Akseli Arvaja

OHJELMISTOTUOTANNON MENETELMÄN VALINNAN VAIKUTUS TYÖMOTIVAATIOON

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Helmikuu 2020

TIIVISTELMÄ

Akseli Arvaja: Ohjelmistotuotannon menetelmän valinnan vaikutus työmotivaatioon (The selection of software engineering methodology and its effect on work motivation)
Kandidaatintyö
Tampereen yliopisto
Helmikuu 2020
Tietotekniikan kandidaatin tutkinto-ohjelma

Tutkimuksessani pyrin ottamaan selvää, miten eri ohjelmistotuotannon menetelmät vaikuttavat työmotivaatioon. Tutkimuksia motivaatiosta ohjelmistotuotannossa on tehty, mutta ohjelmistotuotannon menetelmien eroista työmotivaatiossa ei ole juurikaan tutkittu. Tutkimus toteutetaan kirjallisuuskatsauksena, joten tutkimus kerää tietoa eri lähteistä ja yrittää yhdistellä teorioita muodostaakseen vastaukset tutkimuskysymyksiin.

Tutkimus vastaa kysymykseen "Miten ohjelmistotuotannon menetelmän valinta vaikuttaa työmotivaatioon?". Tutkimus selvitti, että ketterän menetelmän valinta projektiin kasvattaa työntekijöiden työmotivaatiota.

Avainsanat: Ohjelmistotuotanto, työmotivaatio, DevOps, scrum, vesiputous

ALKUSANAT

Kandidaatintutkimus oli mielenkiintoinen ja monenlaisia vaiheita täynnä. Haluan kiittää kihlattuani projektin aikana saamastani tuesta, samoin kuin isääni, kanssaopiskelijoita, työyhteisöä tiedon ja ideoiden jakamisesta, sekä äitiäni oikolukemisesta.

SISÄLLYSLUETTELO

1. JOHDANTO

Ohjelmistoja on tuotettu 1900-luvun puolivälistä saakka, ja ohjelmistotuotannon alkujuuret juontavat NATOn konferenssiin vuoteen 1968, jolloin kiisteltiin, onko ohjelmistotuotanto edes oma tieteenalansa [1]. Sharp et al. kertovat tutkimuksessaan, että ohjelmistotuotanto käsitteenä on ollut käytössä jo yli puolivuosisataa, ja motivaation merkitystä ohjelmistotyössä on tutkittu paljon. Silti, vaikka työntekijän työmotivaatio vaikuttaa ohjelman laatuun ja on yksi ohjelmistotuotannon tärkeimmistä menestyksen tekijöistä, sivutetaan motivaation merkitys usein, sillä motivaatiota on vaikea mitata ja se on ”pehmeää tiedettä” [2].

Ohjelmistoja on tuotettu 1900-luvun puolivälistä alkaen. Ohjelmistotuotannon alkujuuret juontavat NATOn konferenssiin vuoteen 1968, jolloin kiisteltiin, onko ohjelmistotuotanto edes oma tieteenalansa [1]. Sharp et al. kirjoittavat tutkimuksessaan, että ohjelmistotuotanto käsitteenä on ollut käytössä jo yli puolivuosisataa, ja että motivaation merkitystä ohjelmistotyössä on tutkittu paljon. Silti, vaikka työntekijän työmotivaatio vaikuttaa ohjelman laatuun ja on yksi ohjelmistotuotannon tärkeimmistä menestyksen tekijöistä, sivutetaan sen merkitys usein. Motivaatiota on vaikea mitata ja sen tutkimista pidetään ”pehmeänä tieteenä”. [2]

Tutkimuksessani pyrin ottamaan selvää, miten eri ohjelmistotuotannon menetelmät vaikuttavat työmotivaatioon. Tutkimuksia motivaatiosta ohjelmistotuotannossa on tehty, mutta ohjelmistotuotannon menetelmien erojen vaikutusta työmotivaatioon ei ole juurikaan tutkittu. Tutkimus toteutetaan kirjallisuuskatsauksena, joten tutkimus kerää tietoa eri lähteistä ja yrittää yhdistellä teorioita muodostaakseen vastaukset tutkimuskysymyksiin.

Käsiteltävät ohjelmistotuotannon menetelmät ovat vesiputous, scrum, sekä DevOps. Vesiputous valittiin tarkasteluun, koska se on useiden lähteiden mukaan kaikista tunnetuin perinteinen menetelmä [3] [1]. Scrum on puolestaan ketteristä menetelmistä käytetyin [3] ja DevOps on ohjelmistotuotannossa tällä hetkellä yksi suosituimpia menetelmiä [4].

Tutkimuksen päätutkimuskysymys on seuraava:

- Miten ohjelmistotuotannon menetelmän valinta vaikuttaa työmotivaatioon?

Tutkimus pyrkii myös vastaamaan seuraaviin apututkimuskysymyksiin:

1. Mitä on ohjelmistotuotanto ja mitkä ovat tärkeimmät menetelmät?
2. Mitä on työmotivaatio ja mikä motivoi ohjelmistoprojekteissa?
3. Miten motivaatio näkyy ohjelmistoprojektissa?

Luku 3 aloittaa teorian. Siinä käydään läpi mitä ohjelmistotuotanto on, ja mitä erityispiirteitä valituilla menetelmillä on. Luvussa 4 paneudutaan työmotivaatioon ja sen ominaisuuksiin sekä selvitetään mikä tuottaa motivaatiota. Luku 5 yhdistää lukujen 3 ja 4 teorian, vastaten päätutkimuskysymykseen. Viimeinen luku vetää yhteen ja tiivistää pääsisällön.

2. TUTKIMUSMENETELMÄT

Tämä tutkimus on toteutettu kirjallisuuskatsauksena. Tietoa on etsitty ja kerätty monesta lähteestä, sekä jalostettu punnitsemalla lähteiden määrittelyjä eri teorioista. Tietojärjestelminä ja tietokantoina on käytetty TTY:n kirjaston tarjoamaa Andor-tiedonhakupalvelua, Safaribooksonline-verkkokirjastoa, sekä Google scholar -tiedonhakupalvelua. Tietoa on etsitty tutkimukseen liittyvillä hakusanoilla ja yhdistelemällä niitä. Taulukko 1 esittelee hakutuloksia eri tietokannoista.

Hakusana	Andor	Safarobooks
Software Engineering	4356997	24252
Work motivation	1430054	19566
”Work motivation” AND ”Software Engineering”	192721	7838
”Work motivation” AND ”Software Engineering methods”	168287	6693

Taulukko 1. Hakutulokset.

Taulukosta huomaa, että haun rajoituksia lisätessä hakutuloksia tippui hausta pois rajusti. Suurin osa lähteistä oli myös huonolaatuisia tai epäolennaisia. Hakutulokset rajasin tekstien julkaisuvuosien ja johdantojen perusteella. Varsinkin työmotivaatio sekä ohjelmistotuotannon menetelmillä haetut tekstit olivat usein erittäin epäolennaisia.

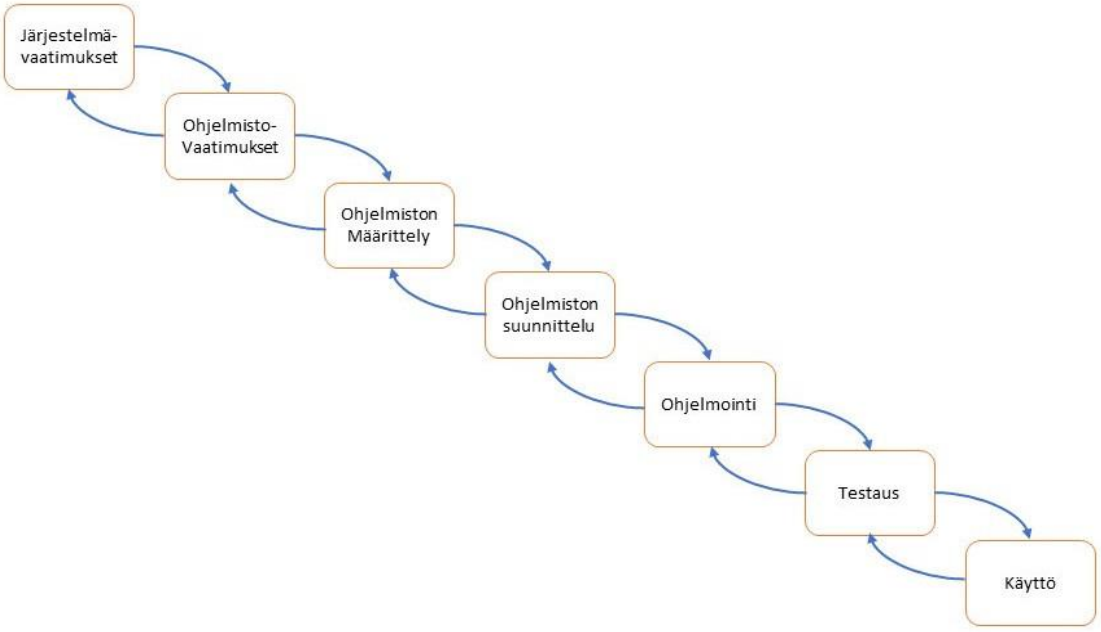
3. OHJELMISTOTUOTANNON MENETELMÄT

Tässä luvussa perehdytään ohjelmistotuotantoon, sen eri malleihin ja niiden ominaisuuksiin. Ohjelmistotuotannon tavoitteena on rakentaa toimiva ohjelma, johon asiakas on tyytyväinen. Vuodesta 1970 alkaen ohjelmaprojektit ovat käyttäneet erilaisia menetelmiä ohjelmien tuottamiseen. Menetelmistä tunnetuimpia ovat vesiputous-, evoluutio- sekä spiraalimalli [5]. vuoteen 2000 mennessä ketterät, varsinkin iteratiiviset, menetelmät olivat alkaneet yleistyä ja niistä tunnetuin oli Kent Beckin kehittämä eXtreme Programming [3] [1].

Perinteiset ohjelmistotuotannon menetelmät ovat olleet laajemmin käytössä vuodesta 1970 alkaen [5]. Näistä menetelmistä vanhin on vesiputousmalli [6]. Vesiputoukseen viitataan yleensä puhuttaessa perinteisistä tai klassisista menetelmistä [3]. Vesiputousmallia käsitellään perinteisenä menetelmänä tässä tutkimuksessa.

3.1 Vesiputousmalli

Vesiputousmallin on kehittänyt Royce vuonna 1970. Malli koostu seitsemästä toisiaan seuraavasta vaiheesta, jotka ovat järjestelmävaatimukset, ohjelmistovaatimukset, ohjelmiston määrittely, ohjelmiston suunnittelu, ohjelmointi, testaus sekä käyttö. Tärkeitä vesiputouksen ominaisuuksia ovat vaiheiden päällekkäinen suorittaminen, sekä eteen- että taaksepäin iterointi vaiheiden välillä. [3]. Vesiputousmalli on kuvattu kuvassa 1.



Kuva 1. *Vesiputousmalli [3].*

Vesiputousmalli on saanut paljon kritiikkiä. Yksi vesiputouksen eniten kritiikkiä saanut ominaisuus on sen hitaus. Ohjelmointi, testaus ja käyttöönotto ovat projektin loppuvaiheissa, ja näin toimiva versio ohjelmasta valmistuu usein vasta projektin loppuvaiheessa [6]. Myöhäinen käyttöönotto viivästyttää mahdollisten vikojen löytämistä [3]. Mikäli vika ohjelmasta löytyy myöhäisessä vaiheessa, esimerkiksi tuotanto- tai testausvaiheessa, viivästyttää se ohjelman valmistumista. Viivästys vaikuttaa kaikkiin vesiputouksen vaiheeseen, koska vaiheet seuraavat toisiaan lineaarisesti. [7]

Myöhäinen testausvaihe ja muutoksille jäykkä projektimalli hidastavat ohjelmasta vikojen löytämistä ja johtavat osien uudelleentekemiseen. Tämän vuoksi ohjelman laatu saattaa kärsiä. Vioistaan ja jäykkyyksistään huolimatta vesiputousmallia käytetään vieläkin paljon ohjelmistotuotannossa. Monet tutkijat ovat myös vakuuttuneita siitä, että vesiputousmallia käytetään vielä pitkään. [8]

3.2 Ketterät menetelmät

Ketterät menetelmät periytyvät kevyistä iteratiivisista menetelmistä. Näiden menetelmien kehittäjät kokoontuivat vuonna 2001 yhteen keskustelemaan käyttämistään kevyistä menetelmistä. Tämän kahden päivän kokouksen aikana luotiin ketteriä menetelmiä ohjaava Agile Manifesto [3]. Haikala ja Mikkonen ovat koonneet manifestista konkretisoituvat arvot taulukkoon.

Taulukko 2. Agile Manifeston arvot [3].

	Periaate
1	Asiakas on pidettävä tyytyväisenä toimittamalla tälle projektin alusta asti tasaisella tahdilla toimivia ohjelmistoversioita.
2	Vaatimuksien muuttuminen tulee hyväksyä projektin aikana, myös myöhäisissä kehitysvaiheissa.
3	Toimivien asiakasversioiden julkaisuväli voi vaihdella muutamasta viikosta muutamaan kuukauteen (mieluummin viikkoja kuin kuukausia).
4	Liiketoiminta- ja kehitysrooleissa olevien työntekijöiden tulee työskennellä yhdessä päivittäin koko projektin ajan.
5	Projektit tulee rakentaa motivoituneiden yksilöiden ympärille. Heille tulee antaa ympäristö ja tuki, jota he tarvitsevat työn tekemiseksi. Luota siihen, että he saavat työn tehtyä.
6	Kaikkein tehokkain kommunikointikeino kehitystiimin ja ulkomaailman välillä on, että itse kehitystiimissä on kasvokkain keskustelu.
7	Toimiva ohjelmisto on projektin edistymisen tärkein mittari.
8	Ketterät menetelmät edistävät tasaista kehitystahtia. Projektin parissa työskentelevien henkilöiden tulisi kyetä pitämään työtahti ja -kuorma mahdollisimman tasaisena, eikä ylitöitä kuulu tehdä.
9	Huomion jatkuva kiinnittäminen tekniseen erinomaisuuteen ja hyvään suunnitteluun tehostavat ketteryyttä.
10	Asiat pitäisi aina pyrkiä tekemään mahdollisimman yksinkertaisella tavalla: minimoidaan vähemmän tärkeät tehtävät.
11	Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät itseohjautuvasti organisoituneissa tiimeissä.
12	Tiimin tulee selvittää säännöllisin väliajoin, miten se voisi tulla aikaisempaa tehokkaammaksi ja hienosäätää käytettävää työtapaa ja prosessia sen mukaisesti.

Ketterä ohjelmistokehitys keskittyy toimittamaan aikaisessa vaiheessa toimivan ohjelman asiakkaalle. Aikainen ohjelmiston toimittaminen mahdollistaa asiakaspalautteen saamisen projektin alkuvaiheessa, nopeamman virheisiin reagoinnin sekä asiakkaan vaatimusten paremman täyttymisen. Asiakas on myös tyytyväisempi, kun pääsee näkemään tuotettaan aikaisemmin. [9]

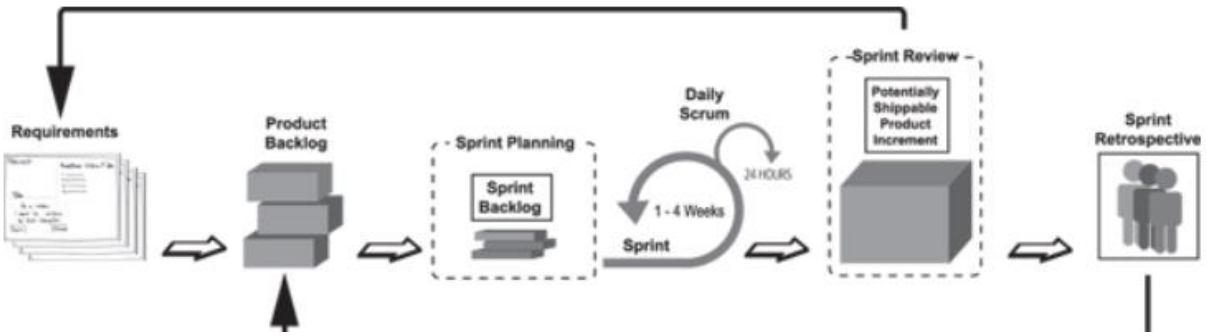
Ketteriä menetelmiä on paljon erilaisia, ja moni yritys käyttää muutettua versiota jostain tunnetusta menetelmästä. Yhteisiä piirteitä menetelmillä on kuitenkin paljon. Asiakkaan

läsnäolo projektissa, iteraatiot ja tuotantosyklit, sekä nopeat palautesyklit ovat kaikissa ketterissä menetelmissä näkyviä ominaisuuksia [7].

3.2.1 Scrum

Scrum on iteroiva ja tuotetta pala palalta valmistava projektinhallintamenetelmä. Se on sopiva menetelmä myös mihin tahansa muihin kuin ohjelmistoprojekteihin. Scrum perustuu iteraatioihin, joita scrum-maailmassa sanotaan pyrähdyksiksi. Yksi pyrähdys sisältää pyrähdyksen suunnittelupalaverin ja tavoitteiden asettamisen, päivittäiset scrum-palaverit, tuotteen kehittämisen mukaan lukien projektinomistajan säännöllisen katselmoinnin, pyrähdyksen katselmointikokouksen, sekä pyrähdyksen arviointipalaverin. [10]

Kuva 2 kuvastaa yhtä scrum-pyrähdystä. Pyrähdys alkaa pyrähdyksen suunnittelukokouksella, johon osallistuu koko scrum-tiimi. Suunnittelukokouksessa käydään läpi tuotteen työlista (product backlog) ja työlistan alkiot (backlog item) jotka otetaan sprinttiin mukaan tehtäviksi pilkottuina. Alkiot voivat olla esimerkiksi tuotteen



Kuva 1. Scrum-malli pyrähdyksineen [10]

ominaisuuksia, käyttötapauksia, käyttötarinoita, asiakasvaatimuksia, virheraportteja, dokumentaation kehittämistä tai arkkitehtuurin parantamista. [3]

Pyrähdyksen pidetään päivittäiset noin 15 minuutin kestävät scrum-palaverit, joissa käydään läpi, mitä kehitystiimi on tehnyt edellisen kokouksen jälkeen, mitä se aikoo tehdä seuraavaksi ja onko tiedossa esteitä, jotka hidastavat työtä. Pyrähdyksen päätyttyä pidetään sekä katselmointikokous että arviointipalaveri. Katselmointikokouksessa tuotantotiimi esittelee sidosryhmille pyrähdyksen aikaansaannoksia sekä kerää heiltä palautetta. Arviointipalaverissa arvioidaan

pyrähdyksen onnistumista sekä jokainen vastaa kysymyksiin, mikä meni pyrähdyksen aikana hyvin ja mitä pitää parantaa seuraavassa pyrähdyksessä. [3]

Scrumissa käytetään scrum-tiimiä. Tiimi koostuu tuotteen omistajasta, kehitystiimistä ja scrum-mestarista. Tuotteen omistajan tehtävä on hallinnoida projektin työlistan alkioita. Tuotteen omistajalla tulee myös olla liiketaloudellista näkökulmaa ja osaamista projektinhallintaan[10]. Hänen tehtäviinsä kuuluu

- osoittaa selkeästi työlistan alkiot
- järjestää työlistan alkiot tärkeysjärjestykseen siten, että se tukee projektin tavoitetta ja päämäärää parhaiten
- varmistua, että työlistan alkiot ovat kaikille tiedossa ja saatavilla
- osoittaa, mitä scrum-tiimi tekee seuraavaksi
- huolehtia, että kehitystiimi ymmärtää työlistan alkiot tarvitulla tasolla
  [11]

Kehitystiimin tehtävä on kehittää julkaistava tuote joka sprintin loppuun mennessä. Kehitystiimin kokoaa projektin tekevä organisaatio. Tiimin koon pitää olla tarpeeksi pieni, jotta tehtävien teko ja organisointi ovat ketterää, mutta tarpeeksi iso, jotta tarvittavat tehtävät saadaan tehtyä sprintin aikana. Kehitystiimin ominaisuuksia ovat seuraavat:

- Itseorganisointi: Kehitystiimillä ei ole ketään johtamassa tai kertomassa mitä tehdä. Kehitystiimi vastaa itse siitä, että se saa tehtäväkasauman tehtyä sprintin loppuun mennessä.
- Kaikki scrumin kehitystiimin jäsenet ovat kehittäjätittelillä huolimatta tehtävästään tiimissä. Tähän sääntöön ei ole poikkeuksia.
- Kehitystiimin sisällä ei pidä olla sisäkkäisiä tiimejä huolimatta erilaisista tehtävistä, kuten testauksesta tai business-analytiikasta. Tähänkään sääntöön ei ole poikkeuksia. Kehitystiimin eri jäsenillä saattaa olla eri taitoja ja keskittymisen kohteita, mutta vastuu kaikesta tekemisestä kuuluu yhdessä koko tiimille. [11]

Scrum-mestari on vastuussa siitä, että scrum-tiimi ymmärtää scrumin teorian, käytännön ja säännöt sekä tukee sekä projektin omistajaa, kehitystiimiä, että projektin

tekevää organisaatiota [11]. Scrum-mestaria voidaan pitää projektipäällikkönä ilman valtaa: hän vastaa pyrähdyksien tuloksista sekä myös tiimin hyvinvoinnista [3].

Scrum tiimillä on yhteinen päämäärä, ja tiimin pitää toimia yhdessä saavuttaakseen sen. Layton (2015) vertaa kirjassaan Scrum for Dummies, scrum-tiimiä amerikkalaisen jalkapallon juoksupeliin. Tuotteen omistaja on pelinrakentaja, joka johtaa koko tiimiä ja päättää mihin juostaan. Tuotantotiimi on keskushyökkääjä (running back), joka juoksee pallon kanssa ja ansaitsee edetyt jaardit. Scrum-mestari on hyökkäyksen linja, joka tekee työtä, että keskushyökkääjän olisi mahdollisimman helppoa ja vaivatonta edetä pallon kanssa. [10]

Scrumillakin huonoja puolia ja haasteita. Scrumin omaksuminen ja oikeaoppinen käyttäminen vaatii aikaa, rahaa sekä koulutusta henkilökunnalle[3]. Scrumin omaksuminen vaatii myös omia tiloja, sillä scrum-tiimin ei tule olla pelkästään samassa rakennuksessa, vaan myös työskennellä samassa tilassa, jotta kommunikaatio tiimin välillä on mahdollisimman ketterää [10]

3.2.2 DevOps

DevOps (Development plus Operations; kehitys ja tuotanto) yhdistää ohjelmistokehityksen kaksi puolta, kehityksen ja tuotannon, hyödyntämällä automatisoituja kehitystä, tuotantoa ja infrastruktuurin monitorointia [12]. DevOps rikkoo perinteistä ohjelmiston elinkaaren mallia (vaatimusten määrittelystä, ohjelman kehityksestä, testauksesta ja lopulta käyttöönotosta) ottaen käyttöön - yleensä avoimen lähdekoodin - sovelluksia. Näillä sovelluksilla pystytään hallitsemaan kehityksen jokaista vaihetta ja varmistamaan, että ne toimivat yhtenäisenä yksikkönä. [13]. Yksi DevOpsin pääfilosofiasta on automatisoida mahdollisimman pitkälle kaikki mekaaninen työ, kuten testaus, lähdekoodin hallinta ja ohjelmiston kääntämiseen liittyvä työ ja siten antaa enemmän aikaa ohjelman kehitykselle ja muille projektin tärkeille tehtäville [14].

DevOpsin juuret juontavat vuoteen 2009, jolloin John Allspawn ja Paul Hammond pitivät konferenssipuheen aiheesta "10 deploys per day: Dev and Ops Cooperation at Flickr" (10 julkaisua päivässä: operaation ja tuotannon yhteistyö Flickr:llä". 10 julkaisua päivässä pidettiin mahdottomana, ja muodostunut DevOps sai nopeasti huomiota. Erityisesti pienemmät web ohjelmia kehittävät start-up firmat ottivat DevOpsin filosofiasta mallia, sillä usein kehittäjät tuottivat ohjelmiin muutoksia nopeaa tahtia,

mutta tuotannoilla oli vaikeuksia julkaista samalla tahdilla muutoksia eteenpäin. Vähitellen myös isommat firmat ottivat käyttöön DevOpsin ja vuonna 2013 IBM ja CA Technologies ilmoitti (ironisesti samana päivänä) käyttävän DevOpsia. [15]

DevOpsi käyttää jatkuvaa integraatiota ja jatkuvaa tuotantoa. Jatkuva integraatio (continuous integration) tukee työkaluillaan ja periaatteillaan yhtä aikaisesti monen tiimin työntekoa samaan projektiin. Sharma luettelee 10 jatkuvan integraation periaatetta kirjassaan The DevOps Adoption Playbook : A Guide to Adopting DevOps in a Multi-Speed IT Enterprise:

1. Vain yksi lähdekoodin hallintajärjestelmä (SCM, Source Code Management): Lähdekoodin hallintajärjestelmän käyttäminen on välttämätöntä, mutta kaikki koodi tulee olla yhdessä CSM:ssä ja vai yksi CSM:tulee olla käytössä.

2. Koodin rakentamisen (build) automatisointi: rakentamisen automatisointi tekee jatkuvasta integraatiosta jatkuvaa.

3. Testien automatisointi: Kuten rakentamisen, myös testien tulee olla automaattisia. Jatkuvan integraation tavoite on integraation lisäksi pitää ohjelma toimivana projektin joka vaiheessa. Aina kun koodi rakennetaan, myös sen testit ajetaan.

4. Jokaisen tulee muuttaa ja julkaista (commit) koodia päähaaralle päivittäin: Tämä auttaa pitämään integraatiota niin yksinkertaisena kuin mahdollista koska tulevat muutokset koodiin ovat kooltaan pienempiä ja niiden hallinta on helpompaa.

5. Jokaisen commitin tulee rakentaa koodi sen päähaaralta: Tämä liittyy myös neljänteen periaatteeseen, näin jokainen työntekijä saa lähes välittömästi palautteen omasta muutoksestaan.

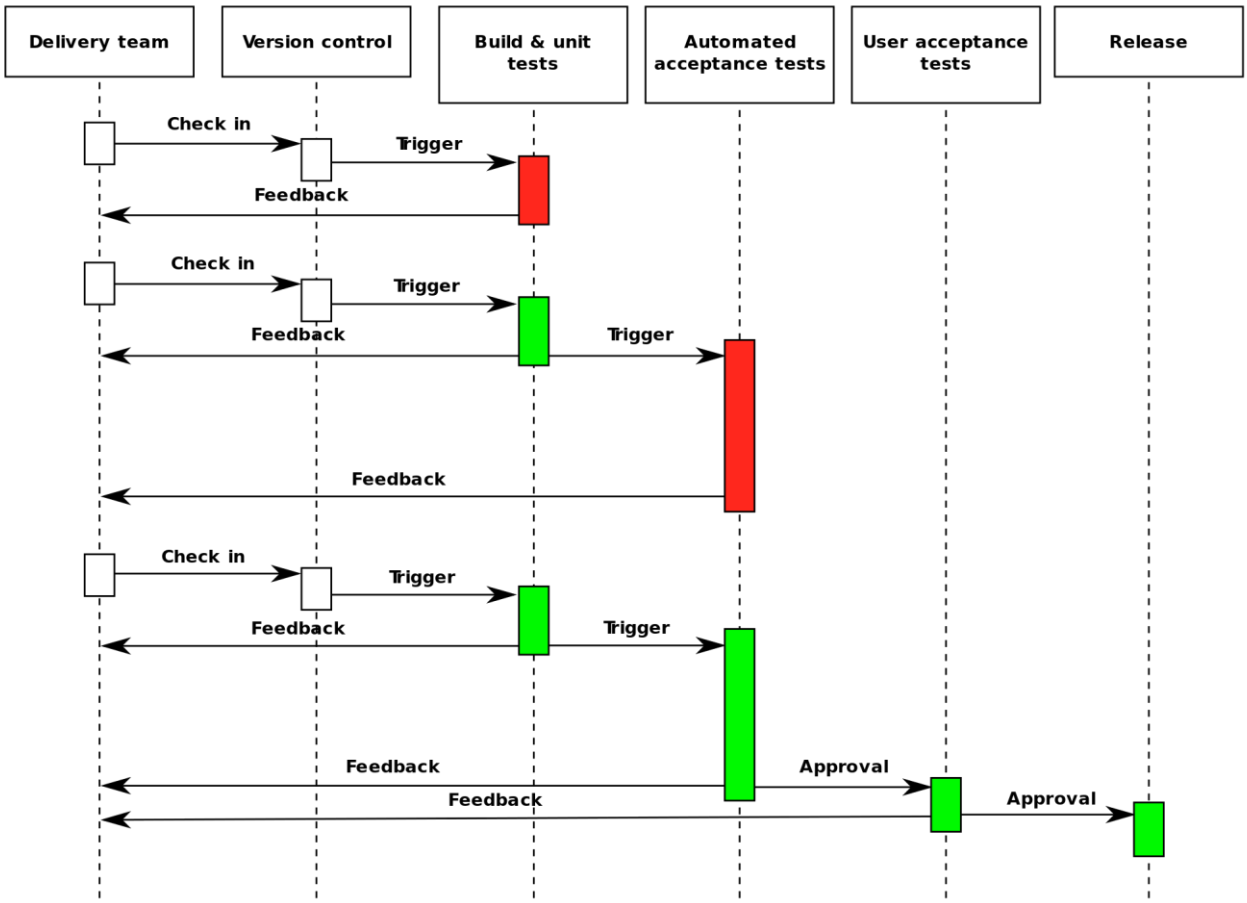
6. Koodin rakennuksen tulee olla nopeaa: Hidas koodin rakennus hidastaa integraatiota liikaa. Nykyaikaiset DevOps työkalut mahdollistavat nopean, vain muuttuneiden tiedostojen pohjalta tehdyn ohjelman rakennuksen.

7. Testaus tuotantoympäristön kopiossa: Mikäli testiympäristö ei vastaa täysin tuotantoympäristöä, voi ohjelmaan jäädä erittäin vakavia virheitä. Tämä ei aina onnistu täydellisesti, joten tuotantoympäristön kaltainen ympäristö tulisi olla mahdollisimman lähellä oikeaa tuotantoympäristöä.

8. Viimeisin rakennettu versio ohjelmasta tulee olla jokaiselle projektissa mukana olevalle helposti saatavilla: Näin myös asiakkaat ja muut projektiin liittyvät tahot pääsevät tarkastelemaan tuotteen kehittymistä.

9. Kaikkien tulee nähdä, mitä on tehty: Tämä helpottaa niin tiimin sisäistä kuin ulkoista kommunikaatiota.

10. Automatisoitu tuotanto: Jatkuva integraatio luonnollisesti johtaa jatkuvaan tuotantoon. [15]



Kuva 3. Jatkuvan tuotannon tuotantoputki [14]

Jatkuva tuotanto on jatkumo jatkuvasta kehityksestä. Se hyödyntää jatkuvan kehityksen kehitysputkea (pipeline), ja pitää sisällään jatkuvan integraation periaatteet ja vaiheet, pitäen huolen, että ohjelman buildi on aina käyttöönotettavissa [14]. Kuvassa 3 on kuvattu jatkuvan kehityksen tuotantoputki, minkä loppupäässä tuote on toimitettavassa.

Jatkuva tuotanto pyrkii kohtelemaan tuotantoympäristöä ja infrastruktuuria kuin ohjelmakoodia, jotta infrastruktuuria pystyy jakaa, testata ja versio kontrolloida helpommin [12]. Esimerkiksi kun tuotantotiimi tekee muutoksen ympäristöön, on kehitystiimin helpompi ottaa sama ympäristö kehitysympäristössä käyttöön [15].

4. TYÖMOTIVAATIO

Tässä osassa perehdytään työmotivaatioon ja siihen, miten työmotivaatio muodostuu työpaikalla ja yksilössä. Business Dictionary määrittää motivaation seuraavalla tavalla: ”Sisäisiä ja ulkoisia tekijöitä, jotka muokkaavat ihmisten halua ja energiaa olla jatkuvasti kiinnostunut ja sitoutunut työhön, rooliin, aiheeseen, tai päämäärän saavuttamiseen.” [16]. Latham ja Pinder (2005) kertovat työmotivaation olevan työntekijän sisäisten ja ulkoisten vaikutteiden muodostama voima, joka aiheuttaa työmyönteistä käytöstä ja määrittää sen muodon, suunnan sekä tehokkuuden [17].

Latham ja Pinder kertovat tutkimuksessaan (2002) työympäristön vaikuttaan ja ottavan vaikutteita ihmisen tarpeista, persoonasta ja arvoista. Työllä on monta eri ominaisuutta jotka vaikuttavat työntekijää. Suuri ero työn ominaisuuksien ja työntekijän arvojen välillä laskee työntekijän työmotivaatiota. Tärkeitä työn ominaisuuksia työmotivaation kannalta Lathamin ja Pinderin mukaan ovat autonomia, oppiminen, suorituskyky ja tyytyväisyys. [17]

Latham ja Pinder nostavat tutkimuksessaan esille autonomian erityisesti työmotivaatioon vaikuttavana tekijänä. Autonomia työssä helpottaa sekä oppimista että kehittymistä, mitkä puolestaan parantavat suorituskykyä. Oman työn (mitä tekee), aikataulun (milloin tekee), sekä tavoitteiden kontrollointi ovat työautonomian tärkeät osa-alueet. Esimiehen tulee puolestaan antaa saavutettavat tavoitteet, antaa valta päättää omista työaktiviteeteista, taata että tarvittavat resurssit ovat käytössä, ja antaa oikea-aikaista palautetta tavoitteen saavuttamisesta. [17]

Furnham ja Treglown kertovat kirjassaan Disencantment: Managing motivation and demotivation at work, että ihmisillä on sisäinen pyrkimys itsenäisyyteen ja oikeuteen päättää omista asioista, sekä pyrkimys olla tekemisissä muiden kanssa. Nykyaikaisen työkulttuurin tulisi keskittyä luomaan yllämainitun mukaista työkulttuuria, jotta työntekijät pystyvät keskittymään paremmin autonomiseen työskentelyyn, oppimiseen ja innovointiin. Furnham ja Treglown luettelevat autonomisen työnteon neljä tärkeää

elementtiä joihin esimiehen tulisi antaa työntekijälle päätäntävaltaa kokonaan tai osittain:

1. Milloin tekee työtä: Esimiehen tulisi keskittyä enemmän työn tulokseen kuin työn aikatauluttamiseen.
2. Miten tekee työtä: Työntekijän tulisi saada päättää miten tekee työnsä, kunhan lopputulos on haluttu.
3. Kenen kanssa tekee työtä: Työntekijöiden tulisi päästä mukaan rekrytoimaan hänen tiimiinsä tulevia työntekijöitä, tai muuten pystyä vaikuttamaan omaan tiimiinsä.
4. Mitä tehtävää tekee: Työntekijän kannattaa antaa silloin tällöin päästä tekemään mitä haluaa. [18]

Jeson Fox kertoo kirjassaan, The Game Changer: How to Use the Science of Motivation With the Power of Game Design to Shift Behaviour, Shape Culture and Make Clever Happen (2011), motivaatiosta työpaikalla. Foxin mukaan tärkein työntekijää motivaatiota lisäävä tekijä on tunne edistymisestä. Edistymisen tunne vahvistuu, kun viive oman työnteon ja palautteen välillä, eli palautesilmukka, on mahdollisimman lyhyt. Työntekijä todennäköisemmin saa enemmän motivaatiota työntekoon, kun hän näkee, miten hänen työpanoksensa on vaikuttanut työprosessissa. [19]

Foxin mukaan nimenomaan tunne edistymisestä motivoi, eikä edistymisen tarvitse olla konkreettista. Mitä nopeammin palautteen saa, sitä paremmin se vaikuttaa edistymisen tunteeseen. Hän ottaa esimerkikseen monet videopelit, missä 80% pelaajan ajastaan käyttämästä ajasta johtaa epäonnistumiseen, mutta silti pelaaja nauttii pelaamisesta. [19]

Weinschenk luettelee kirjassaan, How to Get People to Do Stuff: Master the art and science of persuasion and motivation, seitsemän tekijää, jotka motivoivat ihmistä tekemään asioita. Näistä tekijöistä tärkeimmät ja tutkimuksen kannalta relevantit ovat:

- Tarve kuulua joukkoon: Me ihmiset olemme erittäin sosiaalisia ja meillä on vahva tarve kommunikoida muiden kanssa. Emme pidä yksinolemisesta ja teemme töitä sen eteen, että muut hyväksyisivät meidät. Tarvitsemme tunteen, että kuulumme maailmaan, jossa olemme. Tätä tunnetta voi käyttää hyväkseen ihmisten motivoinnissa.
- Palkitseminen: Vaikka tiedetään, että esimerkiksi vaisto, halu kehittyä sekä halu kuulua yhteisöön on palkitsemista tehokkaampia motivaation lähteitä, ei

palkitsemisen vaikutusta voi sivuuttaa. Oikeaan aikaan oikea palkitseminen motivoi huomattavasti ihmistä.

- Taidon mestarointi: Ihmisillä on vahva halu kehittää itseään ja mestaroida osaamisensa. Tämä halu on huomattavasti vahvempi motivaation lähde kuin esimerkiksi palkitseminen. [20]

5. OHJELMISTOTUOTANNON MENETELMÄN VAIKUTUS TYÖMOTIVAATIOON

Tämä osio yhdistää kahden edellisen osan teorian ohjelmistotuotannon eri menetelmistä, sekä työmotivaatiosta. Osio pyrkii yhdistelemään ja tuomaan esille näkökulmia ja teoriaosuuden teorioita, vastatakseen päätutkimuskysymykseen.

Teoriaosuus osoittaa, että moni aineisto käsittelee samoja asioita. Useat lähteet [17] [18] tukevat väitettä, että mahdollisuus vaikuttaa omaan työhön lisää työmotivaatiota. Eri ohjelmistotuotannon menetelmissä on selviä eroja siihen, miten työntekijä voi itse työhönsä vaikuttaa. Vesiputousmalli ei perinteisenä menetelmänä jousta paljoa siinä, miten itse voi työhönsä vaikuttaa. Vesiputouksessa tehtävät ja vaiheet seuraavat lineaarisesti toisiaan, joten mahdollisuus vaikuttaa omiin tehtäviin on pienempi, kuin esimerkiksi scrumissa, missä tuotantotiimi itse päättää miten suorittaa projektin omistajan laatiman työlistan alkiot. Lähteiden mukaan DevOps ei määrittele erityisemmin, kuinka paljon työntekijä itse pystyy päättämään, mitä tekee ja millä työkaluilla.

Toinen teoriaosuudessa ilmenevä yhtäläisyys on palaute. Oikeaan aikaan ja oikealla tavalla annettu palaute motivoi työntekijöitä paremmin kuin hitaasti tuleva ja huonosti annettu palaute [17][19]. Vesiputouksella tehtävässä projektissa palautteella saattaa kestää pitkäänkin, sillä iso osa ohjelmasta testataan vasta projektin loppuvaiheessa suoritettavassa testausvaiheessa. Palautetta tulee myös tuotantovaiheen aikana, mutta varmuus oman työn onnistumisesta selviää vasta testausvaiheessa. Scrumissa palaute tulee nopeaa tahtia päivittäisten scrum palaverien ansiosta ja varsinkin pyrähdysten lopussa järjestettävässä katselmointikokouksessa. Oma työn ja onnistumisten konkreettinen näkeminen tuotantovalmiina tuotteena lisää myös motivaatiota. Kuten scrumissa, DevOpsissa palaute tulee myös nopeasti automaattisten testien johdosta. Palaute saadaan päivittäin, kun versionhallintaan tallennetaan ohjelmakoodia.

Sosiaalinen vastuu parantaa työmotivaatiota [21]. Vaikka kaikissa ohjelmistotuotannon menetelmissä tavoitteena on saada projekti valmiiksi, on scrumissa yksin

ohjelmointitiimillä vastuu työlistan alkioiden tekemisestä. Tämä vastuu lisää työmotivaatiota scrumin tuotantotiimillä. Scrumissa ja DevOpsissa tavoitteet ja maali ovat selkeästi esillä, ja oman etenemisen näkee selkeästi. Vesiputouksessa puolestaan tavoitteet ja maali voivat olla esillä, mutta omaa etenemistä ei pysty samassa määrin näkemään kuin ketterissä menetelmissä, koska vesiputouksessa vaiheet myöhästyvät usein [7].

Ohjelmistotuotannon menetelmä valitaan projektin alkuvaiheessa, tai ennen projektin aloittamista. Työntekijän motivaatioon voi vaikuttaa myös se, pääseekö hän itse mukaan valitsemaan käytettävää menetelmää, vai pitääkö hänen tuottaa ohjelmaa menetelmällä, mistä hän ei itse pidä.

6. YHTEENVETO

6.1 Tutkimuksen tulokset

Tutkimuksen tarkoitus oli tutkia ohjelmistotuotannon menetelmän vaikutusta työmotivaatioon. Teorioiden pohjalta tutkimuksen tulos on, että scrumilla tai DevOpsilla ohjelman tuottaminen motivoi enemmän työntekijää kuin vesiputouksella. Ketterissä menetelmissä, scrumissa ja vesiputouksessa, on työntekijällä enemmän vapautta vaikuttaa omaan työhönsä kuin vesiputousmallissa. Myös palaute omasta työstä ja etenemisestä tulee nopeammin ketteriä menetelmiä käyttäen.

6.2 Tulosten arviointi

Tutkimus pyrki vastaamaan päätutkimuskysymykseen, kuinka ohjelmistotuotannon menetelmän valinta vaikuttaa työmotivaatioon. Tutkimuksen tieto on kuitenkin kerätty täysin kirjallisuudesta, joten tarkemman ja paremmin todellisuutta kuvaavan tuloksen saisi empiirisesti tutkimalla, esimerkiksi kyselyllä.

LÄHTEET

[1] Haikala, T, Mikkonen I. Ohjelmistotuotannon käytännöt, Talentum Oyj 2011.

[2] Baddoo N, Beecham S, Hall T, Robinson H, Sharp H. Models of motivation in software engineering in: Information and Software Technology, 2019, pp. 219-233.

[3] Munassar, N. & Govardhan, A. A Comparison Between Five Models Of Software Engineering, 2010. Saatavissa: https://pdfs.semanticscholar.org/3a4a/2cb2328e2f416be0be012e5d580975943554.pdf#page=115

[4] Seppänen, J. Scrum – from Theory to Practice in Software Development, 2016. Saatavissa: http://dspace.cc.tut.fi/dpub/handle/123456789/24006

[5] Ji, F. & Sedano, T. Comparing extreme programming and Waterfall project results, 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 2011, pp. 482-486.

[6] Pressman, R.S. Software Engineering: A Practitioner's Approach. Palgrave Macmillan, London, 2005.

[7] Tal, L. Agile Software Development with HP Agile Manager, Apress, Berkeley, CA, 2015.

[8] Baca D, Petersen K, Wohlin CThe Waterfall Model in Large-Scale Developmen, Blekinge Institute of Technology, Karlskrona, Sweden, 2009.

[9] Gerry Coleman . Agile Software Development, Software Quality Professional, Vol. 19(1), 2016, pp. 23. Saatavissa: https://search.proquest.com/docview/1866513328.

[10] Layton, M.C. Scrum for dummies, Wiley, Hoboken, NJ, 2015.

[11] Schwaber K. Sutherland J. Scrum guide, 2013. Saatavissa: _https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

[12] Ebert, C., Gallardo, G., Hernantes, J. & Serrano, N. DevOps, IEEE Software, Vol. 33(3), 2016, pp. 94-100. Saatavissa: http://ieeexplore.ieee.org/document/7458761

[13] Vadapalli, S. DevOps: Continuous delivery, integration, and deployment with devops, Pact publishing, 2018.

[14] Humble, J. & Farley, D. Continuous delivery, Tenth printing ed. Addison-Wesley, Upper Saddle River, NJ [u.a.], 2015.

[15] Sharma, S. The DevOps Adoption Playbook : A Guide to Adopting DevOps in a Multi-Speed IT Enterprise, John Wiley & Sons, Incorporated, Somerset, 2017.

[16] Business dictionary, 2020. Saatavissa: http://www.businessdictionary.com/definition/motivation.html

[17] Latham, G.P. & Pinder, C.C. Work Motivation Theory and Research at the Dawn of the Twenty-First Century, Annual Review of Psychology, Vol. 56(1), 2005, pp. 485-516. Saatavissa: http://search.ebscohost.com/login.aspx?direct=true&db=afh&AN=15888374&site=ehost-live&scope=site

[18] Furnham, A. & Treglown, L. Disencantment: Managing motivation and demotivation at work, Bloombury, 2017.

[19] Fox, J. The Game Changer : How to Use the Science of Motivation with the Power of Game Design to Shift Behaviour, Shape Culture and Make Clever Happen, 1st ed. Wrightbooks, Milton, QLD, 2014.

[20] Weinschenk, S.M. How to get people to do stuff, New Riders, Berkeley, CA, 2013.

[21] Skudiene, V. & Auruskeviciene, V. The contribution of corporate social responsibility to internal employee motivation, Baltic Journal of Management, Vol. 7(1), 2012, pp. 49-67. Saatavissa: http://www.emeraldinsight.com/doi/abs/10.1108/17465261211197421