Tampereen yliopisto

Elias Örmä

HUKKIEN POISTO STARTUP-YRITYKSEN OHJELMISTOKEHITYSPROSESSISSA

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintutkielma
Lokakuu 2020

TIIVISTELMÄ

Elias Örmä: Hukkien poisto startup-yrityksen ohjelmistokehitysprosessissa
Kandidaatintutkielma
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Lokakuu 2020

Lean-ohjelmistokehitys on lean-tuotekehityksestä johdettu ohjelmistokehitysmetodologia. Yksi keskeisimmistä piirteistä lean-ohjelmistokehityksessä on hukkien identifioiminen ja eliminointi ohjelmistokehitysprosessista. Lean ohjelmistokehitykseen kuuluva hukkien identifiointi ja eliminointi esitellään kokonaisvaltaisesti. Jotta lukija voi ymmärtää mitä lean ohjelmistokehityksellä tarkoitetaan, työssä esitellään mitä lean tarkoittaa, mitä lean-tuotekehitys tarkoittaa ja mitä lean-ohjelmistokehitys tarkoittaa.

Startup-yritys on alkuvaiheessa oleva yritys, jolla ei ole aikaisempaa toimintahistoriaa, joka operoi pienillä resursseilla palvellen pientä niche markkinaa. Työssä esitellään mitä Startup-yritys tarkoittaa, sekä mitä ominaispiirteitä startup-yritykset sisältävät. Lisäksi työssä esitellään tarkemmin ohjelmistoalan startup-yritys, sekä siihen kuuluvat ominaispiirteet.

Edellä mainittujen asioiden esittelemisen jälkeen päästään työn oikeaan tarkoitukseen. Tässä kandidaatintutkielmassa analysoidaan lean-ohjelmistokehityksen hukkien identifiointi ja eliminointi mallin käytettävyyttä, sekä sopivuutta ohjelmistoalan startup-yrityksen ohjelmistokehitysprosessissa.

Avainsanat: Lean-ohjelmistokehitys, startup, ohjelmistokehitys.

SISÄLLYSLUETTELO

.

1. JOHDANTO

Nykypäivänä kilpailu ohjelmistotuotannossa kasvaa nopeaa vauhtia. Ohjelmistokehityksen tueksi on kehitetty erilaisia ohjelmistokehityksessä käytettäviä metodologioita. Eri kokoiset yritykset käyttävät erilaisia ohjelmistokehitysmetodologioita organisaation koon, sekä ohjelmistokehitystä toteuttavan ryhmän koon perusteella.

Tämän työn tarkoituksena on esitellä lean-ohjelmistokehitysmetodologia, sekä arvioida lean-ohjelmistokehitysmetodologian hukkien eliminoimisen soveltuvuutta sellaisenaan ohjelmistoalan startup-yritykseen. Lean-ohjelmistokehityksen yksi pääperiaatteista on poistaa hukat ohjelmistokehitysprosessista. Työssä esitellään startup-yrityksen ominaisuudet ja piirteet, minkä pohjalta voidaan arvioida hukkien eliminoimisen lean-ohjelmistokehityksen periaatteita noudattaen sopivuutta ohjelmistoalan startup-yrityksen ohjelmistokehitysprosessiin.

Työn alussa esitellään mitä tarkoittaa lean-tuotanto, lean-tuotekehitys, sekä lean-ohjelmistokehitys. Tämän jälkeen esitellään lean-ohjelmistokehityksessä esiintyvät hukat. Kolmannessa kappaleessa esitellään startup-yritys, sekä ohjelmistoalan startup-yritys käsitteenä. Neljännessä kappaleessa käsitellään hukkienpoisto metodologien soveltamista ohjelmistoalan startup-yritykseen. Viides kappale sisältää yhteenvedon hukkienpoisto metodologien soveltamisesta ohjelmistoalan startup-yritykseen.

2. LEAN JA LEAN-OHJELMISTOKEHITYS

Tässä luvussa esitellään lean-tuotanto, lean-tuotekehitys sekä lean-ohjelmistokehitys. Taustatieto leanista on tärkeää, jotta voidaan ymmärtää, mitä lean-ohjelmistokehityksellä tarkoitetaan, koska lean-ohjelmistokehitys periytyy Lean-tuotekehityksestä ja lean-tuotekehitys periytyy lean-tuotannosta.

2.1 Lean ja Lean-ajattelu

Lean on lähtöisin Toyotan käyttämästä, Tachii Ohnon 1940–1960-luvuilla kehittämästä TPS (Toyota Production System) -tuotantojärjestelmästä [1]. Lean tarkoittaa ajattelutapaa siitä, kuinka toimittaa arvoa asiakkaalle nopeammin eliminoimalla hukkia tuotantoprosessista (Hibbs et al. 2009).

TPS-tuotantojärjestelmä perustuu kahteen perusasiaan: JIT-virtaus ja Jidoka [1]. JIT tarkoittaa, että jokainen prosessi tuottaa ainoastaan sen, mitä tarvitaan seuraavaan prosessiin jatkuvassa virtauksessa [3]. JIT tarkoittaa oikean määrän materiaaleja olevan oikeassa paikassa oikeaan aikaan [2]. JIT-periaatteen päätavoite on eliminoida prosessin sisäisen inventaarion määrää [1]. Jidoka tarkoittaa koneita, jotka toimivat automaattisesti, mutta kykenevät pysähtymään, kun ne huomaavat virheen järjestelmässä [2]. Ihmisiä tarvitaan ainoastaan, kun järjestelmässä tapahtuu virhe ja järjestelmä pysähtyy. TPS-tuotantojärjestelmän kehittäjä Tachii Ohno kutsuikin Jidokaa ihmisavusteiseksi automaatioksi. [1]

TPS-järjestelmän osakehittäjän Shigeo Shignon mukaan tuotannossa esiintyy seitsemän eri hukkaa, joihin kuuluvat viat, ylituotanto, kuljetus, odotus, varasto, liike ja liika jalostus [2]. Vikojen eliminoimiseksi lean yrittää ennaltaehkäistä vikoja ennen kuin ne tapahtuvat. Ylituotanto tarkoittaa tuotantoa, joka on tehty ennen kuin tuotannossa tuotettua tuotetta tarvitaan. Kuljetuksella tarkoitetaan ylimääräistä tavaran liikettä, joka ei tuo lisäarvoa tuotteelle. Odotuksella tarkoitetaan ihmisiä, jotka odottavat tuotannon seuraavaa vaihetta. Varastolla tarkoitetaan kaikkea materiaalia, keskeneräisiä tuotteita ja valmiita tuotteita, joita ei työstetä. Liikkeellä tarkoitetaan ihmisten ja laitteiden liikkumista enemmän kuin on tarpeellista niille tarkoitetun prosessin suorittamiseksi. Liikajalostuksella tarkoitetaan tuotteen jalostamista enemmän kuin asiakas vaatii. [2]

1990-luvulla TPS sai nimekseen lean-tuotanto, minkä jälkeen lean-ajattelua on alettu soveltamaan muun muassa tilausten käsittelyyn, vähittäismyyntiin ja lentokoneiden huoltoon. Lean-periaatetta on laajennettu toimitusketjuihin, tuotekehitykseen ja ohjelmistokehitykseen. [1]

Leanin toteuttamiselle on tärkeää kartoittaa prosessin eri vaiheet ja kategorisoida ne arvoa luoviin ja arvoa luomattomiin vaiheisiin. Tavoitteena on eliminoida hukka sekä kehittää arvoa lisääviä vaiheita. Hukkaa on kahdenlaista: kokonaista hukkaa sekä pakollista hukkaa. [2] Kokonainen hukka tarkoittaa jokaista tuotannossa tapahtuvaa prosessia, joka ei tuota arvoa asiakkaalle, eikä ole pakollinen tuotteen tuottamiseksi. Pakollinen hukka tarkoittaa prosesseja, jotka eivät luo arvoa asiakkaalle, mutta ovat pakollisia.

Nykypäivän lean-tuotanto perustuu viiteen pääperiaatteeseen: arvo, arvovirta, virtaus, veto ja täydellisyys. Arvo on asiakkaiden luoma käsite, johon on pyrittävä. Arvovirta tarkoittaa kartoitusta tuotteen tuottamisen vaiheista. Jokainen vaihe on kategorisoitu arvoa tuottavaksi, arvoa tuottamattomaksi, mutta pakolliseksi tai hukaksi. Virtaus tarkoittaa prosessin jatkuvaa toimintaa. Prosessin pysähtyessä syntyy hukkaa. Veto tarkoittaa tuotteen tekemisen aloittamista vasta sitten, kun asiakas on tilannut tuotteen. Täydellisyydellä tarkoitetaan jatkuvaa täydellisyyteen pyrkimistä löytämällä hukkia ja poistamalla niitä. [2]

Lean-tuotekehitys koostuu samoista ydinajatuksista kuin TPS (Poppendieck & Poppendieck 2006). Lean-kehitys on tuotekehitysmalli, joka keskittyy asiakasarvon luomiseen, hukan eliminoimiseen, arvovirtojen optimointiin, ihmisten vaikutusmahdollisuuksien parantamiseen ja jatkuvaan kehitykseen [2]. Tuotekehitys on tiedonluontiprosessi [1].

2.2 Lean-Ohjelmistokehitys

Ohjelmistokehitys ja tuotanto eroavat toisistaan hyvin paljon. Tuotanto koostuu materiaali-, sekä informaatiovirroista. Ohjelmistokehitys periytyykin lean-tuotekehityksestä.

Lean-ohjelmistokehitys on yksi muoto lean-tuotekehityksestä. Lean-ohjelmistokehityksen ymmärtämiseksi on ymmärrettävä lean-tuotekehityksen perusperiaatteet. [1] Lean-ohjelmistokehitykseen kuuluu seitsemän eri pää osa-aluetta:

1. Hukan eliminointi
2. Oppimisen vahvistaminen
3. Päätä niin myöhään kuin mahdollista
4. Toimita niin nopeasti kuin mahdollista
5. Valtuuta tiimi
6. Rakenna eheys sisään
7. Optimoi kokonaisuus

[1]

Lean-ohjelmistokehityksessä hukan eliminointi tarkoittaa samaa kuin lean-tuotannossa, mutta aikajana voidaan käsittää eri tavalla. Ajan mittaus alkaa, kun tilaus asiakas tarpeesta saadaan. Ajan mittaus loppuu, kun asiakkaan tarpeen täyttävä ohjelmisto toimitetaan. Lean-ohjelmistokehitys keskittyy aikajanan lyhentämistä poistamalla kaiken ei arvoa lisäävän hukan. [1]

Hukan poistamiseksi, se pitää ensiksi tunnistaa. Ensimmäinen askel hukan poistamiseksi on ymmärtää mitä arvo tarkoittaa. Hukka tarkoittaa kaikkea, mikä häiritsee arvon toimittamista asiakkaalle silloin kun asiakas saa tuotteesta eniten arvoa. [1]

Hukan eliminointi on perusteellisin lean-periaate. Kaikki muut periaatteet seuraavat tätä periaatetta. Shigeo Shingno, eräs TPS-järjestelmän kehittäjistä laati seitsemän eri tyyppistä tuotantohukkaa. Taulukko 1 kääntää kyseiset hukat ohjelmointituotannon ongelmiksi:

Taulukko 1. lean-tuotannon ja lean-ohjelmistokehityksen hukat [4]

Lean-tuotanto	**Lean-ohjelmistokehitys**
Varasto	Osittain tehty työ
Ylituotanto	Ylimääräiset ominaisuudet
Ylijalostus	Uudelleen oppiminen
Kuljetus	Tehtävien luovutus
Odotus	Viivästymiset
Liike	Tehtävien vaihtelu
Viat	Viat

1. Osittain tehty työ

Osittain tehdyllä työllä ohjelmistokehityksessä on tapana vanheta. Se tulee muiden kehityskohteiden tielle, joiden pitäisi tulla tehdyksi. Suurin ongelma osittain tehdyssä ohjelmistossa on se, että ei voida tietää, tuleeko ohjelma toimimaan vai ei. Tehdyn koodin laadusta ja ominaisuuksista riippumatta mahdollisia ongelmakohtia ei voida tietää ennen kuin kyseinen ohjelmakoodi integroidaan osaksi isompaa kokonaisuutta. [4]

Osittain tehty ohjelmisto sitoo pääomaa sijoituksiin, jotka eivät ole tuottaneet vielä tulosta. Kyseinen osittain tehty ohjelmisto ei välttämättä ikinä päädy tuotantoon. Tämä johtaa riskiin siitä, että toteutettuun keskeneräiseen ohjelmistoon tehty sijoitus on turha. Minimoimalla keskeneräisen ohjelmistotuotannon minimoidaan riskejä, sekä poistetaan hukkaa. [4]

Osittain tehdyt työt voidaan kategorisoida viiteen eri osa-alueeseen: Toteuttamaton dokumentoitu ominaisuus, synkronoimaton koodi, testaamaton koodi, dokumentoimaton koodi ja käyttämätön koodi. [4]

Toteuttamattomalla koodilla tarkoitetaan koodia, josta on kirjoitettu suunnitteludokumentti, mutta sitä ei ole toteutettu. Mitä pidemmän aikaa jokin dokumentoitu ominaisuus pysyy toteuttamattomana, sitä suuremmalla

todennäköisyydellä kyseistä suunnittelu dokumenttia pitää muuttaa asiakastarpeiden takia. [4]

Synkronoimattomalla koodilla tarkoitetaan koodia, joka muodostuu, kun ohjelmoijat luovat eri versionhallinta haaroja koodatessaan. Tämä luo koodille yhdistämistarpeen. [4]

Testaamaton koodi tarkoittaa koodia, joka on kirjoitettu ilman testien luontia sen ympärille [4]. Todennäköisyys vikojen ilmaantumiselle on suuri, kun koodia ei testata.

Dokumentoimaton koodi tarkoittaa koodia, jolle ei ole saatavilla tarpeellista dokumenttia. Ideaalisesti koodin pitäisi olla itsedokumentoitavaa [4]. Tämä tarkoittaa sitä, että koodia pystyy lukea, sekä ymmärtämään ilman erillistä dokumentti tiedostoa.

Käyttämättömällä koodilla tarkoitetaan koodia, jota ei ole liitetty oikeaan tuotteeseen. Asiakkaan on usein helpompaa omaksua pieninä kokonaisuuksin, koska ominaisuuksien käyttöönotto vaatii tällöin vähemmän opettelua. [4]

2. Ylimääräiset ominaisuudet

Jokainen bitti koodia lisää koodin monimutkaisuutta ja on potentiaalinen virhe kohta. Jos tiettyä koodia ei tarvita heti, sen lisääminen järjestelmää on hukkaa. [4]

Ohjelmistoprojektien on vältettävä sellaisten ominaisuuksien lisäämistä, mikä ei luo arvoa asiakkaalle. Tämän pystyy toteuttamaan käyttämällä maalaisjärkeä, keskittymällä asiakkaiden tarpeeseen, asettumalla uusien ominaisuuksien lisäämistä vastaan päätöstilanteissa, sekä rakentamalla ohjelmisto niin, että ominaisuuksien lisääminen helposti on mahdollista mahdollisimman pitkän aikaa projektin aikana. [4]

3. Uudelleen oppiminen

Uudelleen oppimisella tarkoitetaan jo olemassa olevan tiedon uudelleen hankkimista [4]. Tämä tarkoittaa sitä, että aikaisemmin opittua tietoa jostakin prosessista ei hyödynnetä päätöksenteossa. Suurin osa tiedosta sisältyy subjektiivisiin oivalluksiin, intuitioihin, aavistuksiin ja ajatusmalleihin [5]. Tieto ei ole siis ainoastaan kirjoitettua

dokumentaatiota, vaan ihmiset ovat tärkeä osa sitä. Tietoa voidaan tuhlata siis myös olemalla ottamatta ihmiset, jotka omaavat hyödyllistä tietoa, mukaan kehitysprosessiin [1].

4. Tehtävien luovutus

Tehtävän luovuttaminen toiselle ohjelmoijalle on aikaa vievä prosessi, koska seuraava ohjelmoija, joka saa tehtävän itselleen, joutuu käyttämään huomattavan ajan perehtyäkseen jo osittain tehtyyn tehtävään. Tietoa, jota tehtävän luovuttaja on kartuttanut itselleen, kutsutaan hiljaiseksi tiedoksi. Vain 50 prosenttia tästä tiedosta, siirtyy eteenpäin seuraavalle tehtävän suorittajalle. Jokainen tehtävän luovuttaminen siis vähentää hiljaista tietoa luovutettavasta tehtävästä. Tehtävien luovutusten määrä on siis pyrittävä minimoimaan. [4]

5. Viivästymiset

Yksi suurimmista hukista ohjelmistokehityksessä on jonkin asian odottaminen. Projektin aloituksen viivästyminen, projektin henkilöiden asettamisesta johtuva viivästyminen, viivästykset liian suurien vaatimusten dokumentointiin, tarkistusten ja hyväksyntöjen viivästyksiin, testauksen viivästyksiin ja käyttöönoton viivästyksiin ovat hukkaa.

[4]

Viivästyminen estää asiakasta tunnistamasta arvoa mahdollisimman nopeasti. Perus lean-periaatteisiin kuuluu päätöksien viivyttäminen viimeiseen mahdolliseen hetkeen, jotta voit tehdä tietoisimman päätöksen. [4]

6. Tehtävien vaihtelu

Ihmisten asettaminen moniin projekteihin on hukan lähde. Joka kerta kun ohjelmistokehittäjät vaihtavat tehtävien välillä, aiheutuu huomattava vaihtoaika, kun he saavat heidän ajatuksensa kasattua ja saavat otetta uudesta tehtävästä. Tämä kyseinen vaihtoaika on hukkaa. [4] Kyseinen vaihtoaika vaikuttaakin molempiin tehtäviin, jota ohjelmoija yrittää suorittaa. Asiakasprojektissa tietyn ominaisuuden toteuttaminen alkaa tuottamaan arvoa asiakkaan silmissä vasta sitten, kun ominaisuus on valmis. Tällöin sitä voidaan esitellä asiakkaalle.

7. Viat

Jokaiselle riville koodia pitäisi olla olemassa joukko testejä, jotka todistavat testatun koodin eheyden. Tämän voi kuitenkin toteuttaa vain sellaisille ohjelmiston osille, jonka käyttäytyminen on ennalta tiedossa. Jos vikoja löytyy ohjelman rakennus vaiheessa, se tulisi korjata niin, että kyseinen vika ei voisi toistua enää ikinä uudestaan [1].

Pitkään olemassa olleet viat, jota ei ole löydetty ovat isoja hukkia. Vikojen vaikutusten vähentämiseksi on ne löydettävä silloin kun ne syntyvät. [4]

3. OHJELMISTOALAN STARTUP-YRITYKSET

Tässä kappaleessa esitellään startup-yritys käsitteenä. Startup-yrityksestä ei löydy vain yhtä määritelmää, vaan eri kirjallisuudesta löytyy useita eri määritelmiä. Lisäksi esitellään ohjelmistoalan startup-yritys, koska pelkkä startup-yritys on käsitteenä liian laaja analysoitavaksi lean-ohjelmistokehitys metodologian kanssa. Ohjelmistoalan startup-yritys esitelläänkin niin, että sen piirteitä pystyy analysoimaan lean-ohjelmistokehitys metodologian kanssa.

3.1 Startup-yritys

Startup-yritys on varsin nuori yritys ilman pidempää historiaa. Se, onko tietty yritys startup-yritys, määrittyy kuitenkin yrityksen eri piirteistä. Startup-yrityksien keskeinen saman kaltaisuus löytyy startup-yritysten kyvystä kasvaa. Startup-yrityksen on suunniteltu skaalautuvan todella nopeasti. [6] ”Startup-yritys on yritys joka toimii ratkaistakseen ongelman, jonka ratkaisu ei ole yksinkertainen ja menestys ei ole taattu” – Neil Blumenthal [6] Kuten edellisestä kahdesta määritelmästä startup-yritykselle voidaan nähdä, ei startup-yritystä voi käsittää vain yksiselitteisellä tavalla.

Startupeille on yleistä neljä seuraavaa ominaisuutta.

1. Startupit ovat uusia, epäkypsiä ja epäkokeneita. Tämä tarkoittaa, että startupeilla on todella vähän yhteistä historiaa ja karttunutta kokemusta.
2. Startupeilla on rajoitetut resurssit. Ensimmäiset startuppiin sijoitetut resurssit kohdistetaan yleensä tuotteen julkaisemiselle, tuotteen markkinoinnille, ja strategisten suhteiden luomiselle.
3. Startupeilla on aluksi useita vaikuttajia. Startuppiin voi vaikuttaa sijoittajat, asiakkaat, kumppanit ja kilpailijat.
4. Startupit kohtaavat dynaamisia teknologioita ja markkinoita. Uusia IT yrityksiä perustetaan usein kehittämään teknologisesti innovatiivisia tuotteita. Tämä saattaa edellyttää huippuluokan kehitystyökaluja ja tekniikkoja.

[7]

3.2 Ohjelmistoalan startup-yritys

Ohjelmisto startupit ovat vasta perustettuja yrityksiä, joilla ei ole toimintahistoriaa ja jotka tuottavat nopeasti huipputeknologiaa. Nämä yritykset kehittävät ohjelmistoja erittäin epävarmoissa olosuhteissa ja toimivat nopeasti kasvavilla markkinoilla vakavan resurssipuutteen vuoksi. [8]

Isot ja pienet ohjelmistoalan yritykset kohtaavat samanlaisia ongelmia. Heidän pitää hallita ja kehittää ohjelmisto prosesseja, pysyä mukana nopeassa teknologian kehityksessä, operoida globaalissa ohjelmistoympäristössä. Kuitenkin yleensä, ne tarvitsevat erilaisia tapoja lähestyä asiaa tiettyjen liiketoimintamallien ja tavoitteiden takia, niche marketin takia, koon takia, rahoituksen ja työvoiman saatavuuden takia, prosessien ja hallinnoinnin takia, mm. [9]

Pienet yritykset eivät ole ainoastaan pienennettyjä versioita isoista yrityksistä. Yleisesti ne ovat erittäin responsiivisia ja joustavia, koska nämä ovat pienten yritysten kilpailukyvyn luovia tekijöitä. Tyypillisesti pienissä yrityksissä on ei hierarkinen rakenne, orgaaninen ja vapaasti virtaava hallinto mikä edesauttaa yrittäjyyttä ja innovoimista. Ne yleensä keskittyvät niche markkinaan, jonka isot yritykset ovat jättäneet huomioimatta. Ne esimerkiksi rakentavat komponentteja eri yritysten tuotteille, kehittävät innovatiivisia tuotteita tai tarjoavat palveluita tai huoltoa tuotteille, joita he tai muut tuottavat. [9] Ohjelmistoalan startup-yritykset siis keskittyvät palvelemaan vain pientä niche markkinaa tietyillä ohjelmistoratkaisuilla. Nämä ratkaisut voivat liittyä joko kuluttajan tai yrityksen tarpeeseen. Ohjelmistoalan startup-yrityksessä on erittäin keskeistä, että juuri tiettyä niche markkinaa voidaan palvella siten, että asiakkaat ovat valmiina maksamaan tuotteesta.

Toisin kuin isoissa yrityksissä, pienillä yrityksillä ei ole tarpeeksi henkilökuntaa kehittääkseen funktionaalisia erikoisuuksia, jotka auttavat heitä suorittamaan monimutkaisia tehtäviä toissijaisesti heidän tuotteilleen. [9] Kuitenkin, ne saattavat verkostoitua muiden pienten yritysten kanssa. Tiukat pääomat myös rajoittavat monia pieniä yrityksiä, joten niillä ei ole aina varaa ostaa vaadittua osaamista. [9]

4. HUKKIEN IDENTIFIOINTI JA ELIMINOINTI OHJELMISTOALAN STARTUP-YRITYKSESSÄ

Tässä kappaleessa käsittelen lean-ohjelmistokehityksen hukkien identifiointi ja eliminointi periaatteiden soveltamista ohjelmistoalan startup-yrityksissä. Lean-ohjelmistokehityksen periaatteiden soveltamisen analysointi perustuu aikaisemmassa kappaleessa esiteltyjen Startup-yrityksen piirteiden sopivuuteen aikaisemmin esiteltyjen lean-ohjelmistokehitykseen kuuluvien hukkien eliminointi metodien soveltuvuuteen.

4.1 Osittain tehty työ

Ohjelmistoalan startup-yrityksen resurssit ovat rajalliset, olosuhteet ovat epävarmat, sekä kohdemarkkinat kasvavat nopeasti. Näiden asioiden takia, keskeneräinen ohjelmakoodi on yksi suurimmista hukista ohjelmistoalan startup-yrityksessä. Keskeneräinen ohjelmakoodi ei luo arvoa yhdellekään ohjelmistoalan startup-yrityksen sidosryhmistä.

Ohjelmistonrakentamisen suunnittelu tulisikin täten tehdä niin, että keskeneräistä ohjelmistokoodia ei ole määrällisesti samaan aikaan paljon. Ohjelmistoon toteutettavat ominaisuudet tulisi jakaa pieniin kokonaisuuksiin, priorisoida tärkeys järjestykseen, sekä toteuttaa yksi kerrallaan.

4.2 Ylimääräiset ominaisuudet

Ohjelmistoalan startup-yrityksen tulisi keskittyä luomaan ainoastaan sellaisia ominaisuuksia ohjelmaansa, joka tuottaa asiakkaalle arvoa. Jos ohjelmoijat alkavat rakentamaan ohjelmaan osia, jotka eivät luo suoraan arvoa asiakkaalle, luodaan puhdasta hukkaa. Kuitenkin hyötyä tulevaisuuden ohjelmistokehitystä ajatellen voi kartuttaa rakentamalla ohjelma niin, että siihen on helppo lisätä ominaisuuksia tulevaisuudessa.

4.3 Uudelleen oppiminen

Tiedon oppiminen on tärkeä osa ohjelmistokehitystä. Koska startup-yrityksen tiimi on pieni, tulisi työntekijöiden aikaisemmin opittua tietoa ottaa huomioon päätöksentekoprosesseissa. Vaikka ohjelmistoalan startup-yrityksellä ei ole toimintahistoriaa, se ei tarkoita, etteikö työntekijöillä olisi relevanttia historiaa, joka voisi auttaa päätöksen teon tukena. Työntekijät kuitenkin oppivat koko ajan asioita tehdessään työtä, tämä pitäisi ottaa huomioon päätöksenteko tilanteissa.

4.4 Tehtävien luovutus

Koska startup-yrityksen ohjelmistokehitystiimi on oletettavasti vain pieni joukko ihmisiä, tulisi ohjelmistokehityksen vastuualueiden oltava konkreettisia ja rajattuja. Tällainen toiminta takaa sen, että tietyt opit tietyltä ohjelmistoprojektin osa alueelta säilyy, koska ominaisuuksien toteuttaja ei vaihdu kesken ohjelmistokehitysprosessin.

4.5 Viivästymiset

Asioiden odottaminen on looginen hukka ohjelmistokehityksessä. Odottamisen minimointi voidaan toteuttaa yrityksen sisäisten asioiden suhteen tehokkaalla projekti suunnittelulla. Kuitenkin ohjelmistoalan startup-yrityksen kokemattomuus voi olla asia, joka vaikuttaa asioiden viivästymiseen, koska yritys ei ole toiminut aikaisemmin omana kokonaisuutenaan, eikä yksittäisillä työntekijöillä ole historiaa mihin pohjustaa eri suoritusaika-arvioita.

Sijoittajat, asiakkaat, kumppanit ja kilpailijat voivat luoda odottamisaikoja, joihin ei voida suoraan vaikuttaa yrityksen sisäisellä toiminnalla. Näihin odottamisaikoihin voidaan kuitenkin vaikuttaa epäsuorasti toteuttamalla lean-periaatetta, joka ehdottaa päätöksien viivyttämistä viimeiseen mahdolliseen hetkeen asti tietoisimman päätöksen tuottamiseksi.

4.6 Tehtävien vaihtelu

Monen tehtävän rinnakkain toteuttaminen on epätehokasta vaihtoaikojen takia. Tehtävät tulisikin priorisoida, sekä aikatauluttaa niin hyvin kuin mahdollista, jotta ohjelmistokehittäjien ei tarvitsisi vaihdella eri tehtävien välillä. Koska toteutetut ohjelmisto-ominaisuudet tuottavat arvoa sidosryhmien silmissä vasta silloin kun ne ovat valmiita, tulisi priorisointia korostaa tämänkin takia.

4.7 Viat

Ohjelmiston koodin testaaminen on yksi keskeisimpiä ohjelmointiperiaatteita lean-ohjelmistokehitysmetodologian kanssa. Tämä sopiikin startup-yrityksen ohjelmistokehitysprosessiin erittäin hyvin. Jos startup-yrityksen tuottamaa ohjelmistoa kehitetään jatkuvasti lisäämällä siihen vain testattua, eheää koodia, pienenee vikojen olemassaolon todennäköisyys huomattavasti.

5. YHTEENVETO

Työssä esitelty lean-ohjelmistokehityksen osa liittyen hukkien identifiointiin soveltuu sellaisiin ohjelmistokehitys prosesseihin hyvin, jotka ovat ennalta määrättyjä, sekä omaavat tietyn konkreettisen maalin alusta asti. Ohjelmistoalan startup-yrityksellä saattaa kuitenkin olla hyvin dynaaminen ympäristö, sekä jatkuvasti kehittyvä, sekä muuttuva tarve kehitettävän ohjelmiston ominaisuuksien suhteen. Tämä johtuu ohjelmistoalan startup-yritykseen liittyvien sidosryhmien tarpeista, sekä yrityksen sisäisestä päätöksen teosta.

Edellisen kappaleen analysointi viittasi suurimmalta osalta siihen suuntaan, että kullekin ohjelmoijalle on jaettava hyvin rajattua vastuuta heti ohjelmistokehityksen alusta asti. Tällä eri ominaisuuksien kehittämiseen liittyvät työprosessit voidaan asettaa tietyn ohjelmoijan vastuu alueelle. Tällä tavalla ohjelmistokehittäjän jo oppima tieto pysyy tietyn ominaisuuden kehitysprosessin tukena. Myös työnluovutuksen tarve katoaa tai minimoituu konkreettisen vastuualuejaon takia. Toteutettavat ominaisuudet, sekä ohjelmistokokonaisuudet pitää priorisoida, sekä aikatauluttaa niin hyvin kuin mahdollista, käyttämällä ohjelmistokehittäjien omaa tietämystä työtehostaan. Turhaksi todetut ohjelmistoon suunnitellut ominaisuudet on eliminoitava resurssi puutteen vuoksi. Ohjelmiston toteutus pitää jakaa pieniin kokonaisuuksiin keskeneräisen työn vähentämiseksi. Ohjelmoijien oppimaa tietoa pitäisi hyödyntää päätöksenteossa. Ohjelma pitäisi rakentaa niin, että se pystytään toteamaan eheäksi testauksen avulla, sekä siihen pitäisi pystyä lisäämään lisäominaisuuksia helposti tulevaisuudessa.

Pienikokoinen tiimi voi hyötyä konkreettisesti hyvinkin nopeasti soveltamalla lean-ohjelmistokehityksen hukkien identifiointiin ja eliminointiin liittyviä asioita käyttäen työssä toteutetun analyysin kaltaista analysoimista.

LÄHTEET

[1] M. Poppendieck and T. Poppendieck, “Implementing Lean Software Development: From Concept to Cash,” *The Addison-Wesley Signature Series*, 2006. [Online]. Available: https://learning.oreilly.com/library/view/implementing-lean-software/0321437381/. [Accessed: 01-Mar-2020].

[2] C. Hibbs, S. Jewett, and M. Sullivan, *The Art of Lean Software Development Printing History :*, 1st ed. Sebastopol: O’Reilly Media, 2009.

[3] “Toyota Production System | Vision & Philosophy | Company | Toyota Motor Corporation Official Global Website.” [Online]. Available: https://global.toyota/en/company/vision-and-philosophy/production-system/. [Accessed: 01-Mar-2020].

[4] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit (The Agile Software Development Series)*. Addison-Wesley Professional, 2003.

[5] I. Nonaka, *The knowledge-creating company : how Japanese companies create the dynamics of innovation*. 1995.

[6] “What Is A Startup?” [Online]. Available: https://www.forbes.com/sites/natalierobehmed/2013/12/16/what-is-a-startup/#61b45b374044. [Accessed: 22-May-2020].

[7] J. Stanley M. Sutton, “The role process in a software start-up.”

[8] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, P. Abrahamsson, and N. Paternoster, “In press: Software Development in Startup Companies: A Systematic Mapping Study Software Development in Startup Companies: A Systematic Mapping Study,” doi: 10.1016/j.infsof.2014.04.014.

[9] “(PDF) Guest Editors’ Introduction: Why are Small Software Organizations Different?” [Online]. Available: https://www.researchgate.net/publication/3249286_Guest_Editors’_Introduction_Why_are_Small_Software_Organizations_Different. [Accessed: 13-May-2020].