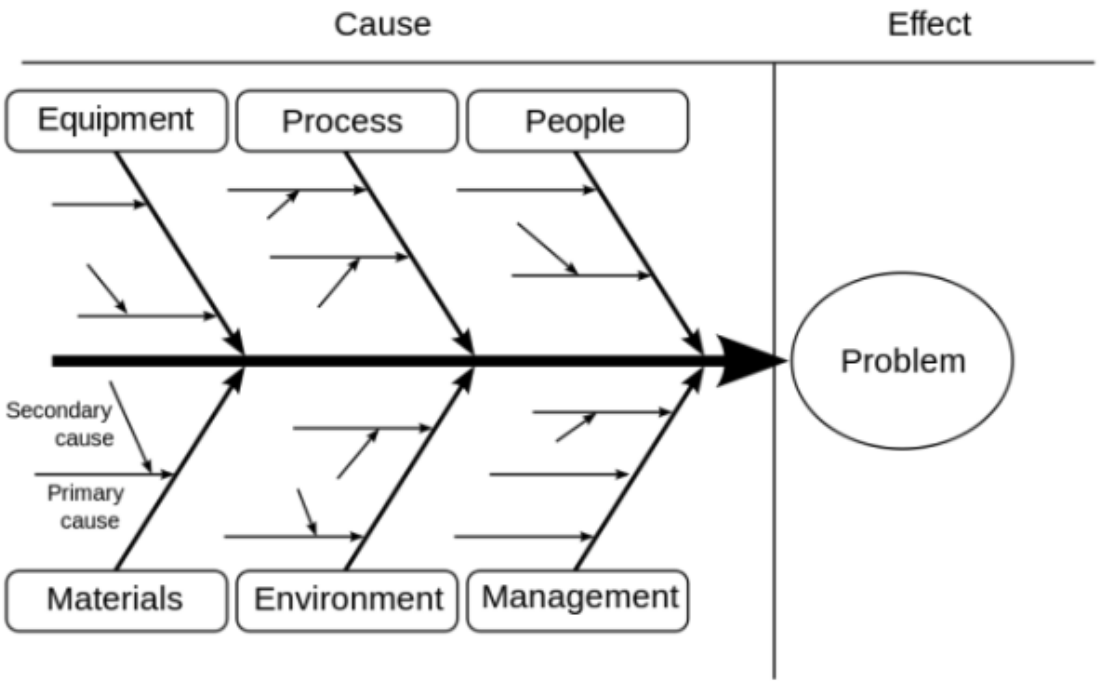
Tyypillisesti Kaizenia toteutetaan seuraavin askelin:

1. Kartoita kehitysmahdollisuudet, tunnista hukkatyö ja esteet.
2. Arvioi nykyinen tilanne.
3. Ideoi kehitysehdotuksia.
4. Suunnittele idean käyttöönotto. Sitoudu parannukseen, oli se miten pieni hyvänsä.
5. Ota parannus käyttöön pienin muutoksin.
6. Arvioi uusi tilanne ja korjaa se, mikä ei toimi.

(Bisiani n.d.)

Kaizen-askeleet tunnetaan myös Demingin ympyränä, eli PDCA-ympyränä (Plan-Do-Check-Act). Askeleet dokumentoidaan usein A3-tekniikkaa hyödyntäen. Tällöin tunnistettu ongelma ja toimenpiteet kirjataan visuaaliseen muotoon A3-paperille, joka kiinnitetään näkyvälle paikalle työtiloissa. (Kangas, Kirjavainen, Fallenius & Mikkonen 2018, 6.)

Toisinaan ongelman todellista alkuperää, juurisyytä on vaikea löytää. Tällöin voi käyttää esimerkiksi 5 x miksi? -tekniikkaa hyväkseen. Tekniikka on hyvin yksinkertainen, siinä kysytään kysymys ”Miksi?” niin monta kertaa, että päästään juurisyyhyn käsiksi. Kysymyksiä voi todellisuudessa olla enemmän tai vähemmän, kuin viisi. Visuaalisempi keino juurisyyn kartoittamiseen on kalanruotokaavio (Ishikawa), jossa ongelma ja mahdolliset syyt kuvataan kalanruotomaiseen kuvioon (ks. kuvio 8). (Kangas ym. 2018, 10.)



Kuvio 8. Esimerkki kalanruotokaaviosta (Kangas ym. 2018, 10)

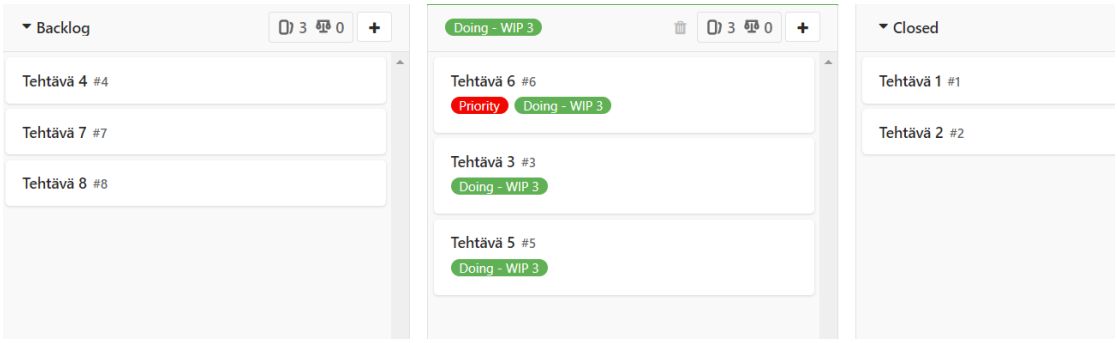
Kaizenissa kyse ei niinkään ole toimintatavoista, kuin asenteesta. Kaizenia voi soveltaa niin työelämässä, kuin kotonakin mihin tahansa elämän osa-alueelle. Parantaminen ei koskaan lopu. Jos olet käyttänyt Kaizenia parantamaan yhden asian, seuraava askel on parantaa sitä uudelleen. (Bisiani n.d.)

Lean ei siis ole vain työkalujen ja menetelmien kokoelma, joita käyttämällä organisaatio voi sanoa olevansa Lean. Lean on pitkäjänteisyyttä ja sitoutumista vaativa kehitystyö, joka koskettaa jokaista yrityksen jäsentä ja vaatii erityistä sitoutumista johdolta.

2.3.2 Kanban

Kanban on lähtöisin Toyotan käyttämistä menetelmistä ja oli alkujaan fyysinen korttijärjestelmä, jota käytettiin visualisoimaan osaston valmiutta vastaanottaa lisää työstettävää raakamateriaalia. Nykypäivänä käytetään esimerkiksi tarralappuja tai

valkotauluja, sekä verkosta löytyy useita erilaisia virtuaalisia palveluita (ks. kuvio 9). (Project Management Guide n.d.)



Kuvio 9. Esimerkki Kanban-laudasta

Työn visualisointi ja työnkulun ymmärtäminen on erittäin tärkeää, jotta oikeanlaisia muutoksia osataan tehdä. Visualisointi tapahtuu usein pystysuuntaisilla sarakkeilla, joilla kuvataan työvaiheita sekä vaakasuuntaisilla riveillä, joilla kuvataan esimerkiksi henkilöitä tai osastoja. Tehtävää työtä rajoitetaan WIP-luvulla (work-in-progress), ettei pullonkauloja pääse syntymään eri työvaiheisiin. Lisäksi täytyy varmistaa, että kaikki noudattavat sovittuja sääntöjä ja käytänteitä samalla tavalla ja ymmärtävät syyt niiden taustalla. (Altman 2017b, Introduction.)

Jos esimerkiksi suunnittelutiimi pystyy kehittämään neljä uutta ominaisuutta tietyn ajanjakson sisällä, mutta testaustiimi voi testata vain kolme ominaisuutta samassa ajassa, tällöin kolme on maksimi. Pullonkauloja voi WIP-luvun lisäksi ehkäistä työntekijöiden uudelleensijoituksilla. (Altman 2017a, Chapter 11: Agile basic strategy.)

Kanbania tulisi käyttää, jotta juuri oikeaan aikaan (just-in-time, JIT) tapahtuva tuotanto saataisiin käytännössä toteutettua. Kanban kertoo henkilölle mitä hänen pitäisi tehdä, kauanko siihen tulisi käyttää aikaa ja kuinka tehtävä on tarkoitus toteuttaa. (Altman 2017b, Chapter 1: Is Kanban dead.)

Kanbanissa henkilöille ei määritetä erityisiä rooleja ja tuotteita voidaan valmistaa jatkuvasti, ilman sprinttimallista ajan jaksottamista. Muutoksia pystytään toteuttamaan joustavasti missä tahansa työn vaiheessa. (Altman 2017b, Chapter 5: Kanban applied in software development.)

2.3.3 Scrum

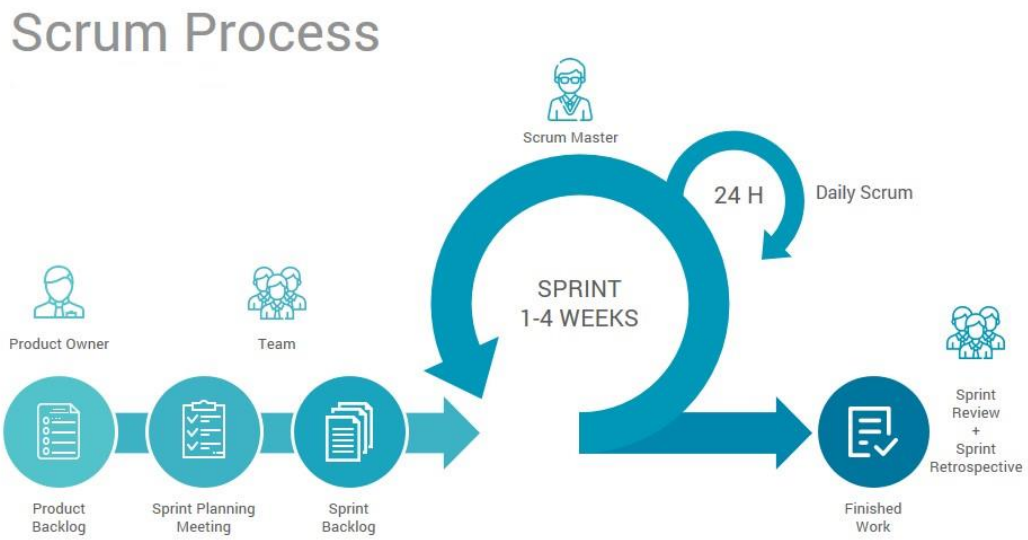
Scrum-sana tulee alun perin rugby-pelistä, jossa pelaajat kokoontuvat tiiviinä rykelmänä yhteen paikkaan ja koittavat saada pallon haltuunsa. Scrum-tiimi käyttäytyy myös rugby-tiimin tavoin, pelissä koko tiimi liikkuu kentällä yhdessä yhteisen tavoitteen saavuttaakseen. Scrum-viitekehyksen pääidea tulee siitä, että asiakas erittäin todennäköisesti muuttaa mieltään sen suhteen, mitä oikeastaan haluaakaan. (Altman 2017c, Introduction.)

Scrumissa tiimiläisille on määritetty kolme erilaista roolia, joissa he toimivat. Scrum-tiimissä on yksi Scrum-mestari (Scrum Master), jonka päätehtävä on poistaa työtä hidastavia ongelmia tiimin tieltä ja valvoa, että Scrumia noudatetaan oikein. Hän ei kuitenkaan varsinaisesti johda tiimiä, vaan toimii pikemminkin tiimin palvelijana ja valmentaa sekä motivoi tarvittaessa. Scrum-mestarin lisäksi määritetään yksi henkilö toimimaan tuotteen omistajana (Product Owner), joka toimii asiakkaan roolissa ja valvoo tuotteen edistymistä haluttuun suuntaan. Hän on yksin vastuussa projektin onnistumisesta. Loput tiimiläiset muodostavat kehitystiimin, joka on vastuussa toimivan tuotteen valmistamisesta jokaisen sprintin päätteeksi. (Altman 2017c, Chapter 1: What is Scrum; Chapter 4: What is Scrum Master?)

Scrumissa kehitystiimin jäsenillä ei ole omia titteleitä, eikä tiimeillä ole alitiimejä vastaamassa työn eri vaiheista (Schwaber & Sutherland 2017, 8).

Scrumissa on neljä vakiintunutta palaverityyppiä, jotka kaikki toteutetaan jokaisen sprintin aikana (ks. kuvio 10). Sprintin suunnittelupalaverissa (Sprint Planning) kaikki Scrum-tiimin jäsenet sopivat yhdessä tulevan sprintin tehtävistä, jotka pitäisi toteuttaa. Päiväpalaverissa (Daily Scrum) kehitystiimi kokoontuu joka päivä samaan aikaan raportoimaan toisilleen tekemisistään, mitä tekevät seuraavaksi ja onko ilmennyt ongelmia. Päiväpalaveri saa kestää maksimissaan viisitoista minuuttia, eikä palaverin aikana lähdetä ratkomaan ongelmia. Sprinttikatselmus (Sprint Review)

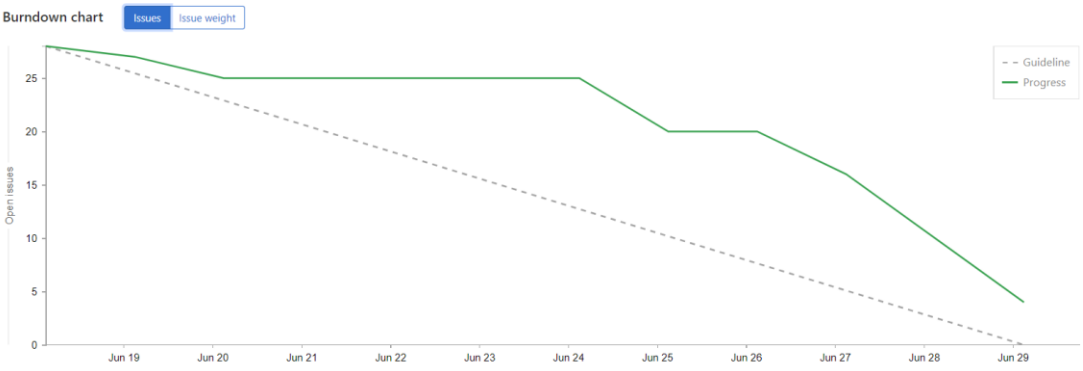
tapahtuu aina sprintin päätteeksi. Kehitystiimi esittelee toimivan tuotteen, jota se on kehittänyt sprintin aikana eteenpäin ja tuotteen omistaja päättää jatkotoimenpiteistä. Retrospektiivissä (Sprint Retrospective) tiimi saa mahdollisuuden tarkastella ja kehittää omaa toimintaansa kuluneen sprintin perusteella. Myös retrospektiivi järjestetään sprintin päättyessä. (Altman 2017c, Chapter 1: What is Scrum.)



Kuvio 10. Scrum-prosessin kuvaus (Warcholinski n.d, muokattu)

Toisinaan saman tuotteen kehitykseen kuitenkin osallistuu useampi kehitystiimi, jolloin osallistujien määrä saattaa kasvaa suureksi. Jokaisen tiimin tulisi pysyä perillä tilanteesta, mutta kaikkien jäsenien ei ole enää järkevää osallistua. Tällöin jokaisesta tiimistä osallistuu yksi henkilö Scrumien Scrumiin (Scrum of Scrums) ja he jakavat tietoa tiimiensä tilanteesta keskenään. Scrumien Scrum pidetään aina tarvittaessa, esimerkiksi päivittäin tai viikottain. (Altman 2017c, Chapter 2: Scrum of Scrums.)

Scrum-viitekehykseen on määritetty neljä erilaista tuotosta, jotka auttavat maksimoimaan tiedon läpinäkyvyyttä ja lisäämään motivaatiota. Tuotteen kehitysjonoon (Product Backlog) listataan ja asetetaan tärkeysjärjestykseen kaikki ne ominaisuudet, toiminnot ja vaatimukset, jotka aiotaan toteuttaa tulevissa sprinteissä. Tuotteen omistaja päivittää kehitysjonoa jatkuvasti, sillä se on elävä dokumentti. Sprintin tehtävälistaan (Sprint Backlog) valitaan kehitysjonosta ne tehtävät, jotka aiotaan työstää kuluvan sprintin aikana. Kehitystiimi vastaa tehtävälistan ylläpitämisestä sprintin kuluessa. Mikäli tehtäviä ei keretä tekemään, ne palautetaan takaisin tuotteen kehitysjonoon sprintin päättyessä. Edistymiskäyrä (Sprint Burndown chart) (ks. kuvio 11) sekä julkaisukäyrä (Release Burndown chart) ilmaisevat, kuinka paljon työtä on tehty tietyn ajanjakson sisällä ja kuinka paljon on vielä jäljellä. (Altman 2017c, Chapter 1: What is Scrum; Schwaber & Sutherland 2017, 15.)



Kuvio 11. Sprintin edistymiskäyrä Kanban-laudalla kesältä 2018

Kaikilla Scrum-tiimin jäsenillä tulee olla yhteinen käsitys siitä, milloin tehtävä työ on ”valmis” (Definition of Done). Jo kehitysjonossa määritettyyn tehtävään on usein kirjattu, milloin tehtävä voidaan katsoa ”valmiiksi”. (Schwaber & Sutherland 2017, 17.)

Huomioitavaa on, että projektin onnistuminen on riippuvainen tiimin jäsenten sitoutumisesta työlle. Tämän lisäksi tuotteen omistajan täytyy olla erittäin hyvin perehtynyt asiakkaan tarpeisiin ja toiveisiin. (Altman 2017c, Chapter 3: Scrum software development methodology.)

2.3.4 Six Sigma ja Lean Six Sigma

Six Sigman keskeinen ajatus on tuotteiden laadun vaihtelun, eli virheiden eliminoiminen tilastollisin keinoin. Laadunvarmistusta pyritään parantamaan mittaamalla prosessissa esiintyvien virheiden määrää ja poistamalla ne radikaalein muutoksin. (Six Sigma n.d.)

Six Sigma hyödyntää ongelmanratkaisuun menetelmää nimeltä DMAIC, joka koostuu viidestä eri vaiheesta:

- Määrittely (Define), eli määritetään ratkaistava ongelma. Tiimi luo prosessista kuvauksen ymmärtääkseen prosessin asiakkaan tarpeet paremmin.
- Mittaus (Measure), eli kuinka hyvin prosessi nykytilanteessa toimii? Toisin sanoen, kuinka suuren luokan ongelmasta on kyse?
- Analysointi (Analyze), eli mistä ongelma johtuu? Todellisen juurisyyn löytäminen on tärkeää, jotta oikea ongelma osataan ratkaista.
- Parannus (Improve), eli ratkaisun kehittäminen todettuun ongelmaan. Tiimi kehittää erilaisia ratkaisuvaihtoehtoja ja mahdollisuuksien mukaan pilotoi niitä. Dataa kerätään todistamaan, että edistystä oikeasti tapahtuu.
- Ohjaus (Control), eli ylläpito. Löydetyn ratkaisun tuloksien mittaamiseksi ja ylläpitämiseksi luodaan suunnitelma. Lisäksi kehitetään varasuunnitelmia mahdollisten tulevien ongelmien varalle.

(DMAIC – The 5 Phases of Lean Six Sigma n.d.)

Six Sigma hyödyntää kymmeniä erilaisia työkaluja ongelmanratkaisun eri vaiheissa ja niitä on vapaasti saatavilla verkossa. Noudattamalla DMAIC -strategiaa ongelmat saadaan poikkeuksetta ratkaistua ja saavutetaan dramaattinen parannus prosessiin.

Six Sigmassa ihmisille on määritetty useita erilaisia rooleja, joihin kuuluu erilainen osaamisen taso:

- Valkoinen vyö (White Belt) ymmärtää Six Sigman rakenteen, tavoitteet ja perussanaston.
- Keltainen vyö (Yellow Belt) ymmärtää Six Sigman konseptin ja osallistuu projektiin tiimin jäsenenä.
- Vihreä vyö (Green Belt) käynnistää ja hallinnoi Six Sigma -projekteja ja tarjoaa koulutusta muille.
- Musta vyö (Black Belt) toimii esimerkiksi valmentajana, opettajana tai projektin vetäjänä.
- Mestari musta vyö (Master Black Belt) on vastuussa Six Sigman käyttöönottamisesta ja yhteisön kulttuurin muuttamisesta. Hän työskentelee johtajien tukena ja auttaa projektien valitsemisessa.
- Mestari (Champion) on yleensä johtaja, joka tukee ja kehittää Six Sigma -kulttuuria ja auttaa valitsemaan oikeanlaisia projekteja. Hän poistaa tiimejä haittaavia esteitä.

(The Roles of Lean Six Sigma n.d.)

Lean Six Sigma on käytännössä Leanin ja Six Sigman risteytys. Lean keskittyy enemmän prosessien nopeuttamiseen hukkaa poistamalla, kun taas Six Sigma virheiden vähentämiseen ja laadun parantamiseen. Näiden yhdistelmällä saadaan käyttöön tehokkaimmat ja laadukkaimmat työskentelytavat, joiden avulla maksimoidaan tuotanto ja vähennetään virheitä. (McKenzie 2009.)

2.3.5 Extreme Programming (XP)

XP-menetelmä soveltuu parhaiten pienempien kehitystiimien käyttöön, joista tavanomaisesti yksi toimii valmentajan roolissa. Optimitapauksessa työskentelytiloissa on jatkuvasti mukana myös ainakin yksi asiakkaan edustaja. Työtä tehdään sykleissä ja jokaisen syklin lopputuloksena on toimivaa, testattua koodia, josta on asiakkaalle hyötyä. (Cockburn 2006, 199-201.)

Valmistettavaan palveluun tai tuotteeseen tarvittavat vaatimukset kootaan yhdessä asiakkaan kanssa käyttäjätarinoiden muotoon ja ne priorisoidaan tärkeysjärjestykseen. Ohjelmoijat määrittävät jokaiselle arvion, kuinka kauan niiden

toteutuksessa oletetaan kestävän. Tarpeen mukaan tehtäviä voidaan uudelleenpriorisoida matkan varrella. (Cockburn 2006, 199-201.)

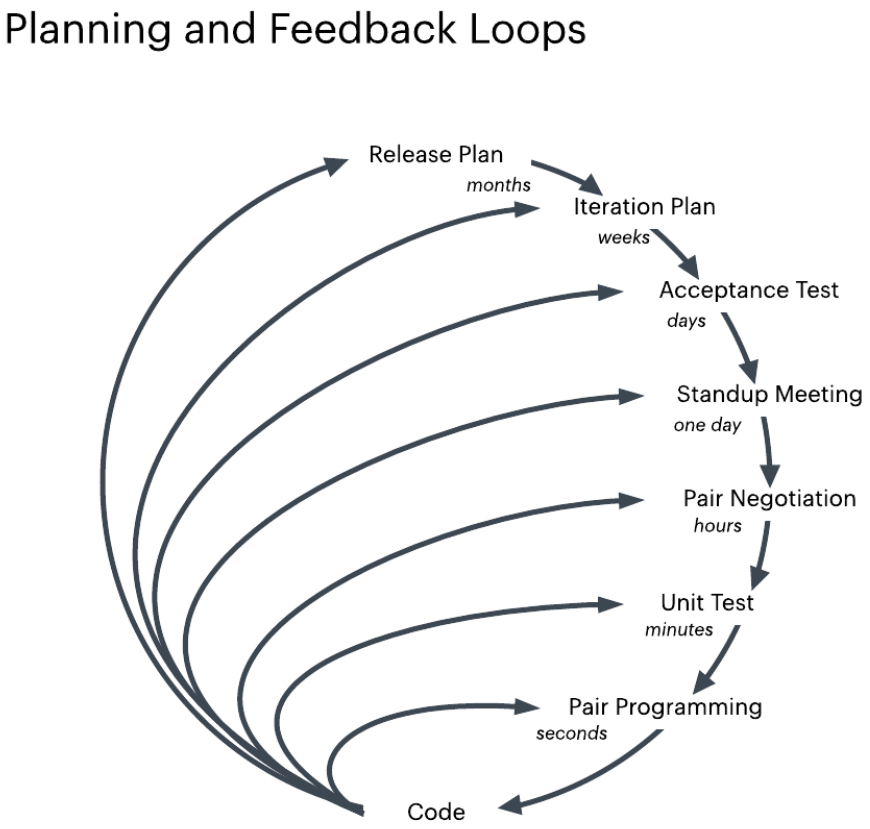
Ohjelmoijat istuvat pareittain yhden koneen äärellä, jota kutsutaan myös parikoodaukseksi. Toinen tekee itse ohjelmointia, toinen katsoo vieressä ja arvioi tuotettua koodia jatkuvasti. Yksikkötestit tehdään kaikelle valmistuneelle koodille ja testien tulee mennä 100%:sti läpi joka kerta, kun aikaansaatu koodi päivitetään pääkoodiin. Tietokoneella istuvan ohjelmoijan ja vieressä istuvan vuoroja vaihdetaan sopivin väliajoin, joka vaihtelee 15 minuutista kahteen tuntiin. Pareja vaihdetaan päivittäin, jotta osaaminen pysyy kaikilla tasaisena. Kuka tahansa voi milloin tahansa muuttaa koodia yksinkertaisemmaksi, mikäli se on liian monimutkaista tai muuten vaikeaselkoista. (Cockburn 2006, 199-201.)

Asiakkaan on vierailtava ohjelmoijien luona vastaamassa kysymyksiin ja selkeyttämässä kokonaiskuvaa mahdollisimman usein, ellei pysty olemaan läsnä jatkuvasti. Asiakkaan vastuulla on kirjoittaa hyväksyntätestejä, joiden mukaan työtä voidaan arvioida syklien päätteeksi ja asiakas myös päättää, mitä seuraavan syklin aikana tehdään. (Cockburn 2006, 199-201.)

Tiimit pitävät päivittäin seisontapalaverin ja jokainen kertoo mitä on tekemässä, mikä toimii ja missä voisivat mahdollisesti tarvita apua. Syklin päättyessä pidetään myös palaveri, jossa käydään läpi koettuja onnistumisia ja seuraavia työskentelykohteita. (Cockburn 2006, 199-201.)

Palautetta saadaankin siis XP-menetelmällä jatkuvasti (ks. kuvio 12).

XP-menetelmää käytettäessä vaaditaan erityisesti rohkeutta tarpeettoman koodin poistamiseen, vaikka sen tuottamiseen olisikin kulunut paljon vaivaa. Jokaisen tiimin jäsenen kokemusta ja ideoita kunnioitetaan yhtä lailla. (Altman 2017a, Chapter 11: Agile basic strategy.)



Kuvio 12. Extreme Programming – menetelmän palautteen saamisen kiertokulku (What Is Extreme Programming... 2018, muokattu)

2.3.6 DevOps ja DevSecOps -tuotekehitysmallit

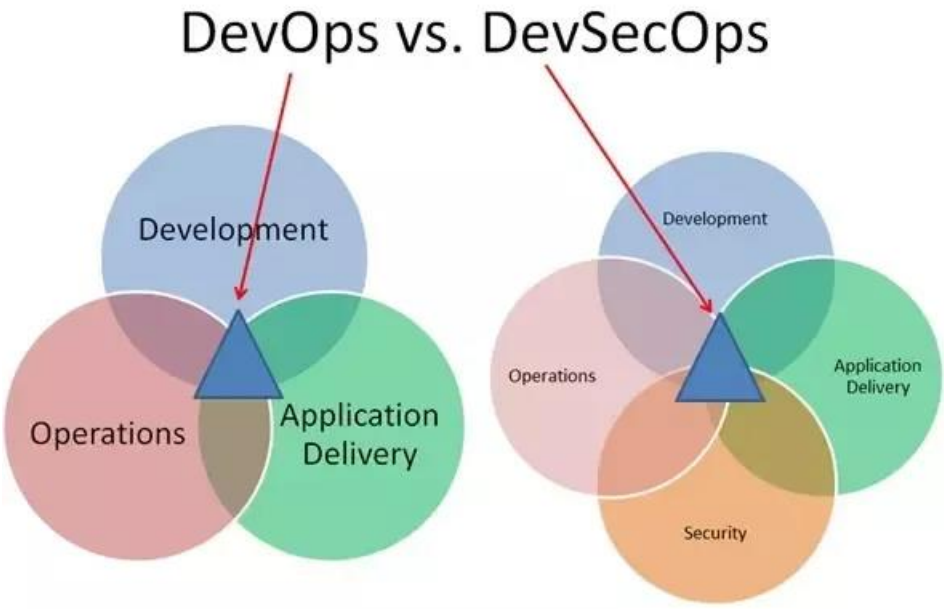
DevOps-nimi tulee sanoista *development* ja *operations*, jotka tarkoittavat kehitystä ja ylläpitoa. Tuotteen vakaus on yhtä tärkeää, kuin sen ominaisuudet.

DevOps-tuotekehitysmallissa ohjelmistokehittäjät ja -testaajat työskentelevät samassa tiimissä varmistaakseen ohjelman nopean kehityksen ja laadunvarmistuksen. Erilaisia käytänteitä hyödynnetään toistuvien, manuaalisten työtehtävien automatisoinnissa, kuten esimerkiksi testaamisessa. Pieniä koodipäivityksiä tehdään usein (Continuous Integration), jolloin vikojen paikallistaminen on helpompaa. (What is DevOps? n.d.)

Sovelluksia voidaan hajottaa toiminnallisuuksien perusteella pienempiin osiin, eli mikropalveluihin. Tämä helpottaa työn koordinointia, kun eri mikropalveluiden kehittäminen voidaan vastuuttaa eri tiimeille. Jokainen mikropalvelu toimii itsenäisenä prosessinaan ja kommunikoi muiden mikropalveluiden kanssa liitäntäkohtien kautta. (What is DevOps? n.d.)

Yksi DevOpsin tärkeimpiä käsitteitä on jatkuva julkaisu (Continuous Delivery), jonka mukaan koodiin tehdyt muutokset rakennetaan, testataan ja valmistetaan toimitettavaksi automaattisesti. Mikäli jatkuva julkaisu toteutetaan oikein, kehittäjillä on aina saatavilla käyttövalmis ja testattu versio sovelluksesta. (What is DevOps? n.d.)

DevSecOps tarkoittaa samaa, kuin DevOps, mutta mukaan on tuotu myös tietoturvan näkökulma (Security) (ks. kuvio 13). Uhkatilanteiden mallinnus ja riskien arviointi auttavat haavoittuvuuksien löytämisessä ja niiden perusteella voidaan tehdä esimerkiksi automatisoituja tietoturvatestejä jo alusta lähtien. Jokainen DevSecOps-tiimin jäsen huolehtii tietoturvasta yhtä lailla, se ei ole vain yhden henkilön vastuulla. (Matteson 2017.)



Kuvio 13. DevOpsin ja DevSecOpsin ero (Sawant 2018)

3 Tiimin toiminnan kehittäminen

3.1 Tiimin merkitys

Ketterissä ohjelmistokehityksen menetelmissä tiimeillä ja niiden itseohjautuvuudella on hyvin suuri merkitys työn onnistumisen kannalta.

Hyvällä tiimityöskentelyllä saavutetaan lukuisia etuja, kuten esimerkiksi:

- Tehokkuus lisääntyy ja virheitä syntyy vähemmän.
- Työntekijöiden tyytyväisyys paranee.
- Muutoksien kohtaaminen helpottuu ryhmän tuen myötä.
- Uusien asioiden oppiminen helpottuu.

(Salminen 2017, 73-79.)

Hyvä tiimi ei kuitenkaan synny itsestään, vaan yleensä tarvitaan kokopäiväinen, asialleen omistautunut valmentaja. Pelkkä tiimin rakentaminen ei vielä riitä, sillä tiimin suorituskykyä tulee ylläpitää jatkuvasti. Muuten tiimin suorituskyky lähtee laskemaan nopeasti. (Salminen 2013, 16-17.)

3.2 Tiimin kehitysvaiheet

Salmisen (2013, 68) mukaan Bruce Tuckman on määrittänyt viisikohtaisen mallin tiimien kehitysvaiheista:

- Perustamisvaihe (Forming)
- Myrskyvaihe (Storming)
- Oppimisvaihe (Norming)
- Suoritusvaihe (Performing)
- Hajoamisvaihe (Adjourning)

Perustamisvaiheessa joukko ihmisiä kootaan yhteen. Vaiheen aikana jäsenet oppivat tuntemaan toisensa ja tiimin tavoitteet sekä määrittävät omat roolinsa. Tässä vaiheessa innostus on yleensä vielä korkealla ja tiimi on hyvin riippuvainen tiiminvalmentajan ohjauksesta. (Salminen 2013, 69-71.)

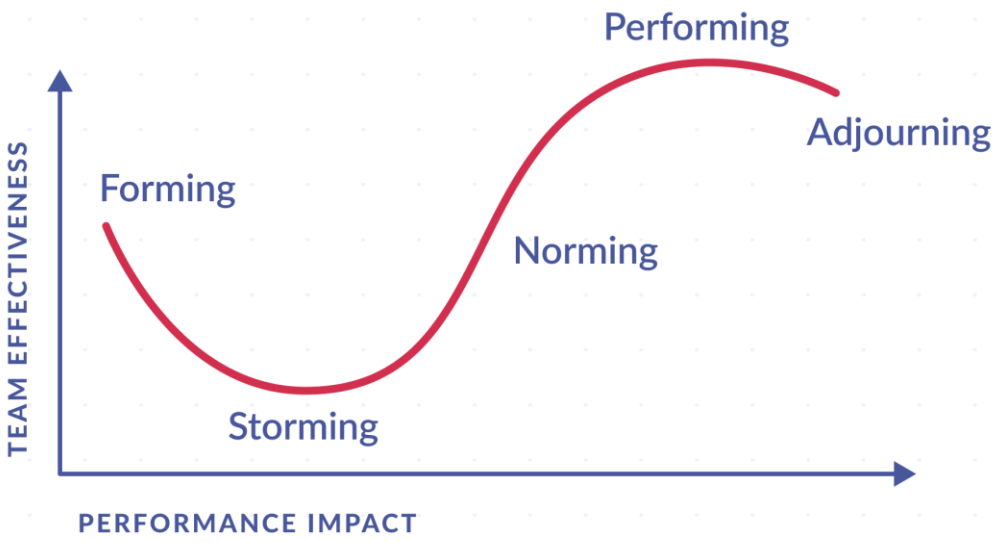
Myrskyvaiheessa tiimin jäsenten välille voi nousta erilaisia ristiriitoja ja joskus jopa valtataistelua. Tiimiläiset eivät välttämättä vielä luota toisiinsa ja näkemykset voivat poiketa toisistaan. Myrskyvaiheessa on erityisen tärkeä korostaa yhteistyön merkitystä ja varmistaa, ettei kukaan jää syrjään. (Salminen 2013, 71-74.)

Oppimisvaiheessa tiimin tavoite alkaa olemaan selvä kaikille jäsenille ja toiminta kehittyy nopeasti. ”Me”-ajattelu alkaa valtaamaan alaa ”minä”-ajattelulta ja tiimin identiteetti vahvistuu erilaisien tarinoiden ja käyttäytymismallien avulla. Tavoitteet saattavat joskus jopa unohtua, kun vietetään hauskaa aikaa yhdessä. (Salminen 2013, 74-75.)

Suoritusvaiheessa tiimin suoritustaso on korkealla ja jäsenet pystyvät tekemään hyvin yhteistyötä keskenään. Tiimi saattaakin alkaa suhtautumaan muihin alentuvasti, joten ympäristön kanssa jatkuva yhteyden pitäminen on tärkeää. Mikäli tiimistä kehkeytyy huipputiimi, se pystyy toimimaan yhteistyössä niin sisäisesti, kuin myöskin ulkopuolisten tahojen kanssa. Huipputiimissä jäsenet tukevat toisiaan motivaation laskiessa, joskin motivaatio on yleensä hyvin korkealla toimivan tiimityön sekä syvän osaamistason vahvistamana. (Salminen 2013, 75-78.)

Hajoamisvaihe tulee vastaan esimerkiksi projektin päättyessä, kun tiimi lopettaa toimintansa (Salminen 2013, 78).

Jokaisella vaiheella on erilainen vaikutus työn tehokkuuteen (ks. kuvio 14). Valmentajan tavoitteena tulisi olla myrskyvaiheen keston lyhentäminen ja suoritusvaiheen keston pidentäminen, jotta tiimin tehokkuus saataisiin maksimoitua. (Paych n.d.)



Kuvio 14. Tiimin kehitysvaiheiden vaikutus tehokkuuteen (Paych n.d.)

3.3 Tiimitoiminnan haasteita

Tiimissä toimiminen on haastavaa ja ongelmilta ei aina voi välttyä. Erilaiset näkemykset ja tavoitteet, sekä sitoutumisen eri tasot voivat aiheuttaa ristiriitoja. Salminen (2013, 231-233) sanoo, että jo yhdenkin jäsenen sitoutumattomuus seitsemän hengen tiimissä heikentää tiimin toimintaa 29 prosenttia. Tämän lisäksi sitoutumaton henkilö ”saastuttaa” ilmapiiriä, joka edelleen laskee tehokkuutta. Siksi tiimin yhteistyökyvyn ylläpitämiseen täytyy panostaa jatkuvasti.

Negatiivisuutta ja ärsyyntymistä muissa tiimiläisissä voivat aiheuttaa esimerkiksi lupausten rikkominen, kritisointi, kilpaileminen, ”vapaamatkustus” ja toisista pahan puhuminen. Valmentajan velvollisuus on tällöin puuttua tiimin motivaatiota ja suorituskykyä alentavaan toimintaan jämäkästi. (Salminen 2013, 233-238.)

Myös työympäristö voi muodostua haasteeksi. Esimerkiksi avokonttorin tarkoituksena on yleensä parantaa työntekijöiden yhteistyötä, kun kaikki tarvittavat henkilöt ovat samassa tilassa. Avokonttorimaisen huoneen sivuoireena on kuitenkin meluisuus. Tutkimuksen mukaan melu vaikuttaa negatiivisesti hyvinvointiin,

tuottavuuteen sekä tulokseen ja tämä lisää työntekijöiden vaihtuvuutta. Lisäksi työntekijät kärsivät lyhytkestoisen muistin toiminnan heikentymisestä ja luovien ratkaisujen tekeminen vaikeutuu. (Jälleen uusi tutkimus osoittaa avokonttorin ongelmat… 2018.)

4 Case: WIMMA Lab

4.1 Hakijoiden valintaprosessi sekä valittujen opiskelijoiden kuvaus

WIMMA Lab on Jyväskylän ammattikorkeakoulun IT-instituutissa oleva suosittu opintokokonaisuus, jonne pääsee vain haastattelun kautta. Haastatteluilla pyritään varmistamaan, että osallistuja pystyy toimimaan itsenäisesti ja sitoutumaan koko kurssin ajaksi WIMMA Lab -toimintaan. Pääpainotus valinnoissa oli positiivisessa asenteessa sekä motivaatiolla oppia uusia asioita.

Kesällä 2018 haastatteluihin osallistui yhteensä 35 henkilöä kokonaisuudessaan 45 hakijan joukosta. Haastattelujen perusteella WIMMA Labiin valittiin 27 osallistujaa Jyväskylän ammattikorkeakoulun Tieto- ja viestintätekniikan puolelta kyberturvallisuuden, mediatekniikan, ohjelmistotekniikan sekä tietoverkkotekniikan suuntautumisvaihtoehdoista. Lisäksi mukaan otettiin yksi Hyvinvointiyksikön toimintaterapeuttiopiskelija.

Opiskelijoista 18 kappaletta oli toisen, 8 kolmannen, yksi viidennen sekä yksi ensimmäisen vuosiluokan edustaja.

Opiskelijat osallistuivat WIMMA Lab -opintokokonaisuudelle 16.5.2018 – 31.7.218 välisenä aikana.

4.2 WIMMA Labin henkilökunnan kuvaus

WIMMA Labin alkuperäinen perustaja, Marko Rintamäki toimii Jyväskylän ammattikorkeakoulussa tuntiopettajana. Hän on toiminut alusta lähtien WIMMA Labin toimeenpanevana voimana.

WIMMA Labin tukihenkilöinä toimivat Juho Pekki sekä Teemu Kontio, myöskin Jyväskylän ammattikorkeakoulun tuntiopettajia. He ovat osallistuneet itse opiskelijoina aikaisempien vuosien WIMMA Lab -toteutuksiin.

Valmentajaksi kesän 2018 WIMMA Labiin valittiin Minttu Mäkäläinen, toisen vuoden ohjelmistotekniikan opiskelija Jyväskylän ammattikorkeakoulusta.

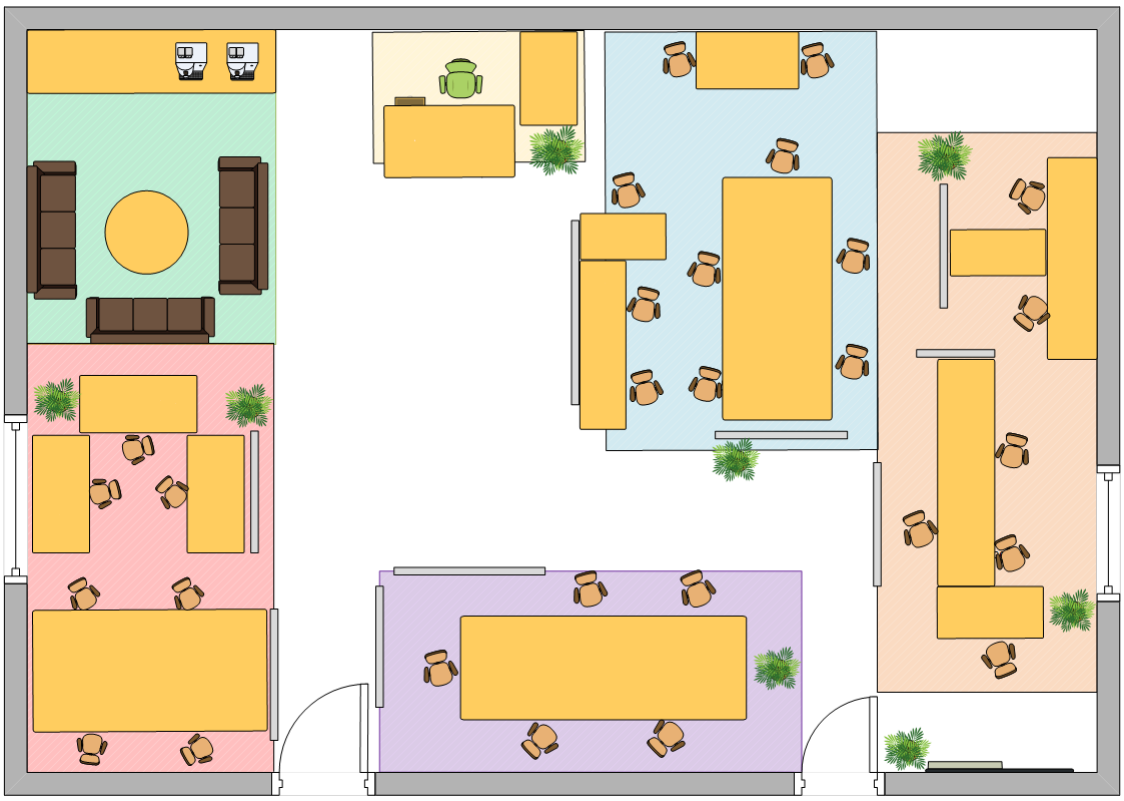
4.3 Työympäristön kuvaus

Kesäksi 2018 Jyväskylän ammattikorkeakoulun IT-instituutista varattiin kaksi luokkahuonetta, jotka yhdistettiin yhdeksi suureksi huoneeksi WIMMA Labin keston ajaksi.

Tiloihin sijoitettiin paikat neljälle tiimille sekä valmentajan työpiste kuvion 15 osoittamalla tavalla. Tiimit pyrittiin sijoittamaan siten, että kullakin tiimillä olisi oma, rauhallinen työympäristönsä sermein ympäröitynä.

Tiloihin järjestettiin myös taukotilat, jonne tuotiin kolme sohvaa rentoutumistarkoitukseen. Taukotilan viereen sijoitetulle pöydälle tehtiin kahvipiste.

Tilojen viihtyvyyden lisäämiseksi sinne haettiin muovisia viherkasveja ja opiskelijat toivat omia julisteitaan seinille.



Kuvio 15. WIMMA Lab 2018 tilojen pohjapiirros (Mäkäläinen 2018, 19)

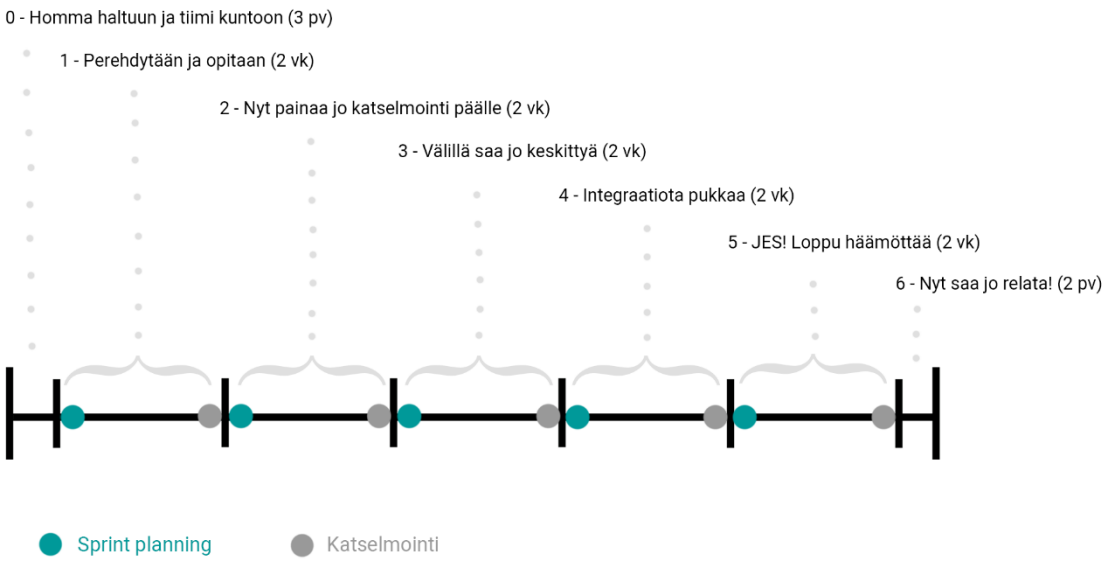
5 WIMMA Lab -prosessin kuvaus

5.1 Alkuvaiheen valmistelut

Ennen WIMMA Labin varsinaista alkamista perehdyttiin tiimiyttämiseen etsimällä verkosta erilaisia harjoituksia, joita voisi käyttää tiimin jäsenten toisiinsa tutustuttamiseen sekä luomaan kaikille yhtenäinen käsitys tiimin saamasta toimeksiannosta. Osa työkaluista saatiin edellisen vuoden WIMMA Lab -toteutuksen materiaaleista, sillä ne oli todettu toimiviksi.

Alkuvaiheessa tutustuttiin myös kesän toimeksiantoihin, jotta niistä saatiin mahdollisimman kattava käsitys. Rintamäen ennalta luomaan sprinttimalliin (ks. kuvio 16) perehdyttiin ja ensimmäiselle sprintille luotiin tehtäviä GitLab-nimisen

verkkopalvelun tarjoamalle Kanban-laudalle, jotta jokainen tiimi pääsisi mahdollisimman nopeasti perehtymään käytettäviin työkaluihin ja menetelmiin.



Kuvio 16. WIMMA Lab 2018 sprintit ja niiden aiheet (Mäkäläinen 2018, 43)

5.2 Sprint 0

WIMMA Lab alkoi virallisesti 16.5.2018, joka oli keskiviikko. Näin ollen ensimmäinen sprintti kesti vain kolme päivää ja sisälsi tiimeihin jakautumisen ennalta määritettyihin kokoonpanoihin. Tiimejä luotiin yhteensä neljä ja ne saivat seuraavanlaisia toimeksiantoja:

- Iotitude, 9 jäsentä
    - Virtuaalikaverin luominen lapselle sairaalaympäristöön.
- Overflow, 7 jäsentä
    - Paikannusjärjestelmä sairaalan lainatavaroille.
- Mysticons, 5 jäsentä

    - Jyväskylän ammattikorkeakoulun LyhytKurssi-järjestelmän automatisointi opintosihteerien työn helpottamiseksi
    - GitLab-palvelun aktiivisuuden seurantajärjestelmä opettajien käyttöön
- Pengwin Media, 6 jäsentä
    - WIMMA Labille uudet kotisivut
    - Grafiikkaa muille tiimeille

Hyvinvointiyksikön opiskelija toimi konsulttina sekä Iotituden että Overflown tiimeissä vuorotellen.

Jokaiselle tiimille oli valittu jo etukäteen tiiminvetäjät valintaprosessissa tehtyjen haastattelujen perusteella, jotta tiimien toiminta saataisiin mahdollisimman sulavasti liikkeelle (Mäkäläinen 2018, 34).

Ensimmäiseen sprinttiin kuului myös toimeksiantojen esittely. Eri toimeksiantojen tilaajat tulivat esittäytymään ja sopivat tarpeen mukaan heidän toimeksiantonsa saaneen tiimin kanssa jatkotapaamisista. Jokainen tiimi alkoi kirjoittamaan toimeksiannostaan vaatimusmäärittelyä.

Työpisteet saatuaan kuntoon tiimit aloittivat myös GitLabiin ja Kanbaniin tutustumisen aiemmin luotujen tehtävien avulla (ks. kuvio 17). Jokainen tiimi loi aluksi itselleen oman projektiympäristönsä, jonne kaikki ottivat saman tehtäväalustan oppimisen pohjaksi. Tiimit tutustuivat itsenäisesti mm. Team Canvasiin, User Storyihin sekä Agile Essenceen ja sopivat keskenään roolien ja vastuiden jakamisesta.

Kuvio 17. GitLabiin luotuja tehtäviä kesältä 2018

Orientaatioviikolla käytiin läpi myös WIMMA Labin yhteiset säännöt, kuten työskentelyajat ja poissaoloista ilmoittaminen sekä sovittiin kahvinkeittovuorot.

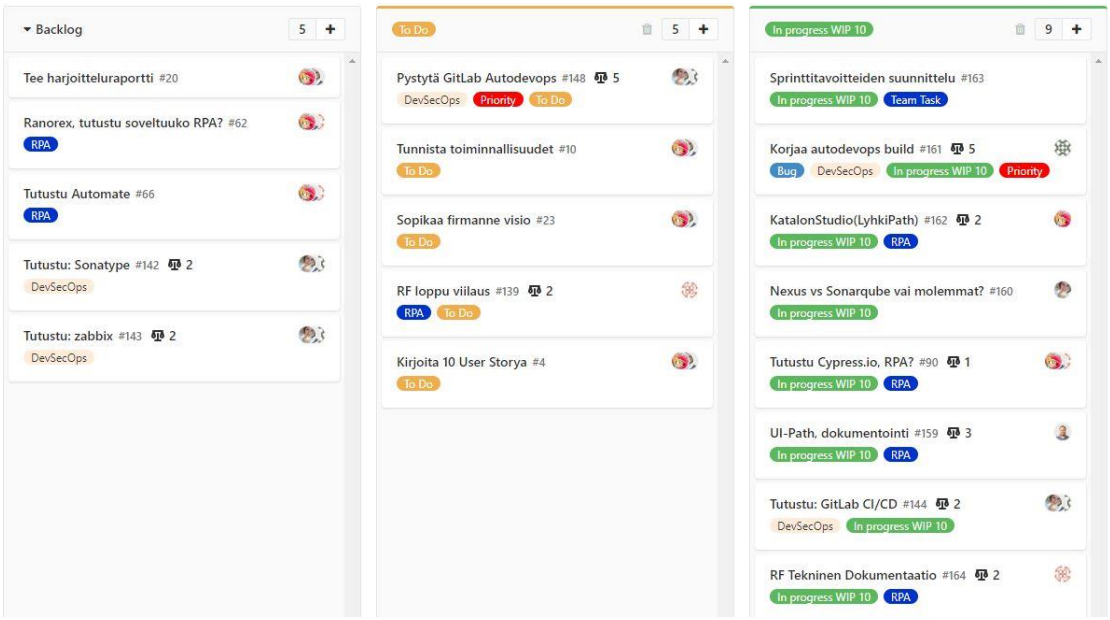
5.3 Sprintit 1 – 5

Sprintit 1- 5 olivat kaikki kahden viikon mittaisia ajanjaksoja, joiden teemat olivat:

- Sprint 1 - Perehdytään ja opitaan
- Sprint 2 - Nyt painaa jo katselmointi päälle
- Sprint 3 - Välillä saa jo keskittyä
- Sprint 4 - Integraatiota pukkaa! On saatava hommat yhteen
- Sprint 5 - JES! Loppu häämöttää

(Mäkäläinen 2018, 41-42)

Jokainen sprintti (joka toinen maanantai) alkoi sprintin tavoitteiden suunnittelemisella, jonka tiimit tekivät itsenäisesti. Tavoitteet kirjattiin tehtäviksi tiimin omille Kanban-laudoille ja niille pyrittiin määrittämään tehtävän kuvaus, tekijä ja tekemiseen kuluva aika. Tiimit loivat omia leimoja (label) helpottamaan töiden lajittelua (ks. kuvio 18). Töiden määrää rajoitettiin WIP-luvuilla.



Kuvio 18. Mysticonsin Kanban-lauta GitLabissa kesällä 2018

Tavoitteiden määrittelemisen jälkeen tiimit työstivät toimeksiantojaan itsenäisesti.

Jokaisen sprintin päätteeksi (joka toinen perjantai) tärkeimmät työt katselmoitiin. Katselmointiin osallistuivat alkuvaiheessa Rintamäki, Mäkäläinen sekä kyseisen tiimin johtaja. Loppuvaiheessa katselmoinnin suoritti Mäkäläinen yhdessä tiiminvetäjän kanssa. Mikäli tehtävissä havaittiin puutteita, ne siirrettiin takaisin tehtäväksi. Muutoin ne voitiin sulkea.

Joka aamu kello 8.15 tiimien vetäjät kokoontuivat seisontapalaveriin Mäkäläisen kanssa keskustelemaan toimeksiantojen edistymisestä ja mahdollisista ongelmista. Jokainen kertoi vuorollaan oman tiiminsä tilanteesta. Tämän jälkeen tiiminvetäjät pitivät vastaavan seisontapalaverin omille tiimeilleen ja kävivät samat asiat läpi yksilötasolla. Viikottain pidettiin myös isompi seisontapalaveri, johon osallistuivat kaikki WIMMA Labilaiset. Tällöin jokainen kertoi parilla sanalla mitä oli tehnyt ja oppinut viikon aikana.

WIMMA Labissa koitettiin myös retrospektiivin pitämistä. WIMMA Labin lyhyen keston vuoksi retrospektiivejä pidettiin joka viikon perjantaina mahdollisten hyötyjen maksimoimiseksi. Aluksi retrospektiiviin osallistuivat vain tiimien johtajat sekä

Mäkäläinen ja he kävivät läpi kuluneen viikon ongelmia koko WIMMA Labin tasolla ja pyrkivät keksimään parannuksia mm. melun tasoon ja työrauhan ylläpitämiseen. Muutaman kerran jälkeen osallistumaan pyydettiin kaikki halukkaat WIMMA Labilaiset. Loppupeleissä retrospektiivistä ei koettu olevan erityisesti hyötyä ja ne päätettiin lakkauttaa.

Jokaista tiimiä rohkaistiin esittelemään työnsä etenemistä toimeksiantajalle mahdollisimman usein ja kysymään palautetta. Haasteita tähän loi kesä, sillä toimeksiantajat katosivat kesälomille ja erityisesti loppuvaiheessa palautetta ei enää saatu. Tiimien tuotteista tehtiin videot, koska lopussa töitä ei pystytty esittelemään kadonneille toimeksiantajille.

5.4 Sprint 6

Viimeinen sprint oli vain kahden päivän mittainen. Näistä ensimmäisenä päivänä tiimit viimeistelivät tuotteidensa dokumentaatiot ja käyttöohjeet luovutuskuntoon. Tämän jälkeen kaikki kokoontuivat yhteen katsomaan tiimien tuotevideoita ja saivat mahdollisuuden esittää kysymyksiä ja antaa palautetta.

Viimeisenä päivänä tilat siivottiin ja luokkahuoneet palautettiin alkuperäiseen tilaansa (Mäkäläinen 2018, 43).

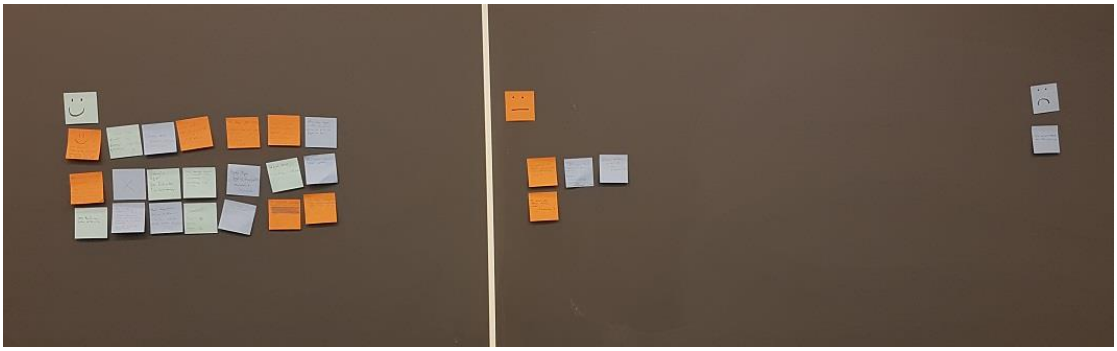
5.5 Käytännön harjoituksien kuvaus ja toteutus

WIMMA Labin aikana toteutettiin useampia harjoituksia mm. tiimiyttämiseen, tuotteeseen tutustumiseen sekä viestintätaitojen harjoitteluun (Mäkäläinen 2018, 44-61).

Sprinttien 0 ja 1 aikana toteutettiin seuraavat harjoitukset:

- Supersankariesittely, jonka tarkoituksena oli tiimin jäseniin tutustuminen rennolla, humoristisella tavalla.
- Team Canvas, jossa kartoitettiin tiimin jäsenten osaamisalueita ja yhteisiä tavoitteita. Täytetyt Team Canvas -paperit laitettiin tiimien seinille.

- Product Canvas ja speksauskävely, jonka aikana tiimi tutustui tuotteeseen ja jäsenet loivat yhtenäisen käsityksen tavoitteesta. Myös Product Canvas -paperit laitettiin tiimien seinille näkyville.
- Moving Motivators, jonka avulla Mäkäläinen kartoitti yksitellen jokaisen WIMMA Labilaisen motivaatiotekijöitä. Mäkäläinen koosti kaikista tuloksista Excel-taulukon, jonka perusteella WIMMA Labilaisten kolme tärkeintä motivaattoria olivat tärkeysjärjestyksessä kyvykkyys, uteliaisuus ja yhteenkuuluvuus.
- Tiimin omat pelisäännöt, jokainen tiimi määritti keskenään yhteiset pelisäännöt, joita jokainen tiimin jäsen sitoutui noudattamaan WIMMA Labin keston ajan.
- Pitchaus- ja kättelyharjoitus, jonka aikana opeteltiin kertomaan omasta tekemisestä lyhyesti ja harjoiteltiin kättelemistä.
- Happiness Wall, jolla testattiin palautteen antamista välittömästi pitchaus- ja kättelyharjoituksen jälkeen. Pitchaus- ja kättelyharjoitus sai 21 myönteistä, 4 neutraalia sekä yhden negatiivisen palautteen (ks. kuvio 19).



Kuvio 19. Happiness Wall -harjoituksen tulokset kesältä 2018

Näiden lisäksi koko WIMMA Labin keston ajan kokeiltiin Lean Coffee -nimistä menetelmää, jonka piti helpottaa avun saamista ei-niin-kiireellisissä asioissa.

Ensimmäisien viikkojen aikana se toimi, kun oli enemmän kysyttävää ohjaajalta. Lopulta se kuitenkin unohtui kaikilta, eikä sitä enää käytetty.

WIMMA Labin alussa kaikille luotiin LinkedIn-tilit ja koko WIMMA Labin keston ajan WIMMA Labilaisia rohkaistiin verkostoitumaan sekä toisiensa, että yrityksistä vierailemaan tulleiden asiantuntijoidenkin kanssa.

Mob Programming -harjoitus pidettiin vasta sprintin 4 aikana, sillä se vaati tiimeiltä jo sisäistä kommunikaatiokykyä ja toisten osaamisen tunnistamista. Harjoituksen tarkoituksena oli kehittää kommunikaatiokykyjä edelleen ja parantaa tiimin toimimista kokonaisuutena. Suurin osa WIMMA Labilaisista piti harjoituksesta kovasti, esimerkiksi Iotituden jäsenet jäivät vielä työajan loputtua puoleksitoista tunniksi tekemään harjoitusta eteenpäin. Iotitude myös otti Mob Programming -tyyppisen tavan projektin loppuvaiheessa käyttöön laajemmin ja tekivät mm. koodinsa refaktorointia tällä tavalla. Mysticons oli alkuun harjoitusta vastaan, koska kokivat olevansa huonoja ohjelmoinnissa. He kuitenkin olivat ainoa tiimi, joka lopulta sai ratkaistua tarjotun ongelman. Mob Programming -harjoitus todisti erinomaisesti, mihin tiimi kokonaisuutena pystyy.

6 Analyysi

6.1 Yleistä

WIMMA Labiin on sulautettu kuluneiden vuosien aikana huikea määrä erilaisia toimintatapoja monista eri menetelmistä, joiden määrää ei ulkopuolelta katsottuna voisi kuvitellakaan. Toimintaa on pyritty systemaattisesti parantamaan aina seuraavaa toteutuskertaa silmällä pitäen. Parhaat työkalut siirretään seuraavalle vuodelle mukaan, kun taas huonommat ovat jääneet matkan varrelle.

Tutkittujen menetelmien mukaisia käytänteitä tunnistettiin lukuisia. Analysointi perustuu Mäkäläisen tekemiin havaintoihin WIMMA Labista kesältä 2018, joita vertailtiin luvussa 2 kuvailtuihin ohjelmistokehityksen menetelmiin ja luvun 3 tiimin kehittämisen malliin.

6.2 WIMMA Lab ja ketterä ohjelmistokehitys

WIMMA Labista löytyy hyvin paljon samoja elementtejä, kuin ketteristä ohjelmistokehityksen menetelmistäkin. WIMMA Lab ei noudata mitään yhtä, tiettyä menetelmää, vaan on ottanut vuosien varrella työkaluja ja käytänteitä monista eri menetelmistä. Luvun 2.1 mukaisesti tämän voidaan todeta olevan ketterää, sillä sopeutuminen on ketterän kehityksen ydin.

Syklien mukainen työskentely toteutui täysin myös WIMMA Labissa. Ensimmäinen ja viimeinen sprintti olivat muita lyhyempiä, itse kehitysvaiheeseen keskittyneet sprintit olivat kukin kahden viikon pituisia. Sprinttien aiheet auttoivat hahmottamaan paremmin tuotteen kehitystä kokonaisuuden kannalta.

WIMMA Labiin pyrittiin valitsemaan mahdollisimman omatoimisia, motivoituneita opiskelijoita. Tiiminvetäjät valittiin jo haastatteluiden perusteella, koska aiempien vuosien perusteella se koettiin tärkeäksi. Muille tiimiläisille ei määritetty erityisiä rooleja, tiimit saivat keskenään päättää sisäisen työnjakonsa. Tiimeille annettiin milteipä vapaat kädet toimeksiannon toteuttamisessa ja aikataulun suunnittelemisessa. Sprinttien aiheet ja henkilökunnan tai toimeksiantajan tekemät teknologiavalinnat määrittivät työn toteutusta tietyllä tavalla, mutta muuten tiimit olivat täysin omatoimisia. Tiimiyttämiseen panostettiin varsinkin ensimmäisien sprinttien aikana huomattavan paljon, jotta tiimin sisäinen yhteistyö saataisiin mahdollisimman luontevaksi. Alkuvaiheessa eri työkalujen käyttöä valvottiin enemmän, sillä suurin osa opiskelijoista ei ollut koskaan käyttänyt esim. Kanban-taulua. Loppuvaiheessa siihen ei enää juurikaan puututtu, kun työkalut kävivät kaikille tutuiksi. Nämä ovat täysin luvun 2.1 itseohjautuvuuden mukaisia menetelmiä; tiimit pyrittiin alusta pitäen rakentamaan ketteriksi ja tiimit toteuttivat työnsä itsenäisesti. Tiimit myös työskentelivät omilla, yhteisillä paikoillaan, jotta kommunikointi ja avun saanti olisi mahdollisimman vaivatonta.

Yksi ketterien kehitysmenetelmien olennaisimpia asioita on asiakkaalta tuleva jatkuva palaute ja sen perusteella työn parantaminen oikeaan suuntaan, kuten luvussa 2.1 kerrotaan. WIMMA Lab toteutettiin kesällä, jolloin yrityksien kesälomien ajoitukset tuovat tähän hyvin suuria haasteita. Kaikki tiimit saivat varsinkin alkuun

tehtävän määrittelyssä apua asiakkaan suunnalta, mutta suurin piirtein juhannuksen tienoilla asiakaskontaktit katkesivat täysin kesälomien vuoksi. Loppuaika jouduttiin kehitystyötä jatkamaan pelkästään alkuvaiheessa saadun informaation perusteella.

Iotitude sai yllättäen mahdollisuuden kerätä palautetta varsinaisilta tuotteen loppukäyttäjiltä. Kurssin ohjaaja tuli katsomaan tiimien tilannetta ja toi omat lapsensa mukaan vierailulle. Lapset pääsivät pelaamaan Iotituden valmistamaa virtuaalikaveria ja heiltä pyrittiin saamaan mahdollisimman monipuolista palautetta erilaisin kysymyksin. Loppukäyttäjien konsultointi kuuluu luvun 2.1 mukaan ketterään ohjelmistokehitykseen.

6.3 WIMMA Lab ja ohjelmistokehityksen kaksitoista periaatetta

Ketterän ohjelmistokehityksen kaksitoista periaatetta ilmenevät erittäin hyvin WIMMA Labin toiminnasta, joskin puutteitakin tietyillä osa-alueilla on.

- Asiakkaan tarpeet (oletettavasti) täyttävä versio tuotetaan erittäin nopeasti ja sitä muokataan saadun palautteen mukaisesti (kohdat 1 ja 2 täyttyvät).
- Saatavilla on jatkuvasti uusin versio tuotteesta (kohta 3 täyttyy).
- Opintokokonaisuuteen ei kuulu liiketoimintaa, joten ainoastaan ohjelmiston kehittäjät työskentelevät yhdessä päivittäin WIMMA Labin keston ajan (kohta 4 täyttyy).
- WIMMA Labiin pyritään haastatteluiden perusteella ottamaan ainoastaan motivoituneita opiskelijoita. Tukea on aina tarjolla ja tiimit toimivat itsenäisesti (kohta 5 täyttyy).
- Niin valmentaja kuin opiskelijatkin työskentelevät kaikki samassa huoneessa ja pitävät säännöllisesti palavereita (kohta 6 täyttyy).
- Toimivan prototyypin valmistaminen on erinomaista ja tavoiteltavaa, mutta se ei varsinaisesti ole opintokokonaisuuden edellytys (kohta 7 ei täyty).
- Loppuvaiheessa opiskelijoiden kiinnostus alkoi vähitellen hiipumaan, projekteja ei olisi voinut jatkaa enää kauaa (kohta 8 ei täyty).
- Tekninen suunnittelu ja toteutus tehtiin tiimin parhaan osaamistason mukaisesti (kohta 9 täyttyy).

- Työtä tehtiin toisinaan puhtaasti kokeilumielessä ja tiimin ideoiden pohjalta, joten epäilemättä osa siitä oli turhaa ja ylimääräistä (kohta 10 ei täyty). Oppimismielessä kokeilut ovat kuitenkin erittäin oleellisia.
- Tiimit saivat tehdä toteutuksensa itsenäisesti (kohta 11 täyttyy).
- Tiimit eivät tarkastelleet omaa tehokkuuttaan tai sen parantamista (kohta 12 ei täyty).

Täyttymättä jääneiden kohtien taustasyynä on pääasiassa projektin luonne, eli kyseessä on koulun järjestämä opintokokonaisuus, jolloin vasta opetellaan työelämässä käytettyjä toimintatapoja.

6.4 WIMMA Lab ja Lean

Luvun 2.3.1 mukaan Lean perustuu virtauksen maksimointiin ja hukan poistamiseen.

Hukan eri muotojen poistoon käytettävää Kaizenia ei ainakaan tietoisesti saatu toteutettua. Mäkäläinen toimi valmentajan roolissaan tiimien ulkopuolella, kun taas esitellyt hukan muodot kohdistuvat pääsääntöisesti tiimin sisäiseen työskentelyyn. Tiimien jäsenten olisi pitänyt itse hoitaa tietoisesti omien käytänteidensä tarkkailu ja näiden kautta pyrkiä parantamaan toimintojaan.

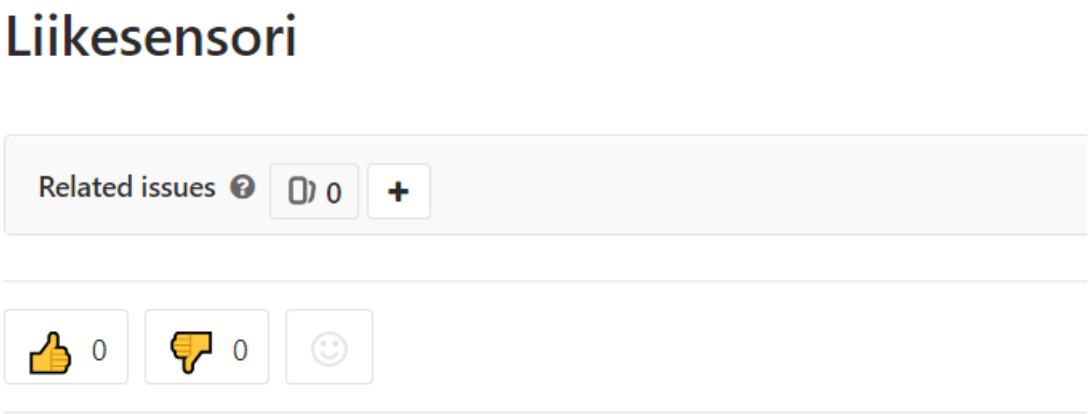
Mäkäläisen työ WIMMA Labin valmentajana kesän aikana muistutti hieman luvussa 2.3.1 kuvailtua Gemba-kävelyä. Hän oli koko kesän paikalla tarkkailemassa ja opettelemassa WIMMA Lab -prosessia ja pyrki kehittämään toimintaa pääsääntöisesti tiiminvetäjien kanssa.

Mäkäläinen sai kutsuttua käymään Jyväskylän alueelta useampia henkilöitä, joilla oli monen vuoden kokemus projektien vetämisestä sekä ketterien menetelmien käyttämisestä ohjelmistokehityksessä. He jakoivat asiantuntemustaan ja toivatkin useita erilaisia ideoita, joita kokeiltiin käytännössä kesän aikana. Mob Programming sekä retrospektiivit olivat tulosta heidän käynneistään. Tämä kuvastaa hyvin luvussa 2.3.1 kuvattua johdon sitoutumista kehitystyöhön.

6.5 WIMMA Lab ja Kanban

Luvussa 2.3.2 esitelty Kanban oli käytössä WIMMA Labilaisilla koko kesän ajan GitLab-verkkopalvelun kautta. Kanbanin käyttöä opeteltiin heti ensimmäisestä päivästä lähtien ja kaikki töiden hallinta kulki laudan kautta. Tiimit määrittivät kukin itselleen oman WIP-lukunsa ja seurannan mukaan pysyivät sen asettamissa rajoissa erinomaisesti.

Katselmointien perusteella korjattavaksi palautetut tehtävät olivat pääasiallisesti sisällöltään puutteellisia (ks. kuvio 20), sillä osa tiimien jäsenistä ei kokenut tärkeänä kirjoittaa vaadittuja tietoja ylös ennen tehtävän tekemistä. Tehtävään tulisi kirjoittaa luvun 2.3.2 mukaan mitä pitää tehdä, tekemiseen arvioitu aika sekä miten tehtävä on tarkoitus tehdä. Jokaisen tiimin jäsenen pitäisi näistä ymmärtää samalla tavalla, miten kyseenomainen tehtävä tehdään. Tätä koitettiin käydä yhdessä läpi ja selventää, miksi tehtävät tulisi kirjoittaa auki, mutta huonolla menestyksellä. Koko tiimi työskenteli fyysisesti hyvin tiiviissä tilassa ja osa heistä koki sanallisen kommunikoinnin tehokkaammaksi, kun he kerta kuitenkin tekivät tehtävät itse.



Kuvio 20. Esimerkki huonosti täytetystä tehtävästä Kanban-laudalla kesältä 2018

Työtä pyrittiin visualisoimaan myös kaikkien WIMMA Lab -projektien yhteisellä, fyysisellä Kanban-taululla, jolla kuvattiin projektien etenemistä sprinttitasolla. Se kuitenkin jäi miltei huomiotta, eikä siitä ollut erityisesti hyötyä.

Visualisointia pyrittiin parantamaan myös viikoittaisen kalenterin sijoittamisella huoneen keskeiselle paikalle, heti taukotilan viereen. Ensin koko WIMMA Labin yhteinen kalenteri oli verkossa, mutta jonkin ajan jälkeen huomattiin, etteivät sitä pitäneet silmällä kuin tiiminvetäjät. WIMMA Labilaisen ehdotuksesta kalenterista tehtiin fyysinen versio tiloihin, joka toimikin saadun palautteen mukaan erinomaisesti. Tämä vastaa hyvin niin luvussa 2.3.2 kerrottuun visualisoinnin parantamiseen, kuin myös luvussa 2.3.1 kuvailtuun työn kehittämiseenkin.

6.6 WIMMA Lab ja Scrum

WIMMA Lab -tasolla eniten yhtäläisyyksiä oli Scrumin mukaisessa tiimijaossa. Rintamäki oli lähinnä Product owneria, kun taas Mäkäläinen toimi enemmän Scrum-mestarin -tapaisessa roolissa. Tiimit toimivat kehitystiimin roolissa. Ne eivät kuitenkaan vastanneet Scrumia täydellisesti, sillä Scrumissa ei luvun 2.3.3 mukaan tunnusteta kehitystiimin jäsenien omia titteleitä (kuten tiiminvetäjä). Myöskään tiimien sisäisiä alitiimejä ei tunnusteta, mutta Mysticons jakautui saamiensa kahden erilaisen toimeksiannon perusteella kahteen osaan. Tiiminvetäjää piti välillä muistuttaa, että hänen pitäisi tietää koko tiiminsä tekemisistä, ei vain oman alitiiminsä tekemisistä.

Vakiintuneet palaverit vastasivat miltei täysin Scrumissa käytettyjä palavereita. WIMMA Labin alussa jokainen tiimi kirjoitti summittaisia tehtäviä oman Kanban-lautansa Backlogiin ja jokaisen sprintin alussa suunnittelupalaverin jälkeen niitä tarkennettiin ja otettiin sprintille tehtäväksi. Usein uusia tehtäviä lisättiin sitä mukaa, kun niitä keksittiin. ”Valmiin” määritelmää ei kirjattu tehtäviin kertaakaan, vaan tehtävän todettiin yleensä olevan valmis, kun tekijä itse sekä toinen tiimin jäsen päättivät niin yhdessä.

Alkuvaiheessa tiimit eivät pitäneet omia päiväpalavereja ollenkaan, mutta suurin piirtein sprint 1:n aikana käytänne vakiinnutettiin.

Sprinttikatselmukset pidettiin jokaisen sprintin päätteeksi, joskin vain tärkeimmät työn osa-alueet käytiin yhdessä läpi. Varsinaiset tiimien asiakkaat eivät osallistuneet yhteenkään katselmointitilaisuuteen, vaan tiimien jäsenet päättivät jatkosta keskenään.

Retrospektiivejä koitettiin muutaman viikon ajan, kunnes niiden todettiin olevan hyödyttömiä. Retrospektiivit toimivat vielä silloin, kun vain tiiminvetäjät osallistuivat niihin. Heillä tuntui olevan parempi käsitys WIMMA Labin kehittämisestä kokonaisuutena. Kun tiimin jäsenet tulivat mukaan, keskustelujen aiheiksi nousivat usein työkaverin huonot käytöstavat ja muut, yleisesti ilmapiiriä laskevat aiheet. Varsinaista toimintaa parantavia päätöksiä ei enää saatu tehtyä, joten retrospektiivit päätettiin lopettaa. Retrospektiivit järjestetään luvun 2.3.3 mukaan Scrumissa aina sprintin päätteeksi, mutta WIMMA Labissa ne pidettiin joka viikko opintokokonaisuuden lyhyen keston vuoksi.

Scrumissa määritetty Scrumien Scrum pidettiin heti alusta asti päivittäin. Tämän koettiin olevan erittäin hyödyllinen tapa jakaa tietoa eri tiimien välillä. Toisinaan yhden tiimin ongelmiin saatiin välittömästi apua toisen tiimin puolelta ja näin nopeutettiin ongelmanratkontaa huomattavasti. Mäkäläinen sai myös erinomaisen mahdollisuuden välittää kaikkia WIMMA Labilaisia koskevia tiedotteita tiiminvetäjien välityksellä eteenpäin.

Luvussa 2.3.3 kuvaillut Scrumin tuotokset vastasivat miltei täysin myös WIMMA Labin tuotoksia. Tuotteen kehitysjonoon tehtiin erityisesti WIMMA Labin alussa suuremman tason tehtäviä, jotka tultaisiin toteuttamaan kesän aikana. Tiimi itse ylläpiti kehitysjonoaan, kun taas Scrumissa se on tuotteen omistajan vastuulla. Tehtäväksi siirrettiin aina kuluvalle sprintille valitut tehtävät ja tarpeen mukaan niitä luotiin lisää. Edistymiskäyrä oli jokaisella tiimillä olemassa, mutta sitä ei juurikaan pidetty silmällä. Sprinttikatselmuksissa sitä yleensä katsottiin yhdessä ja pohdittiin sen tarkoitusta.

6.7 WIMMA Lab ja Six Sigma

Luvussa 2.3.4 kuvailtua Six Sigmaa, tai sen tapaisia käytänteitä ei ollut havaittavissa missään WIMMA Labin vaiheessa.

Mikäli Six Sigman rooleja vertaillaan WIMMA Labissa esiintyneisiin rooleihin, tiimien jäsenet olisivat vastanneet lähinnä keltaista vyötä. Mäkäläinen olisi valmentajan roolissaan ollut vihreä vyö ja Rintamäki ohjaajan roolissaan musta vyö. Todellisuudessa tämä ei kuitenkaan ole lähelläkään Six Sigman tarkoittamia roolikuvauksia.

6.8 WIMMA Lab ja Extreme Programming

Luvussa 2.3.5 kerrottiin Extreme Programming -menetelmässä käytettävästä parikoodauksesta, jota ilmeni tietyllä tapaa myös WIMMA Labin aikana. Menetelmä koettiin erittäin hyödylliseksi varsinkin niiden henkilöiden mielestä, joilla oli hieman vähemmän kokemusta ohjelmoinnista. Pääsääntöisesti koodauksen tekikin vahvempi ohjelmoija, kun taas kokemattomampi istui vieressä katsomassa. Esitellyn parikoodauksen idean mukaan vuoroja tulisi kuitenkin vaihtaa tietyin väliajoin, jota ei WIMMA Labissa tehty.

WIMMA Labin alussa jokainen tiimi koitti luoda omasta toimeksiannostaan käyttäjätarinoita (ks. kuvio 21), jotka auttoivat toimeksiannon ymmärtämisessä. Käyttäjätarinoita ei kuitenkaan erikseen priorisoitu tärkeysjärjestykseen, eikä niiden toteuttamiselle arvioitu aikaa.

As a...	I want...	So that...
Child	To be able to change my virtual friend to a different one	I can make it look the way I want
Software Developer	to be able to shake the RuuviTag and have the system react to it in some way	I can control the game scenes with it
Patient	My friend to greet me	I feel welcome
Developer	features, pretests and user stories in mind map	I can document more easily
Patient	my virtual friend to move	it feels more alive
developer	features, pretests, and user stories in mind map	i can document more easily
Patient	my friend to greet me	i feel welcome

Kuvio 21. Iotituden luomia käyttäjätarinoita kesältä 2018

XP:ssä asiakkaan tulisi olla mukana ohjelmiston kehityksessä päivittäin, mutta WIMMA Labissa näin ei ollut. Asiakkaalta saatiin toimeksiantoon riittävät tiedot jo alkuvaiheessa, osa tiimeistä sai palautetta muutamaan otteeseen myös myöhemmin. Näin ollen asiakkaat eivät myöskään kirjoittaneet minkäänlaisia hyväksyntätestejä, joilla työtä olisi voitu arvioida. Tiimit vastasivat itse tehtävien määrittämisestä sykleille, kun XP:ssä tämäkin olisi ollut asiakkaan vastuulla.

Sen sijaan sykleissä työskentely ja päivittäiset seisontapalaverit toteutuivat myös WIMMA Labissa. Erityisiä tiimikohtaisia palavereita syklien päätteeksi kuitenkaan ei pidetty.

WIMMA Labin aikana kokeiltiin myös Mob Programmingia (Mäkäläinen 2018, 59-61). Suurin ero parikoodaukseen verrattuna on siinä, että parikoodaukseen osallistuu kaksi henkilöä, kun taas Mob Programmingia tekee koko tiimi samaan aikaan. Mob Programmingin aikana jokainen henkilö pääsi koodaamaan useamman kerran, vaikka harjoitukseen osallistuikin Iotituden ja Overflown tapauksissa kahdeksan henkilöä. Osa Iotituden jäsenistä käytti varsinkin WIMMA Labin loppuvaiheessa Mob Programming -tyyppistä työskentelytapaa tekemänsä koodin refaktoroinnissa oppimisen maksimoimiseksi.

6.9 WIMMA Lab ja DevOps

Iotitude sekä Overflow käyttivät koodinsa hallinnoimiseen GitLabin tarjoamaa versionhallintaa, joten päivän päätteeksi saatavilla oli aina uusin versio palvelusta. Tämä vastasi eniten DevOps-tuotekehityksessä käytettyä mallia, josta kerrottiin luvussa 2.3.6.

Iotituden toimeksiantona oli rakentaa virtuaalikaveri sairaalaympäristöön. Virtuaalikaverin tuli tehdä erilaisia asioita eri tilanteissa, joka toteutettiin luomalla useita, yksittäisiä näkymiä. Eri näkymien toteuttaminen jaettiin tiimin jäsenien kesken, joka vastaa melko pitkälti DevOpsin itsenäistä mikropalveluiden kehittämistä.

Iotitude lisäsi mukaan vielä turvallisuusnäkymän, sillä yksi tiimiläinen keskittyi alusta alkaen pääasiassa koodin analysointiin ja automaattisten testien tekemiseen.

DevSecOpsia se ei kuitenkaan täysin vastaa, sillä silloin kaikkien tiimin jäsenten tulisi pitää huoli tietoturvan säilyttämisestä.

Mysticonsin jäsenet pitivät koodinsa pääosin omilla koneillaan. Tästä aiheutui ongelmia, kun yhden tiimiläisen kahden viikon työ korruptoitui, eikä sitä saatu palautettua. Työ jouduttiin tekemään alusta alkaen uudelleen.

6.10 WIMMA Labin tiimitoiminta

Luvussa 3.2 kuvaillut tiimin kehitysvaiheet olivat enimmäkseen havaittavissa WIMMA Labin aikana.

Perustamisvaiheen kuvaus vastaa sprinttien 0 ja 1 vaiheita erinomaisesti. Näillä sprinteillä opiskelijat saapuivat taloon ja heidät sijoitettiin eri tiimeihin. Yhteisten työkalujen käyttöä opeteltiin ja tiimit osallistuivat useisiin tiimiintymisharjoituksiin valmentajan opastuksella.

Myrskyvaihe oli havaittavissa oikeastaan vain kahden osallistujan kohdalla. Yhteistyö tuntui olevan haastavaa ja erilaisia konflikteja nousi esiin. Molemmat päätyivät jättämään WIMMA Labin kesken erilaisten syiden vuoksi myöhemmin.

Oppimisvaihe ilmeni käytännössä parhaiten sprinttien 2 ja 3 aikana, jolloin tiimit tutustuivat syvemmin tarvitsemiinsa teknologioihin. Moni tiimi kävi myös viettämässä yhdessä iltaa vahvistaen samalla itsenäisesti tiiminsä identiteettiä.

Sprintit 4 ja 5 vastasivat eniten suoritusvaiheen kuvausta, sillä valmistettavien palveluiden raamit oli jo rakennettu ja nyt keskityttiin pääasiassa paranteluihin ja viimeistelyyn. Mikään tiimi ei alkanut käyttäytymään ylimielisesti, vaan loppuvaiheessa esiintyi pikemminkin enemmän tiimien välistä yhteistyötä.

Hajoamisvaihe oli selkeästi sprintillä 6, kun paikat siivottiin ja opintokokonaisuus päättyi.

Pengwin Median kanssa oli alkuvaiheessa suuria ongelmia motivaation kanssa. Tiimiin oli valittu neljän hengen tiivis kaveriporukka, jotka olivat tottuneet tiettyyn ”lauman arvojärjestykseen”. Erityisesti yksi henkilö aiheutti koko tiimiin motivaatio-

ongelmia käyttäytymisellään ja tämä näkyi töiden tekemättömyytenä. Tiiminvetäjällä oli suuria vaikeuksia saada kaveriporukan jäsenet tekemään töitä. Ratkaisuna tämä yksi ongelmia aiheuttanut henkilö jätti WIMMA Labin kesken. Asia käsiteltiin saman päivän aikana Mäkäläisen ja tiimin jäsenten kesken ja kaikille saatiin yhteisymmärrys tilanteesta. Motivaatio-ongelmat katosivat ja tiimi työskenteli huomattavasti aikaisempaa tehokkaammin. Tämä vastaa täysin luvun 3.3 kuvaamaa sitoutumatonta henkilöä, joka ”saastuttaa” tiimin ilmapiiriä ja laskee tehokkuutta.

Yksi suurimmista haasteista koko kesän aikana oli WIMMA Labin tiloissa esiintynyt meluisuus. Yhteen luokkahuoneeseen oli sijoitettu yhteensä 29 henkilöä, joiden aiheuttama puheensorina oli omiaan aiheuttamaan hermojen kiristymistä. Meluisuutta vähentämään tiloihin oli tuotu sermejä, mutta niistä ei juurikaan ollut hyötyä. Useat opiskelijat toivatkin mukanaan omia kuulokkeita, joita käyttivät työpäivän ajan. Retrospektiivin tuloksena yhden viikon ajan koitettiin ”hiljaisen tunnin” pitämistä, eli kaikki olivat puhumatta joka päivä kello 13-14 ja kaikki viestintä hoidettiin sähköisesti. Jälkikäteen pidetyn äänestyksen tuloksena 15 ihmistä halusi jatkaa hiljaisen tunnin pitämistä, kun taas 11 halusi lopettaa sen. Vähitellen seuraavan viikon aikana se kuitenkin hiljalleen katosi ja meluisuus palasi. Luvun 3.3 mukaan tämä on yleisiä avokonttorimaisen työskentelytilan tuomia haasteita, jolla on useita negatiivisia vaikutuksia.

7 Tulokset

7.1 Miten hyvin WIMMA Lab seuraa aikaansa?

Kuten jo luvussa 1.1 todettiin, ketterien tiimien ohjaukseen käytettävät mallit tulevat jatkossakin säilyttämään vahvan asemansa ohjelmistokehityksessä. Ketteryyden todellinen merkitys tulisi sisäistää ja periaatteet pitäisi ymmärtää, pelkkä työkalujen käyttäminen ei riitä. Ohjelmistot tulisi myös saada nopeasti valmistettua ja jäljen pitäisi olla laadukasta.

Tähän verrattuna WIMMA Labin pitäisi parantaa hieman. Vaikka tiimien ohjauksessa sovelletaan lukuisia erilaisia ketteriä käytänteitä, niiden taustalla olevia periaatteita ei käydä erikseen läpi missään vaiheessa. Ei riitä, että vain henkilökunta pyrkii

ymmärtämään ketteryyttä, vaan myös opiskelijat tulisi sisällyttää mukaan. Taustalla olevan filosofian ymmärtämisen puolesta puhuvat myös useammat luvussa 2.3 esitellyt menetelmät.

Scrum on kiistattomasti käytetyin ketterä menetelmä, joten siihen nähden WIMMA Lab on sisällyttänyt Scrumin toimintatapoja mukaan erinomaisesti.

7.2 Millaisia ovat ohjelmistokehityksen trendit nyt ja millaisia uusia metodeja on tullut tarjolle?

Kuviossa 1 esitetyn aikajanan perusteella uusimmat trendit ovat Lean, Kanban ja DevOps. Näistä erityisesti DevOps tulee luvun 1.1 mukaan yleistymään jatkossa laajemmin.

Suurimpia trendejä ovat nykypäivänä jatkuva testaus, tiimityöskentely, nopea palaute sekä mobbaus. Käyttäjien tyytyväisyys on yksi tärkeimmistä menestyksen mittareista.

Vanhempiin luvussa 2.3 tutkittuihin malleihin kuuluvat Scrum, Six Sigma sekä Extreme Programming. Vaikka Scrum ei varsinaisesti uusi olekaan, se on silti käytetyin ketterä menetelmä vuonna 2018.

7.3 Mitä näistä olisi mahdollista kokeilla tulevissa WIMMA Lab -toteutuksissa?

WIMMA Lab hyödyntää jo nykyisellään miltei jokaisen opinnäytetyössä tutkitun menetelmän parhaiksi koettuja käytänteitä ja työkaluja.

Näistä kuitenkin Extreme Programmingin osuutta voisi laajentaa, sillä se koettiin erityisen hyväksi opiskelumuodoksi. Mobbaus on yksi luvussa 1.1 tunnistetuista trendeistä ja käytännön kokeilun kautta se sai myös positiivista palautetta, joten se tulisi tavalla tai toisella sisällyttää mukaan. Tämä vahvistaa myös tiimin yhteisvastuun ottamista sekä parantaa kommunikointia. Kenties tämän suhteen voisi suorittaa kokeiluja tekemällä työtä Mob Programming -tyyppisesti vaikka yhden päivän viikossa.

DevOps- sekä DevSecOps -tuotekehitysmalleihin kannattaa jatkossakin panostaa, sillä tuotteiden laadun varmistus ja turvallisuus ovat nykyaikana tärkeitä. Turvallisuudesta huolehtiminen pitää tuoda kaikkien tiimin jäsenten tehtäväksi jo periaatetasolla.

Palautteen saamista asiakkaalta tulisi parantaa, sillä nykykäytäntö ei ole paras mahdollinen. Opiskelijoita tulisi myös totuttaa asiakkaan kanssa työskentelyyn paljon nykyistä laajemmin, sillä heiltä tahtoi toisinaan unohtua, että toimeksiantoa tehdään oikealle ihmiselle. Tämä auttaa niin kommunikoinnin kuin paremman tuotteenkin kehittämisessä ja varmistaa tyytyväisemmän asiakkaan. Asiakkaan voisi vähintäänkin kutsua katselmointeihin mukaan, jotta he oikeasti tulisivat paikan päälle katsomaan työn edistymistä. Mikäli asiakas on suostuvainen, niin miksei häntä voisi kutsua myös sprintin suunnittelupalavereihinkin osalliseksi.

Scrumin käyttöä tulisi edelleen jatkaa, sillä se on ketteristä menetelmistä ylivoimaisesti käytetyin.

8 Pohdinta ja jatkokehitys

Aihe kokonaisuudessaan oli erittäin laaja. Ketterät menetelmät ovat saaneet alkunsa miltei kolmekymmentä vuotta sitten, joten ”nykyaikaisuuden” määritelmää jouduttiin pohtimaan uudelleen läpi työn.

Aineistoa oli tarjolla tietyissä tapauksissa jopa liikaakin, joten oli erittäin vaikeaa löytää yksi ”oikea” linjaus. Oikea se ei varmasti kaikkien mielestä olekaan, sillä kuten luvussa 1 todettiin, ketteristä ohjelmistotuotannon menetelmistä puhuessaan ihmiset usein tarkoittavat eri asioita. Ketteryys tuntuu toisinaan olevan milteipä uskontoon verrattavissa oleva asia, joten mielipiteitä riittää.

Mahdollisia tutkittavia menetelmiä ja työkaluja on olemassa nykypäivänä todella paljon, mutta johonkin raja oli vedettävä. Kaikkia ei yksinkertaisesti voitu valita, jotta työn määrä pysyisi realistisena. Jatkon kannalta kuitenkin kannattaisi vähintäänkin tutustua menetelmiin, nimeltä Scaled Agile Framework (SAFe), Disciplined Agile Framework (DA), Crystal Methodology, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM) sekä LeSS Framework.

Tutkimus toteutettiin hieman väärin päin, eli ensin tutkittiin ketterien menetelmien käyttöä WIMMA Labin aikana ja seuraavaksi tutustuttiin lähdemateriaaliin. Tämä on epäilemättä vaikuttanut osaltaan kirjoitettuun tekstiin ja pohdintoihin. Mikäli lähdemateriaaliin olisi tutustuttu ensin, tutkittavien menetelmien otanta olisi epäilemättä ollut satunnaisempi.

Esitettyihin kysymyksiin löydettiin vastaukset. Eri lähteistä löytyi monenlaista eri tietoa vastauksia etsittäessä, mutta yhteistä näille kaikille oli ketteryyden ymmärtäminen. Ketteryyden ajatusmaailmaa kannattaisi avata opiskelijoille enemmän, sillä pelkkien työkalujen käytön opetteleminen ei riitä. Luultavasti kuitenkaan tämä ei tule kokonaisuudessaan onnistumaan, sillä se on erittäin haastavaa yrityselämänkin puolella.

Ketterän tiimin jäsenten tulisi ideaalitilanteessa vastata yhdessä kaikesta tekemisestään. Mikäli jatkuvan parantamisen periaatteita ei sisäistetä, yhteisvastuun ottaminen saattaa jäädä huomiotta. Sitoutuminen ei välttämättä ole opiskelijan pääprioriteettina koulun järjestämällä opintokokonaisuudella.

Mikäli opiskelijoiden ajatusmaailmaan saataisi iskostettua Kaizenin periaatteita, hukan poistoa voisi koittaa järjestelmällisemmin eri työkalujen avulla pitkin WIMMA Labia. Tämä vaatisi erityistä huomiota, jota ei kesällä 2018 järjestetyn WIMMA Labin aikana ymmärretty osoittaa. Myös virtauksen maksimointia voisi tietyillä osa-alueilla yrittää. Kuitenkin on otettava huomioon, että kyse on opiskeluympäristöstä, jossa kokeilut ja virheiden tekeminen on sallittua, joten pääarvon ei pitäisi olla virtauksessa.

Ketterä kehitys vaikeutuu huomattavasti asiakkaan lähtiessä kesälomalle. Yritysmaailman edustajat saattavat myös laiminlyödä velvollisuuksiaan, koska kyseessä on ”vain opiskelijaprojekti” ja oikeat työt odottavat. Tämä todennäköisesti tulee olemaan myös tulevien WIMMA Lab -toteutuksien haasteena. Tässä voi käydä myös toisinkin päin, eli opiskelijat jatkavat projektia alussa saatujen tietojen perusteella ja unohtavat näyttää välituloksia tavoitettavissa olevalle asiakkaalle. Tätä tulee pitää silmällä ja muistuttaa asiasta tarvittaessa.

Tutkituista menetelmistä Six Sigmaa tuskin voidaan käyttää WIMMA Lab -kontekstissa ollenkaan. Six Sigma vaatii mitattavissa olevia tuloksia sekä tilastollisten työkalujen hallintaa. Kumpaakaan ei näillä näkymin ole saatavilla.

Tiimeissä työskenteleminen on ehdottomasti paras toteutustapa, sillä näin opiskelija ei ole missään vaiheessa yksin. Ympärillä oleva tiimi tukee ja motivoi tarvittaessa ja osaamisen jakaminen onnistuu paremmin. Tämä myös tukee Extreme Programmingin ideaa, jota kannattaisi jatkossa hyödyntää laajemmin.

Opinnäytetyön aikana opin eniten tiimin toiminnasta sekä sen erilaisista kehitysvaiheista. Oli hämmentävää huomata, miten hyvin kesän tapahtumat vastasivat Bruce Tuckmanin viisikohtaiseen tiimin kehitysvaihemalliin. Epäilemättä teoriaan olisi kannattanut kokonaisuudessaan tutustua paremmin jo etukäteen, niin ketterien menetelmien kuin tiimitoiminnan kehittämisen osaltakin. Tällöin olisin kenties voinut tehdä havaintoja täsmällisemmin kuluneen kesän aikana. Nykyisellään olin lähinnä tarkastelemassa WIMMA Labin prosessia sekä sen aikana käytettyjä menetelmiä ja pyrin oppimaan niistä mahdollisimman paljon käytännön työskentelyn kautta. Eniten harmittamaan jäi se, ettei kaikkiin menetelmiin pystynyt opinnäytetyön puitteissa tutustua, sillä muuten työmäärä olisi paisunut liikaa. Kaikkinensa olen tyytyväinen saavuttamaani osaamiseen, sillä olen saanut hyvän peruskäsityksen erilaisten ketterien menetelmien luonteesta ja miten niitä voisi parhaiten hyödyntää erilaisissa projekteissa.

Lähteet

Agile Practices Timeline. N.d. Aikajana Agile Alliancen sivuilla. Viitattu 6.9.2018. https://www.agilealliance.org/agile101/practices-timeline/.

Altman, H. 2017a. Agile Project Management: Quick-Start Guide For Beginners And How To Implement Agile Step-By-Step. CreateSpace Independent Publishing Platform.

Altman, H. 2017b. Kanban: Step-by-Step Agile Guide Designed to Help Teams Working Together More Effectively. CreateSpace Independent Publishing Platform.

Altman, H. 2017c. Scrum: The First Agile Methodology For Managing Product Development Step-By-Step. CreateSpace Independent Publishing Platform.

Barrios, J. N.d. The Software Development Life Cycle: An Introduction for Business Analysts. Artikkeli. Viitattu 2.9.2018. https://www.joebarrios.com/the-software-development-life-cycle-for-the-beginning-business-analyst/.

Bisiani, B. N.d. Kaizen-tekniikalla hiljaa hyvä tulee. Kirjoitus Genesta Magazinen sivuilla. Viitattu 1.9.2018. https://mag.genesta.eu/fi/kaizen-tekniikalla-hiljaa-hyv%C3%A4-tulee.

Cockburn, A. 2006. Agile Software Development: The Cooperative Game. 2. p. Upper Saddle River (NJ): Addison-Wesley.

Dawson, B. 2015. Hybrid Development Workshop Presentation. Kuva San Franciscossa pidetyn työpajan diashowsta. Viitattu 6.9.2018. https://www.slideshare.net/brianvdawson/hybrid-development-workshop-presentation-san-francisco.

DMAIC – The 5 Phases of Lean Six Sigma. N.d. Blogikirjoitus Go Lean Six Sigma -sivulla. Viitattu 2.9.2018. https://goleansixsigma.com/dmaic-five-basic-phases-of-lean-six-sigma/.

Hämäläinen, H. 2016. Ohjelmistotuotekehitys-organisaatioiden tulevien vuosien trendit. Kirjoitus Contribyten sivuilla. Viitattu 2.9.2018. https://contribyte.fi/2016/01/10/ohjelmistotuotekehitys-organisaatioiden-tulevien-vuosien-trendit/.

Julistuksen takana olevat periaatteet. 2011. Ketterän ohjelmistokehityksen julistuksen periaatteet. Viitattu 30.8.2018. http://agilemanifesto.org/iso/fi/principles.html.

Jälleen uusi tutkimus osoittaa avokonttorin ongelmat, jotka ovat pahenemaan päin – Juuri kukaan ei pysty työskentelemään häiriöttä ilman erityisiä toimenpiteitä. 2018. Artikkeli Talouselämän sivuilla. Viitattu 6.9.2018. https://www.talouselama.fi/uutiset/jalleen-uusi-tutkimus-osoittaa-avokonttorin-ongelmat-jotka-ovat-pahenemaan-pain-juuri-kukaan-ei-pysty-tyoskentelemaan-hairiotta-ilman-erityisia-toimenpiteita/64b762e6-c0b7-3ca9-a4b9-f76949ee244e.

Kangas, M., Kirjavainen, A., Fallenius, E. & Mikkonen, K. 2018. Yle Lean Culture Toolkit. Ylen tekemä Lean-työkalupaketti. Viitattu 1.9.2018. https://drive.google.com/file/d/1NkGRe-YACIcxextpkZLD-HTydZ1ifPyY/view.

Laanti, M., Similä. J. & Abrahamsson, P. 2013. Definitions of Agile Software Development and Agility. Konferenssikirjoitus. Viitattu 30.8.2018. https://www.researchgate.net/publication/263853224_Definitions_of_Agile_Software_Development_and_Agility.

Liker, J. 2006. Toyotan tapaan. Jyväskylä: Gummerus Kirjapaino.

Lopresti, J. 2018. What is a Gemba Walk and Why is it Important? Kirjoitus Six Sigma Dailyn sivuilla. Viitattu 2.9.2018. https://www.sixsigmadaily.com/what-is-a-gemba-walk/.

Matteson, S. 2017. DevSecOps: What it is and how it can help you innovate in cybersecurity. Artikkeli ZDNet-sivuilla. Viitattu 2.9.2018. https://www.zdnet.com/article/devsecops-what-it-is-and-how-it-can-help-you-innovate-in-cybersecurity/.

Maurer, R. 2013. The Spirit of Kaizen: Creating Lasting Excellence One Small Step At a Time. New York: McGraw-Hill.

McKenzie, E. 2009. Lean vs Six Sigma: What's the Difference? Kirjoitus Ultimus-sivustolla. Viitattu 2.9.2018. http://www.ultimus.com/blog/bid/33875/lean-vs-six-sigma-what-s-the-difference.

Mäkäläinen, M. 2018. WIMMA Lab Black Book 1.0. WIMMA Labin opaskirja. Viitattu 14.9.2018. http://www.wimmalab.org/static/media/WIMMALab-BlackBook-1.0.ec5daf40.pdf.

Opintojakson kuvaus. 2016. Challenge factory -opintojakson kuvaus ASIO:ssa. Viitattu 27.8.2018. https://asio.jamk.fi/pls/asio/asio_ectskuv1.kurssin_ks?ktun=TTVW0110&knro=&noclose=+&lan=f&ark=.

Paych, M. N.d. 5 stages of team development every leader should know. Kirjoitus Medium-sivuilla. Viitattu 9.9.2018. https://medium.com/swlh/team-development-stages-51df5606c0a2.

Project Management Guide. N.d. Projektin johtamisen opas Wrike:n sivuilla. Viitattu 28.8.2018. https://www.wrike.com/project-management-guide/methodologies/.

Salminen, J. 2013. Taitava tiimivalmentaja. Helsinki: J-Impact.

Salminen, J. 2017. Onnistu tiimityössä: Tiimin jäsenen kirja. 3. uud. p. Helsinki: J-Impact.

Sawant, S. 2018. What is the difference between DevOps and DevSecOps? Kuva Quora-sivustolla. Viitattu 11.9.2018. https://www.quora.com/What-is-the-difference-between-DevOps-and-DevSecOps.

Schaeffer, C. N.d. Agile versus Waterfall for CRM Implementation Success. Kuva CRM Search -sivustolla. Viitattu 6.9.2018. http://www.crmsearch.com/agile-versus-waterfall-crm.php.

Schwaber, K. & Sutherland, J. 2017. Scrum-opas. Scrumin määritelmä ja pelisäännöt. Viitattu 31.8.2018. https://scrumwell.files.wordpress.com/2018/03/2017-scrum-guide-fi-v1-02.pdf.

Shanholtzer, H. 2018. Agile Trends to Watch in 2018. Artikkeli Agile Connectionin sivuilla. Viitattu 9.9.2018. https://www.agileconnection.com/article/agile-trends-watch-2018.

Six Sigma. N.d. Six Sigman infosivusto. Viitattu 1.9.2018. http://www.sixsigma.fi/index.php/fi/six-sigma/.

Six Sigman kehitysvaiheet. N.d. Kirjoitus Quality Knowhow Karjalainen Oy:n sivuilla. Viitattu 6.9.2018. http://www.sixsigma.fi/index.php/fi/six-sigma/roolit/.

Stine, M. 2010. The Seven Wastes of Software Development. Kirjoitus DZone-sivulla. Viitattu 2.9.2018. https://dzone.com/articles/seven-wastes-software.

The 12th Annual State of Agile Report. 2018. Raportti CollabNet VersionOnen sivuilla. Viitattu 9.9.2018. https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report.

The Roles of Lean Six Sigma. N.d. Blogikirjoitus Go Lean Six Sigma -sivulla. Viitattu 2.9.2018. https://goleansixsigma.com/roles-lean-six-sigma/.

Warcholinski, M. N.d. Differences Between Lean, Agile and Scrum. Kuva Brainhub-sivuilla. Viitattu 9.9.2018. https://brainhub.eu/blog/differences-lean-agile-scrum/.

What is DevOps? N.d. Artikkeli AWS:n sivuilla. Viitattu 2.9.2018. https://aws.amazon.com/devops/what-is-devops/.

What Is Extreme Programming? An Overview of XP Rules and Values. 2018. Kuva Lucid Chart -sivuilla. Viitattu 11.9.2018. https://www.lucidchart.com/blog/what-is-extreme-programming.

Väisänen, J. 2013. VSM (Value Stream Mapping) - Arvovirtakuvaus. Kirjoitus Quality Knowhow Karjalainen Oy:n sivuilla. Viitattu 2.9.2018. http://www.qk-karjalainen.fi/fi/artikkelit/vsm-value-stream-mapping-arvovirtakuvaus/.