Jari Mäyrä

LEAN-TOIMINTAMALLI JA KPI-MITTARIT KETTERÄSSÄ OHJELMISTOKEHITYKSESSÄ

Opinnäytetyö
CENTRIA AMMATTIKORKEAKOULU
Tuotantotalouden koulutusohjelma
Huhtikuu 2015

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö	Aika	Tekijä/tekijät
YLIVIESKA	Huhtikuu 2015	Jari Mäyrä
Koulutusohjelma Tuotantotalous		
Työn nimi Lean-toimintamalli ja KPI-mittarit ketterässä ohjelmistokehityksessä		
Työn ohjaaja Tapio Malinen		Sivumäärä 44
Työelämäohjaaja Tomi Korpela		

Tutkimuksen tavoitteena oli tuoda uusia menetelmiä ja mittareita ketterien ohjelmistotiimien toiminnan tehokkuuden ja laadun kehittämiseksi. Toinen tavoite oli kehittää ohjelmistokehitysprosessia.

Tutkimus pohjautuu kirjallisuuteen, internet-lähteisiin ja tutkimuksiin. Teoreettinen viitekehys perustuu Scrumiin, joka on ketterän projektinhallinnan viitekehys sekä Lean-ajatteluun. Scrumilla ja Lean-ajattelulla on omat luonteenomaiset tavat mitata suorituskykyä ja laatua, vaikkakin niillä on paljon yhteistä.

Tutkimuksen tietoperusta koostuu kolmesta osuudesta: Lean, Scrum sekä KPI-mittarit. Lisäksi työssä esitellään toimeksiantajan nykyiset KPI-mittarit, joilla ketteriä ohjelmistokehitystiimejä mitataan. Tutkimuksen lopussa esitän menetelmiä ja KPI-mittareita, joiden avulla ketterien ohjelmistotiimien tehokkuutta ja laatua voidaan parantaa. Tärkein havainto työssäni oli, että Lean-ajattelun ja siihen perustuvien mittarien ja työkalujen avulla voidaan tehostaa ja nopeuttaa ohjelmistokehitysprosessia sekä parantaa laatua.

Asiasanat

KPI, Lean, mittarit, prosessikehitys, Scrum

ABSTRACT

CENTRAL OSTROBOTHNIA UNIVERSITY OF APPLIED SCIENCES	Date April 2015	Author Jari Mäyrä
Degree programme Industrial Management		
Name of thesis Lean and KPI-metrics in Agile Software Development		
Instructor Tapio Malinen		Pages 44
Supervisor Tomi Korpela		
The objective of this study was to bring new ideas of measuring agile development team performance and quality. Another aim was to improve software development process. The theoretical part was based on literature and internet sources. The theoretical frame consists Scrum iterative project methodology and Lean process improvement model. Scrum and Lean methodologies have their own ways of measuring performance and quality, although they have a lot of common. The theoretic part consists of three sections: Lean, Scrum and KPI's and measurement. Also, this study demonstrates current ways on measuring agile team performance. In the end, the study demonstrates specific key performance indicators and operations to improve quality and performance.		

Key words KPI, Lean, metrics, process development, Scrum

KÄSITTEIDEN MÄÄRITTELY

KPI	Key performance indicator, käytetään usein suomennosta: keskeinen suorituskykymittari. Mittari, joka mittaa organisaation suorituskykyä suhteessa sille asetettuihin tavoitteisiin.
VSM	Value stream map eli arvoketju. Kuvaa graafisesti prosessissa tapahtuvaa arvon tuottamista ja hukkaa.
Hukka	Lean-filosofian keskeinen käsite, tarkoittaa toimintaa joka ei lisää lopputuotteen arvoa asiakkaalle.
DoD	Definition of done, valmiin määritelmä Scrumissa.
Sprintti	Scrumin työskentely tapahtuu sprinteissä. Sprintti on enintään kuukauden pituinen tai sitä lyhyempi aikaraja, jonka sisällä tuotetaan ”valmis” –määritelmän täyttävä käyttökelpoinen ja potentiaalisesti julkaisukelpoinen tuoteversio.
Retro	Scrumin-tiimin palaveri, jossa tiimi keskittyy oman toimintansa kehittämiseen jatkuvan parantamisen hengessä.
Velositeetti	Velositeetti (engl. velocity) on Scrumin vakiomittari jolla mitataan tiimin suorituskykyä sprintin aikana. Yksikkönä on story point tai tunti. Velositeettiä käytetään sprinttien mittaamisen ja sprinttien suunnitteluun.
WIP	Work in progress, työn alla olevien kohteiden määrä. Käytetään tulkittaessa kumulatiivista vuokaaviota.
Läpimenoaika	Läpimenoaika (cycle time) on keskeinen Lean-mittari. Se mittaa keskimääräistä toimitusaikaa.
PDCA	Plan – Do –Check –Act. Jatkuvan parantamisen periaatteeseen perustuva laadunparannusmenetelmä.
User Story	Käyttäjätarina on lyhyt toiminnallinen vaatimusmäärittely toteutettavasta ominaisuudesta. Käyttäjätarina on Scrum-termi.
Jira	ohjelmistokehityksen hallintaan yleisesti käytettävä järjestelmä, joka tukee Scrumia.
ROI	Return on Investment. Sijoitetun pääoman tuotto. Mittaa suhteellista kannattavuutta.
ITIL	IT Information Library. Kokoelma IT-palvelutuotannon parhaita käytäntöjä.

SISÄLLYSLUETTELO

KUVIOT

TAULUKOT

1. JOHDANTO

Liiketoiminnan vaatimat nopeat muutokset ja tarve nopeaan reagointiin on edistänyt ketterien menetelmien käyttöä tuotekehitystiimien projektinhallinnan viitekehyksenä. Suosittu ketterä menetelmä on Scrum. Lean-ajattelu on levinnyt muille toimialueille autoteollisuudesta. Lean-ajattelu on keino kehittää prosesseja sekä tuotannon kokonaisuutta. Lean-ajattelua on sovellettu myös ohjelmistokehitykseen. Yhteistä Lean-ajattelulle ja Scrumille on tavoite arvon maksimoimiseen asiakkaalle, pyrkimys jatkuvaan parantamiseen sekä työntekijöiden ottaminen mukaan kehitystyöhön.

Tämän työn tavoitteena on kehittää KPI-mittareita ketterien ohjelmistokehitystiimien toiminnan mittaamisen ja keinoja tuotekehitystiimien toiminnan kehittämiseen. Mittaamisen ja toimenpiteiden tavoitteena on parantaa laatua, tehokkuutta ja toiminnan arvoa asiakkaalle. Tutkimuksen teoreettisen viitekehyksen muodostavat Scrum ja Lean-ajattelu.

Tutkimustyössä käsitellään ohjelmistokehitystiimien mittaamista ja toiminnan kehittämistä. Ohjelmistojen ylläpitotoiminnot on rajattu pois, samoin kuin sellainen viankorjaus, jolle on sovittu toimittajien kanssa erillinen korjausprosessi.

Tutkimusprojekti on edennyt seuraavasti:

1. Tutkimusaiheeseen liittyviin teorioihin perehtyminen.
2. Nykytilanteen kartoittaminen KPI-mittareiden osalta.
3. Toimenpide-ehdotusten laatiminen toimeksiantajan kanssa käytyjen keskusteluiden pohjalta.

Tutkimusraportissani esittelen ensin Lean-ajattelun taustaa ja teoriaa. Pääpaino on Lean-ajattelun ohjelmistotuotannon sovelluksissa. Teoriaosassa perehdyn myös Scrum-projektinhallintamenetelmään. Kolmas kokonaisuus teoria osuudessa on mittaamiseen liittyvä kokonaisuus. Tutkimuksen pohjatiedoksi perehdyin tuotekehitystiimien nykyisiin mittareihin ja niistä on kerrottu myös tässä raportissa. Teorian, nykytilanteen ja toimeksiantajan edustajan kanssa käytyjen keskusteluiden perusteella päädyin raportissa ehdottamaan uusia mittareita ja toimenpiteitä tuotekehitystiimien toiminnan kehittämiseksi.

Tämän tutkimuksen toimeksiantaja on Elisa oyj.

2. LEAN-AJATTELU

Teoriaosuudessa käsittelen Lean-toimintamallia ja Scrum-projektinhallintamenetelmää. Näiden pohjalta olen määrittänyt työssä toimenpiteitä kehitystiimien toiminnan kehittämiseen ja toiminnan mittaamiseen. Scrum on ketterä menetelmä, jota toimeksiantajan ohjelmistokehitystiimit käyttävät. Lean on eri toimialoille levennyt filosofia ja ajattelumalli, jonka ajatuksia tässä työssä hyödynnettiin toiminnan kehittämisessä.

Lean-ajattelulla ja Scrumilla voidaan havaita monia yhtäläisyyksiä. Keskeistä molemmissa on lisäarvon tuottaminen asiakkaalle nopeasti. Molemmissa kannustetaan työntekijöitä ratkaisemaan ongelmat ja osallistumaan kehitystyöhön. Samoin molemmissa painotetaan jatkuvaa parantamista. Lean-ajattelussa ja Scrumissa työskentely tapahtuu tiimeissä joille annetaan vastuuta ja joita kuunnellaan sekä arvostetaan ihmisiä yksilöinä. Kokonaisuutena voidaan sanoa, että Lean-ajattelu ja Scrum täydentävät hyvin toisiaan.

Lean-ajattelulla ja Scrumilla on oma paikkansa. Scrumin vahvuus on siinä, että se luo puitteet tehokkaalle tiimityöskentelylle ja tekee näkyväksi tiimin tekemän työn. Lean puolestaan korostaa kokonaisuuden hallintaa ja prosessin kehittämistä, työn tehostaminen tapahtuu poistamalla hukkaa.

2.1. Lean-ajattelun taustaa

Lean tuotanto- ja johtamismalli on alun perin kehitetty Toyotan toimesta Japanissa. Toyotan menestys pohjautuu näihin malleihin. Lean-toimintamalli on sittemmin levinnyt useille toimialoille. Lean-periaatteita noudattavat yritykset ovat tavallisesti toimialansa kannattavimpia ja nopeimmin kasvavia.

Lean on alkanut saada jalansijaa myös IT-palveluissa ja muussa palvelutuotannossa. Tuotteita valmistavan teollisuuden ja IT:n välillä on havaittu monia yhtäläisyyksiä. Teollisuus valmistaa tavaroita jotka tuottavat arvoa asiakkaalle, IT-toiminnot vastaavasti valmistavat palveluita jotka tuottavat arvoa organisaatiolle ja asiakkaalle.

Keskeisiä asioita Lean-toiminnassa ovat tinkimätön laatuajattelu, henkilöstön sitouttaminen kehitystyöhön ja jatkuva parantaminen. Tuotteen tai palvelun arvo määritetään asiakasnäkökulmasta. Asiakaslähtöisyys ja lisäarvon tuottaminen asiakkaalle kiteytyy siihen, että yrityksen sisällä hahmotetaan ne toiminnot, jotka lisäävät arvoa asiakkaalle ja kohdistetaan yrityksen voimavarat yksinomaan näihin toimintoihin. Arvon lisääminen tarkoittaa toimenpidettä jonka seurauksena palvelu vastaa paremmin asiakkaan vaatimuksia. Kun arvoa kasvatetaan suhteessa toiminnan kustannuksiin, parannetaan

yrityksen kilpailukykyä ja varmistetaan kannattava toiminta myös tulevaisuudessa. (Kouri I. 2010 s.6)

Lean-filosofian viisi periaatetta (Kouri I. 2010 s.7)

1. Arvo
   Tuotteen tai palvelun arvo määritetään asiakasnäkökulmasta
2. Arvoketju
   Kuvataan yrityksen arvoketju jossa asiakkaan kokema arvo muodostuu.
3. Virtautus
   Virtautus tarkoitta sitä että tuotteet tai palvelut virtaavat valmistusprosessissa pysähtymättä.
4. Imuohjaus
   imuohjaus tarkoittaa tuotteen tai palvelun valmistamista todellisen tarpeen mukaan.
5. Pyri täydellisyyteen
   Prosesseja kehitetään poistamalla hukkaa ja panostamalla jatkuvaan parantamiseen.

Lean-kehitystoiminta perustuu jatkuvaan parantamiseen. Jokaisella työntekijällä on vastuu sekä toiminnan että tuotteen laadusta. Jokainen virhe tai ongelma on nähtävä tilaisuutena kehittää laatua, tehokkuutta ja työturvallisuutta.

2.2. Seitsemän Lean-pääperiaatetta ohjelmistotuotannossa

Lean ohjelmistokehityksen konseptin ovat luoneet Mary ja Tom Poppendieck jotka tunnistivat seitsemän Lean-periaatetta ja ehdottivat kuinka ne voidaan hyödyntää ketterässä ohjelmistokehityksessä. Näiden periaatteiden on tarkoitus tarjota perusta paremmalle ohjelmistokehitykselle.

Eliminoi hukka (Eliminate Waste)

Hukan poistamisessa keskitytään siihen aikaväliin joka kuluu asiakkaan tilauksesta siihen hetkeen kun häneltä kerätään rahat pois. Leanissa keskitytään lyhentämään tätä aikaväliä poistamalla hukkaa. Kaikki mikä ei ole tarpeellista arvon tuottamisen kannata tai suoraan luo arvoa on hukkaa ja tulee poistaa. (Poppendieck M. & Poppendieck T. 2008)

Hukka täytyy ensin tunnistaa jotta se voidaan poistaa. Koska hukka tarkoittaa kaikkea sitä mikä ei lisää arvoa, täytyy tietää mikä oikeasti luo arvoa asiakkaalle. Mikään ei korvaa syvällistä asiakkaan toiminnan ja tarpeiden ymmärrystä. Varmista siis että sinulla on syvällinen ymmärrys millaista arvoa teknologia voi asiakkaalle tuottaa.

Poppendieckien kirjan mukaan suurin hukka ohjelmistokehityksessä on ohjelmistoihin kehitetyt turhat ominaisuudet. Heidän mukaansa keskimäärin 20 prosenttia tyypillisen ohjelmiston ominaisuuksista käytetään säännöllisesti. Tämän vuoksi sellaisten turhien

ominaisuuksien kehittäminen on hukka koska ne eivät tuota lisäarvoa asiakkaalle. Ohje tiivistettynä: kirjoita vähemmän koodia, keskity arvon luomiseen. (Poppendieck M. & Poppendieck T. 2008 s. 24).

Rakenna laatu prosesseihin (Buid quality in)

Laatu vaatii kurinalaisuutta. Ohjelmistovirheiden etsimiseen ja testaamiseen käytetään resursseja, yleisesti on tiedossa että mitä myöhemmässä vaiheessa virhe löydetään, sitä kalliimmaksi sen korjaaminen tulee.

Vikojen etsimisen sijasta huomio kannatta kohdistaa siihen että ohjelmistoon ei tule virheitä. Virheet täytyy löytää aikaisemmin kuin loppukatselmoinnissa. Prosessit täytyy rakentaa siten että laatu on sisäänrakennettu niihin alusta lähtien. (Poppendieck M. & Poppendieck T. 2008 s. 24).

Lean-filosofian mukaan virheen löytyessä, muu toiminta keskeytetään siksi aikaa kunnes koodia on korjattu ja testattu. Virheiden lisääminen virhelistoille lisää jonoja jotka taas ovat osittain tehtyä työtä eli hukkaa. Tavoitteena on siis tilanne jossa uutta koodia ei järjestelmään lisätä ennen kuin vanha on korjattu ja testattu. Käytännössä tämä ei monessakaan organisaatiossa näin toteudu.

Automatisointi yleensä parantaa laatua, tämä pätee myös ohjelmistokehitykseen. Rutiinityövaiheita kannattaa automatisoida niin paljon kuin mahdollista. Erityisesti testaus sellainen ohjelmistokehityksen työvaihe jota voidaan automatisoida.

Laatua voidaan parantaa kun pyritään tekemään kerralla valmiita kokonaisuuksia ja osissa tehtävää työtä vältetään.

Tehosta oppimista (Create knowledge)

On tärkeää että kehitysprosessi rohkaisee järjestelmälliseen oppimiseen koko kehitysketjun alusta loppuun. lisäksi kehitysprosessia täytyy kehittää jatkuvan parantamisen periaatteiden mukaisesti. Joskus standardiprosesseihin lukittautuminen estää tiimejä kehittämästä jatkuvasti prosesseja. Lean-organisaation toimintaan kuuluu olennaisena osana jatkuva prosessien parantaminen koska monimutkaisessa ympäristössä on aina ongelmia. Prosessien kehitysvastuun kuuluu olla kehitystiimeillä. Olennaista on rakentaa organisaatioon jatkuvan parantamisen kulttuuri. (Poppendieck M. & Poppendieck T. 2008).

Tiimitasolla oppimista tehostaa monitaitoisten ja – toimisten tiimien käyttö jotka kykenevät toteuttamaan toiminnallisuudet alusta loppuun.

Ongelmanratkaisumenetelmien käyttäminen ja hyödyntäminen tehostaa oppimista ja lisää tietoa prosessista. Yleisesti käytettävä ongelmanratkaisumenetelmä on Plan-Do-Check-Act -malli eli nk. Demingin pyörä. PDCA-malli on jatkuva kehityssykli, joka etenee suuremmista ongelmista pienempiin. Plan-vaihessa tunnistetaan ja analysoidaan ongelma.

Do-vaihessa kehitetään ongelmaan ratkaisu. Check-vaiheessa tarkistetaan ratkaisun vaikutus ja mahdollisen parannustarpeet. Act- vaiheessa ratkaisu sovelletaan käytäntöön.

Päätä mahdollisimman myöhään (Defer commitment)

Käytännössä tämä merkitsee peruuttamattoman päätöksen lykkäämistä niin myöhään kuin mahdollista. kaikkia mahdollisia päätöksiä ei lykätä vaan peruuttamattomat päätökset tehdään niin myöhäisessä vaiheessa kuin se on mahdollista ilman ongelmia. Ennen peruuttamattoman päätöksen tekemistä kannattaa hankkia mahdollisimman paljon tietoa ymmärryksen lisäämiseksi ja monimutkaisuuden ja riippuvuuksien hallitsemiseksi. Käytännön esimerkki tästä on Scrumin sprint planningit joissa kehitettävät ominaisuudet valitaan juuri ennen kuin niitä aletaan toteuttaa. (Poppendieck M. & Poppendieck T. 2008).

Kaiken ei tarvitse olla valmiiksi määriteltyä ennen kuin työ voidaan aloittaa. Iteratiivisessa kehityksessä päätöksiä voidaan tehdä ja toimintaa ohjata myös matkan varrella.

Toimita nopeasti (Deliver fast)

Tuotteen saaminen markkinoille nopeasti on huomattava kilpailuetu koska asiakkaat pitävät nopeasta toiminnasta. Nopeat toimitukset estävät sen että vaatimukset eivät ehdi vanhentua eivätkä asiakkaat ehdi muuttaa mieltään. Lisäksi iteratiivinen, nopea toiminta mahdollistaa nopean palautteen saamisen asiakkailta. (Poppendieck M. & Poppendieck T. 2008).

Tuotettavien yksiköiden koon pienentäminen nopeuttaa toimitusaikaa parantaa arvovirtausta. Läpimenoa voidaan myös tehostaa minimoimalla keskeneräisten töiden määrää. Oleellista on suunnitella työt vastaamaan tiimin kapasiteettia.

Lean-periaatteiden mukaan toimittaessa keskitytään läpimenoaikaan, ei henkilöresurssien käyttöasteeseen.

Arvosta ihmisiä (Respect people)

Ohjelmistokehittäjiä motivoi vastuu, haasteet ja mahdollisuus itsensä kehittämiseen sekä työ jolla on merkitystä. Tämä tarkoitta myös itse organisoituvia ja –ohjautuvia tiimejä, joille annetaan kohtuulliset tavoitteet ja mahdollisuudet sekä työkalut saavuttaa nämä tavoitteet.

Johtamistaitojen ja ihmissuhdeosaamisen kouluttaminen tiiminvetäjille ja ohjaajille vaikuttaa positiivisesti työyhteisön ilmapiiriin ja tuottavuuteen.

Vastuu kannattaa viedä alimmalle mahdolliselle organisaatiotasolle koska vastuu motivoi työntekijöitä.

Työntekijöillä on oikeus olla ylpeitä omasta ammattitaidostaan, tällaista ilmapiiriä kannattaa rohkaista.

Optimoi kokonaisuus (Optimize the whole)

Huomion täytyy kohdistua koko arvoketjun kehittämiseen, ei siis vain tiettyjen tiimien tai toimintojen kehittämiseen. Pelkästään tietyn osan optimointi ei johda toivottuun lopputulokseen ja voi olla kokonaisuuden kannalta jopa haitallista. Huomion täytyy olla koko tuotteessa, ei pelkästään ohjelmistossa. Yrityksen sisällä voi olla eri organisaatioyksiköiden välisiä muureja joiden ylittäminen on hankalaa ja kallista, samoin voi olla myös alihankkijoiden ja yhteistyökumppaneiden välillä. Näissä tilanteissa on tärkeää varmistaa että kaikki osapuolet ovat sitoutuneet lisäarvon tuottamiseen loppuasiakkaan näkökulmasta. (Poppendieck M. & Poppendieck T. 2008).

Oleellista on järjestää mittarit siten että ne kohdistuvat koko arvoketjuun.

Prosessia voidaan analysoida arvovirtakuvauksen avulla (value stream map). Arvovirtakuvaus esitellään tarkemmin kappaleessa 2.4.

2.3. Hukka

Lean-filosofian mukaan hukka tarkoittaa arvoa tuottamatonta toimintaa, joka ei lisää lopputuotteen arvoa asiakasnäkökulmasta.

Taulukossa on 1. on verrattu valmistusteollisuuden hukkia ohjelmistoteollisuudessa tapahtuvaan haaskaukseen.

TAULUKKO 1: Hukat valmistus vs. ohjelmistotuotanto. (Poppendieck M. & Poppendieck T. 2008 s. 74.)

Manufacturing	**Software development**
In-process inventory	Partially done work
Over-production	Extra features
Extra processing	Relearning
Transportation	Handoffs
Motions	Task Switching
Waiting	Delays
Defects	Defects

Keskeneräinen työ (Partially done work)

Keskeneräinen työ on yksi hukan muoto. Tavoite on että ohjelmistokehitys etenee nopeasti käyttöönottokelpoiseksi jatkuvana virtana johon kuuluu integrointi, testaus, dokumentointi ja käyttöönotto. Paras tapa saada työ kerralla valmiiksi on toteuttaa se iteraatioissa joka sisältää kaikki nämä vaiheet. (Poppendieck M. & Poppendieck T. 2008).

Työ on osittain tehtyä, mikäli sen osia on jonoissa tai varastoissa. Esimerkiksi testaamaton koodi on osittain tehtyä työtä, sillä se on odottamassa pääsyä testaukseen.

Määrittelyt joita ei toteuteta, muuttuvat hyvin pian hyödyttömiksi. Nämä toteuttamattomat vaatimukset, ominaisuudet ja kehitystarpeet ovat keskeneräisen työn muotoja.

Ohjelmistokehityksessä hukkaa ovat myös testaamaton ja dokumentoimaton ohjelmakoodi. Myös sellainen ohjelmistokoodi, jota ei oteta käyttöön tai yhdistetä kokonaisuuteen on hukkaa.

Ylimääräiset/turhat ominaisuudet (Extra features)

Ylimääräiset/turhat ominaisuudet ovat ohjelmiston ominaisuuksia joille asiakkaalla ei ole tarvetta nykyisten töiden suorittamiseksi. Mary ja Tom Poppendieckin kirjan mukaan tämä on pahin seitsemästä hukasta. Jos ominaisuudelle ei ole tällä hetkellä ei ole selvää taloudellista perustetta, sitä ei ole tarpeen kehittää. (Poppendieck M. & Poppendieck T. 2008 s. 75).

Uudelleen opettelu (Relearning)

Asioiden uudelleen opettelu tuottaa hukkaa. Jo kertaalleen opittujen asioiden unohtaminen on inhimillistä, samoin tehtyjen päätösten unohtaminen. Pahimmassa tapauksessa palataan huomaamatta kokeilemaan jotain jo aiemmin toimimattomaksi todettua asiaa uudestaan. Jo päätetyistä asioista tulisi pitää kirjaa. (Poppendieck M. & Poppendieck T. 2008).

Tehdyt päätökset ja asiat olisi syytä kirjata johonkin järjestelmään, esimerkkinä ohjelmistokehityksessä käytettävä Jira-järjestelmä. Poppendieckit ovatkin sitä mieltä, että aiemmin tuttujen ja unohdettujen asioiden uudelleen keksiminen on kenties paras määritelmä uudelleen opettelulle kehitystyössä. Toisaalta, mikäli ihmisten osaamista ei hyödynnetä parhaalla mahdollisella tavalla, aiheutetaan vielä merkittävämpää hukkaa kuin asioiden uudelleen opettelulla.

Tehtävien vaihtelu (Task switching)

Tehtävien vaihtelu vie aikaa ja häiritsee keskittymistä. Vaihdon jälkeen tehtävään täytyy syventyä uudestaan. Eri asioiden tekeminen samaan aikaan ohjelmistokehityksessä ei useinkaan ole järkevää vaan aiheuttaa turhaan hukkaa. (Poppendieck M. & Poppendieck T. 2008).

Viivytykset ja odottelu (Delays)

Ohjelmistokehittäjät tarvitsevat muiden asiantuntijoiden ja liiketoimintaihmisten vastauksia voidkseen tehdä kehitystyössä tarvittavia päätöksiä ja ratkoa ongelmia. Hukkaa aiheutuu sitä kun muita odotetaan asioista tietäviä ihmisiä vapautuviksi jotta näiltä voidaan saada vastauksia. On naiivia olettaa että kaikki tarvittava informaatio löytyy dokumenteista. (Poppendieck M. & Poppendieck T. 2008).

Virheet (Defects)

Ohjelmakoodin virheet on yksi verrattain yleinen hukan muoto. Testauksen tarkoitus ei ole löytää virheitä vaan prosessissa olevia ongelmia jotka näitä virheitä aiheuttavat. Testaus sisältää sekä yksikkötestauksen ja hyväksymistestauksen. Virheen löytämisen täytyy johtaa toimenpiteisiin jotka estävät ettei ohjelmakoodiin pääse jatkossa virheitä. Mikäli verifioinnissa löytyy koko ajan virheitä - on koko prosessi virheellinen. (Poppendieck M. & Poppendieck T. 2008).

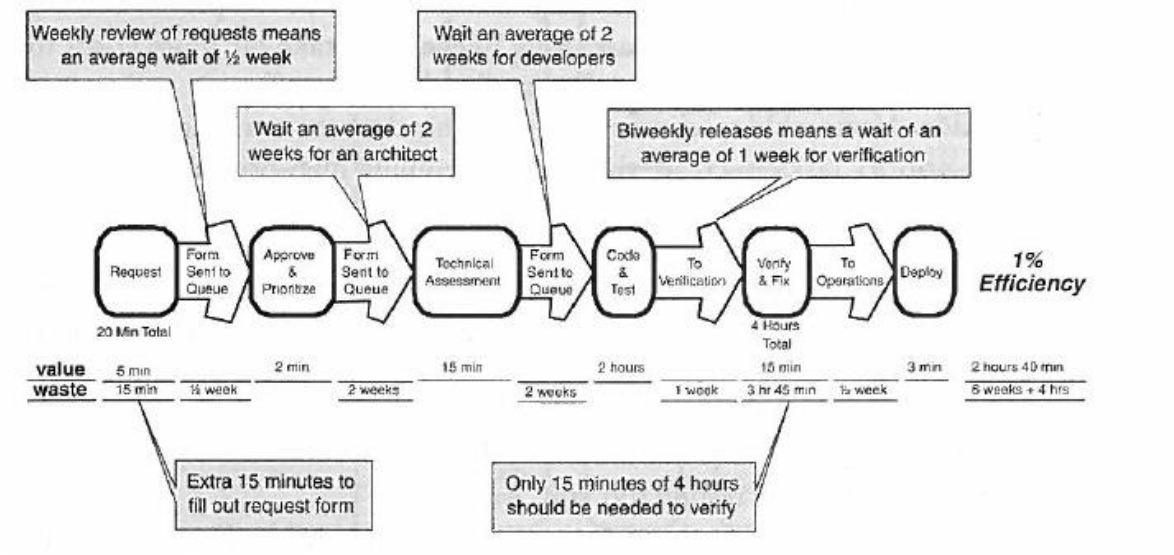
2.4. Arvovirtaketju (Value stream map)

Toyotan tuotantojärjestelmässä tarkastellaan koko tuotantoketjua lähtien siitä pisteestä kun asiakas tekee tilauksen päättyen siihen kun asiakkaalta ”kerätään rahat pois”. Tätä aikaa pyritään lyhentämään poistamalla tekijät jotka aiheuttavat hukkaa. Poistettavat tekijät ovat sellaista toimintaa joka ei lisää tuotteen arvoa asiakkaalle. Tämä mainittu aikaväli voidaan kuvata ”value stream mapping” –menetelmällä. Arvovirtaketju alkaa aina asiakkaasta ja päättyy asiakkaaseen. (Poppendieck M. & Poppendieck T. 2008 s. 24).

Ohjelmistokehityksessä arvovirtaketjun alkupiste on asiakkaan tekemä tilaus – organisaatiosta riippuen tämä saattaa tarkoittaa eri asioita. Vastaavasti kello pysähtyy siihen hetkeen kun ratkaisu on onnistuneesti toimitettu tai asiakkaan ongelma ratkaistu. ”Value stream map” on aikajana jolle kuvataan merkittävät tapahtumat kehitysjakson ajalta.

Leanin tavoite on lyhentää kehitykseen kuluvaa aikaa poistamalla hukkaa – eli sellaista toimintaa joka ei lisää tuotteen arvoa asiakkaalle. Viivytykset ja odottelut aiheuttavat huomattavasti hukkaa ohjelmistokehityksessä. Arvovirtaketjun tavoite on lisätä ymmärrystä prosesseista ja toimintaan vaikuttavista asioista.

Kuviossa 1. on ohjelmistokehitykseen liittyvä esimerkki arvovirtakuvauksesta. Kysymyksessä on yksinkertaisen, korkean prioriteetin ominaisuuden toimitusketju. Esimerkissämme varsinaista arvoa tuottavaa työtä on 2h 40 min, mutta tuotantoon vienti kestää yli kuusi viikkoa.



KUVIO 1. Value stream map (Mary and Tom Poppendieck: Implementing Lean Software s.87)

3. SCRUM PROJEKTINHALLINNAN VIITEKEHYS

3.1. Scrumin historia

Scrumin kehitykseen ovat vaikuttaneet useat henkilöt ja julkaisut.

Scrumia on käytetty ja kehitetty vuodesta 1993 lähtien erilaisten ohjelmistojen ja muiden tuotteiden kehityksessä. Merkittävä vaikuttaja Scrumin takana oli 1986 Harward Business Review:ssa julkaistu artikkeli jossa Takeuchi ja Nonaka julkaisivat artikkelin itseorganisoituvista tiimeistä autoteollisuudessa. Muita Scrumin kehitykseen vaikuttaneita henkilöitä ovat Ken Scwaber ja Jeff Sutherland, jotka esittelivät ja Scrumin OOPSLA-seminaarissa 1995. (Schwber K. & Sutherland J. 2013)

3.2. Yleistä Scrumista

Scrum on projektinhallinnan viitekehys. Scrum on ketterä menetelmä jonka tarkoituksena on tuoda läpinäkyvyyttä tuotteen kehitykseen ja erityisesti sen edistymisen seurantaan. Scrumia käytetään pääasiassa ohjelmistokehitykseen mutta sitä voi hyödyntää myös muissa tuotekehityshankkeissa.

Scrumin määritelmä:

Scrum on projektinhallinnan viitekehys, jossa ihmiset voivat ratkaista monimutkaisia ongelmia kehittäessään tuotteita tuottavasti ja luovasti mahdollisimman korkealla lisäarvolla. Scrum on (Schwber K. & Sutherland J. 2013):

- Kevyt
- Yksinkertainen ymmärtää
- Vaikea hallita hyvin

Scrum on hyvä keino ratkoa monimutkaisia ongelmia, jolloin lähtötilanteessa ei tunneta hyvin ongelmaa eikä sen ratkaisua. Yksinkertaisten, toistuvien ongelmien ratkaisuun Scrum ei ole sopiva. Monimutkaisuus tarkoittaa tässä yhteydessä sitä että emme tiedä asioiden välisiä riippuvuussuhteita, se saattaa tarkoittaa myös ”liikkuvaa kohdetta”, joka voi olla uusi tuote, uudet markkinat tai uusi teknologia.

Scrum koostuu Scrumtiimeistä rooleineen, tapahtumista, tuotoksista ja säännöistä. Jokainen elementti palvelee tiettyä tarkoitusta ja on tärkeä osa Scrumin onnistumista.

Scrum soveltuu parhaiten monimutkaisiin projekteihin joissa projektin lopputulosta ja vaatimuksia ei ole naulattu valmiiksi aloitusvaiheessa. Monimutkaisissa aloitteissa ei

useinkaan ole mahdollisuutta määritellä alkuvaiheessa millainen projektin lopputuloksena syntyvä tuote tulee olemaan.

Luonteenomaista Scrumille on että projektin kuluessa vaatimukset ja ennusteet lopputuloksesta tarkentuvat. Tämä johtuu siitä että projektin kuluessa tietoja kertyy ja sitä hyödynnetään projektin ohjaamiseen oikeaan suuntaan. Lyhyisiin kehityssykleihin perustuvan menetelmän etuna on myös se että tavoitetta ja edistymistä seurataan sovituissa tarkistuspisteissä jolloin sitä ohjataan oikeaan suuntaan.

Scrum on siis keino ratkoa ongelmia joihin ei tunneta etukäteen ratkaisua. Se ei ole hyvä menetelmä yksinkertaisten, toistuvien tehtävien hoitoon. Scrum soveltuu tietotyöhön jossa työn tarkoitus on arvon (value) luominen. Scrumin täytyy siis tuottaa arvoa liiketoiminnalle ja sidosryhmille. Lähtökohta on siis käyttäjän, asiakkaiden ja liiketoiminnan tarpeissa. Scrum luo kehitystyölle kehykset joilla näihin tarpeisiin vastataan. Monimutkaisille ongelmille on luonteenomaista että emme aloitusvaiheessa tiedä asioiden välisiä riippuvuussuhteita.

Lähtökohtaisesti Scrum on kehitetty ohjelmistoprojektien tarpeisiin. Ohjelmistoprojektin tarkoitus on luoda arvoa. Todellista arvoa syntyy silloin kun luodaan uusia ja ainutkertaisia tuotteita jotka perustuvat käyttäjien tarpeeseen ja liiketoiminnan visioon.

3.3. Ketterän ohjelmistokehityksen julistus

Vuonna 2001 17 merkittävää ketterän kehityksen (jota silloin kutsuttiin ”kevyiksi menetelmiksi”) puolestapuhujaa kokoontui Snowbirdin hiihtokeskukseen Utahissa keskustelemaan menetelmiensä yhteisestä perustasta. Tarkoitus oli luoda yhteistä pohjaa ketterille menetelmille ja edistää näin ketterän ajattelun leviämistä. Tuon kokoontumisen tuloksena he julkaisivat julistuksen nimeltä Agile Manifesto (’Ketterä manifesti’), jota pidetään ketterän kehityksen perusmääritelmänä. Manifestissa määritellään ketterille menetelmille neljä tyypillistä arvoa, sekä 12 periaatetta, joita menetelmät noudattavat.

Manifestin sisältö on seuraava:

> *Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:*
>
> *Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja*
>
> *Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota*
>
> *Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja*
>
> *Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa*

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän. (Jeffries R. & Kern J. & Marick B. & Martin R. C. & Mellor S. & Schwaber K. & Sutherland J. & Thomas D)

3.4. Työskentelytapa

Scrumissa työskennellään toistavasti ja lisäävästi (engl. iterative-incremental) ennustettavuuden optimoimiseksi ja riskien kontrolloimiseksi. Tavoitteena oleva tuote kehittyy pikkuhiljaa täydellisemmäksi ja valmiimmaksi useiden kehitysjaksojen aikana. Kehitysjaksoa kutsutaan sprintiksi (engl. Sprint). Sprintti on 1-4 viikon mittainen aikaraja, jonka sisällä tuotetaan "valmiin" määritelmän täyttävä, käyttökelpoinen ja potentiaalisesti julkaisukelpoinen tuoteversio. (Larman C. & Vodde B. 2013).

3.5. Sertifiointi

Scrum Alliance -yhdistys on organisoinut maailmanlaajuisen Scrum-koulutuksen. Scrum Alliance luotiin levittämään tietoutta ja ymmärrystä Scrumista, ja he tarjoavat virallisia Scrum-sertifiointeja kouluttajille ja käyttäjille. Sertifioidut Scrum-kouluttajat järjestävät ScrumMaster-, tuoteomistaja ja Scrum-kehittäjä -kursseja, joilla käydään läpi Scrumin perusperiaatteita ja opastetaan kyseisissä rooleissa toimimisessa. Kurssin käyneet voivat suorittaa yhdistyksen sivuilla testin, jonka perusteella saavat oikeuden kutsua itseään sertifioiduiksi Scrum-osaajiksi.

3.6. Scrumin taustateoria

Scrum perustuu empiiriseen prosessinhallintateoriaan tai empirismiin. Empirismin mukaan tieto perustuu kokemukseen ja päätösten tekemiseen tunnettujen tosiasioiden pohjalta. Scrum hyödyntää iteratiivis-inkrementaalista (toistavaa ja lisäävää) lähestymistapaa ennustettavuuden optimoimiseen ja riskien kontrolloimiseen. Empiirisellä prosessinhallinnalla on kolme tukijalkaa: läpinäkyvyys, tarkastelu ja sopeuttaminen (Schwber K. & Sutherland J. 2013).

Empiirinen prosessikontrolliteoria tarkoittaa säännöllistä (syklistä) tutkimista ja mukautumista, jolloin optimoidaan tavoitteet viimeisimmän tiedon pohjalta.

Scrum edellyttää läpinäkyvyyttä ja nopeaa palautetta. Tällainen lähestymistapa edellyttää seuraavia asioita:

1. Järjestelmässä täytyy olla riittävästi läpinäkyvyyttä jotta sitä voidaan havainnoida.

2. palautteen täytyy olla riittävän nopeaa jotta voimme mukautua sopivasti.

3.7. Scrumtiimi ja roolit

Scrumitiimi (Scrum Team)

Scrumtiimi eli tuotekehitystiimi on monitaitoinen yksikkö jonka tehtävä kehittää tuotteita joilla on arvoa liiketoiminnalle. Kehitystiimi koostuu ammattilaisista jotka muuttavat sprinttiin valitun kehitysjonon sisällön potentiaalisesti julkaisukelpoiseksi ”valmiiksi” tuoteversioksi jokaisessa sprintissä. Ainoastaan kehitystiimin jäsenet osallistuvat tuoteversion kehitykseen. Ulkoisen ohjauksen sijaan Scrumtiimit päättävät itse miten tekevät työnsä. Monitaitoisuus tarkoittaa sitä että tiimillä on kaikki työn tekemiseen vaadittava osaaminen ilman riippuvuuksia tiimin ulkopuolisiin tekijöihin. Jatkuvan parantamisen hengessä Scrumtiimit pyrkivät kehittämään koko ajan omaa toimintaansa ja osaamistaan.

Scrumin tiimimallilla tavoitellaan joustavuutta, luovaa toimintaa ja toiminnan optimointia. (Schwber K. & Sutherland J. 2013).

Kehitystiimin optimikoko on 7 +/-2 henkilöä. Scrumtiimin täytyy olla riittävän pieni jotta se pystyy ketteränä mutta on riittävän suuri jotta se saa valmiiksi merkittävän määrän työtä valmiiksi sprintin aikana.

Tuoteomistaja (Product owner)

Tuoteomistaja on vastuussa siitä että kehitystyön panokset kohtaavat liiketoiminnan tarpeet. Hänen vastuullaan on myös tuotekehityspäätökset yrityksen strategian mukaisesti ja varmistaa että tuotekehitystiimi tekee oikeita asioita. Tuoteomistaja tuntee tuotteen liiketoiminnan ja edustaa käyttäjiä ja asiakkaita.

Tuoteomistaja on vastuussa tuotteen arvon ja kehitystiimin työn arvon maksimoimisesta. Näiden toteutus voi vaihdella suuresti organisaatioiden, Scrumtiimien ja yksilöiden välillä.

Tuoteomistaja on vastuussa tuotteen kehitysjonon hallinnasta. Tämä sisältää:

- Tuotteen kehitysjonon kohtien selkeän kirjallisen ilmaisun.

- Tuotteen kehitysjonon järjestämisen siten, että tavoitteet saavutetaan parhaalla tavalla.

- Kehitystiimin työn arvon varmistamisen ja maksimoimisen.

- Tuotteen kehitysjonon avoimuuden, läpinäkyvyyden ja ymmärrettävyyden varmistamisen siten että tuotteen kehitysjonoista selviää mitä Scrumtiimi tulee tekemään seuraavaksi

- Sen varmistamisen että kehitystiimi ymmärtää tuotteen kehitysjonon riittävällä tarkkuudella.

Tuoteomistaja on yksi henkilö, ei työryhmä tai tiimi. Kaikki tuotekehitystiimin tekemä kehitystyö tulee olla tuoteomistajan hyväksymää, tuoteomistaja osoittaa kehitystehtävät tiimille eikä kenelläkään muulla ole oikeutta sitä tehdä. Tiimin ei myöskään tarvitse reagoida muiden kuin tuoteomistajan esittämiin kehityspyyntöihin. (Schwber K. & Sutherland J. 2013).

Tekninen tuoteomistaja (Technical Product Owner)

Technical product owner –rooli tullut viime aikoina mukaan joidenkin Scrumtiimien toimintaan. TPO-roolista on Organisaatiokohtaisia sovelluksia.

Technical Product Owner –rooli ei ole virallisesti osa Scrumia ja sen työtapaa. TPO:lle ei myöskään ole sertifioituja koulutuksia kuten Scrummaster ja Product Owner –koulutukset.

Eräs määritelmä TPO-roolille on että se on Scrummasterin ja PO:n yhdistelmä, lisättynä teknisellä osaamisella. Yksi sovellutus on sellainen jossa TPO ja PO työskentelevät rinnakkain ja samanarvoisina. Tämän yhteistyön on ajateltu toimivan siten että PO jalostaa ja priorisoi kehitysideoita yhdessä liiketoiminnan kanssa, TPO:n tehtävä painottuu siihen että hän käsittelee kehitysideoita tiimin kanssa. Eli PO:n rooli painottuu prosesseihin ja liiketoimintaan ja TPO:n tekniseen puoleen. (Vaughn S. 2013).

TPO-rooli vaatii monipuolista osaamista. TPO:n tausta voi olla Tekniseltä puolelta, myöhemmin täydennettynä business- ja tuoteosaamisella tai hän voi olla taustaltaan Business/Tuotepuolelta, myöhemmin täydennettynä teknisellä osaamisella.

Erään määritelmän mukaan TPO tuo mukaan teknisen näkökulman tuotetta koskeviin strategisiin keskusteluihin. Tämä puolestaan auttaa tiimiä saavuttamaan sprintin tavoitteet koska tällöin kehittäjäresursseja ei tarvitse sitoa pitkän tähtäimen road mapien tekoon.

Toinen tärkeä tehtävä TPO:lle on se että hän on vastuussa Product backlogin ennen sprinttiä tapahtuvasta valmistelusta (grooming)

TPO:n tärkein tehtävä on rakentaa siltaa asiakastarpeiden ja teknisten mahdollisuuksien välille. TPO edustaa ja puolustaa teknistä näkökulmaa liiketoimintatavoitteiden puristuksessa. Monessa tapauksessa teknisen ja liiketoiminnallisen näkemyksen yhdistäminen auttaa parantamaan kehitettävien tuotteiden laatua ja tuomaan tätä kautta arvoa liiketoiminnalle. (Vaughn S. 2013).

Scrummaster

Scrummasterin tehtävä on helpottaa kehitystiimin ja tuotevastaavan välistä yhteistyötä. Samoin Scrummasterin työtä on huolehtia että ympäröivä organisaatio omaksuu ja käyttää Scrumia tehokkaasti.

Scrummasterin vastuulla on että tiimi käyttää Scrumia ja se että tiimin sidosryhmät ymmärtävät Scrum-toimintamallia. Scrummaster varmistaa tämän siten että Scrumtiimit pitäytyvät Scrumin teoriassa, käytännöissä ja säännöissä. (Schwber K. & Sutherland J. 2013).

Scrummasterin tehtävät koostuvat suurelta osin ryhmän työskentelyä haittaavien esteiden poistamisesta ja ryhmän valmentamisesta itseohjautuvuuteen.

Scrummaster auttaa Scrumtiimin sidosryhmiä ymmärtämään mitkä heidän tavoistaan toimia Scrumtiimin kanssa ovat hyödyllisiä ja mitkä taas eivät. Scrummaster auttaa muuttamaan näitä toimintatapoja maksimoidakseen Scrumtiimin käytännön arvon. (Schwber K. & Sutherland J. 2013).

Scrummaster suojaa tiimiä ulkopuolisilta häiriöiltä ja keskeytyksiltä sekä sprintin aikana pyydetyiltä uusilta vaatimuksilta. Scrummaterin tehtävä on osaltaan taata tiimilleen työrauha.

3.8. Scrumin tapahtumat

Scrumissa käytetään ennalta sovittuja tapahtumia luomaan säännöllisyyttä ja minimoimaan muiden kuin Scrum-palavereiden tarve. Scrumin tapahtumat ovat aikarajattuja (time-boxed) eli jokaisella niistä on maksimipituus. Tämä varmistaa sen että suunnittelulle jää riittävästi aikaa, mutta suunnitteluprosessissa ei pääse tapahtumaan hukkaa.

Sprintti

Scrumin iteraatiota kutsutaan sprintiksi.

Scrum-työskentely tapahtuu sprinteissä. Sprintti on enintään kuukauden pituinen tai sitä lyhyempi aikaraja, jonka sisällä tuotetaan ”valmis” – määritelmän täyttävä käyttökelpoinen ja potentiaalisesti julkaisukelpoinen tuoteversio. Sprinteillä on sama pituus koko kehityksen ajan. Edellisen päätyttyä aloitetaan välittömästi uusi sprintti. (Schwber K. & Sutherland J. 2013).

Hyvin yleinen sprintin pituus nykyään on kaksi viikkoa. Tärkeä tekijä sprintin pituudessa on että se pysyy koko ajan samana. Säännöllinen rytmi tekee helpommaksi oikeaan työrytmiin pääsemisen ja tekee kapasiteettitarpeen arvioinnin luotettavammaksi.

Ennustettavuuden lisäksi säännöllinen rytmi parantaa tiimin työmoraalia ja vähentää ulkopuolisia häiriötekijöitä.

Sprinttien aikajaksottamisen syy on se että vahvistaa oppimista ja rohkaisee kohtaamaan tosiasiat eli tarkastelemaan tiimin tilannetta suhteessa sprintin tavoitteisiin.

Sprintit koostuvat sprintin suunnittelupalaverista, päiväpalaverista, kehitystyöstä, sprintin katselmoinnista ja sprintin retrospektiivista. (Schwber K. & Sutherland J. 2013).

Sprintin suunnittelupalaveri (Sprint Planning Meeting)

Sprinttia alkaa aina suunnittelupalaverilla. Sprintin suunnittelupalaverissa suunnitellaan sprintin aikana tehtävä työ. Tämä suunnitelma luodaan yhteistyössä koko Scrumtiimin kesken. Sprintin suunnittelupalaveri on ajallisesti rajattu ja jakaantuu kahteen osaan. Yhden viikon mittaisen sprintin suunnittelupalaveri ei saa kestää yhtä tuntia enempää, kahden viikon sprintin suunnittelupalaverin maksimipituus on kaksi tuntia jne.

Sprintin aikana tehtävä työ suunnitellaan sprintin suunnittelupalaverissa. Tämä suunnitelma luodaan yhteistyössä koko Scrumtiimin kesken. Sprintin suunnitelupalaverin lähtökohtana on tuotteen järjestetty kehitysjono, kehitystiimin kapasiteetti ja aiempi suorituskyky. Päätös sopivasta työmäärästä on täysin kehitystiimin, koska vain kehitystiimi voi arvioida, mitä se pystyy toteuttamaan alkavassa sprintissä.

Palaverin vaiheet:

Osa 1. valitaan tavoite

- Tuoteomistaja esittelee tuotteen roadmapin viimeisimmän vision.

- Kehitystiimi arvioi oman nopeutensa ja kapasiteettinsa seuraavalle sprintille.

- Kehitystiimi esittelee nykyisen määritelmään ”valmiille” ja mahdolliset muutokset siihen.

- Kehitystiimi valitsee tuotteen kehitysjonosta kohteet jotka se aikoo toteuttaa tuotteiksi seuraavassa sprintissä.

Osa 2. laaditaan suunnitelma

- Kehitystiimi hajottaa valitusta tuotekehitysjonon kohdan pienempiin tehtäviin joiden toteuttaminen kestää vähemmän kuin päivän.

- Kehitystiimi arvioi työmäärän joka kuluu valittujen tuotekehitysjonon kohteiden toteuttamiseen.

- Kehitystiimi päättää mitkä tuotekehitysjonon kohteet valitaan sprintin tavoitteeseen ja esittelee tämän tavoitteen tuote-omistajalle.

Sprintin keskeyttäminen. Sprintti voidaan joissain tapauksissa keskeyttää ennen aikarajan täyttymistä. tyypillisesti se johtuu seuraavista syistä: kehitystiimi havaitsee että he eivät pysty tavoittamaan sprintin tavoitetta tai tuoteomistaja havaitsee että ulkopuoliset olosuhteet ovat muuttuneet niin paljon että sprintin tavoite on muuttunut merkityksettömäksi.

Päiväpalaveri

Päiväpalaveri on enintään 15 minuutin mittainen aikarajattu tapahtuma, jossa kehitystiimi tahdistaa keskinäiset työnsä ja luo suunnitelman seuraavalle 24 tunnille. Tämä tapahtuu tarkastelemalla edellisen päiväpalaverin jälkeen tehtyä työtä ja ennustamalla mitä voidaan tehdä ennen seuraavaa päiväpalaveria.

Scrum-kokouksessa (Daily Scrum), jokainen projektiryhmän jäsen vastaa seuraaviin kysymyksiin

- Mitä olen tehnyt eilisen kokouksen jälkeen?

- Mitä aion saada tehdyksi seuraavaan kokoukseen mennessä?

- Onko esiin nousseita ongelmia, jotka estävät minua suoriutumasta tehtävistäni?

- Kokous on lyhyt (15 min) ja pidetään mielellään seisaaltaan.

Vähintään kerran päivässä tuotekehitystiimi pysähtyy arvioimaan omaa edistymistään verrattuna sprintin tavoitteeseen. Samalla tiimi miettii onko syytä muuttaa suunnitelmaa ja jos on niin miten sitä muutetaan.

Usein palaveri pidetään tiimin tehtävätaulun ympärillä, joka mahdollistaa töiden hallitsemisen visuaalisesti. Tämä mahdollistaa myös muutosten tekemisen tuotekehitysjonoon tarvittaessa.

Mikäli keskustelua halutaan jatkaa 15 minuutin jälkeen, on tarkoituksenmukaista sopia erillinen palaveri, joka voi alkaa heti päiväpalaverin jälkeen.

Sprintin katselmointi (Sprint review)

Sprintin katselmoinnista käytetään myös nimitystä demo. Sprintin lopussa pidetään sprintin katselmointi, jossa tarkastellaan kehitetty tuoteversio ja sopeutetaan tarvittaessa tuotteen kehitysjonoa. Sprintin katselmoinnin aikana Scrumtiimi ja sidosryhmät katselmoivat yhdessä, mitä sprintissä kehitettiin. Perustuen tähän tietoon ja mahdollisiin sprintin aikaan tuotteen kehitysjonoon tehtyihin muutoksiin osallistujat keskustelevat siitä, mitä voitaisiin kehittää seuraavaksi arvon optimoimiseksi.

Sprintin katselmointi on vapaamuotoinen palaveri, jossa esitettävän tuoteversion demon tavoitteena on saada palautetta, edistää keskustelua ja luoda realistinen pohja seuraavan sprintin suunnittelupalaverille. Demo on merkittävä siksi että se tuo liiketoiminnan näkökulman kehitystyöhön. (Schwber K. & Sutherland J. 2013).

Sprintin katselmointi alkaa sillä että tiimi esittelee että mitä sprintin tavoitteita se saavutti ja mitkä jäivät tavoittamatta, samalla tuoteomistaja hyväksyy tiimin tulokset.

Sprintin katselmointiin kutsutaan tärkeimmät sidosryhmien edustajat.

Sprintin retrospektiivi (Sprint retrospective)

Sprintin retrospektiivi antaa Scrumtiimille tilaisuuden tarkastella työskentelyään ja tehdä suunnitelman kehitysprosessin parannuksille, jotka toteutetaan seuraavassa sprintissä

Sprintin retrospektiivi on jatkuvan parantamisen työkalu. Sprintin retrospektiivi pidetään sprintin katselmoinnin jälkeen ja ennen seuraavaa sprintin suunnittelupalaveria. Palaveri rajataan enintään kolmeen tuntiin kuukauden sprintille. Lyhyemmille sprinteille varataan vähemmän aikaa.

Restropektiivi edustaa kysymystä miten me teimme. Retrospektiivi siis tarkastelee kuinka edellinen sprintti sujui liittyen ihmisiin, yhteistyöhön prosessin ja työkaluihin. Retrossa tunnistetaan asiat jotka sujuivat hyvin sekä määritellään tärkeimmät parannuskohteet. Retrospektiivissä luodaan suunnitelma kehitystiimin työskentelytapojen parantamiseksi.

Retrospektiivillä tavoitellaan tiimin tuottavuuden ja kyvykkyyden kasvattamista. Sillä on myönteinen vaikutus myös laatuun ja tiimin sisäiseen moraaliin.

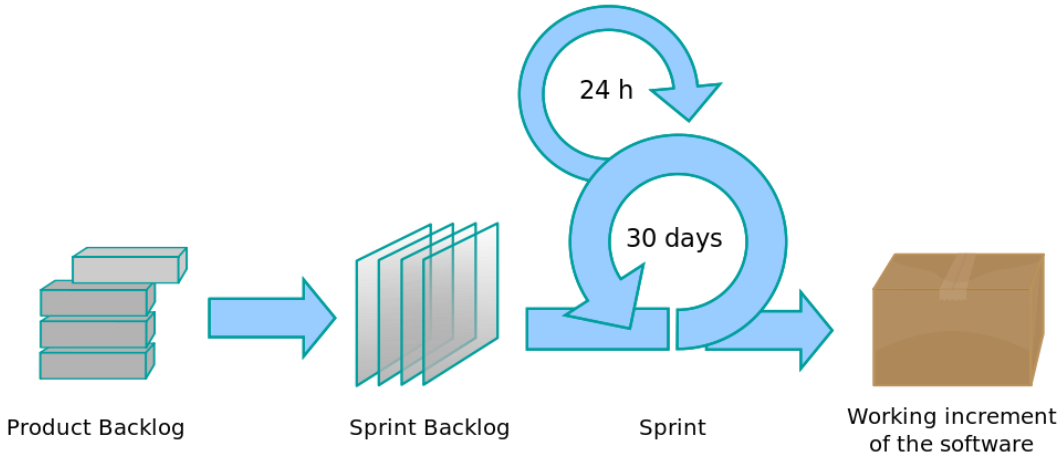
Jotta retron oppimistavoite täyttyy, täytyy retropektiivin johtaa konkreettisiin toimenpiteisiin. Joskus tosin saattaa riittää että ongelma tiedostetaan ja siitä keskustellaan.

Tuotteen kehitysjonon työstö (Grooming, refining)

Puhekielessä tuotteen kehitysjonon työstöstä käytetään nimitystä groomaus. Tuotteen kehitysjonon työstö tarkoittaa yksityiskohtien, työmääräarvioiden ja kehitysjonon kohtien keskinäisen järjestyksen lisäämistä. Kyseessä on toistuva prosessi, jossa tuoteomistaja ja kehitystiimi lisäävät yhteistyössä yksityiskohtia tuotteen kehitysjonoon. Työstön aikana tuotteen kehitysjonon kohtia katselmoidaan ja arvioidaan. Niitä voidaan kuitenkin päivittää milloin tahansa muulloinkin tuoteomistajan toimesta tai toiveesta. Työstöön käytetään 5-10% kehitystiimin kapasiteetista. Kehitystiimi vastaa kaikista työmääräarvioista. Tuoteomistaja voi vaikuttaa kehitystiimiin auttamalla sitä ymmärtämään vaatimuksia ja tekemään kompromisseja, mutta ihmiset, jotka tekevät työn, antavat lopullisen työmääräarvion.

3.9. Scrumin tuotokset

Scrumin tuotokset on suunniteltu maksimoimaan läpinäkyvyyttä, jolloin kaikilla on yhdenmukainen käsitys tuotoksesta.



KUVIO 2. Scrumprosessi (Schwber K. & Sutherland J. 2013).

Tuotteen kehitysjono (Product Backlog)

Tuotteen kehitysjono on tuotevastaavan ensisijainen työkalu jolla hän ohjaa tuotekehitystä vastaamaan asiakastarpeita.

Tuotteen kehitysjono on järjestetty lista tuotteeseen mahdollisesti tulevaisuudessa tehtävistä muutoksista. Se on ainoa lähde tuotteeseen toteutettaville muutoksille. Nämä muutokset voivat olla uusia ominaisuuksia, vaatimuksia, parannuksia ja korjauksia.

Tuotteen kehitysjono ei ole koskaan valmis.

Tuotejonon alkupäässä ovat korkeamman prioriteetin kehityskohdat. Tuotteen kehitysjonossa korkeammalla prioriteetilla olevat kohdat [story] ovat yleensä selkeämpiä ja yksityiskohtaisempia kuin matalammalle järjestetyt kohdat. Seuraavaan sprinttiin valittavat tuotekehitysjonon kohdat tulee työstää niin pieniksi että kukin valittu yksittäinen kohta voidaan tehdä valmiiksi sprintin loppuun mennessä. (Schwber K. & Sutherland J. 2013).

Tuoteomistajan vastuulla on tuotteen kehitysjonon järjestäminen ja täydentäminen. Tuotteen kehitysjonon kohtiin täytyy sisältyä vähintään seuraavat tiedot: kehityskohteen kuvaus, kohteiden järjestys ja työmääräarvio. (Schwber K. & Sutherland J. 2013).

Tuotejonon järjestämisessä täytyy huomioida erilaisia asioita, kuten:

- työmäärä
- riippuvuudet
- riskit
- arvo (value)
- viivästymisen kustannukset

Tuotteen tuotekehitysjonon täytyy sisältää työtä jolla on arvoa loppukäyttäjille ja sidosryhmille.

Kehitystiimin tehtävä on antaa työmääräarviot kehitysjonon kohteille. Tuoteomistaja voi vaikuttaa kehitystiimiin auttamalla sitä ymmärtämään vaatimuksia ja tekemään kompromisseja mutta ihmiset jotka tekevät työn, antavat lopullisen työmääräarvion. (Schwber K. & Sutherland J. 2013).

Sprintin kehitysjono (Sprint Backlog)

Sprintin kehitysjono muodostuu sprinttiin valituista kehitysjonon kohdista (story) sekä suunnitelmasta niiden toteuttamiseksi. Sprintin kehitysjono on kehitystiimin antama ennuste siitä, mitä toiminnallisuutta seuraavaan tuoteversioon tulee sisältymään sekä tarvittavasta työstä toiminnallisuuden toteuttamiseksi. (Schwber K. & Sutherland J. 2013).

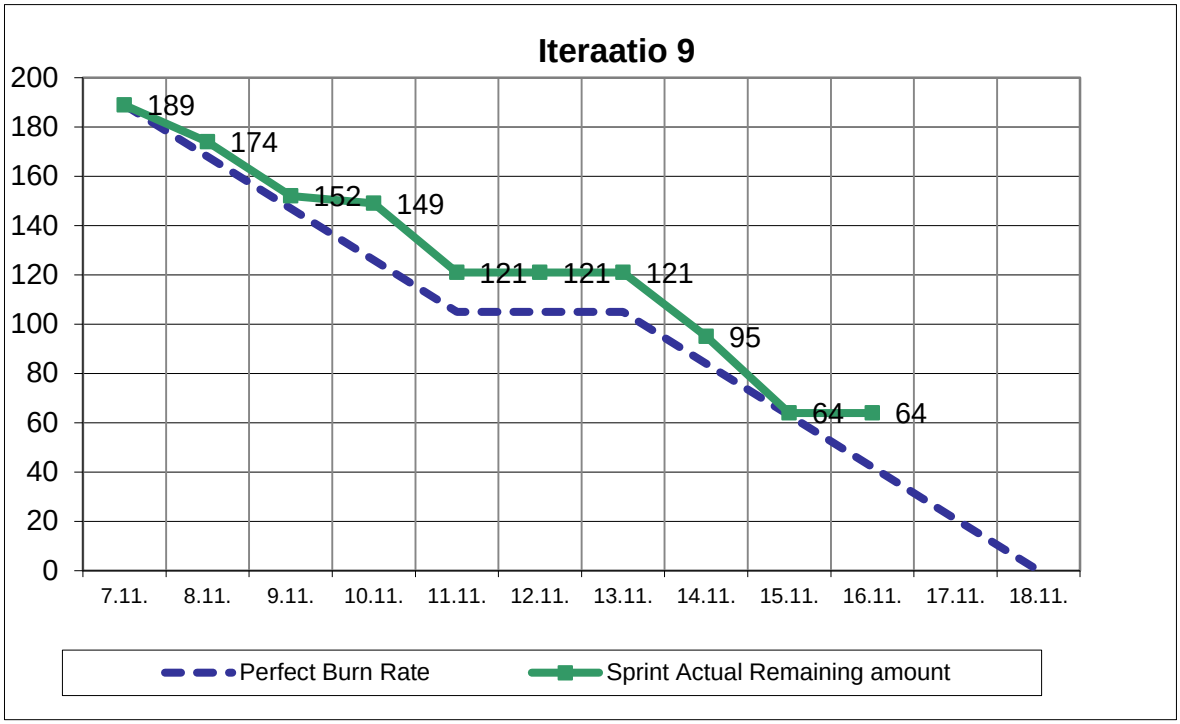
Sprintin kehitysjonon luo, omistaa ja ylläpitää kehitystiimi. Sprintin tuotekehitysjono on kehitystiimin työkalu jolla se voi ennustaa edistymistä kohti sprintin tavoitteita.

Suositeltu tekniikka samaan tilaan sijoitetuille tiimeille on ylläpitää sprintin tehtävälistaa fyysisellä tehtävätaululla, joka sijaitsee tiimin työtilassa. Tämä helpottaa sprintin tilanteen seuraamista.

Sprintin edistymisen seuraaminen

Sprintin kehitysjonon jäljellä oleva kokonaistyömäärä on milloin tahansa laskettavissa yhteen. Kehitystiimi päivittää tätä kokonaistyömäärää vähintään jokaisessa päiväpalaverissa arvioidakseen todennäköisyyttä sprintin tavoitteen saavuttamiselle. Tiimi tarkastelee omaa edistymistään tarkkailemalla jäljellä olevaa sprintin kokonaistyömäärää. (Schwber K. & Sutherland J. 2013).

Edistymiskäyrä eli burndown chart (Kuvio 3.) kuvaa tuotteen tai sprintin jäljellä olevaa työmäärää ajan funktiona. X-akseli kuvaa sprintin jäljellä olevaa aikaa päivinä ja Y-akseli jäljellä työmäärää tunteina.



KUVIO 3. Burndown chart

Tuoteversio (increment)

Tuoteversio on kokonaisuus joka muodostuu kaikista tuotteen kehitysjonon kohdista, jotka ovat valmistuneet sprintin ja aiempien sprinttien aikana. Sprintin lopussa kyseisellä sprintillä valmistuneenvalmistuneen tuoteversion tulee olla “valmis”, joka tarkoittaa että se täyttää Scrumtiimin “valmiin” määritelmän ja on käyttökelpoisessa kunnossa. Tuoteomistaja tekee lopullisen päätöksen että julkaistaanko valmis tuoteversio. (Schwber K. & Sutherland J. 2013).

Sprintin aikana tuotetaan ”valmis” –määritelmän täyttävä käyttökelpoinen ja potentiaalisesti julkaisukelpoinen tuoteversio.

”Valmiin” määritelmä (definiton of done)

Kun tuotteen kehitysjonon kohdan tai tuoteversion sanotaan olevan ”valmis”, kaikkien tulee ymmärtää mitä ”valmis” tarkoittaa. Tämä määritelmä saattaa vaihdella eri Scrumtiimien välillä mutta tiimin jäsenillä tulee olla yhteinen ymmärrys siitä mitä valmis työ tarkoittaa. (Schwber K. & Sutherland J. 2013).

Esimerkkejä ”valmiin” määritelmästä:

- Implementoitu
- Katselmoitu ulkopuolisen toimesta
- Testattu ja testit läpäisty
- Ominaisuutta koekäytetty onnistuneesti
- Asiakas hyväksynyt tuotoksen
- Tuotos sisällytetty seuraavaan julkaisuun

Kehitystiimin ohjaaminen ja vision merkitys

Käytännössä kehitystiimin ohjaaminen voi olla hyvinkin haastavaa mutta koetellut käytännöt auttavat tässä työssä. Visio auttaa johdattamaan kehitystä oikeaan suuntaan.

Kaikilla kehitystiimin jäsenillä pitäisi olla näkemys siitä mikä on visio, ketkä ovat tärkeimmät asiakkaat, miten voimme tuottaa parhaiten arvoa ja mitä ovat pahimmat kilpailijamme. Visio auttaa PO:ta tekemään päätöksiä ja sanomaan mitkä kohdat kehitysjonossa toteutetaan ja mitä ei toteuteta.

Scrum ja laatu

Scrum tukee jatkuvaa parantamista. Scrumin ajatusmaailmaan kuuluva läpinäkyvyys auttaa myös organisaatiota ja siellä työskenteleviä ihmisiä kehittämään työmenetelmiä ja omaa työtään. Yksi jatkuvan parantamisen työkalu on retrospektiivi jossa keskustellaan työskentelykäytännöistä ja siitä mitä edellisessä sprintissä tehtiin hyvin ja mitä voitaisiin parantaa. Iteratiivinen malli itsessään tukee jatkuvaa parantamista – näin organisaatio pystyy analysoimaan omaa toimintaansa ja tekemään tarvittavia korjausliikkeitä. Scrumin etuna pidetään sitä että tiimissä työskentelevät ihmiset ottavat itse vastuuta jatkuvasta parantamisesta ja laadun kehittämisestä, tällöin voidaan saavuttaa nopeasti näkyviä tuloksia työmotivaatioon ja tuottavuuteen.

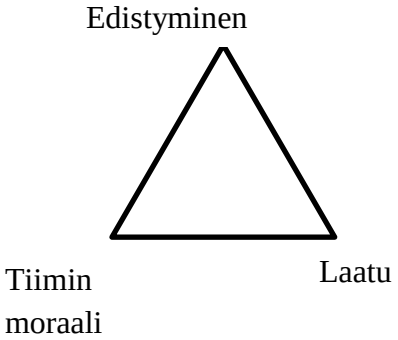
4. MITTARIT JA MITTAAMINEN

KPI eli Key performance indicator, käytetään usein suomennosta ”keskeinen suorituskykymittari”. Se mittaa organisaation suorituskykyä suhteessa sille asetettuihin tavoitteisiin. Mittarin tarkoitus on antaa palautetta kehitettävästä tuotteesta, laadusta, tiimin toiminnasta tai prosessista. KPI:t ovat mittareita mutta kaikki mittarit eivät ole keskeisiä suorituskyvyn mittaamisen kannalta. KPI on yhdessä kumppanien kanssa valittu ja sovittu indikaattori joka kertoo saavutetaanko sovitut tavoitteet. Kumppani voi olla organisaation sisäinen osasto tai ulkopuolinen palveluntarjoaja.

Kehitysprojektit eivät ole erityksissä muuta maailmasta. Ketterien tuotekehitysprojektien mittaamisella on kaksi tarkoitusta. Ulkopuolisille sidosryhmille annetaan säännöllisesti tietoa kehitystoiminnasta. Liiketoiminta ja projektien rahoittajat ovat luonnollisesti kiinnostuneita kehitystoiminnan saavutuksista. Toinen tarkoitus mittaamisella on antaa tuotekehitystiimille tietoa tietoa edistymisestä ja saavutuksista.

Ketterän raportoinnin kolme ulottuvuutta.

Ketterän ohjelmistokehityksen raportoinnissa on havaittavissa kolme ulottuvuutta (Kuvio 4.). Tuotekehityksen edistymisen mittaamista varten on olemassa omat mittarit, näistä tyypillinen esimerkki on velositeetti. Toinen oleellinen mittari on laatu, sitä mitataan esimerkiksi mittaamalla ohjelmistovirheiden määrää tietyllä ajanjaksolla. Kolmas, vaan ei vähäisin, mitattava kokonaisuus on tiimin moraali. Tiimin moraali on sidoksissa ihmisten johtamiseen, yleiseen työhyvinvointiin ja muihin ”pehmeiksi” koettuihin asioihin, jotka saattavat olla haastavia teknisille ihmisille. Tärkeää on kuitenkin havaita että näiden kolmen asian välillä on selvä yhteys, yhden asian muuttuminen vaikuttaa väistämättä muihin. (Krebs 2009, 110.)



KUVIO 4. Ketterän raportoinnin kolme ulottuvuutta (mukaillen Krebs 2009, 111)

Laadun mittaamisella voidaan arvioida joko teknistä laatua tai asiakastyytyväisyyttä. Tehokkuutta mittaavat mittarit mittaavat resurssien käyttöä. Arvon mittaamiseen on kehitetty mittareita jotta saadaan selvitettyä asiakkaalle toimitettu lisäarvo. Ketteryyttä voidaan myös mitata, tällöin mitataan töiden etenemistä ja ennustettavuutta sekä suunnittelun tarkkuutta.

Hyvän mittarin ominaisuuksia

Heuristics for wise agile measurement (Hartmann & Dymond, 2006)

A good metric or diagnostic:

1. *Affirms and reinforces Lean and Agile principles.*
2. *Measures outcome, not output.*
3. *Follows trends, not numbers.*
4. *Belongs to a small set of metrics and diagnostics.*
5. *Is easy to collect.*
6. *Reveals, rather than conceals, its context and significant variables.*
7. *Provides fuel for meaningful conversation.*
8. *Provides feedback on a frequent and regular basis.*
9. *May measure Value (Product) or Process.*
10. *Encourages "good-enough" quality.*

Jotta mittaaminen johtaa toivottuun lopputulokseen, täytyy ensin miettiä mikä on ongelma jota yritetään ratkaista. Tämän jälkeen voidaan asettaa tavoite johon pyritään. Mikäli tavoite on mitattavissa, kerätään mittaustieto joka kertoo missä kohtaa ollaan menossa suhteessa tavoitteeseen.

Mittaustuloksilla ei ole tarkoitus heikentää tiimin moraalia tai musertaa sen tahtoa vaan voimistaa tiimin omaa kehittämistä ja parantaa moraalia. Hyvä mittari tarjoaa tietoa tiimin toiminnasta ja eväitä kehittämiseen, sen tarkoitus ei ole asettaa paineita tiimille tai yksilöille.

Mittaaminen ei pelkästään riitä vaan mittaustulokset täytyy tehdä näkyviksi tiimille ja kaikille jotka asiaan osallistuvat. Mittaustuloksia täytyy käydä läpi ja niiden tulee johtaa toimenpiteisiin.

Haasteena mittaamisessa on löytää mittareita jotka rohkaisevat oikeanlaisen toimintaan. Tavoitteeksi kannattaa asettaa mittareiden määrän vähentäminen ja sellaisten korkean tason mittareiden löytäminen jotka johtavat oikeanlaiseen toimintaan tiimi- ja yksilötasolla. (Leffingwell D. 2007 s.237).

Esimerkki huonosta mittarista: Tiimille oli asetettu mittariksi "koodiriviä per tunti" tämä sai aikaan sen että laatu kärsi. Laatu ja tehokkuus paranivat kun uudeksi tavoitteeksi asetettiin virheiden minimoiminen.

4.1. Lean: mittaaminen ja tunnusluvut

Lean-ajattelun mukaan mittaamisen tarkoitus on seurata prosessien toimivuutta ja havaita poikkeamat. Mittaamisen ja mittareiden tarkoitus ei myöskään ole asettaa paineita työntekijöille. (Kouri I. 2010. s. 28).

Hyvien mittareiden löytäminen kehitystiimeille on haastavaa. Ohjelmistokehittäjät eivät pidä mittaamisesta ja kokevat sen uhka heidän itsenäiselle työlleen. Vääränlaiset mittarit aiheuttavat vääränlaista toimintaa eivätkä rohkaise oikeanlaista käyttäytymistä.

Läpimenoaika (cycle time) on keskeinen Leanin mittari. Tärkeintä ei ole mitata kaikkein nopeinta toimitusaikaa eli nopeinta mahdollista toimitusta vaan kuinka nopeasti voidaan toimittaa luotettavasti ja toistuvasti uusi asiakkaan tilauksen mukainen ominaisuus.

Ensin täytyy mitata määritellyille töille läpimenoaika, joka on luotettava ja toistettava. Tämän jälkeen tätä läpimenoaikaa parannetaan jatkuvan parantamisen keinoilla, yksi hyvä malli tähän on PDCA-laatuympyrä. Keskittyminen läpimenoajan pienentämiseen kiinnittää työntekijöiden huomion oikeisiin asioihin: laatuun, yhteistyöhön, standardointiin ja hyvien käytäntöjen jalkauttamiseen. (Leffingwell D. 2007. s.238).

Useimpien kehitystoimenpiteiden rahoitus perustuu liiketoiminnan tarpeisiin. Poppendiecien kirjassa suositellaan kehitystoiminnan mittaamista liiketoiminnan tarpeiden realisoitumisella eli nk. business casen toteutumisella. Tavoite on mitata kehitystoiminnan todellista taloudellista kannattavuutta.

Kuten on todettu, Leanin perusajatus on arvon tuottaminen asiakkaalle. Asiakkaiden perustarpeiden tyydyttäminen ei riitä, vaan asiakkaille täytyy tarjota enemmän arvoa. Tällainen arvo voi olla asiakkaan liiketoiminnan tarpeiden syvällinen ymmärtäminen ja tehokkuutta lisäävien ratkaisujen tarjoaminen asiakkaille. Kirjassa ehdotetaan yksinkertaisen ja selkeän kyselyn tekemistä jolla mitataan asiakastyytyväisyyttä. Suositteluaste on hyvä mittari asiakastyytyväisyyden mittaamiseen. (Leffingwell D. 2007. s.241).

Leanin tyypillisiä tunnuslukuja ovat: (Kouri I. 2010. s. 28)

- Tuottavuus

- Laatu

- Läpäisyaika

- Keskeneräinen tuotanto

- Hukka

4.2. Ketterä ohjelmistokehitys ja mittaaminen

Ketterän ohjelmistokehityksen mittaamisen perustana on toimitettu ohjelmakoodi. Yksinkertaistettuna arvioinnin perusteena on se onko ohjelmakoodi toimivaa vai ei. Scrum-ideologiaan liittyy kokeellinen ja iteratiivinen lähestymistapa ennustettavuuden ja tarkan suunnittelun sijaan. Vaikka ketterä kehitys varsin vapaamuotoiselta toiminnalta mutta sitä voidaan mitata monelle eri tasolla. Erilaisia mittaustietoja voidaan kerätä tiimitasolla, projektitasolla ja prosessitasolla. Yritystasolla suositeltava lähestymistapa on balanced scorecard –tyyppinen mittaaminen. (Leffingwell D. 2007. s. 311).

> *The primary metric for agile software development is wheather or not working software actually exists and is demonstrably suitable for use in its intended purpose. In agility, that key indicator is determined empirically, by demonstration, at the end of iteration and every release. (Dean Leffingwell: Scaling software agility).*

Monet ketterän kehityksen mittarit on sisällytetty tiimien päivittäiseen sisäiseen toimintaan, tämä hankaloittaa mittaustiedon keräämistä ja asettaa haasteita nopealle prosessimonitoroinnille. Esimerkkinä tästä on tiimien pitävä retrospektiivit joita tiimi pitää säännöllisesti kehittääkseen omaa toimintaansa.

Agile-tiimien suorituskyvyn mittaaminen voidaan jakaa kahteen osaan: projektimittarit ja prosessimittarit.

Ketterät projektimittarit

Ketterien tiimien projektimittarit mittaavat tiimien suorituskykyä projektitasolla. Eli tiimin kykyä kehittää komponenttia ja kykyä viedä nämä muutokset tuotantoon. Tämäntyyppiset projektimittarit voidaan jakaa iteraatiomittareihin (Iteration metrics) ja versiokohtaisiin mittareihin (Release metris). Iteraatiokohtaiset prosessimittarit ovat nopeasyklistä kehitystä mittaavia mittareita jotka tuottavat nopeaa palautetta tiimien toiminnan kehittämiseen. Versiomittarit mittaavat toimintaa hitaammalla syklillä ja näiden mittareiden painotus on korkeamman asiakasarvon ohjelmistoversioissa. Nämä mittarit ovat siis tiimien välineitä oman toimintansa laadun ja suorituskyvyn parantamiseen kuten esimerkiksi velositeetti per sprintti ja toimitettujen versioiden määrä per sprintti.

Ketterät prosessimittarit

Lisäksi on olemassa prosessimittareita mittaamaan ketterän ohjelmistotiimin prosessin kypsyyttä. Periaatteena on jakaa ketterä ohjelmistokehitys osiin ja arvioida jokaisen osa-alueen kypsyyttä erikseen. Arvioitavia kyvykkyyksiä osa-alueittain ovat: tuoteomistajuus, versionsuunnittelu, iteraatiosuunnittelu, tiimin tehokkuus, testauskäytännöt ja kehitystyön käytännöt. (Leffingwell D. 2007. s. 313).

5. NYKYISET KPI-MITTARIT

Scrumtiimit käyttävät oman toimintansa mittaamiseen sekä burndown chart -kaaviota ja velositeettia. Nämä ovat Scrumin vakiomittareita, luonnollisesti Scrumittimien toimintaa mitataan sillä että miten ne pystyvät toimittamaan valmiita julkaisukelpoisia ohjelmistoversioita.

Tämän lisäksi on yhdessä sovittu erilaisia KPI-mittareita joilla ketterien ohjelmistokehitystiimien toimintaa tällä hetkellä mitataan.

5.1. Julkaisuvalmiiden ominaisuuksien määrä kuukausittain

Number of tested and accepted features available for releasing

Tämä mittari mittaa uusien ominaisuuksien kehittämisen edistymistä kuukausittain. Toimittajan tulee toimittaa testattuja ja hyväksyttyjä asiakkaalle arvoa tuottavia ominaisuuksia jotka ovat valmiita käyttöönotettavaksi. Mittari kertoo kuinka monta uutta ominaisuutta on hyväksytty kuukauden aikana. Tavoite on pitää hyväksyttyjen ominaisuuksien määrä suunnitellulla tasolla. Mittari mittaa paitsi suunnittelun onnistumista niin myös tiimien kykyä toimittaa valmiita ominaisuuksia. Yksi ominaisuus saattaa koostua useista eri tiimien tekemistä ”user storyista”. (Company internal material. 2014).

Haasteena tässä mittaamistavassa on se että kehitettävät ominaisuudet voivat olla hyvinkin erikokoisia ja vaativuudeltaan sekä vaikuttavuudeltaan erilaisia, tällöin niiden laskeminen ei anna hyvää kokonaiskuvaa tilanteesta.

Toinen tähän aiheeseen liittyvä haaste on että kehitettävillä ominaisuuksilla on erilaisia merkityksiä liiketoiminnan kannalta. Ratkaisu olisi arvottaa kehitettävät ominaisuudet merkittävyyden ja koon suhteen siten että ensin tehdään sellaiset pienet ominaisuudet joiden painoarvo liiketoiminnalle on suuri

TAULUKKO 2. Number of tested and accepted features available for releasing. (Company internal material. 2014)

Scope	Number of tested and accepted features available for releasing
Period	Monthly
Level	Product area
Type	Absolute number, cumulative progress
Target	Keep the number of accepted features on the planned level
Responsible	AD leads

5.2. Toimitettujen “user storyjen” määrä

Number of backlog items delivered (user stories)

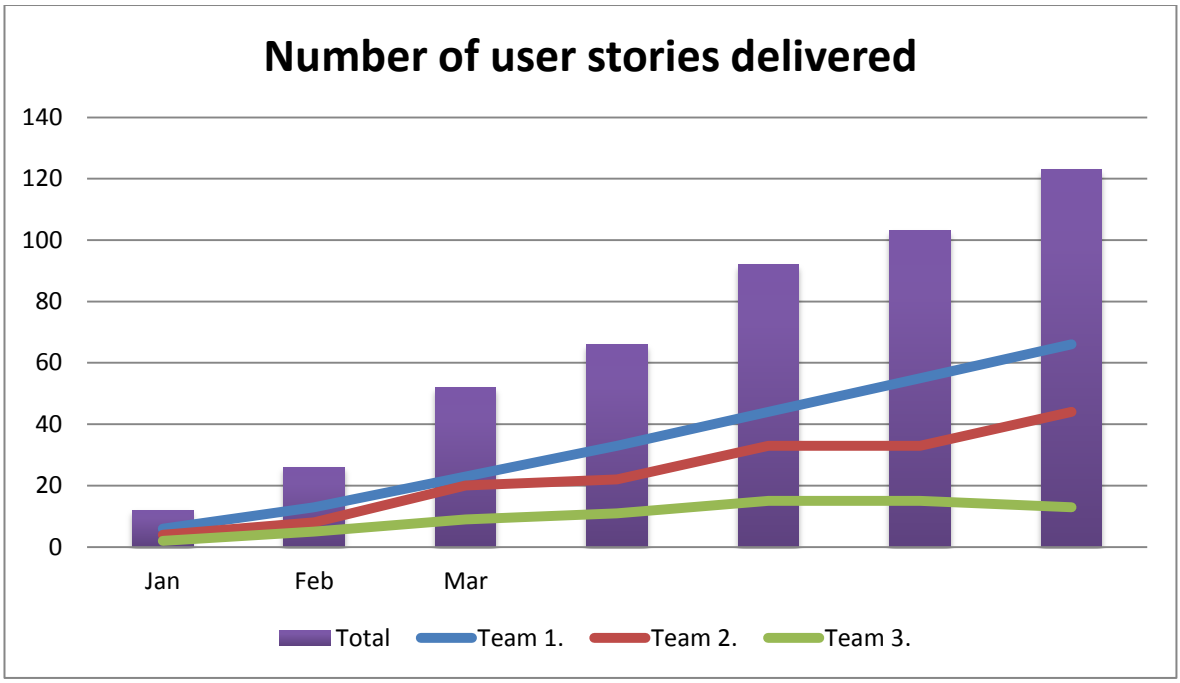
Mittari mittaa kuukausittaista ohjelmistokehityksen edistymistä tiimitasolla ja yhteenlaskettuna tuotealuetasolla. Tiimin täytyy toimittaa täysin käyttöönotettuja ja testattuja ”user storyjä” sprintistä toiseen. Mittaus kertoo kuinka monta ”user storya” tiimi on onnistunut toimittamaan ja mikä toimitusten trendi on. Tavoite on että sprintin lopussa kaikki suunnitellut ”storyt” on tehty ja osittain valmiita ei ole. (Company internal material. 2014).

Toimitettujen käyttäjätarinoiden määrä on selkeä tiimikohtainen mittari, joka tarjoaa eväitä tiimeille oman toimintansa kehittämiseen ja tulevien sprinttien suunnitteluun. Haasteena tässäkin on se että kehityskohteet voivat olla hyvin erikokoisia ja niiden määrän laskeminen suoraan ei kerro tiimin todellista tulosta.

TAULUKKO 3. Number of backlog items delivered (user stories) (Company internal material. 2014.)

Scope	Number of backlog items delivered (user stories)
Period	Yearly, reported monthly
Level	Team
Type	Absolute number, trend
Target	x% increase from year to year
Responsible	Scrum teams

Kuvio 5. esittää kumulatiivisesti kuukausittain toimitettujen ”user storyjen” määrän. X-akselilla on kuukausi ja Y-akselilla toimitettu määrä kappaleina.



KUVIO 5. Number of user stories delivered.

5.3. Mittaustyökalun ilmoittama testauksen automatisointisuhde

Test automation ratio reported by test coverage (%) in test tool

Mittari kertoo täysin automatisoitujen testitapausten määrän suhteessa kaikkiin testitapauksiin. Testiautomaatioprosenttia mitataan joka kuukausi tiimi- ja tuotealuetasolla. Testiautomaatioprosentin kuukusitrendi kertoo testiautomaation käyttöönoton edistymisen.

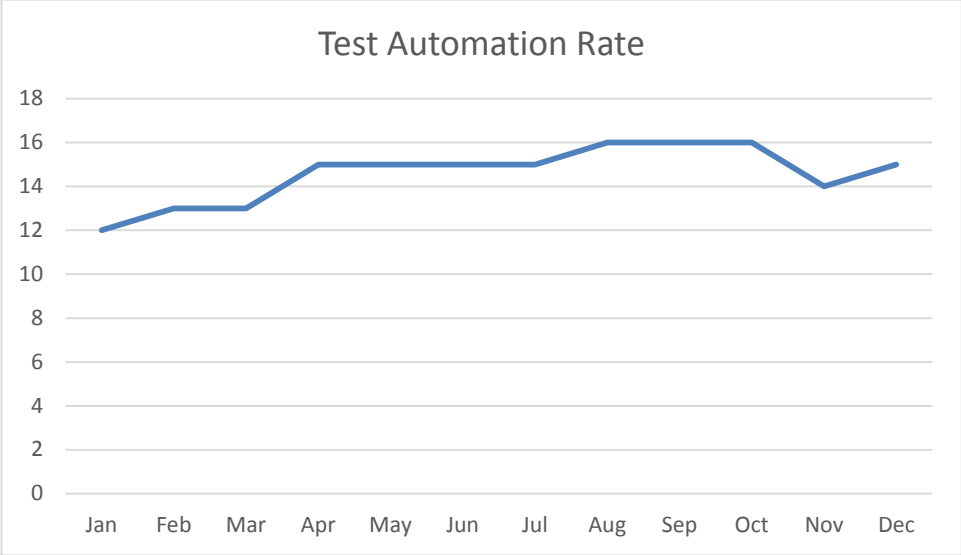
Tavoite on lisätä testiautomaation osuutta testaamisesta ja samalla vähentää manuaalitestaukseen käytettävää aikaa. ja samalla nopeuttaa ohjelmistokehitystä. Kannatta huomioida että automatisoidun testauksen määrä ei nouse jatkuvasta, manuaalitestausta tarvitaan edelleen ”päästä-päähän” testaamiseen ja käytettävyyden testaamiseen. 100% testiautomaatio-osuus ei ole realistinen eikä mahdollinen. (Company internal material. 2014).

Tämä mittari on Lean-ajattelun mukainen. Lean kannustaa automatisoimaan mahdollisimman paljon. testauksen automaatiosuhteen nostaminen vaikuttaa positiivisesti ennen kaikkea laatuun. Haasteena tässä on edelleen yleisen tietoisuuden lisääminen testiautomaatiosta ja sen hyödyistä.

TAULUKKO 4. Test automation ratio reported by test coverage (%) in test tool [1]

Scope	Test automation ratio reported by test coverage (%) in test tool
Period	Yearly, reported monthly
Level	Team
Type	Absolute number, trend
Target	Increase automation level x% until end of 2015 (versus end of 2014)
Responsible	Scrum teams

Kuvio 6. esittää automaatiotestauksen prosenttiosuutta kokonaistestauksesta kuukausitasolla. X-akselilla on kuukausi ja Y-akselilla automaatiotestauksen prosenttiosuus.



KUVIO 6. Test Automation Rate

5.4. Bugikorjaukseen kuluva keskimääräinen aika

Time from “new” to “closed” state for bug fixes

Mittari mittaa ohjelmistovikojen eli ”bugien” korjausnopeutta, käytännössä mitataan tilojen “new” ja “closed” –tilojen välistä aikaa. Sprintin lopussa ilmoitetaan sprintin aikana bugien korjaamisen käytettyjen aikojen keskiarvo. Mittausten kuukausittaisen keskiarvon trendi kertoo miten ratkaisuaika kehittyy. Tavoite on pienentää keskimääräistä ratkaisuaikaa vuoden aikana. Mittaus kertoo ohjelmistovikojen korjauksen tehokkuudesta.

Mikäli ohjelmistovirheellä on palvelusopimukseen perustuva erillinen käsittelyprosessi, sitä ei huomioida tässä KPI-mittauksessa. (Company internal material. 2014).

Bugikorjaukseen liittyy useita haasteita. Jatkuvasti joudutan käymään rajanvetoa siitä mikä on ohjelmistovirhe ja mikä on normaali kehityskohde. Prosessimielessä tietyt bugit ovat oman käsittelyprosessin piirissä ja alihankkija hoitaa ne tietyn sopimuksen mukaisesti. Nämä määrittelyt ja jaottelut asettavat haasteita periaatteessa selkeälle mittarille.

TAULUKKO 5. Time from “new” to “closed” state for bug fixes (Company internal material. 2014.)

Scope	Time from “new” to “closed” state for bug fixes
Period	Sprint – reported in the end of sprint
Level	Team
Type	Absolute number (time in days), trend
Target	Decrease correction time from x days to y days until end of an agreed time period.
Responsible	AD leads, Scrum teams

5.5. Tekninen velka

Technical debt measured by test tool

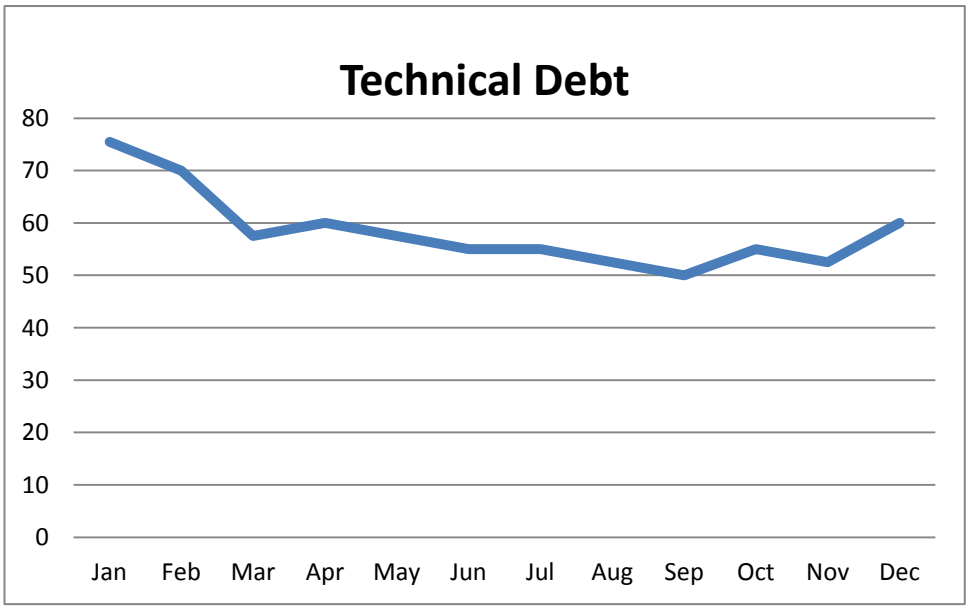
Mittaus kertoo teknisen velan määrän. Teknistä velkaa mitataan erillisellä työkalulla, joka käyttää määrättyjä algoritmeja teknisen velan mittaamiseen. Teknistä velkaa mitataan kuukausittain. Tavoitteena on lyhentää teknistä velkaa joka vuosi. Kehitystiimi on vastuussa teknisen velan pienentämisestä. Tekninen velka (technical debt) on ohjelmistokehityksessä käytettävä termi, joka tarkoittaa järjestelmän rapautumista, koodin monistamisesta johtuvaa huonoa laatua tai automaation puutetta. Esimerkki teknisestä velasta on tunnetun virheen jättäminen tarkoituksella (korjaamatta) järjestelmään. (Company internal material. 2014).

Teknisen velan käsite voi olla haastava. Termi aukeaa parhaiten esimerkkien kautta. Tähän liittyy mittarin hyödyntämisen haaste, tekninen velka täytyy viedä ymmärrettävästi tiimeille ja kehittäjille. Kun sovelluskehittäjät ymmärtävät mitä tekninen velka on ja miten siihen voidaan vaikuttaa – mittaus konkretisoituu toimenpiteiksi jotka parantavat laatua ja tuottavuutta. Keinona tähän on automaatiotyökalujen kouluttaminen kehittäjille.

TAULUKKO 6. Technical debt measured by test tool (Company internal material. 2014.)

Scope	Technical debt measured by test tool
Period	Year, reported monthly
Level	Team
Type	Absolute number (time in days), trend
Target	Decrease technical debt x% in test tool measurements until end of 2015 (versus end of 2014)
Responsible	AD leads, Scrum teams

Kuvio 7. esittää testaustyökalun ilmoittaman teknisen velan määrän kuukausitasolla. X-akselilla on kuukausi ja Y-akselilla testaustyökalun ilmoittama tekninen velka.



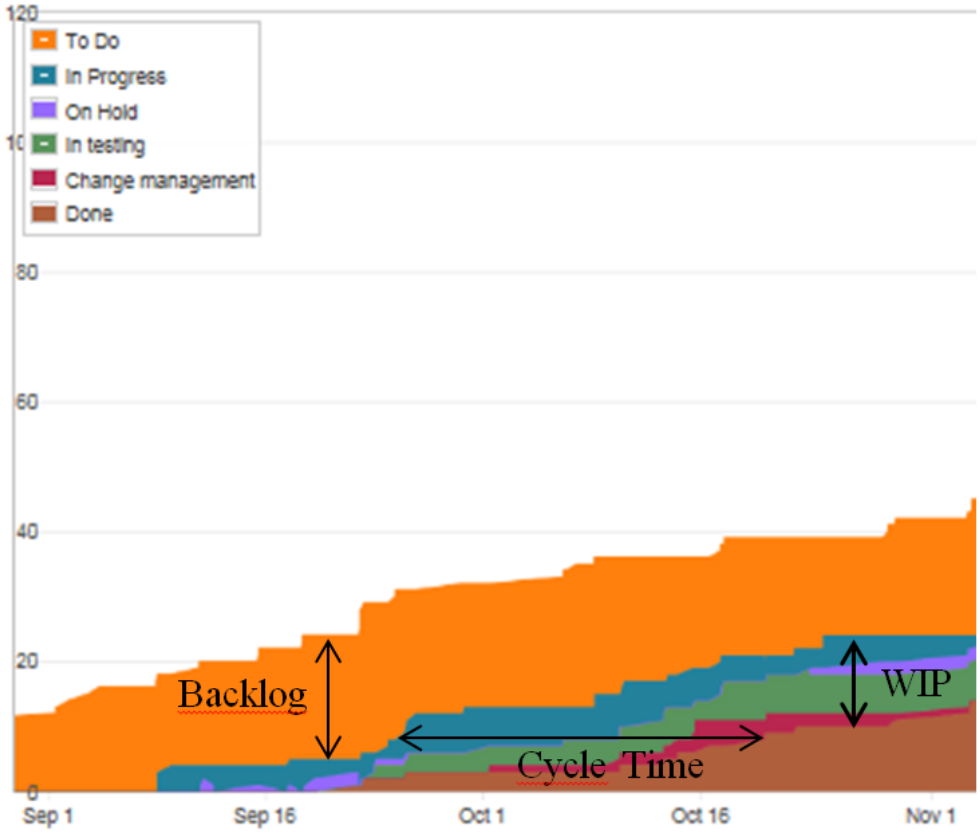
KUVIO 7. Technical Debt

6. EHDOTETUT TOIMENPITEET JA KPI:T PROSESSIEN PARANTAMISEKSI

6.1. Kumulatiivinen vuokaavio

Tehokas tapa visualisoida toimitusprosessin toimivuutta on kumulatiivinen vuokaavio (cumulative flow diagram). Tämä tekniikka pohjautuu Lean-toimintamalliin. Lean-periaatteen mukaisesti tavoitteena on tasaisesti etenevä vuo, jossa mihinkään työvaiheeseen muodostu pullonkauloja. CFD eli cumulative flow diagram on aluegrafiikka, joka näyttää töiden virtauksen suhteessa aikaan. Kuvio X. on esimerkki kumulatiivisesta vuokaaviosta. Kaaviosta voi todeta tietyn hetken tilanteen, montako kehityskohdetta eli ”user storya” on kulloinkin eri tilassa. Nämä tilat ovat:

- Backlogilla (To do)
- Työn alla (In progress)
- Odottamassa (On hold)
- Testauksessa (In testing)
- Muutoshallinnassa (Change management)
- Valmiina (Done)



KUVIO 8. CFD

Kumulatiivinen vuokaavio (Kuvio 8.) sisältää tietoa:

- Läpimenoajasta (Cycle Time)
- Työn alla olevista kehityskohteista (WIP=work in progress)
- Jäljellä olevasta työmäärästä (Backlog)
- Eri vaiheiden jonoista
- Pullonkauloista

Toimeksiantajan ketterien tiimien ohjaamiseen käyttämä työkalu sisältää CFD:n raportointityökalut. Se piirtää jatkuvasti koko uutta kaaviota ja historiatiedon tarkastelu tietyltä aikaväliltä on mahdollista.

CFD:n kuvaamisen ja tarkastelun mahdollistavat työkalut ovat siis kunnossa. Kehitettävää on sen sijaan siinä miten kumulatiivista vuokaaviota hyödynnetään.

CFD on tehokkaimmillaan silloin kun tunnistetaan toimitusprosessissa olevia pullonkauloja. Kaavio paljastaa eri kohdissa olevan jonomuodostuksen ja mahdollistaa näin juurisyiden selvittämisen ja korjaamisen. Jos testauksessa ongelmia tietyllä ajanhetkellä vihreä osuus paljon suurempi, kts. Kuvio 7.

Kumulatiivista vuokaaviota voidaan käyttää yleensä työn etenemisen seurantaan. Pitkältä aikaväliltä kuvattu CFD antaa arvosta tietoa tiimin kehittymisestä ja työn virtaamisesta. kaaviosta voi päätellä myös mahdollisia resurssikapeikkoja ja työmäärän kehittymistä.

Toimeksiantajan tapauksessa kumulatiivista vuokaaviota voi hyödyntää kahdella tasolla. Johto voi käyttää sitä työvuon seurantaan ja resurssien kohdentamiseen. Tiimitasolla kumulatiivista informaatiota voidaan käyttää tiimin retroissa keskustelun herättäjänä ja jatkuvan parantamisen välineenä.

6.2. Läpimenoaika ja arvovirtakuvaus

Toimeksiantajan tapauksessa kehitystiimit käyttävät scrumia ja niiden toiminta ja prosessit ovat vakiintuneita. Prosesseja kehitetään jatkuvasti organisaatiossa käytössä olevilla vakiintuneilla käytännöillä. Keskusteluissa on käynyt esille että kehitystiimien toiminnassa tärkeimmät kehityskohteet liittyvät toimitusten nopeuttamiseen. Tiimien toimintaa halutaan vieläkin nopeammaksi ja ketterämmäksi. uusia ominaisuuksia halutaan jatkuvasti tuotantoon, ilman viiveistä. tavoitteena tasainen vuo pieniä muutoksia jotka viedään tuotantoon saman tien.

Jotta tämä onnistuisi täytyy prosessia kehittää edelleen. Keinoja toimitusten nopeuttamiseksi on muutosten pitäminen niin pieninä että ne voidaan toteuttaa ja viedä tuotantoon yhden sprintin aikana. Scrumin periaatteena on viedä tuotantoon nopeasti sellaisia pieniä muutoksia joiden arvo asiakkaalle on mahdollisimman suuri.

Toinen keino tavoitteiden saavuttamiseksi on ennakointi. Jokaisen muutoksen vienti vaatii muutoshallintaprosessin läpikäynnin. Muutoshallinta on eräänlainen ITIL-mutka ketterässä ohjelmistokehityksessä ja muodostaa hitaustekijän tuotantoon viennissä mikäli sitä ei huomioida oikealla tavalla. Tiimin täytyy arvioida ennakoida muutoshallinnan vaatima aika ja tehdä muutoshallintapyyntö mahdollisimman aikaisessa vaiheessa, sprintin alussa.

Kolmas keinotoimitusten nopeuttamiseksi löytyy Lean-ajattelun työkalupakista. kaikenlaisen hukan tunnistamiseksi poistamiseksi täytyy tehdä pitkäjänteisesti työtä. asiakasarvon tunnistaminen ja siihen keskittyminen luonnollisesti nopeuttaa toimitusaikaa.

Ehdotan siis uudeksi mittariksi läpimenoaikaa. Läpimenoaikaa mitattaessa ei mitata lyhyintä aikaa, vaan keskimääräistä aikaa jolla uusia toiminnallisuuksia voidaan luotettavasti toimittaa. Kehitystiimien töidenhallintaan käytettävässä järjestelmässä on ominaisuuksia joilla läpimenoaika voidaan mitata. Toisaalta täytyy todeta että tämä Control chart - työkalu (Kuvio 9.) ei anna kovin selkeää kuvaa läpimenoajan kehityksestä, käytännössä raportointityökalu vaatii tulkintaa ja tendenssin seuraaminen on hankalaa. Puutteena voidaan pitää sitä että Jiran ei ole satavissa widget-raportointityökalua läpimenoaikojen seurantaan. Kehityskohteena esitän että kannattaa seurata, josko toimittaja sellaisen tulevaisuudessa kehittää tai löytyykö markkinoilta hyvä työkalu läpimenoaikojen seurantaan.

Name	Mean	Median	Standard Deviation	Min Time	Max Time
Cycle Time (Working days only)	7 days 2 hours 43 minutes	7 days 12 hours 26 minutes	5 days 20 hours 55 minutes	0 minutes	14 days 21 hours 42 minutes

KUVIO 9. Läpimenoaika

Läpimenoaika on Lean-ajattelun perusmittareita. Läpimenoaika mittaa aikaa joka keskimäärin kuluu tilauksesta toimitettuun asiakkaan tarpeen mukaisen ohjelmiston toimittamiseen. Läpimenoaikaa kertoo prosessin kyvykkyydestä järjestelmätasolla. Lisäksi se tuo esille prosessissa ilmenevän hukan. Jokainen puuttuva taito, heikko kyvykkyys ja viallinen käyttöönotto lisäävät läpimenoaikaa. Läpimenoaikaa lisää jos yritetään tehdä liian paljon, samoin monimutkaisuus, tarpeettomat riippuvuudet ja kyvyttömyys muutokseen.

Arvovirtakuvaus

Kehitystoimenpiteenä ehdotan arvovirtakuvauksen piirtämistä riittävän tarkalla tasolla. Kohteeksi ehdotan uuden keskikokoisen ominaisuuden toimitusprosessia. Tilaus alkaa siitä kun tiimi ottaa ”user storyn” työn alle ja päättyy kun se on onnistuneesti käyttöönotettu. Oikein toteutettuna arvovirtakuvaus eli VSM on erittäin tehokas kehitystyökalu. Arvovirtakuvauksen piirtämisen jälkeen parannustoimenpiteet kannatta aloittaa siitä kohdasta jossa eniten tulee hukkaa. Seuraavaksi työn alle otetaan kohde jossa esiintyy toiseksi eniten hukkaa. Koko ketju käydään näin läpi jatkuvan parantamisen hengessä.

Kehitystoimenpiteiden vaikutukset voidaan arvioida läpimenoajan avulla. hukan poistaminen nopeuttaa läpimenoaikaa ja toimituksia.

Prosessin päästä-päähän toimivuus voidaan kuvata piirtämällä siitä arvovirtakuvaus (value stream map eli VSM). VSM tekee näkyväksi prosessissa tapahtuvaa hukkaa ja sen missä lisäarvon tuottaminen tapahtuu.

Jotta arvovirtaketjun kuvaaminen tuottaa hyötyä, täytyy prosessien ja tiimien toiminnan olla vakiintunutta. Mikäli toimintoketju ja prosessi hakevat muotoaan, ei VSM:n kuvaaminen anna oikeaa kuvaa tilanteesta.

Toinen haaste arvovirtaketjun kuvaamisessa on organisaatiorajat. Mikäli prosessissa on mukana useita eri organisaatioyksiköitä, on riski että arvovirtojen kuvaaminen aiheuttaa kitkaa organisaatioyksiköiden välillä.

Arvoketjun laatiminen (Poppendieck M. & Poppendieck T. 2008. s.85).

1. Valitse kuvattava prosessi. Yksittäisten toimintojen sijaan pyritään kuvaamaan prosesseja. Esimerkki kuvattavasta arvoketjusta on ”kauanko kestää uuden keskikokoisen ominaisuuden kehittäminen nykyiseen ohjelmistoon”.

2. Määrittele milloin kehittäminen alkaa ja mihin se päättyy.

3. Tunnista arvoketjun omistaja.

4. Yksinkertaista. Pyri hahmottamaan asiat asiakasnäkökulmasta. Kysymyksiä: Kauanko aikaa kuluu tuotteen kehittämiseen tai käyttäjän tarpeen täyttämiseen. Tarkoitus ei ole laskea laskutettavia tunteja vaan käytettyä aikaa. Montako prosenttia käytetystä ajasta käytetään varsinaiseen arvoa lisäävään työhön?

Paras menetelmä VSM:n kuvaamiseen on kysellä henkilökunnalta ja tarkkailla työntekoa eri vaiheessa. VSM:n piirtämistä on lähes mahdoton automatisoida. VSM:n kuvaaminen on sen tekijälle hyvä keino oppia prosessin toimintaa.

VSM:n piirtämiseen on olemassa graafisia työkaluja.

6.3. Tiimin viremittari

Tavoittaakseen päämääränsä tiimi tarvitsee haasteita, yhteisen päämäärän ja yhteisen sitoutumisen. Viisas organisaatio kohdistaa huomion, koulutukset ja resurssit sellaisen ympäristön luomiseen jossa työntekijät yltävät parhaaseen mahdolliseen suoritukseen. (Poppendieck & Poppendieck 2008, 127)

Työmoraalin ja yhteishengen vaikutusta tuottavuuteen ja laatuun ei kannata aliarvioida. Työhyvinvointiin vaikuttavat monet asiat: työntekijöiden kokema stressi, johtaminen ja tiimin yhteishenki. Lisäksi tiimin työmoraaliin vaikuttavat erilaiset tiimin kokemat haasteet esimerkiksi työkalujen toimivuudessa tai tietojen saamiseen liittyvissä asioissa.

Organisaation kannattaa tukea kaikin mahdollisin keinon sitoutumista ja innostusta. Ihmiset tekevät hyvää työtä silloin kun he kokevat että työ on mielekästä ja heillä on mahdollisuus vaikuttaa omaan työhönsä.

Paras keino selvittää tiimin työmoraalia on kysyä sitä suoraan työntekijöiltä. Esittelen tässä kaksi menetelmää joilla tiimin työmoraalia voidaan mitata.

Yksinkertaisimmillaan tiimin työmoraalia voidaan pyytämällä heiltä arvio tämänhetkisestä työtyytyväisyydestään asteikolla 1-10. 10 tarkoittaa että työntekijä on täysin tyytyväinen tiimiin ja omaan toimintaansa, 1 tarkoittaa sitä että henkilö on tyytymäyön ja haluaa tiimistä pois. Tämä kysely toteutetaan retrossa suljettuna lippuäänestyksenä. Moraalibarometri kertoo tiimin työtyytyväisyyden kehittymisestä eri sprinttien välillä, mutta yksityiskohtaisia tietoja ongelmakohdista se ei kerro. (Krebs, 2009. s.134).

Tiimin työtyytyväisyyskysely

Yksityiskohtaisempia tietoja tiimin tilasta antaa jokaisen sprintin lopussa toteutettava tiimin työtyytväisyyskysely. Jokainen tiimin jäsen vastaa anonyymisti kysymyksiin jotka koskevat

- Työtyytyväisyyttä
- Tiimin yhteishengestä
- Tiimin menestystä ja saavutuksia
- Oppimista

Lisäksi annetaan mahdollisuus antaa vapaamuotoista palautetta jotka antavat enemmän palautetta tiimin toiminnasta. Kyselyn yhteenvedot jaetaan tiimille ja käsitellään groomingeissa.

Lähteiden mukaan tiimin ja yksilön työmoraalin ja välillä on vahva yhteys. Samoin tiimin suorituskyky on riippuvainen sen työmoraalista. On havaittu että mitä pidempään tiimi työskentelee yhdessä, sitä parempi sen suorituskyky on. Tiimin vireen mittaaminen on iteraatioiden välillä antaa hyödyllistä tietoa tiimin tilasta. (Krebs, 2009. s.134).

6.4. Asiakastyytyväisyys ja arvon tuottaminen

Asiakkaiden perustarpeet täytyy pystyä tyydyttämään ja suorituskyvyn täytyy olla linjassa kilpailijoiden kanssa. Parhaiten menestyvät kuitenkin yritykset jotka ylittävät asiakkaiden odotukset ja ennen kaikkea ymmärtävät näiden liiketoiminnan tarpeet. Asiakkaiden odotusten ymmärtäminen luo huomattavan edun kilpailijoihin nähden.

Ketterässä ohjelmistokehityksessä tavoitteena on luoda arvoa liiketoiminnalle. Scrum-mallissa työt priorisoidaan liiketoiminnallisen arvon mukaan. Mitä nopeammin arvoa tuottavia ominaisuuksia toimitetaan, sitä enemmän investointi tuottaa arvoa. Tämän arvon mittaaminen sisältää oletuksia. Ohjelmistokehittäjät eivät yksin pysty arvioimaan kehitettyjen ominaisuuksien arvoa liiketoiminnalle vaan siihen tarvitaan apua liiketoiminnalta.

Haasteena tässä on kehitystoimenpiteiden ”storyjen” arvottaminen liiketoiminnan kannalta. Kaikkein suurin hyöty liiketoiminnalle saavutetaan silloin kun kehitystiimi tekee ensin sellaiset pienet kehityskohteet joiden arvo liiketoiminnalle on mahdollisimman suuri. Käytännössä tämä ominaisuus voidaan ottaa käyttöön lisäämällä uusi kenttä järjestelmään, johon toiminnallisuuden liiketoiminnallinen arvo kirjataan. Toiminnallisuuden arvon määrittävät tilaaja, eli liiketoiminnan edustaja ja tuoteomistaja yhdessä. Liiketoiminnallisen arvon kirjaamisen tavoitteena on lisätä tiimin ymmärrystä kehitettävän toiminnallisuuden merkityksestä ja helpottaa priorisointia.

Hyvä tapa arvioida kehitetyn ominaisuuden arvoa liiketoiminnalle on kysyä asiakkaan mielipidettä kehitetystä toiminnallisuudesta ja onko kehitettyä ominaisuutta hyödynnetty. Oheinen kuvio esittää menetelmän jolla asiakaskyselyiden tuloksia voidaan havainnollistaa. Tässä mallissa asiakkaalle esitetään sprintin onnistumiseen liittyviä kysymyksiä joihin asiakas vastaa arvioimalla kunkin kohteen onnistumista asteikolla 1-5. Asiakaskyselyn tuloksi voidaan käsitellä yhdessä tiimin retrospektiivissä ja miettiä siellä yhdessä toimenpiteitä toiminnan kehittämiseksi.

Asiakaskyselyn kysymykset, Kuvio 10. (Agilepearls 2014.):

- Tuottiko päättynyt sprintti lisäarvoa tuotteelle?
- Olivatko kaikki toiminnallisuudet (storyt) hyväksyttäviä?
- Olivatko uudet toiminnallisuudet odotetunlaisia?
- Tuletko käyttämään uutta toiminnallisuutta?
- Koetko että mielipiteitäsi ja kommenttejasi kuultiin?
- Onko sinulla mielessä uusia tarpeita julkaisuun?
- Arviosi kokonaisuutena päättyneestä sprintistä?

KUVIO 10. Asiakastyytyväisyyskysely (Agilepearls 2014).

Taloudellinen näkökulma

Parhaiten arvon liiketoiminnalle pystyvät arvioimaan liiketoiminnan henkilöt ja kehittäjät yhdessä. Tällöin tietylle ominaisuudelle tai projektille lasketaan kustannus jota sitten verrataan sen asiakkaalle tuotettuun arvoon.

Liiketoiminnan näkökulmasta on olemassa useita tapoja mitata arvoa. Yksi niistä on sijoitetun pääoman tuotto (ROI). Muita käyttökelpoisia liiketoiminnallisia mittareita ovat NPV (net present value eli nettonykyarvo) ja IRR (internal rate of return eli sisäinen/efektiivinen korko). Hartmann ja Dymond suosittelevat tutkimuksessaan että business määrittää parhaan menetelmän investoinnin arvoin mittaamiseen valitsee liiketoiminta. Iteratiivinen lähestymistapa on tähän mittaamiseen suositeltava. (Hartmann D. & Dymond R. 2006.)

7. YHTEENVETO JA POHDINTA

Tutkimuksessa perehdyin Lean-ajatteluun ja Scrumiin, joka on ketterän projektihallinnan viitekehys. Näiden teorioiden pohjalta pohdin KPI-mittareita ketterien ohjelmistokehitystiimien toiminnan mittaamiseen sekä menetelmiä toiminnan kehittämiseen.

Scrumista on olemassa viljalti materiaalia, kirjallisuutta, internet-lähteitä ja koulutusta. Ketterät menetelmät ja erityisesti Scrum ovat vakiintuneet ohjelmistokehityksen projektihallinnan menetelmäksi.

Lean on osoittanut tehonsa teollisuudessa ja nyttemmin Leanin ohjelmistokehitykseen suunnattu sovellus on aloittanut voittokulkunsa. Lean ohjelmistokehityksestä löytyy jonkin verran kirjallisuutta ja erilaisia internet-lähteitä. Aiheen perusteoksia ovat Mary ja Tom Poppendieckien kirjat.

Kuten aikaisemmin tutkimuksessa kirjoitin, on molemmille toimintamalleille oma paikkansa, ne täydentävät oivasti toisiaan. Scrum luo toimintamallin ketterälle tiimille ja Lean on menetelmä jolla prosesseja voidaan tehostaa ja laatua parantaa. Huono laatu johtaa asioiden uudelleen tekemiseen ja pahimmillaan siihen että tuote ei täytä asiakkaan tarpeita, jolloin joudutaan maksamaan sanktioita ja hyvityksiä.

Mielestäni inhimillinen ulottuvuus on erittäin tärkeä. Leanin-ajattelu pohjautuu henkilön kunnioittamiseen. Työntekijät ovat oman alansa asiantuntijoita, eivät siis tuotannontekijöitä joita voi käskeä ja kohdella miten tahansa. Henkilön kunnioittaminen korostuu ohjelmistokehityksessä, koska tehokkuus ja laatu ovat suoraan riippuvaisia työntekijöiden motivaatiosta. Kun tekemisissä ihmisten kanssa, on syytä olla korostaa myös kommunikaation merkitystä. Ongelmista täytyy keskustella rakentavasti ja ihmistä arvostavasti. Kun ongelmista keskustellaan, mietitään samalla toimintatapaa siten että motivoidaan työntekijöitä etsimään itsenäisesti ratkaisuja. Tällöin työntekijöiden motivaatio säilyy ja heidän kapasiteetti saadaan täysimääräisesti käyttöön. Tuon työssäni esille keinoa joissa tiimin työhyvinvointia ja tiimin yhteistyön toimivuutta voidaan mitata sekä päästä kiinni ongelmiin ja suorittaa korjaavia toimenpiteitä.

Vaativaa kehitystyötä tekevien työntekijöiden kannalta Scrum ja Lean tuovat parannuksia työn tekemisen menetelmiin. Kummankaan mallin tarkoituksena ei ole asettaa lisää paineita tiimi- tai yksilötasolla vaan luoda sellaiset raamit jotka mahdollistavat tavoitteellisen ja mielekkään työnteon. Keskeistä on työntekijän kunnioittaminen ja arvostaminen. Scrum luo raamit siihen miten työtä tehdään ja pyrkii vähentämään ylimääräistä hälyä tiimin ympäriltä jotta se voi keksittyä täysipainoisesti kehitystehtäväänsä. Lean puolestaan poistaa hukkaa, odottelua ja turhauttavaa byrokratiaa.

Mielestäni Lean on oikein käytettynä erittäin tehokas tapa kehittää yrityksen toimintaa ja laatua. Oleellista on arvoketjun kehittäminen ja asiakkaan tarpeisiin keskittyminen. Ei ole

ihme että menestyvät yritykset hyödyntävät Lean-ajattelua. Varoituksen sana on kuitenkin kaiken hehkutuksen jälkeen paikallaan. Yrityksen prosessien täytyy olla riittävän kypsiä jotta arvovirtaketjun optimoinnista saadaan hyöty irti. Toinen varottava asia on alioptimointi, huomion täytyy keskittyä koko arvoketjun kehittämiseen.

Yksi osuus työssä oli organisaation KPI-mittareihin ja toiminnan mittaamiseen liittyviin teorioihin perehtyminen. Työni KPI-mittaamiseen liittyvä osuus osoittautui haastavaksi koska mittaamiseen liittyy monenlaisia asioita. Lean-ajattelussa ja Scrumissa on omat vakiintuneet tavat mitata suorituskykyä ja laatua. Mittaamiseen voidaan liittää myös erilaisia taloudellisia malleja, joita en kuitenkaan lähtenyt työssäni avaamaan.

Toimeksiantajan prosessit ovat vakiintuneita ja niitä kehitetään jatkuvasti. KPI-mittarit ovat mielestäni tällä hetkellä hyvällä tolalla mutta toimintaa voidaan silti kehittää eteenpäin. Yksi kehityskohde on kehitysprosessin nopeuttaminen ja ketteryyden lisääminen. Parhaita keinoja tähän on Lean-ajattelun mukaisesti hukan poistaminen. Erityisesti sprintin alussa tapahtuva ennakointi vähentää odottelua, esimerkiksi muutoksenhallintatilauksen tekeminen hyvissä ajoin.

Leaniin ja Scrumiin on vahvasti sisäänrakennettuna laatuajattelu ja jatkuvan parantamisen kulttuuri. Viat ja laatupoikkeamat nähdään mahdollisuutena kehittää ja parantaa prosessia. Huono laatu aiheuttaa kustannuksia. Nämä kustannukset voivat olla esimerkiksi epäselvyyksiä, sähläystä ja asioiden uudelleen tekemistä. Tärkeä havainto että laadun tekeminen ei maksa mitään, se päinvastoin säästää kustannuksia kun asiat tehdään kerralla oikein ja asiakkaiden kokema laatu paranee ja tuotteet tuottavat asiakkaille lisäarvoa.

Työn lopputuloksena esitän erilaisia toimenpiteitä ketterän ohjelmistokehitysprosessin kehittämiseksi. Uskon että Lean-ajatteluun pohjautuvilla toimenpiteillä on mahdollista kehittää kehitystiimien toimintaa edelleen. Lisäksi esitän uusien Lean-ajatteluun ja Scrumiin pohjautuvien mittareiden ja visualisointitapojen käyttöönottamista. Vaikka mittaaminen on hyvällä tasolla, voidaan edelleen kehittää tapaa jolla mittaamisen tulokset tehdään näkyviksi ja esitetään tiimeille.

Ehdotan työssäni Lean-ajattelun jalkauttamista ja yleensäkin Leanin hyödyntämistä paremmin. Käytännössä tämä tarkoittaa toiminnan solakoittamista edelleen ja arvovirtakuvaksen hyödyntämistä prosessikehityksessä. Läpimenoaika on selkeä toiminnan tehokkuutta kuvaava mittari. Arvovirtakuvaus on jatkuvan parantamisen periaatteeseen pohjautuva työkalu, joka säännöllisesti hyödynnettynä on erittäin tehokas menetelmä hukan poistamiseen. Ehdotan läpimenoaikojen jatkuvaa seurantaa ja arvovirtakuvauksen piirtämistä puolen vuoden välein. Kehittämistoimet kohdistetaan niihin vaiheisiin joissa eniten tulee hukkaa.

Luonteva jatkoprojekti päättötyölleni on ehdotettujen toimenpiteiden ja mittareiden jalkauttaminen käytäntöön.

Insinöörityön tekeminen ajoittui samaan ajankohtaan kun perehdyin uusiin tehtäviin. Tutkimustyö oli siis osa uusiin tehtäviin perehtymistä ja uusien asioiden opettelua. Työn tärkeintä antia on uusien asioiden oppiminen ja mielestäni työssä toteutui opiskelu ja työelämän yhteensovittaminen parhaalla mahdollisella tavalla.

LÄHTEET

Agilepearls: Principles of Lean Software Development for Agile Methodology. Www-julkaisu. Saatavissa: https://agilepearls.wordpress.com/tag/customer-satisfaction-survey/

Company internal material. 2014.

Deemer P. The distributed Scrum primer, version 1.0. Www-dokumentti. Saatavissa: http://www.goodagile.com/distributedScrumprimer/DistributedScrumPrimer.pdf

Gustafsson J. 2011. Model of Agile Software Measurement: A Case Study. Master of Science Thesis in the Programme Software engineering and Technology. Chalmers University of Technology. University of Gothenburg. Www-julkaisu. Ladattavissa: http://publications.lib.chalmers.se/records/fulltext/143815.pdf

Hartmann D. & Dymond R. 2006. Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value. Www-julkaisu. Ladattavissa: http://www.itu.dk/~kenn/sasu/hand-in/articles/IEEE/Appropriate%20Agile%20Measurement%20-%20Using%20Metrics%20and%20Diagnostics%20to%20Deliver%20Business%20Value%20%2801667571%29.pdf

Hayes W. & Miller S. & Lapham M. A. & Wrubel E & Chick T. 2014, Agile Metrics: Progress Monitoring of Agile Contractors, January 2014, Carnegie Mellon University. Www-julkaisu. Ladattavissa: http://resources.sei.cmu.edu/asset_files/TechnicalNote/2014_004_001_77799.pdf

Jeffries R. & Kern J. & Marick B. & Martin R. C. & Mellor S. & Schwaber K. & Sutherland J. & Thomas D. Ketterän ohjelmistokehityksen julistus. Www-dokumentti. saatavissa: http://www.agilemanifesto.org/iso/fi/

Ketterä ohjelmistokehitys. 2015. Wikipedia. Www-julkaisu. Ladattavissa: http://fi.wikipedia.org/wiki/Ketterä_ohjelmistokehitys

Kouri I. 2010. Lean-taskukirja. Helsinki. Teknologiainfo Teknova.

Krebs, J. 2009. Agile Portfolio Management. Redmond, Microsoft Press.

Larman C. & Vodde B. 2013. Scaling agile development, Large and multisite Product with large-Scale Scrum. Www-dokumentti. Saatavissa: http://www.crosstalkonline.org/storage/issue-archives/2013/201305/201305-Larman.pdf

Leffingwell D. 2007. Scaling software agility. Best practices for Large Enterprises. Pearson education inc.

Lindstöm J. 2011. Scrum improves transparency and maneuverability. Www-julkaisu. Saatavissa http://reaktor.com/blog/Scrum-improves-transparency-and-maneuverability/

Mazzanti G. 2010. Agile KPIs. Www-julkaisu. Saatavissa: http://www.slideshare.net/mgaewsj/agile-kpis-5853270?related=1

Poppendieck M. & Poppendieck T. 2008. Implementing Lean Software development: From Concept to Cash. 6. painos. Addison-Wesley.

Schwber K. & Sutherland J. 2013 The Scrum Guide. Www-julkaisu. Saatavissa https://Scrumwell.files.wordpress.com/2014/03/Scrum-guide-2013-fi-v1-1.pdf

Scrum. 2015. Wikipedia. Www-julkaisu. Saatavissa: http://fi.wikipedia.org/wiki/Scrum

Tabaka, J. 2006. Collaboration explained : facilitation skills for software project leaders. Upper Saddle River, NJ : Addison-Wesley

Vaughn S. 2013. How Agile Led to the Creation of the Technical Product Owner. Www-dokumentti. Saatavissa: http://www.techwell.com/2013/09/how-agile-led-creation-technical-product-owner

Yeret Y. 2010. Explaining Cumulative Flow Diagrams – CFD. Www-julkaisu. Ladattavissa: http://www.slideshare.net/yyeret/explaining-cumulative-flow-diagrams-cfd