INNOVAATIOPROSESSIA TUKEVAT MENETELMÄT – CASE LIIKETOIMINNAN LAAJENTAMINEN OHJELMISTOTUOTANTOON

Methods for supporting innovation process – a case study of company expansion into software engineering

Lappeenrannan–Lahden teknillinen yliopisto LUT

Tuotantotalouden kandidaatintyö

2022

Aino Liukkonen

Tarkastaja: Dosentti Lea Hannola

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUT Teknis-luonnontieteellinen

Tuotantotalous

Aino Liukkonen

Innovaatioprosessia tukevat menetelmät – case liiketoiminnan laajentaminen ohjelmistotuotantoon

Tuotantotalouden kandidaatintyö

2022

45 sivua, 3 kuvaa ja 1 taulukko

Tarkastaja: Dosentti Lea Hannola

Avainsanat: innovaatioprosessi, innovaatioprosessin alkuvaihe, tuotekehitys, NPD, ketterät menetelmät

Keywords: innovation process, front end of innovation, fuzzy front-end, new product development, agile

Perinteisiltä insinöörialoilta ohjelmistotuotantoon laajentaminen vaatii yritykseltä uuden toimintakulttuurin omaksumista. Ohjelmistotuotannon alaa määrittävät erityispiirteet, kuten uudelleenkäytettävien ohjelman osien hyödyntäminen, nopea kehitystahti sekä prototyyppien laaja hyödyntäminen. Viime vuosien ja vuosikymmenten aikana ohjelmistotuotannon menetelmät ovat myös kehittyneet paljon.

Innovaatioprosessi koostuu kolmesta osasta, jotka ovat innovaatioprosessin alkuvaihe, tuotekehitys sekä kaupallistaminen. Tämän kandidaatintyön tavoite on vertailla innovaatioprosessin alkuvaihetta ja tuotekehitystä tukevia menetelmiä sekä löytää niistä kohdeyrityksen tarpeisiin parhaiten sopivat. Työ on toteutettu kirjallisuuskatsauksena. Lisäksi se sisältää case-osuuden, jossa tarjotaan ehdotus kohdeyritykselle innovaatioprosessia tukevista menetelmistä kirjallisuuskatsauksessa esiin nousseiden havaintojen pohjalta.

Tutkimuksessa kävi ilmi, että ketterät menetelmät ovat suuressa suosiossa ohjelmistotuotannon alalla. Ne mahdollistavat lyhyet iteraatiokierrokset sekä asiakkaan osallistamisen kehitykseen jo projektin aikaisessa vaiheessa. Innovaatioprosessin alkuvaiheeseen löydettiin lukuisia menetelmiä, joista tässä työssä esitellään muutamia. Ideoiden luomiseen kannattaa käyttää menetelmiä, jotka yhdistävät yksilö- ja ryhmätyöskentelyä. Kohdeyritykselle ideoiden arviointiin sopivimmiksi nähtiin kevyet menetelmät. Kokonaisratkaisuna kohdeyritykselle esitetään yhdenmukaisten tekniikoiden käyttämistä innovaatioprosessin alkuvaiheen ja tuotekehitysvaiheen välillä.

Sisällysluettelo

1 Johdanto

1.1 Tutkimuksen taustaa

Ohjelmistot ovat tulleet entistä tärkeämmiksi nykypäivänä myös teollisuudessa. Koneiden ja laitteiden älykkäitä ominaisuuksia sekä simulointia voidaan käyttää hyödyksi lähes alalla kuin alalla. Perinteisten ohjelmistotuotannon menetelmien rinnalle on viime vuosina ja vuosikymmeninä noussut paljon uusia menetelmiä (Broy, 2018). Nopeasti kehittyvällä alalla myös käytettävät menetelmät kehittyvät.

Tämä kandidaatintyö on toteutettu yritystoimeksiannosta. Kohdeyritys on monialainen insinööritoimisto, jonka tavoitteena on laajentaa toimintaansa ohjelmistotuotannon alalle tarjoten asiakkailleen kokonaisvaltaisempaa palvelua. Yrityksellä ei ole aiempaa liiketoimintaa alalla tai ohjelmistotuotantoon vaadittavaa organisaatiota. Kilpailu alalla on kovaa, ja tästä syystä on erityisen tärkeää löytää yrityksen tarpeisiin parhaiten soveltuvat menetelmät sujuvaan ohjelmistojen innovaatioprosessiin ja tuotekehityksen projektinhallintaan. Tässä työssä yrityksestä käytetään nimeä Yritys X.

Innovaatioprosessi määritellään eri tavoin eri lähteissä. Koen et al. (2001) jakavat innovaatioprosessin kolmeen vaiheeseen. Nämä vaiheet ovat innovaatioprosessin alkuvaihe (front-end of innovation), uuden tuotteen ja prosessin kehitys (new product and process development) sekä kaupallistaminen (commercialization). Tässä tutkimuksessa innovaatioprosessilla viitataan Koenin et al. (2001) määritelmään. Tutkimus keskittyy innovaatioprosessin alkuvaiheeseen sekä tuotekehitykseen. Näihin vaiheisiin etsitään keinoja ja menetelmiä ohjelmistotuotannon alalle laajentavalle yritykselle.

1.2 Tutkimuksen tavoitteet ja tutkimuskysymykset

Tämä kandidaatintyö kartoittaa ja vertailee keskeisimpiä menetelmiä innovaatio- ja tuotekehitysprosessien hallintaan. Ohjelmistotuotannon tuotekehitystä ja projektinhallintamenetelmiä on tutkittu paljon jo vuosikymmenten ajan. Myös innovaatioprosessin alkuvaiheeseen

on kiinnitetty kasvavassa määrin huomiota 2000-luvulla. Tämä tutkimus pyrkii löytämään keinoja sovittaa yhteen innovaatioprosessin alkupään ja tuotekehityksen keinoja yrityksessä.

Tutkimuskysymykset ovat seuraavat:

1. Mitä ohjelmistotuotantoon laajentaminen vaatii yritykseltä?
2. Millaisia menetelmiä innovaatioprosessin alku- ja tuotekehitysvaiheiden hallintaan on olemassa?
3. Mitä yrityksen tulee ottaa huomioon menetelmien valinnassa?
4. Millaisia menetelmiä Yritys X:n kannattaa ottaa käyttöön laajentaessaan ohjelmistotuotantoon?

Tutkimuksen tavoitteena on vertailla erilaisia innovaatioprosessia tukevia menetelmiä ja antaa lukijalle yleiskuva innovaatioprosessista sekä siihen liittyvien menetelmien hyvistä ja huonoista puolista. Lisäksi tutkimus tarjoaa kirjallisuuteen pohjautuen Yritys X:lle ehdotuksen menetelmistä, jotka sopivat yrityksen tarpeisiin sekä sen tavoitteeseen laajentaa toimintaansa ohjelmistotuotannon alalle.

1.3 Tutkimusmenetelmät, rajaukset ja rakenne

Tutkimus on toteutettu integroivana kirjallisuuskatsauksena. Integroiva kirjallisuuskatsaus on kuvailevan kirjallisuuskatsauksen muoto, jonka tavoitteena on kuvata tutkittavaa ilmiötä laajasti sekä tuottaa uutta tietoa jo tutkitusta aiheesta (Salminen, 2011). Kirjallisuuskatsauksen lisäksi kandidaatintyön lopussa on case-osuus, jonka tavoitteena on tarjota Yritys X:lle ehdotus käytettävistä menetelmistä kirjallisuuskatsaukseen pohjaten.

Tässä tutkimuksessa hyödynnetään pääasiassa vertaisarvioituja tieteellisiä julkaisuja sekä joitakin kirjoja, seminaarijulkaisuja ja verkkolähteitä. Tieteellisiä sekä kirjalähteitä on haettu LUT Primo-, SCOPUS- ja Google Scholar -tietokannoista. Tärkeimpiä hakusanoja ovat muun muassa *front-end of innovation, idea generation, idea selection, new product development, agile software engineering* ja *waterfall*.

Tämä kandidaatintyö käsittelee innovaatioprosessin alkuvaihetta sekä tuotekehitystä. Innovaatioprosessin kolmatta osaa eli kaupallistamista ei käsitellä. Innovaatioprosessia

tarkastellaan ohjelmistokehityksen kontekstissa ja erityisesti alalle laajenemista tavoittelevan yrityksen näkökulmasta. Tutkimus esittelee useita innovaatioprosessia tukevia menetelmiä sekä ohjelmistotuotannon alalle laajenemiseen liittyviä seikkoja. Tavoitteena on tarjota lukijalle yleiskäsitys niistä sekä informaatiota, jonka perusteella lukija voi arvioida esimerkiksi eri menetelmien soveltuvuutta omaan yritykseen.

Työ koostuu kuudesta luvusta. Johdannon jälkeen seuraavassa luvussa kerrotaan lyhyesti ohjelmistokehityksen erityispiirteistä sekä ohjelmistokehitykseen laajentamiseen liittyvistä vaatimuksista yritykselle. Kolmas ja neljäs luku käsittelevät innovaatioprosessia sekä sitä tukevia menetelmiä ja työkaluja kirjallisuuskatsauksen avulla. Kolmas luku keskittyy innovaatioprosessin alkuvaiheeseen ja neljäs luku tuotekehitykseen. Viides luku esittää kirjallisuuskatsauksen perusteella ehdotuksen käytettävistä menetelmistä Yritys X:lle. Viimeisessä luvussa kootaan yhteen tutkimuksen johtopäätökset.

2 Liiketoiminnan laajentaminen ohjelmistotuotantoon

Ohjelmistokehitys on nopeasti kasvava ala. Nykypäivänä ohjelmistokehityksen erityispiirteenä ovat nopeasti muuttuvat vaatimukset, aiempaa nopeammat toimitusajat ja kehityssyklit sekä suuri kilpailu alalla. Ala on laaja ja kehittyy koko ajan. (Sommerville, 2016, s. 18–27). Kullakin alalla on omat erityispiirteensä. Tämä kappale kertoo lyhyesti ohjelmistokehityksen erityispiirteistä sekä alalle laajentamisen vaatimuksista.

Erilaisten prototyyppien rakentaminen on ohjelmistokehityksessä helpompaa kuin fyysisten laitteiden kohdalla eikä vaadi yhtä paljoa resursseja. Niiden avulla tuotteen toiminnallisuutta voidaan esitellä asiakkaalle jo kehityksen alkuvaiheessa. Erityisesti parin viime vuosikymmenen aikana asiakkaan osallistaminen projektiin jo tuotekehitysvaiheessa on noussut suureen suosioon ohjelmistokehityksen alalla. Prototyypit auttavat myös kehitystiimiä hahmottamaan tuotteen toimintaa ja kehityskohteita. Prototyypeille on kaksi päävaihtoehtoa, jotka ovat evoluutioprototyypit (evolutionary prototype) ja kertakäyttöiset prototyypit (throwaway prototype; rapid prototype). Evoluutioprototyyppien tarkoitus on toimia oikeiden komponenttien tapaan, ja kehityksessä lähdetään kehittämään tuotetta prototyypin päälle. Niiden avulla voidaan testata ohjelman suunnitteluratkaisuita. Kertakäyttöiset prototyypit puolestaan vain auttavat tuotteen mallintamisessa, ja prototyypin jälkeen varsinaisen tuotteen kehitys aloitetaan alusta. Kertakäyttöisiä prototyyppejä käytetään erityisesti käyttöliittymäsuunnittelussa. (Haikala & Mikkonen, 2011, s. 38–39).

Verrattuna fyysisten tuotteiden tai laitteiden tuotekehitykseen ohjelmistokehityksessä tuotetta on helppo muokata ja kehittää sen julkaisun jälkeen. Alalla voidaan kehittää tuotteita julkaisten ensimmäisenä versiona Minimum Viable Product. Minimum Viable Product (MVP) on sellainen versio tuotteesta, joka sisältää kaikki välttämättömät toiminnallisuudet julkaisemista varten. MVP:n avulla voidaan myös esitellä tuotetta sidosryhmille kehityksen aikaisessa vaiheessa sekä sitä voidaan käyttää apuna tuotteen suunnittelussa. Ohjelmistoliiketoimintaa aloittavalle yritykselle voi olla erityistä hyötyä MVP:stä. (Duc & Abrahamsson, 2016). Ohjelmistotuotteista julkaistaan tyypillisesti uusia versioita, ja uusiin versioihin sisällytetään uusia tai paranneltuja ominaisuuksia aiempiin versoihin verrattuna.

Ohjelmistokehitykselle tyypillistä on rakentaa ohjelmia helposti uudelleenkäytettävistä palasista. Koodin uudelleenkäytettävyys säästää yritykseltä rahaa ja työntekijöiden aikaa. (Sandhu et al., 2010). Erityisesti moduulien keskinäinen riippuvuus, koheesio, ohjelmakoodin monimutkaisuus, periytyminen ja ohjelman koko ovat tärkeitä koodin uudelleenkäytettävyyteen vaikuttavia tekijöitä (Mehboob et al., 2021). Tyypillistä on, että yritys kehittää alustan, jonka päälle se rakentaa useita tuotteitaan. Joskus myös kolmannet osapuolet voivat kehittää lisäosia alustan päälle, kuten esimerkiksi älypuhelinten käyttöjärjestelmissä tai internetin hakukoneissa. (Tiwana et al., 2010).

Tärkeänä vaatimuksena yritykselle ohjelmistotuotantoon laajennettaessa on uusien menetelmien omaksuminen jokaisella organisaation tasolla. Ohjelmistokehitystiimin lisäksi esimerkiksi yritysjohdon ja myynnin tulee ymmärtää käytettyjä menetelmiä sekä niiden hyötyjä ja haittoja yritykselle. Jos ohjelmistokehityksessä otetaan käyttöön yrityksessä entuudestaan tuntemattomia menetelmiä, on tärkeää, myös yritysjohdon, myynnin ja markkinoinnin työntekijät koulutetaan menetelmään. Esimerkiksi ohjelmistoalalla nykyään suuressa suosiossa olevat ketterät kehitysmenetelmät voivat olla organisaatiossa entuudestaan tuntemattomia. Dikert et al. (2016) mainitsevat ketterien kehitysmenetelmien toteuttamisen suurimmiksi haasteiksi muun muassa jäykän organisaatiokulttuurin ja muutosvastarinnan organisaatiossa, osaamisen puutteen käyttöön otettaviin menetelmiin liittyen sekä heikon johtoportaan tuen.

Toiminnan laajentaminen uudelle toimialalle on aina investointi yritykselle. Henkilöstön ja koulutusten lisäksi ohjelmistotuotannon alalla täytyy tehdä investointeja työkaluihin, kuten kehitysympäristöä varten tarvittaviin palvelimiin ja ohjelmistolisensseihin. Tässä kandidaatintyössä keskitytään pääosin ohjelmistokehityksen menetelmiin sekä sivutaan myös niiden toteuttamiseen vaadittavia henkilöstöresursseja ja kyvykkyyksiä.

3 Innovaatioprosessin alkuvaihe

3.1 Innovaatioprosessin alkuvaiheen kuvaus

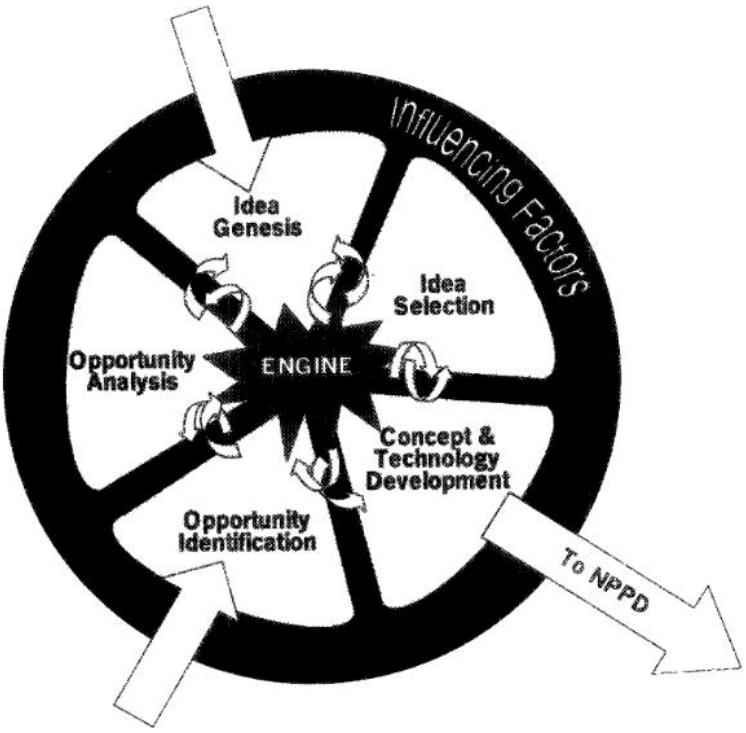
Innovaatioprosessin alkuvaihe (front-end of innovation; fuzzy front end) on innovaatioprosessin ensimmäinen vaihe. Kim ja Wilemon (2002) määrittelevät käsitteen *fuzzy front-end* jaksoksi, joka alkaa siitä, kun uusi mahdollisuus löydetään ja päättyy, kun idean katsotaan olevan valmis kehitykseen. Ohjelmistoprojektissa se käsittää siis alustavan tutkimuksen ja suunnittelun ennen ohjelmistokehitysvaihetta. Innovaatioprosessin alkuvaihe on erittäin merkityksellinen projektin onnistumisen kannalta, ja siksi siihen tulee kiinnittää huomiota.

Koen et al. (2001) esittävät artikkelissaan New Concept Development -mallin (NCD), jonka avulla voidaan vertailla ja tutkia eri yritysten innovaatioprosessin alkupään toimintoja. Kuvassa 1 kuvataan NCD-mallin toiminta. Malli koostuu kolmesta osasta:

1. Mallin sisäosassa on viisi tärkeää innovaation alkupään toimintoa. Nämä ovat mahdollisuuksien tunnistaminen, mahdollisuuksien analysointi, ideoiden luonti, ideoiden valinta sekä konseptien ja teknologian kehitys. Näiden välillä voidaan liikkua innovaatioprosessin alkuvaiheessa missä tahansa järjestyksessä.

2. Mallin keskiössä on prosessin ”moottori”, joka on prosessin kantava voima. NCD-mallissa moottori tarkoittaa organisaation johtamista ja yrityskulttuuria.

3. Mallin ulkoreunalla ovat vaikuttavat tekijät, joihin kuuluvat organisaation kyvykkyydet, liiketoimintastrategia, yrityksen ulkoinen ympäristö sekä tiede. (Koen et al., 2001).



Kuva 1. NCD-malli (Koen et al., 2001, s. 47)

Mahdollisuuksien tunnistaminen (Opportunity Identification)

Mahdollisuuksien tunnistaminen tarkoittaa sitä, kun organisaatio päättää mahdollisuuksista, joita se haluaa hyödyntää. Esimerkkejä tällaisista mahdollisuuksista ovat kilpailulliseen uhkaan vastaaminen tai kilpailuedun tavoittelu uusilla tuotteilla. Yritys voi päättää tehdä joko pieniä uudistuksia johonkin olemassa olevaan tuotteeseen tai kehittää toimintaansa aivan uudelle liiketoiminta-alueelle. Mahdollisuuksien tunnistaminen pohjautuu yleisesti yrityksen tavoitteisiin ja strategiaan. (Koen et al., 2001).

Mahdollisuuksien analysointi (Opportunity Analysis)

Tunnistettuja mahdollisuuksia sekä niiden markkinapotentiaalia tutkitaan. Metodit voivat vaihdella yksinkertaisista monimutkaisiin. Usein mahdollisuuksien analysointi sisältää muun muassa markkina-analyysin ja trendien analysointia (Koen et al., 2001). Nykyään trendien ja olemassa olevien tuotteiden analysoinnissa voidaan käyttää hyödyksi kasvavassa määrin myös data-analytiikkaa ja tekoälyä (Lee et al., 2020).

Ideoiden luominen (Idea Genesis)

Tässä mallin vaiheessa jalostetaan tunnistetusta mahdollisuudesta konkreettinen idea. Ideaa voidaan työstää useaankin otteeseen ja parannella sitä. Yrityksellä on ideoiden luomiseen usein virallinen prosessi, joka kannustaa työntekijöitä luomaan uusia ideoita. Ideat voivat myös syntyä prosessin ulkopuolella esimerkiksi asiakkaan toiveiden tai epäonnistuneiden kokeiluiden seurauksena. (Koen et al., 2001). Useat yritykset myös keräävät ideoita ehdotuslaatikkoon. Työntekijöiden motivoiminen auttaa laadukkaiden ideoiden syntymiseen.

Ideoiden valinta (Idea Selection)

Jatkojalostettavien ja tuotantoon otettavien ideoiden valinta on tärkeä osa innovaatioprosessin alkuvaihetta. Useimmilla yrityksillä on jonkinlainen systeemi ideoiden ja konseptien arvioimiseen. Tavoitteena on löytää sellaiset yrityksen tavoitteisiin ja strategiaan sopivat ideat, joilla on eniten potentiaalia tuottaa voittoa ja kasvumahdollisuuksia yritykselle tulevaisuudessa. Martinsuo ja Poskela (2011) tuovat tutkimuksessaan esille, että ideoiden arviointi on tärkeää erityisesti tulevaisuuden mahdollisuuksien kannalta. Kock et al. (2015) nostavat tärkeäksi myös tuoteportfolion hallinnan ideointivaiheessa, jossa yrityksen tavoitteena on koota tasapainoinen portfolio potentiaalisista innovaatioprojekteista. Tuoteportfolion hallinta tarkoittaa käytännössä sen hallintaa, millaisen yhdistelmän tuotteita yritys kehittää ja ylläpitää. Cooper et al. (2001) jakavat tähän liittyvät ideoiden arviointimenetelmät viiteen merkittävimpään kategoriaan: taloudelliset menetelmät, liiketoimintastrategiaan perustuvat menetelmät, kuplakaavio, pisteytys sekä tarkastuslistat. Tässä tutkimuksessa esitellään pääasiassa sellaisia menetelmiä, joita voidaan käyttää myös yksittäisiin tuotteisiin lisättävien ominaisuuksien arviointiin.

Konseptien ja teknologian kehitys (Concept & Technology Development)

Tässä mallin vaiheessa kehitetään business case, joka perustuu kilpailija-analyysiin ja riskiarvioihin sekä arvioihin markkinapotentiaalista ja asiakastarpeista. Vaiheen muodollisuus vaihtelee riippuen esimerkiksi projektin luonteesta sekä yrityskulttuurista. Konseptien ja teknologian kehityksen jälkeen projekti voidaan siirtää tuotekehitykseen tai tarvittaessa voidaan palata muihin NPD-mallin vaiheisiin. (Koen et al., 2001).

3.2 Innovaatioprosessin alkuvaihetta tukevia menetelmiä ja työkaluja

Innovaatioprosessin alkuvaihetta tukemaan on kehitetty erilaisia menetelmiä ja työkaluja, niin ideointiin ja ideoiden keräämiseen, jatkojalostukseen otettavien ideoiden valitsemiseen kuin myöskin ideoiden kehittämiseen konsepteiksi. Tässä kappaleessa esitellään muutamia tällaisia menetelmiä. Suurin osa käsitellyistä menetelmistä liittyy Koenin et al. (2001) NPD-mallin vaiheisiin *Ideoiden luominen (Idea Genesis)* ja *Ideoiden valinta (Idea Selection)*.

Ideointi voi tapahtua joko yksilöinä tai ryhmissä. Perinteisesti ryhmien on uskottu tuottavan laadukkaampia ideoita yksilöihin verrattuna. Uudessa tutkimuksessa tätä näkemystä on myös kyseenalaistettu ja puhtaan yksilö- tai ryhmätyöskentelyn rinnalle mielenkiintoisena vaihtoehtona on nostettu näiden kahden yhdistelmä. Esimerkiksi Korden ja Pauluksen (2017) tutkimus viittaa siihen, että hybridimenetelmissä eli yhdistettäessä yksilö- ja ryhmätyöskentelyä saadaan aikaan parhaita ideoita.

Tässä kappaleessa käsiteltäviksi on valittu joitakin yleisimpiä yritysten käyttämiä menetelmiä innovointiin ja ideoiden arviointiin. Lisäksi tavoitteena on esitellä muutamia vähemmän tunnettuja menetelmiä ja työkaluja. Kappaleessa käydään läpi eri menetelmien etuja ja huonoja puolia sekä mihin kukin menetelmä soveltuu.

Brainstorming

Brainstorming on eräs yleinen ja yksinkertainen tekniikka, jota useat yritykset käyttävät innovaatioprosessin alkuvaiheessa. Alex Osborn kehitti brainstorming-tekniikan ja esitteli sen ensimmäistä kertaa käytännössä vuonna 1939. Tekniikassa ryhmä ihmisiä kokoontuu yhteen ja keksii yhdessä mahdollisimman paljon ideoita käsiteltävään ongelmaan. Fasilitaattori on vastuussa kokouksen kulusta ja pyrkii inspiroimaan osallistujia. (Osborn, 1953, s. 297–307). Osborn mainitsee neljä sääntöä brainstorming-sessioille. Ne ovat seuraavat:

1. Ideoita ei arvostella. Arvostelu jätetään myöhempään vaiheeseen ideoiden työstämistä.

2. Myös epärealististen ideoiden esittäminen on tervetullutta, sillä ne ovat usein luovia, herättävät ajatuksia ja niistä voi myöhemmin jalostaa myös toteutuskelpoisia ideoita.

3. Tavoitteena on muodostaa mahdollisimman suuri määrä ideoita.

4. Tavoitteena on lisäksi keksiä uusia ideoita, jotka kehittävät ja parantavat jo esitettyjä ideoita. Myös aiemmin esitettyjen ideoiden yhdistely on toivottavaa. (Osborn, 1953, s. 300–301).

Brainstorming on saanut osakseen myös paljon kritiikkiä, esimerkiksi sellaisten tutkimusten perusteella, joiden mukaan ryhmässä ideoinnin tuotteliaisuus kärsii. Useiden kognitiivisten prosessien arvellaan selittävän tätä ilmiötä. Ryhmässä työskennellessä esimerkiksi kritiikin pelko voi estää ideoiden sanomista ääneen. Toisaalta tuotteliaisuutta voi heikentää myös se seikka, että ääneen ideoita kertoessa osallistujien täytyy vuorotella, mikä rajoittaa esitettyjen ideoiden maksimimäärää tietyssä ajanjaksossa. Tästä huolimatta menetelmä on edelleen vuosikymmeniä kehittämisensä jälkeen laajasti käytetty. (Kalargiros et al., 2019).

Myöhemmin brainstormingista on kehitetty monia erilaisia versioita ja sitä sovelletaan eri tavoin. Brainwriting ja elektroninen brainstorming ovat esimerkkejä brainstormingin sovelluksista. Brainwritingissa osallistujat kirjoittavat ideansa paperilapuille hiljaisuudessa. Elektroninen brainstorming toimii samalla tavalla kuin brainwriting, mutta siinä apuna käytetään kynän ja paperin sijasta sähköistä järjestelmää. Brainwriting on alun perin kehitetty ratkaisuna brainstormingin ongelmaan, jossa ryhmässä ideoiden laatu ja määrä heikkenevät. (Michinov, 2012).

Kalagiros et al. (2019) pitävät fasilitaattorin aktiivista roolia keskeisenä brainstorming-tekniikassa ja esittävät tämän puutteellisen toteutuksen eräänä tärkeänä syynä brainstormingin osakseen saamalle kritiikille. Osbornin (1953) alkuperäisen kuvauksen mukaan fasilitaattori ei vastaa pelkästään kokouksen johtamisesta, vaan hänen tehtävänään on myös inspiroida ja kannustaa ideointiin. Kalagiros et al. (2019) kuvaavat artikkelissaan inspiroivaa fasilitointia. Fasilitaattorin kehonkieli, puheen viestit ja niistä välittyvä innostus asiaan vaikuttavat positiivisesti osallistujien riskinottoon, luovien ideoiden syntymiseen ja näin ollen myös ideoiden laatuun. (Kalargiros et al., 2019).

Johnson ja D’Lauro (2018) tutkivat idean valintaa brainstorming-session päätteeksi. Tutkimuksessa ryhmä valitsi aikaisessa vaiheessa esitetyn idean parhaaksi ideaksi. Kun ryhmää pyydettiin seuraavassa tutkimuksen vaiheessa valitsemaan toteutuskelpoisin idea, valituksi tuli useimmiten erittäin aikaisessa vaiheessa esitetty idea; kun taas etsittiin omaperäisintä ideaa, valitut ideat olivat tasaisemmin jakautuneet session koko ajalle. Osallistujien arviot

omaperäisyydestä ja toteutuskelpoisuudesta korreloivat negatiivisesti, mikä viittaa siihen, että omaperäisyyttä ja toteutuskelpoisuutta pidettiin toisensa poissulkevina ominaisuuksina. Tutkimuksen tuloksia voidaan käyttää hyväksi, kun suunnitellaan brainstorming-sessiota. (Johnson ja D'Lauro, 2018).

Tuplatiimi

Tuplatiimi on Helinin kehittämä menetelmä ongelmanratkaisuun ryhmässä. Helin (1987) listaa tuotekehityksen yhdeksi tuplatiimi-menetelmän käyttökohteista. Menetelmässä keskeisessä asemassa on pari-ideointi ja ristiinarviointi. Tuplatiimikokouksessa käytetään sekä yksilö-, pari-, että ryhmätyöskentelyä ja pyritään näiden avulla tuottamaan mahdollisimman hyviä ideoita. Kokous jakaantuu viiteen päävaiheeseen. Ne ovat käsiteltävän asian jäsennys, ideoiden ja ehdotusten tuottaminen, ideoiden ja ehdotusten seulonta, ratkaisumallien rakentaminen sekä jatkotoimenpiteistä sopiminen. (Helin, 1987, s. 102–103).

Tuplatiimimenetelmää käytettäessä yksi osallistujista on ohjaaja. Hänen tulee olla perehtynyt menetelmään ja hänellä on päävastuu kokouksen etenemisestä. Tuplatiimikokous alkaa käsiteltävän asian jäsennyksellä. Analyysi aloitetaan joko ongelmien tai tavoitteiden käsittelyllä. Ensin työskennellään yksin ja jokainen kirjoittaa ylös valitun lähestymissuunnan mukaan joko ongelmia tai tavoitteita. Tämän jälkeen ryhmän jäsenet jaetaan työpareiksi, ja parit valitsevat omista ylös kirjaamistaan ongelmista tai tavoitteista kaksi tai kolme tärkeintä. Tämän jälkeen kaikkien parien tulokset esitellään ja laitetaan näkyville, ja jokainen työpari merkkaa omasta mielestään kolme tärkeintä muiden parien ehdotusta. (Helin, 1987, s. 102–107).

Ideoiden ja ehdotusten tuottaminen etenee samankaltaisesti kuin käsiteltävän asian jäsennys, lähtien ensin liikkeelle yksilötyöskentelystä ja edeten seuraavaksi parityöskentelyyn. Yksilötyöskentelyn tavoitteena on, että jokainen keksii viisi ideaa. Parityöskentelyvaiheessa tavoitteena on löytää ja laittaa esille viisi sellaista ideaa, joita muut parit eivät ole vielä kirjanneet ylös. (Helin, 1987, s. 109–110).

Ideoiden seulontavaiheessa jokainen työpari valitsee ensin omista ideoistaan kaksi parasta ja esittelee omat ideansa toisille osallistujille aloittaen itse parhaiksi kokemistaan. Seuraavaksi suoritetaan ensimmäinen ristiinarviointi, jossa jokainen työpari valitsee toisten esittämistä ideoista mielestään neljä parasta. Toisessa ristiinarvioinnissa työparit kertovat ja

perustelevat omat valintansa ja valitsevat muiden perustelut kuultuaan vielä kaksi lisävalintaa. Viimeisellä valintakierroksella työparit valitsevat kolme mielestään kehityskelpoisinta ideaa kaikkien ideoiden joukosta, ja näistä enintään yksi saa olla parin oma idea. (Helin, 1987, s. 111–113).

Kokouksen neljännessä vaiheessa rakennetaan ratkaisumalleja valittujen parhaiden ideoiden pohjalta. Työparit hajotetaan ja osallistujat jaetaan uudelleen pienryhmiin, joissa työstetään yhdessä ratkaisumalleja. Sen jälkeen samankaltaisia ratkaisumalleja pyritään yhdistämään. Kokouksen lopussa seuraa viides vaihe eli jatkotoimista sopiminen. Tässä vaiheessa sovitaan muun muassa ehdotettujen ideoiden jatkotyöstämisestä, vastuuhenkilöistä ja aikataulusta. (Helin, 1987, s. 115–119).

Tuplatiimi-menetelmästä ei löydy paljoa tieteellistä tutkimusta, mutta menetelmän periaatteita ja sen käyttöä innovaation alkuvaiheessa voidaan tutkia myös tuoreen tieteellisen tiedon valossa. Muun muassa tutkimukset yksilö- ja ryhmätyöskentelyn yhdistämisestä tukevat tuplatiimin käyttöä ideoinnissa. Tuplatiimissä yksilö- ja ryhmätyöskentely vuorottelevat. Korde ja Paulus (2017) esittävät tutkimuksessaan, että erityisesti ryhmätyöskentelyjakson jälkeen yksilötyöskentelyn tulokset paranevat, sillä ryhmätyöskentelyssä yksilöille syntyy assosiaatioita, joita he pystyvät hyödyntämään yksilötyöskentelyn aikana.

Tuplatiimiä on kuvattu samankaltaiseksi Nominal Group Technique -nimisen kansainvälisen tekniikan kanssa. Nominal Group Techniquessa osallistujat ideoivat ensin hiljaa yksin, seuraavaksi ideat luetaan anonyymisti ääneen ja niistä äänestetään ryhmässä (Delbecq & Van de Ven, 1971). Menetelmää käytetään laajasti monilla aloilla ongelmien ratkaisuun. Nominal Group Techniquesta poiketen tuplatiimissä ryhmä- ja yksilötyöskentelyn lisäksi tärkeässä osassa ovat parityöskentelyvaiheet.

ABC-analyysi

ABC-analyysi on tunnettu erityisesti inventaarion hallinnasta (Flores & Whybark, 1986), mutta sitä on käytetty laajasti myös ideoiden valintaan innovaatioprosessin alkuvaiheessa (Rebernik & Bradač, 2008). Menetelmässä asiat, eli tässä tapauksessa ideat, järjestetään A-, B-, ja C-luokkiin. (Flores & Whybark, 1986) Ideat järjestetään luokkiin sen perusteella, kuinka tärkeiksi ne arvioidaan. A-luokan ideat ovat erittäin tärkeitä, B-luokan ideat melko tärkeitä ja C-luokan ideat vähemmän tärkeitä. (Rebernik & Bradač, 2008). Menetelmä

soveltuu tilanteisiin, joissa useampia ideoita halutaan laittaa tärkeysjärjestykseen. Ohjelmistokehityksen alalla esimerkiksi uusien ominaisuusideoiden priorisointi on tällainen tilanne.

Ideoiden arvottaminen

Ideoista parhaan valitseminen tapahtuu yrityksissä useimmiten pisteyttämällä ideat tai laittamalla ne paremmuusjärjestykseen. Kun ideoita pisteytetään, kaikki ideoiden arviointiin osallistuvat antavat kullekin idealle numeerisen arvon tietyllä välillä, esimerkiksi 0–10. Paremmuusjärjestykseen laitettaessa arviointiin osallistuvat henkilöt laittavat ideat paremmuusjärjestykseen parhaasta huonoimpaan. Sen jälkeen kaikkien osallistujien rankit summataan ja suurimman pistemäärän saanut idea valitaan. (Cui et al., 2019).

Cuin et al. (2019) tutkimus viittaa siihen, että paremmuusjärjestykseen laittaminen ei toimi yhtä tehokkaasti kuin jokaisen idean numeerinen arvioiminen, mahdollisesti siksi, että arvioijat käyttävät rankatessa vähemmän aikaa kunkin idean ja sen mahdollisuuksien ja ongelmien tarkkaan miettimiseen.

Ideoiden arviointiin käytettävät attribuutit ja niiden valinta

Ideoita voidaan arvioida erilaisten ennalta määriteltyjen ominaisuuksien perusteella, joita hyvältä idealta odotetaan. Näiden ominaisuuksien eli attribuuttien valinnassa täytyy olla tarkka, jotta attribuutit oikeasti kuvaavat ideoiden laatua ja yritykselle hyödyllisiä piirteitä. Ideoiden arviointiin käytetään yleensä useita attribuutteja, joihin voivat kuulua esimerkiksi uutuus, käytännöllisyys, haluttavuus ja kaupallistettavuus (Kudrowitz & Wallace, 2013).

Eräs yrityksissä usein käytetty mittari ideoiden arvioinnin avuksi on NUF (novel, useful, feasible). Tiimi arvioi ideat uutuusarvon, käytettävyyden ja toteuttamiskelpoisuuden perusteella. Jokainen kategoria arvioidaan numeroarvosanalla, esimerkiksi 1–10. Kaikkien kolmen kategorian pisteet lasketaan yhteen, ja ideoita voidaan vertailla keskenään yhteenlasketun tuloksen perusteella. (Kudrowitz & Wallace, 2013).

Tarkistuslistat

Useat yritykset käyttävät yksinkertaisia tarkistuslistoja apuna ideoiden arvioimiseen. Tarkistuslistat sisältävät ominaisuuksia tai kriteereitä, joita idean tulisi sisältää läpäistäkseen testin. Tarkistuslistat sisältävät joukon kyllä/ei -kysymyksiä, ja jokaisen projektin täytyy joko läpäistä kaikki kysymykset tai ennalta määritelty prosentuaalinen osuus kysymyksistä, jotta idean toteuttamista harkitaan. Menetelmää voidaan käyttää apuna joko päätettäessä, otetaanko tietty idea jatkojalostukseen tai järjestettäessä useampia ideoita paremmuusjärjestykseen. (Cooper et al., 2001). Tarkistuslistat ovat yksinkertaisia ja niiden käyttö on helppo oppia, mutta haasteena on oikeiden kysymysten valinta.

Pugh-matriisi

Niin kutsuttu Pugh-matriisi on Stuart Pugh'n (1991) kehittämä menetelmä, jonka avulla voidaan vertailla keskenään saman ongelman vaihtoehtoisia ratkaisuja ja valita niistä paras. Alussa valitaan verrokki, johon ideoita verrataan. Jos kehitetään parannusideoita jo olemassa olevaan tuotteeseen tai konseptiin, tämä valitaan verrokiksi. Matriisissa jokaiseen sarakkeeseen tulee yksi idea ja jokaiselle riville kriteeri, jonka mukaan ideoita vertaillaan. Jokaista ideaa vertaillaan verrokkiin ennalta määriteltyjen kriteerien perusteella. Jos idea täyttää vaatimuksen verrokkia paremmin, matriisiin merkitään sille kohdalle +, jos huonommin, matriisiin merkitään -, ja jos idean ja verrokin välillä ei ole eroa, merkitään S. (Pugh, 1991, s. 74–79). Käytännössä usein käytetään myös sellaista sovellusta matriisista, jossa vertailu merkataan matriisiin numeerisesti arvoilla -1, 0 ja +1. Lopussa kunkin idean saama pistemäärä lasketaan yhteen. Eri kriteereitä voidaan myös painottaa ja kertoa rivin pistemäärä tällä painokertoimella, mikäli jotkut kriteerit ovat toisia tärkeämpiä.

Tämän jälkeen Pugh (1991) suosittelee tutkimaan vahvojen ideoiden huonoja ominaisuuksia ja etsimään niille ratkaisuehdotuksia. Jos löydetään keino, jonka avulla verrokkia heikompia ominaisuuksia saadaan parannettua heikentämättä muita ominaisuuksia, tämä uusi idea kirjoitetaan ylös matriisiin. Seuraavaksi sama toistetaan heikoilla ideoilla. Lopulta esiin noussutta parasta ideaa voidaan lähteä jatkokehittämään. (Pugh, 1991, s. 78).

Menetelmä soveltuu esimerkiksi uusien tiettyä asiakastarvetta varten ideoitujen toiminnallisuuksien vertailemiseen keskenään. Pugh (1991, s. 76) painottaa, että menetelmässä

kaikkien ideoiden tulee olla ratkaisuja samaan ongelmaan. Matriisi ei siis sovellu esimerkiksi satunnaisten tuoteinnovaatioiden vertailemiseen.

Strategic buckets -menetelmä

Strategic buckets -menetelmää käytetään hyväksi tuoteportfolion hallinnassa. Menetelmä auttaa yritystä valitsemaan omaan innovaatiostrategiaansa sopivat tuotteet. Uudet innovaatiot jaetaan erikokoisiin ”astioihin”, jotka kuvaavat eri tyylisiä innovaatioita. Jaottelu voi olla esimerkiksi seuraava: uudet tuotteet, parannukset vanhoihin tuotteisiin, uudet teknologiat ja kustannusvähennykset. Näiden astioiden suhteellinen koko kuvaa sitä, kuinka paljon resursseja yrityksen strategiassa niille on määritelty. Kunkin astian sisällä siihen kuuluvat ideat arvioidaan ja järjestetään keskenään paremmuusjärjestykseen. Toteutettavaksi otetaan projekteja eri astioista strategian mukaan. (Chao & Kavadias, 2008).

Yleisesti käytettävät erityisesti taloudelliset menetelmät tuoteportfolion hallintaan jättävät usein innovatiivisia uuden teknologian ideoita vähemmälle huomiolle, sillä tällaisissa tuotteissa riski on suuri ja niistä saatava tuotto voi tulla vasta pitkällä aikavälillä. Strategic buckets puolestaan ottaa myös korkeamman riskin projekteja mukaan portfolioon sopivassa suhteessa yrityksen strategiaan verrattuna. (Chao & Kavadias, 2008).

Strategic buckets -menetelmää suositellaan käyttämään, jos eri ideoita ja projekteja ei voida tyydyttävästi vertailla keskenään niiden erilaisuuden takia. Toisaalta menetelmän käyttö ei hyödytä yritystä, mikäli vaihtoehtoisten projektien määrä on pieni tai projektit ovat hyvin samankaltaisia. (Santiago & Soares, 2020). Yrityksen aloittaessa ohjelmistoliiketoimintaa ideoiden ja toteutettavien projektien määrä on todennäköisesti melko pieni, jolloin menetelmän käyttö ei ole perusteltua. Toisaalta myöhemmin liiketoiminnan sekä projektien määrän kasvaessa menetelmä voi osoittautua hyödylliseksi. Cooperin et al. (2001) tutkimuksen mukaan parhaiten menestyvät yritykset, jotka nojaavat toteutettavien projektien valinnassa eniten liiketoimintastrategiaan perustuviin keinoihin. Strategic buckets on eräs tällaisista keinoista.

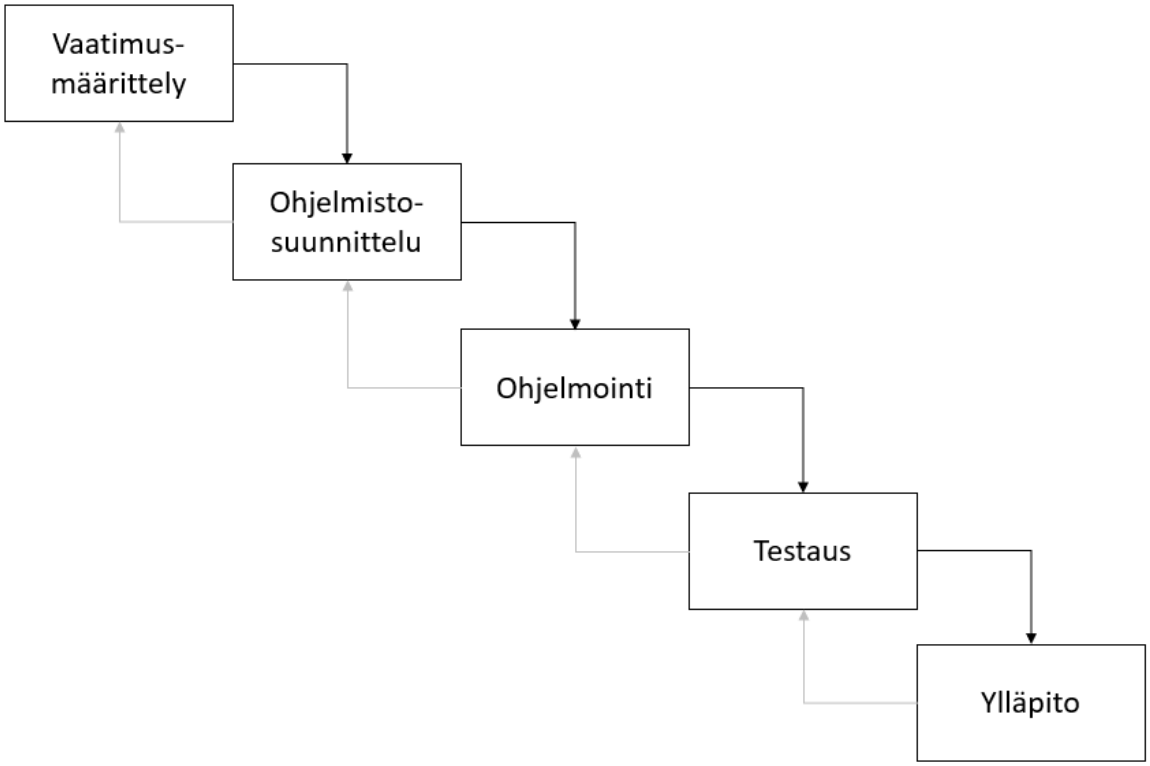
Strategic buckets-menetelmän lisäksi tai rinnalle tuoteportfolion hallintaan on monia muitakin menetelmiä. Esimerkiksi portfoliokartta on eräs visuaalinen työkalu, joka auttaa hahmottamaan organisaation tuoteportfoliota (Ulonska & Welo, 2014). Tässä tutkimuksessa ei kuitenkaan perehdytä tarkemmin muihin tuoteportfolion hallintamenetelmiin.

4 Tuotekehitys ja sitä tukevat menetelmät

Tuotekehitys on innovaatioprosessin alkuvaiheen jälkeen seuraava vaihe. Tässä vaiheessa idea tuotteesta on jo olemassa. Hyviä käytäntöjä onnistuneeseen tuotekehitysvaiheen aloittamiseen on, että projekti sopii yhteen yrityksen strategian kanssa, käyttäjien tarpeet on eritelty ja dokumentoitu tarkasti, tuotekonsepti on määritelty hyvin, on olemassa hyvä projektisuunnitelma, ylempi johto tukee projektia ja projektitiimi on motivoitunut. (Koen et al., 2001).

Tässä luvussa esitellään erilaisia ohjelmistotuotannon projektinhallintamenetelmiä. Aluksi perehdytään lyhyesti perinteisiin projektinhallintamenetelmiin ja sen jälkeen käsitellään ketteriä menetelmiä. Käsiteltävät menetelmät on valittu perustuen niiden yleisyyteen alalla sekä suosion kehitykseen 2000-luvulla. Menetelmien yleisyyttä on arvioitu kirjallisuuteen perustuen. Aiemmissa tutkimuksissa (Vijayasarathy & Butler, 2016) tärkeimmiksi menetelmiksi ovat nousseet vesiputous, Scrum, testivetoinen kehitys sekä Kanban. Digital.ai:n (2021) toteuttaman ketteriin menetelmiin keskittyvän kyselytutkimuksen perusteella erityisesti Scrum on kasvattanut suosiotaan viime vuosina, kun taas esimerkiksi 2000-luvun alkupuolella suositun eXtreme programming -menetelmän suosio on hiipunut merkittävästi. Lisäksi raportin (digital.ai, 2021) pohjalta tässä luvussa käsiteltäväksi nostetaan DevOps. Luvun lopussa käsitellään lyhyesti ohjelmistotuotannon hybridimalleja eli useampien ohjelmistotuotannon projektinhallintamenetelmien yhdistelmiä. Suurta osaa tässä luvussa esitellyistä menetelmistä voidaan käyttää koko innovaatioprosessin ajan innovaatioprosessin alkuvaiheesta asti.

4.1 Vesiputousmalli



Kuva 2. Vesiputousmalli (mukaillen Fagarasan et al., 2021, s. 3)

Vesiputousmalli on perinteinen malli ohjelmistoprojektien hallintaan. Vesiputousmallin mukaisesti toteutettava projekti alkaa järjestelmävaatimusten ja ohjelmistovaatimusten määrittelyllä. Sen jälkeen suunnitellaan ohjelmisto. Seuraavaksi toteutetaan ohjelmisto suunnitelman mukaisesti. Viimeiset kaksi vaihetta vesiputousmallissa ovat testaus ja ylläpito. Mallissa edetään siis järjestelmällisesti eteenpäin niin, että edellisen vaiheen ollessa valmis siirrytään seuraavaan vaiheeseen. (Fagarasan et al., 2021).

Vesiputousmalli perustuu Winston Roycen artikkeliin, jossa Royce kertoo henkilökohtaisia näkemyksiään suurten ohjelmistoprojektien hallinnasta (Hohl et al., 2018). Royce (1987) esittelee artikkelissaan mallin, joka kuvaa ohjelmistotuotannon prosessia alusta loppuun asti. Roycen esittelemää mallia on myöhemmin kutsuttu vesiputosmalliksi. Royce pitää mallissa tärkeänä iterointia taaksepäin, mutta malli on usein ymmärretty väärin.

Vesiputousmallia on kritisoitu erityisesti sen lineaarisen etenemisen takia. On kuitenkin olemassa projekteja, joihin vesiputousmalli sopii hyvin. Esimerkiksi pieniin sisäisiin ohjelmiin, joiden toteuttaminen vaatii vain vähän aikaa ja henkilöstöä, vesiputousmallin soveltaminen voi olla järkevää. Vesiputousmalli toimii, mikäli vaatimukset ohjelmistolle voidaan määrittää tarkasti jo projektin alussa. Vijaysarathyn ja Butlerin (2016) tekemästä kyselytutkimuksesta käy ilmi, että perinteisiä menetelmiä käytetään erityisesti kriittisissä suuren budjetin projekteissa. Kritiikistä huolimatta vesiputousmallia käytetään edelleen paljon ja se on eräs käytetyimmistä ohjelmistotuotannon projektinhallintamenetelmistä (Vijayasarathy & Butler, 2016).

4.2 Ketterät menetelmät

Ketterät menetelmät ovat vaihtoehto edellä esitellylle vesiputousmallille. Ketterien menetelmien ydinajatuksena on, että ohjelmistotuotetta kehitetään lyhyissä iteraatioissa. Jokaiselle iteraatiolle määritellään sen alussa tavoitteet. Iteraation lopussa asiakkaalle toimitetaan toimiva inkrementti eli versio tuotteesta. Joka iteraatiossa tuotteeseen lisätään toiminnallisuuksia. Useiden pienten julkaisuiden ansiosta yrityksellä on mahdollisuus saada asiakkaalta palautetta koko tuotekehitysprosessin ajan sekä reagoida asiakkaan muuttuviin toiveisiin joustavammin kuin perinteisiä ohjelmistokehitysmenetelmiä käyttäen. Ketterät menetelmät ovat kasvattaneet suosiotaan ohjelmistoyrityksissä perinteiseen vesiputousmalliin verrattuna, ja viimeisten 20 vuoden aikana ketterien menetelmien mukaisia projekteja on valmistunut kolme kertaa enemmän kuin vesiputousmallilla toteutettuja projekteja. (Fagarasan et al., 2021).

17 alan ammattilaista kerääntyi yhteen vuonna 2001 tavoitteena määritellä ”paremman”, kevyemmän ohjelmistokehityksen menetelmiä ja arvoja. Tämän tapaamisen seurauksena syntyi ketterän ohjelmistokehityksen perustana laajasti pidetty ”Manifesto for Agile Software Development”. (Hohl et al., 2018). Manifestissa (Beck et al., 2001) kuvataan ketterän ohjelmistokehityksen arvoja seuraavasti:

“Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

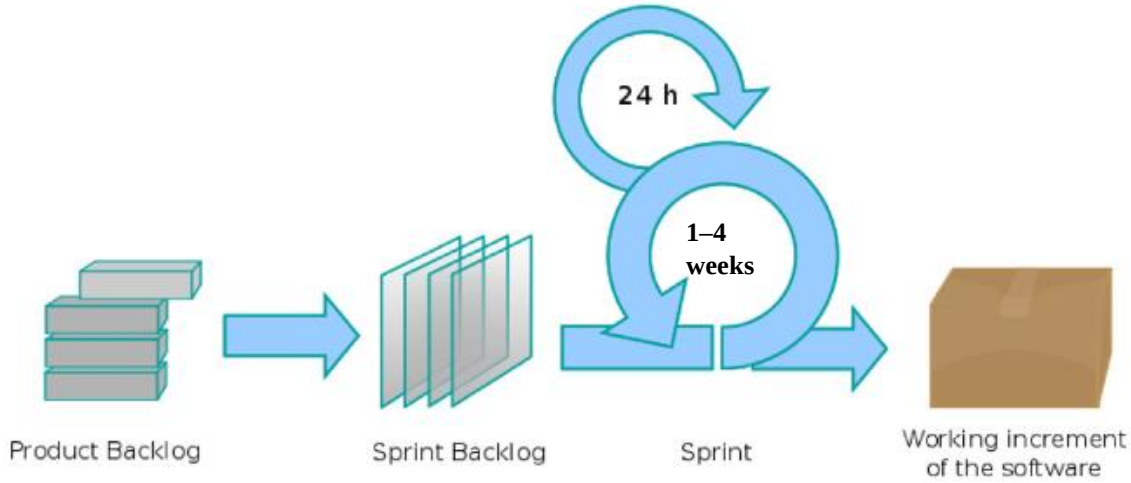
***Yksilöitä ja kanssakäymistä** enemmän kuin menetelmiä ja työkaluja*
***Toimivaa ohjelmistoa** enemmän kuin kattavaa dokumentaatiota*

Asiakasyhteistyötä *enemmän kuin sopimusneuvotteluja*
Vastaamista muutokseen *enemmän kuin pitäytymistä suunnitelmassa*
Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän."

Hohl et al. (2018) haastattelivat manifestin luonnissa mukana olleita henkilöitä heidän näkemyksistään manifestiin sekä yleisesti ketterään ohjelmistokehitykseen liittyen. Tutkimuksesta käy ilmi, että manifestin tekijät pitävät tärkeänä mukautumiskykyä sekä ketterien menetelmien soveltamista tapauskohtaisesti jokaisen yrityksen tarpeisiin. Yleisenä kompastuskivenä ketterien menetelmien käytössä erityisesti nykypäivänä he näkivät sen, että monissa yrityksissä ketteriä menetelmiä yritetään käyttää valmiina malleina ja lähinnä "trendikkyyden" takia, ymmärtämättä kunnolla ketteryyden perusajatusta. (Hohl et al., 2018).

Ketteriä menetelmiä voidaan soveltaa myös innovaatioprosessin alkuvaiheeseen. Monet innovaatioprosessin alkuvaiheen ongelmat ovat samankaltaisia kuin ohjelmistokehityksen ongelmat, ja usein saman menetelmän käyttäminen innovaatioprosessin alkuvaiheesta tuotekehitykseen asti auttaa tiedon kulussa prosessin vaiheiden välillä sekä menetelmien omaksumisessa organisaatiossa.

4.2.1 Scrum



Kuva 3. Scrum. (mukaillen Lei et al., 2017, s. 61)

Sutherland ja Schwaber kehittivät Scrumin ketterän ohjelmistotuotannon malliksi, ja Schwaber kuvasi mallia ensimmäisen kerran virallisesti vuonna 1995. Schwaber (1997) mainitsee

artikkelissaan vesiputousmallin suureksi haasteeksi mallin lineaarisen luonteen, eli kuinka se esittää ohjelmistokehitysprosessin lineaarisena jatkumona alusta loppuun. Vesiputousmalli ei anna kunnollisia ratkaisuja siihen, kuinka reagoida odottamattomalla tavalla muuttuviin tilanteisiin ohjelmistokehityksen kuluessa. Schwaber havainnollistaa artikkelissaan, kuinka ohjelmisto-organisaatioiden joustava reagointi ennustamattomiin muutoksiin ja tämän myötä todennäköisyys projektin onnistumiselle heikkeni rajusti mitä monimutkaisemmasta projektista oli kysymys (niin vesiputous-, spiraali- kuin iteratiivisissakin prosesseissa).

Tähän ongelmaan Scrum pyrkii tarjoamaan ratkaisuja. Scrumin tavoitteena on olla mahdollisimman joustava muutoksille. Scrumissa ohjelmistokehitys tapahtuu ennalta määritellyn mittaisissa sprinteissä. Tyypillinen sprintin pituus on yhdestä neljään viikkoa. (Schwaber, 1997). Tuotteelle vaadittavat ominaisuudet sekä toivottavat ominaisuudet ovat priorisoituna tuotebacklogissa. Scrum-tiimi suunnittelee seuraavan sprintin tavoitteet suunnittelutapaamisessa, ja siellä sovitut tehtävät kirjataan sprintin backlogiin. Joka päivä pidetään lyhyt, noin vartin mittainen päivittäispalaveri. Päivittäispalaverien tavoitteena on informoida muita tiimin jäseniä omasta etenemisestä, seuraavan päivän tavoitteista sekä kohtaamistaan haasteista. (Lei et al., 2017).

Sprintin lopussa ohjelmistotuotteesta tulee olla ajettavissa oleva versio. Tiimit kokoontuvat sprintin katselmointiin, jossa käydään läpi sprintin aikana tuotteeseen tehdyt muutokset, lisätään tehtäviä backlogiin seuraavaa sprinttiä varten sekä arvioidaan riskejä ja ratkaistaan ongelmia. (Schwaber, 1997). Katselmoinnissa käydään läpi, mitä kukin on saanut aikaan kuluneen sprintin aikana. Katselmoinnin jälkeen ja ennen seuraavan sprintin alkua pidetään retrospektiivi, jossa tarkoituksena on käsitellä sitä, kuinka sprintti onnistui esimerkiksi kommunikaation, henkilöstöresurssien ja prosessien näkökulmasta sekä löytää kehitysehdotuksia seuraavaa sprinttiä ajatellen. (Lei et al., 2017).

Scrum-tiimiin kuuluu Scrum master, tuotteen omistaja sekä ohjelmistokehitystiimi. Tiimin ideaalikoko on seitsemän henkilöä. Scrum masterin tehtäviin kuuluu projektinhallinta, ja roolissa työskentelevä henkilö huolehtii siitä, että tiimillä on kaikki tarvittavat resurssit ja informaatio. Tuotteen omistaja on yhteydessä asiakkaan kanssa ja kommunikoi asiakastarpeet kehitystiimille. Tuotteen omistajan tehtävänä on olla päävastuussa määrittämässä, mitkä ovat tärkeimpiä arvoa luovia elementtejä tuotteelle. (Gloger, 2010). Unger-Windeler

et al. (2021) kuvaavat tuotteen omistajan roolia kommunikoijan rooliksi, joka on yhteydessä eri henkilöihin tiimin sisäisesti ja sen ulkopuolella.

Scrum skaalautuu hyvin myös suuremmalle organisaatiolle. Erilaisista skaalautumisvaihtoehdoista löytyy kaupallisia koulutuksia. Scrumin päälle suunniteltuja skaalautuvia agilemalleja ovat esimerkiksi Scrum of Scrums (SoS), Scaled Agile Framework (SAFe) ja Large-Scale Scrum (LESS). (Ebert & Paasivaara, 2017).

Annosi et al. (2016) tunnistavat tutkimuksensa pohjalta Scrum-menetelmän heikkouksia. Tulokset osoittavat, että Scrum voi aiheuttaa ajallista painetta ja tämän myötä ylimääräistä stressiä työntekijöille. Menetelmässä on tärkeää tiimien itseohjautuvuus ja ryhmätyöskentely. Backlog ja usein toistuvat palaverit, joissa tarkastellaan projektin etenemistä, voivat yhdessä aiheuttaa painetta suorittaa tehtäviä nopeasti. Tiimissä voi myös esiintyä negatiivista sisäistä valvontaa tehtävien etenemisestä, mikä saattaa heikentää työntekijöiden tyytyväisyyttä. Paine saada backlogissa olevia tehtäviä valmiiksi vähentää useissa organisaatioissa työntekijöiden mielenkiintoa kouluttautumiseen, vaikka pitkällä aikavälillä se on tärkeää henkilöstön kompetenssin ylläpitämiselle ja näin ollen myös projektien laadulle. (Annosi et al., 2016). Gaete et al. (2021) mainitsevat, että Scrum on usein jäykkä muihin ketteriin menetelmiin verrattuna, sillä menetelmään sisältyy paljon tiukkoja sääntöjä.

4.2.2 Kanban

Kanban-projektinhallintamenetelmä on lähtöisin Japanista ja perustuu autovalmistaja Toyotan tuotantomenetelmään. Myöhemmin Kanban on otettu laajasti käyttöön myös ohjelmistotuotannon alalla. (Lei et al., 2017). Menetelmän pääperiaatteena on juuri oikeaan aikaan tapahtuva tuotanto ja toimitus. Kanban kertoo tarkasti, mikä työ on tehtävä ja milloin se on tehtävä. Se pyrkii hyödyntämään yrityksen käytössä olevia resursseja, kuten työntekijöiden työaikaa, mahdollisimman tehokkaasti. (Sugimori et al., 1977). Ohjelmistotuotannossa Kanbanin tärkeänä tavoitteena on minimoida prosessiin sisältyvä odotusaika ja muu hukka sekä priorisoida työtehtäviä, jotta tärkeimmät tehtävät tulevat varmasti tehtyä (Lei et al., 2017). Kirovska ja Koceski (2015) mainitsevat seitsemän Kanbanin peruspilaria ohjelmistokehityksessä. Ne ovat hukan minimointi, oppimisen tehostaminen, päätösten tekeminen

mahdollisimman myöhäisessä vaiheessa, toimitukset mahdollisimman aikaisessa vaiheessa, tiimin voimaannuttaminen, laadun rakentaminen ja kokonaiskuvan näkeminen.

Kanban-menetelmässä käytetään backlogia. Tehtävät esitetään visuaalisesti korttiseinällä. Visuaalinen esitystapa tuo selkeästi esille, kuinka projekti etenee. Kaikki projektin toteuttamiseen tarvittavat vaiheet identifioidaan, ja tarvittavat tehtävät kirjoitetaan korteille ja sijoitetaan backlogiin. Backlogista otetaan tehtäviä järjestyksessä. Tehtävät kulkevat vaiheesta toiseen korttiseinällä. Keskeneräisiä tehtäviä rajoitetaan. Kussakin vaiheessa voi olla enimmillään ennalta päätetty määrä keskeneräisiä tehtäviä ja uusia tehtäviä voidaan ottaa vastaan vain, jos kyseisellä hetkellä keskeneräisiä tehtäviä on tätä enimmäismäärää vähemmän. Muussa tapauksessa ensin täytyy saada yksi aiemmista tehtävistä valmiiksi ennen kuin otetaan uusia tehtäviä vastaan aiemmasta vaiheesta. (Lei et al., 2017). Eri vaiheita voivat olla esimerkiksi kehitys, testaus ja implementointi. Yritykset käyttävät nykypäivänä korttiseinän luomiseen useimmiten ohjelmistoa, jonka avulla backlogia hallitaan.

Kanbanin avulla toteutetussa projektissa ei ole ennalta määriteltyjä rooleja. Project managerin rooliin voidaan halutessa nimittää yksi henkilö, mutta myöskään project managerin rooli ei ole välttämätön. Toisin kuin Scrumissa, projektin tehtävillä ei ole erikseen omistajia, vaan kaikki tehtävät ovat tiimin yhteisiä. Kanban toimii hyvin myös suurissa tiimeissä. (Fagarasan et al., 2021).

Lei et al. (2017) tutkivat Kanbanin ja Scrumin tehokkuutta ohjelmistokehitysprojekteissa. Tutkimus vertailee näiden kahden menetelmän toimivuutta aikatauluun, laajuuteen, budjettiin, riskeihin, resursseihin sekä laatuun liittyen kyselytutkimuksen avulla. Tulokset viittaavat siihen, että projektin aikataulu ja muutoksiin vastaaminen toimivat paremmin Kanban-menetelmällä tehdyissä projekteissa. (Lei et al., 2017). Scrumista poiketen Kanban ei toimi sprinteissä vaan aikataulut elävät jatkuvasti tarpeen mukaan. Menetelmä soveltuu erityisen hyvin pieniin työtehtäviin sekä ylläpitoon. (Fagarasan et al., 2021).

Kanban suunniteltiin alun perin erityisesti Toyotan tarpeisiin. Mallilla on joitakin rajoitteita, joita on myös tutkittu kirjallisuudessa. Esimerkiksi rajusti vaihteleva kysyntä tai epätäsmälliset prosessit voivat aiheuttaa sen, että Kanban ei toimi toivotun kaltaisesti kyseisessä organisaatiossa. (Braglia et al., 2020) Ohjelmistotuotannon kontekstissa Kanbanin ongelmina on mainittu aikatauluihin liittyvä ennalta-arvaamattomuus, minkä takia aikataulutus voi olla haastavaa. Projektin aikataulu voi muuttua tarpeen mukaan. (Fagarasan et al., 2021).

Toisaalta tämä tekee aikataulusta joustavamman. Gaete et al. (2021) mainitsevat Kanbanin heikkoutena sen, että keskeneräisen työn rajoittaminen voi johtaa pullonkauloihin.

4.2.3 Testivetoinen kehitys

Siinä työskennellään lyhyissä sykleissä ja testit kirjoitetaan ennen varsinaista ohjelmakoodia. Periaatteena on kirjoittaa testit, jotka ohjelman tulisi läpäistä ja kirjoittaa sen jälkeen ohjelmakoodi, jota muokataan niin kauan, kunnes se menee läpi kaikista kirjoitetuista testeistä. Tämän jälkeen jatketaan uusien testien kirjoittamisella ja ohjelman muokkaamisella näiden mukaan. (Karac & Turhan, 2018).

Testivetoinen kehitys on alun perin Ward Cunninghamin eXtreme programming -malliin liittyvä menetelmä. Sitä on myöhemmin alettu käyttää myös muiden erityisesti ketterien projektinhallinnan mallien kanssa. Testivetoinen kehitys ei ole pelkkä testaustekniikka. Menetelmän nimi viittaa siihen, kuinka testivetoinen kehitys ohjaa analyysi-, design- ja ohjelmointipäätöksiä. Kirjoitettavat testit auttavat hahmottamaan myös ohjelman vaatimuksia. (Janzen & Saiedian, 2005).

Testivetoinen kehitys on aiheuttanut paljon mielipiteitä sekä puolesta että vastaan, niin akateemisessa maailmassa kuin myöskin yritysmaailmassa. Testivetoiseen kehitykseen liittyvät tutkimukset antavat ristiriitaisia tuloksia. Suuressa osassa tutkimuksista kirjoitetun ohjelmakoodin laadun on havaittu paranevan testivetoisen kehityksen avulla. Testivetoinen kehitys ohjaa kehittäjiä kirjoittamaan laajemmat testisetit, joiden avulla myös bugien tunnistaminen helpottuu. (Munir et al., 2014). Toisaalta joissakin tutkimuksissa laadun on raportoitu pysyneen jotakuinkin samana (Baldassarre et al., 2021) tai jopa heikentyneen (Wilkerson et al., 2012).

Ohjelman ulkoisen laadun lisäksi testivetoisen kehityksen vaikutuksia projekteihin on tutkittu myös monen muun mittarin avulla. Siinä missä suurin osa tutkimuksista raportoi ulkoisen laadun parantuneen, useat tutkimustulokset viittaavat siihen, että tehokkuus laskee yrityksissä testivetoista kehitystä käytettäessä. Joissakin tapauksissa testivetoinen kehitys voi viedä runsaasti enemmän kehittäjien aikaa kuin perinteinen test last -metodi. (Bissi et al., 2016). Kasvavan ajankäytön lisäksi muun muassa kehittäjien puutteelliset tiedot testivetoisesta kehityksestä tai ohjelmistotestauksesta hankaloittavat usein menetelmän käyttöönottoa

yrityksessä (Causevic et al., 2011). Testivetoinen kehitys voi olla vaikea ajatusmalli omaksua monille ohjelmistokehittäjille. Tämän on tosin arveltu olevan seurausta pinttyneistä toimintamalleista. Santosin et al. (2021) tutkimus vertailee ohjelmistotekniikan ammattilaisten sekä opiskelijoiden koodin laatua test-last ja test-first -metodien mukaan. Sekä opiskelijat että ammattilaiset olivat aloittelijoita testivetoisessa kehityksessä. Siinä missä ammattilaisten koodin laatu heikkeni test-first -menetelmällä, opiskelijoilla samanlaista ilmiötä ei ollut havaittavissa. (Santos et al., 2021).

4.3 DevOps

Erityisesti viimeisten reilun kymmenen vuoden aikana yrityksissä on kiinnitetty huomiota ongelmiin IT-osastojen tarkkaan vastuunjakoon liittyen. Ohjelmistokehitys ja ylläpito toimivat usein erillään toisistaan, jolloin yksikköjen välinen tiedonkulku ja yhteistyö heikkenevät. Tämä aiheuttaa monesti viivästyksiä ohjelmistojen julkaisuissa sekä laadun heikkenemistä. DevOpsin tavoitteena on ratkaista näitä ongelmia organisaatiossa. Se on menetelmä, jossa tärkeää on yhteistyö ohjelmistokehityksen ja ylläpidon välillä. Ylläpidolla tarkoitetaan tässä kontekstissa ylläpidon IT-palvelutoimintoja, eli käytännössä ylläpidon työntekijät huolehtivat asiakkaan IT-ympäristöstä ja sen toimivuudesta. (Díaz et al., 2021). Tämän tiiviin yhteistyön lisäksi DevOpsin toinen kulmakivi on automaatio (Riungu-Kalliosaari et al., 2016). DevOpsille ominaista on automaation hyödyntäminen laajasti koko ohjelmiston elinkaaren ajan. Tähän kuuluvat muun muassa testiautomaatio, ohjelmiston suorituskyvyn mittaaminen, lokitiedostojen hallinta ja kehitysinfrastruktuurin hallinta.

DevOpsia käyttävillä yrityksillä tavoitteena on menetelmän avulla saada tehtyä julkaisuja aikaisemmin, parantaa tuotteen laatua, parantaa yhteistyötä organisaation sisällä sekä lisätä tehokkuutta. Nopea julkaisutahti ja parantunut kommunikaatio organisaation sisällä mahdollistavat myös nopean palautteen ohjelmistoversioista. (Díaz et al., 2021). DevOpsin periaatteet liittyvät yrityskulttuuriin, automaatioon, mittaamiseen ja jakamiseen. Liiketoimintoja ja organisaation tehokkuutta mitataan. Mittaamalla myös pyritään ymmärtämään koodia ja ympäristöä paremmin. (de França et al., 2016).

DevOpsin perusteina toimivat ”kolme polkua”. Ensimmäisen polku mahdollistaa työn nopean kulkemisen ohjelmistokehityksestä ylläpitoon. Yksi mahdollinen keino päästä

tavoitteeseen on esimerkiksi ylläpidon työntekijöiden integroiminen ohjelmistokehitystiimeihin. Toinen polku liittyy palautteenantoon. Kaikissa prosessin vaiheissa palautteen tulee kulkea ohjelmistokehityksen ja ylläpidon välillä. Tähän polkuun liittyvät menetelmät keskittyvät operaatioissa kerätyn tiedon välittämiseen palautteena ohjelmistokehittäjille. Kolmas polku on jatkuva oppiminen ja uuden kokeileminen. Tämä polku liittyy erityisesti organisaatiokulttuuriin. DevOpsin mukaisessa organisaatiokulttuurissa panostetaan jatkuvaan oppimiseen ja palautteeseen, ja kulttuurin tavoitteena on olla virheitä salliva ja kannustaa uusien ideoiden testaamiseen ja toteuttamiseen. (König & Kugel, 2019).

Teknisessä mielessä DevOpsissa hyödynnetään jaettuja sähköisiä palveluita osastojen välillä. Sekä ohjelmistokehittäjillä että ylläpidon työntekijöillä on pääsy palveluihin ja työkaluihin. Ne on ideaalitilanteessa automatisoitu niiltä osin kuin mahdollista. Näin vältetään turhia pullonkauloja, jotka voivat syntyä esimerkiksi, jos tietyn tiimin työntekijän täytyy aina ”avata tiketti”. Jaetut työkalut tiimien välillä voivat olla esimerkiksi käyttöönottoputkiin, automaattitestaukseen ja tuotantoympäristön kaltaisten ympäristöjen luomiseen liittyviä. (König & Kugel, 2019).

Organisaation tasolla kommunikaation parantamiseksi ja tiedonkulun nopeuttamiseksi tiimien välillä on olemassa vaihtoehtoisia tapoja. Niin kutsutussa Embedded Ops -mallissa keskitetyn IT-ylläpidon sijaan ylläpidon työntekijöitä integroidaan tuotekehitystiimeihin. Näin ylläpidon työntekijät ovat tiukemmin yhteydessä sisäisiin ja ulkoisiin asiakkaisiin. Prioriteetit määritetään tällaisessa organisaatiossa lähes yksinomaan tuotekehitystiimin näkökulmasta. Toinen vaihtoehto organisaation järjestämiselle on yhteyshenkilön käyttäminen. Jokaista tuotekehitystiimiä kohden on yksi yhteyshenkilö ylläpitotiimistä, joka pitää tiiviisti yhteyttä tiimiin ja toimii linkkinä projektitiimin ja oman ylläpitotiiminsä välillä. Mallin tavoitteena on havaita mahdolliset ajankäytölliset konfliktit sekä resurssi- ja priorisointikonfliktit aikaisessa vaiheessa. Sekä Embedded Ops -mallissa että yhteyshenkilömallissa ylläpidon työntekijä osallistuu ohjelmistokehittäjien kokouksiin. Niissä hän voi raportoida tuotekehitystiimille uusimpien julkaisuiden perusteella havaituista kehittämiskohteista. Näin tuotekehitystiimi oppii ymmärtämään tekemiensä ratkaisuiden vaikutusta asiakkaalle paremmin. Lisäksi ylläpidon työntekijät pysyvät paremmin kärryillä tuotekehityksen tehtävistä. (König & Kugel, 2019).

DevOpsin käyttöön liittyviä haasteita on tutkittu kirjallisuudessa. Monet haasteista liittyvät menetelmän soveltamiseen yrityksessä. Ohjelmistokehittäjillä ja ylläpidon työntekijöillä on

lähtökohtaisesti erilaiset tavoitteet, ja näiden tavoitteiden yhteensovittaminen voi osoittautua haastavaksi käytännössä. Myös tekniseen puoleen liittyy haasteita. Eri työvaiheiden automaatio on menetelmän perusajatuksia. Järkevästi toteutettuna automaatio voi säästää pitkällä aikavälillä työntekijöiden aikaa. Sen laajamittaiseen toteuttamiseen tarvitaan kuitenkin paljon resursseja, mikä on osoittautunut ongelmaksi joillekin yrityksille. Lisäksi myös ylläpitoon liittyvät toistuvat prosessit on tarkoitus DevOpsissa automatisoida. Tämä vaatii organisaatiossa myös ylläpidossa hyvää teknistä ohjelmointiosaamista. (de França et al., 2016).

Riungu-Kalliosaari et al. (2016) tutkivat DevOpsin etuja ja haasteita suomalaisissa yrityksissä. He nimeävät haastatteluiden pohjalta menetelmän eduiksi muun muassa paremman kommunikaation ja tiedon jakamisen organisaation sisällä sekä laadukkaammat tuotteet. Automaatiosta havaittiin olevan hyötyä erityisesti testauksessa, sillä testiautomaatio mahdollistaa nopeamman julkaisutahdin ja vähentää lisäksi bugeja. Nopea julkaisutahti puolestaan auttaa yritystä parantamaan tuotteidensa laatua ja asiakastyytyväisyyttä, sillä asiakkailta saadaan palautetta tiheään tahtiin. Näin asiakastarpeisiin pystytään reagoimaan nopeasti. Toisaalta haastatteluissa nousi esille myös haasteita ja ongelmia DevOpsin käyttöön liittyen. Näistä osa liittyy menetelmän epäselkeyteen, sillä DevOpsiin kuuluville menetelmille ei ole selkeää määritelmää. Haastateltavat raportoivat myös ongelmia menetelmän mukaisen kommunikaation omaksumisessa ja yrityskulttuurin muuttamisessa. DevOps ei myöskään sovi kaikkiin ympäristöihin, esimerkiksi mikäli yrityksen tietokannat ovat liian monimutkaisia niiden toisintamiseksi automaatiotestauksessa. (Riungu-Kalliosaari et al., 2016).

DevOpsin käyttöönotto vaatii organisointia ja työntekijöiden koulutusta. Olemassa olevassa organisaatiossa tämä tarkoittaa käytännössä koko organisaatiota koskevaa muutosta (Kasteleiner & Schwartz, 2019). Kun ohjelmistokehitysorganisaatiota luodaan tyhjästä, samanlaista tarvetta muutosjohtamiselle ei ole. DevOps vaatii kuitenkin osaavaa henkilöstöä. Nopeasti suosioon nousseen menetelmän osaajista on suurta kilpailua, minkä johdosta tulee varautua kouluttamaan henkilökuntaa DevOpsiin.

DevOpsin käyttöönotosta on kirjoitettu paljon ja tehty myös paljon tieteellistä tutkimusta. Suurin osa näistä keskittyy olemassa oleviin organisaatioihin, joissa siirrytään DevOpsiin usein esimerkiksi Scrumista. Tutkimusten tuloksia voi hyödyntää myös uuden ohjelmistoliiketoiminnan luomiseen.

4.4 Hybridimenetelmät

Asiantuntijat painottavat, että projektinhallintamenetelmissä ei ole yhtä ratkaisua, joka sopisi kaikkiin tarpeisiin (Hohl et al., 2018; Vijayasarathy & Butler, 2016). Eri yrityksille ja eri projekteihin sopii erilainen lähestymistapa. Valittavat menetelmät riippuvat muun muassa projektin koosta ja kriittisyydestä. Näin ollen myöskään valmiita ohjelmistohallinnan ratkaisuita ei tule nähdä kaikkiin tarpeisiin sopivina ratkaisuina. Usein yritykset muokkaavat menetelmiä omien tarpeidensa mukaan tai yhdistelevät useampia menetelmiä jopa samassa projektissa. Erilaisista hybridimallit eli useamman projektimallin yhdistelmät ovat suosittuja yrityksissä. Kuhrmann et al. (2019) nostavat näistä esille erityisesti niin sanotun Water-scrum-fallin eli vesiputousmallin ja Scrumin yhdistelmän.

Myös DevOpsin periaatteita yhdistetään usein muihin ketteriin menetelmiin. Tiimit käyttävät usein Scrumin periaatteita ja menetelmän mukaisia sprinttejä. Lisäksi tiimien tehtävien havainnollistamisessa saatetaan käyttää Kanban-korttiseinää. (König & Kugel, 2019).

5 Ehdotus yritykselle innovaatioprosessia tukevista menetelmistä

Tässä luvussa käsitellään edellisiin lukuihin pohjautuen innovaation alkuvaihetta ja tuotekehitystä kohdeyrityksessä sekä niiden saumatonta yhteensovittamista. Tavoitteena on löytää ehdotus yritykselle sopivista menetelmistä. Yritys X:llä ei ole entuudestaan ohjelmistokehitysorganisaatiota ja tavoitteena on laajentaa ohjelmistotuotannon alalle aloittaen mahdollisimman pienellä organisaatiolla. Käytettäviltä menetelmiltä tämä vaatii keveyttä. Valittavien menetelmien tulee olla mahdollisia toteuttaa pienellä organisaatiolla. Toisaalta yrityksen kannattaa valita menetelmiä, jotka skaalautuvat helposti organisaation kasvaessa.

Aloittavalle organisaatiolle on hyödyllistä valita menetelmiä, jotka ovat laajasti tunnettuja. Tämä auttaa yritystä löytämään osaavaa työvoimaa, joille käytettävät menetelmät ovat tuttuja jo entuudestaan, ja he osaavat soveltaa niitä yrityksen ongelmiin oma-aloitteisesti. Lisäksi suosituista menetelmistä löytyy laajasti tietoa ja esimerkiksi erilaisia koulutuksia on paljon saatavilla. Laaja käyttö ei itsessään ole tae menetelmän toimivuudesta kohdeyrityksessä, joten tulee kiinnittää huomiota siihen, että menetelmä soveltuu muiltakin osin yrityksen tarpeisiin.

Innovaatioprosessin alkuvaiheeseen on olemassa monia erilaisia menetelmiä sekä ideoiden luomiseen että niiden arvioimiseen ja parhaiden ideoiden valitsemiseen. Uuden tutkimuksen valossa ideoiden luontivaiheessa on hyvä käyttää sekä yksilötyöskentelyä että ryhmätyöskentelyä. Tuplatiimin kaltaiset tekniikat voivat auttaa hyödyntämään näitä havaintoja yrityksen arjessa. Toisaalta esimerkiksi tuplatiimitekniikasta saadaan eniten irti suuremmalla osallistujamäärällä ja se voi olla raskas pienelle ohjelmistokehitysorganisaatiolle. Tekniikasta voi kuitenkin hakea inspiraatiota tähän vaiheeseen. Suositeltavaa ideointivaiheessa on joka tapauksessa hyödyntää yksilö- ja ryhmätyöskentelyä vuorotellen. Mikäli päädytään käyttämään esimerkiksi brainstorming-tekniikkaa, organisaatiossa kannattaa kokeilla perinteisen brainstormingin sijasta brainwritingia.

Ideoiden valintaa varten käytettäviä menetelmiä voi käyttää niin uusien tuoteideoiden arviointiin kuin kehitettävään tuotteeseen lisättävien ominaisuuksien arviointiin. Ennen kuin uutta tuotetta aletaan kehittämään, kannattaa sen toteutuskelpoisuutta arvioida useilla eri tavoilla ja tehdä lisäksi muun muassa kannattavuuslaskelmat projektista. Ominaisuusideoiden

arviointiin parhaita ovat usein kevyet menetelmät. Suuremmista uusista ominaisuuksista voi tehdä demon, jonka avulla ominaisuutta on helppo esitellä asiakkaalle.

Jotta innovaatioprosessista tulee sujuva, ideoiden valintaprosessiin ja niiden jalostamiseen tulee kiinnittää huomiota. Ideoita tulee useasta lähteestä. Osa ideoista on yrityksen sisäisiä ja osa syntyy keskusteluissa asiakkaan kanssa. Muun muassa Bosch et al. (2013) esittävät idea backlogia ideoiden hallintaan. Backlogiin kirjataan kaikki ideat. Backlogista vastaava henkilö kirjoittaa kaikki ideat samankaltaisessa muodossa, jotta ne ovat mahdollisimman hyvin vertailtavissa keskenään. Tämän jälkeen ideat arvioidaan ja priorisoidaan. (Bosch et al., 2013). Idea backlogia voi käyttää hyväksi myös sellaisten ideoiden kanssa, jotka ehdottavat uusia ominaisuuksia olemassa olevaan tai kehitettävään tuotteeseen. Tällaisesta idea backlogista ideoita voi nostaa tuotebacklogiin ketterän kehityksen periaatteiden mukaan esimerkiksi viikoittaisissa palavereissa. Näin ideat kulkevat innovaation alkuvaiheesta tuotekehitykseen.

Valittavan tuotekehitysmenetelmän tulee mahdollistaa työntekijöiden roolien eläminen. Toisaalta myös rekrytoitavilta työntekijöiltä vaaditaan joustavuutta ja halua tarttua myös oman varsinaisen alueensa ulkopuolisiin työtehtäviin tarpeen vaatiessa. Pienellä porukalla ohjelmistoja kehitettäessä työntekijöiltä vaaditaan laaja-alaisuutta, itseohjautuvuutta ja kokonaiskuvan hahmottamista. Työntekijöiden käytännön kokemuksesta käytettävistä menetelmistä on hyötyä siinä, että projekteja saadaan tehtyä pienellä työntekijämäärällä. Erityisesti aloittelevan organisaation voi kannattaa harkita osan työvoimasta hankkimista alihankintana, jonka avulla päästään nopeammin kiinni projekteihin ja voidaan täyttää alun osaamisvajetta tiimissä.

Taulukossa 1 on tiivistetty eri projektinhallintamenetelmien etuja ja heikkouksia kirjallisuuskatsaukseen perustuen. Perinteisen vesiputousmallin huonona puolena on se, että muutoksiin reagoiminen on hidasta ja kallista. Ketterät kehitysmenetelmät ja DevOps vastaavat tähän ongelmaan. Jokaisella projektinhallintamenetelmällä on omat hyvät ja huonot puolensa, ja kukin menetelmä sopii eri käyttötarkoituksiin.

Taulukko 1. Tuotekehityksen projektinhallintamenetelmien vertailu.

Menetelmä	**Positiivista**	**Negatiivista**	**Mihin soveltuu erityisesti?**
Vesiputousmalli	Konkreettiset tavoitteet.	Muutoksiin reagointi on jäykkää. Ongelmien korjaaminen on kallista mallin myöhäisemmissä vaiheissa.	Esimerkiksi pienet sisäiset ohjelmistot, joiden vaatimukset on helppo määritellä etukäteen.
Scrum	Tiivis yhteistyö asiakkaan kanssa. Jokaisen sprintin tuotteena toimiva inkrementti ohjelmasta. Soveltuu pienellä organisaatiolla toteutettavaksi, mutta skaalautuu myös hyvin organisaation kasvaessa.	Jäykkä moniin muihin ketteriin menetelmiin verrattuna. Voi lisätä työntekijöiden stressiä ajallisen paineen takia.	Ohjelmistotuotteet, joihin halutaan julkaista tiheästi uusia ominaisuuksia ja joiden jakelu asiakkaille on helppoa.
Kanban	Tieto projektin kulusta on aina ajan tasalla ja visuaalisesti esitettynä. Aikataulu ja muutoksiin vastaaminen onnistuvat hyvin.	Keskeneräisen työn rajoittaminen voi johtaa pullonkauloihin.	Pienet työtehtävät. Ohjelmistojen ylläpito.
Testivetoinen kehitys	Useammat testit menevät läpi. Parantaa ohjelmistojen laatua.	Hyödyt kiisteltyjä niin käytännössä kuin akateemisessa maailmassakin. Joidenkin tutkimusten mukaan käytettävä työaika on pidempi. Omaksuminen vie kauan aikaa.	Projektit, joissa halutaan panostaa laadukkaaseen lopputulokseen kattavien ennalta määriteltyjen testien avulla.
DevOps	Parempi kommunikaatio tuotekehityksen ja ylläpidon välillä. Vähentää sisäisiä konflikteja. Automaation avulla tihentää julkaisutahtia.	Menetelmä on usein epäselvä organisaatiossa.	Ohjelmistotuotteet, joiden kehityksen aikana halutaan asiakkaan palautetta mahdollisimman usein.

Tässä kandidaatintyössä käsitellyistä tuotekehitysmenetelmistä Scrum vastaa Yritys X:n tarpeita parhaiten, sillä menetelmä sopii hyvin pienille tiimeille. Toisaalta tulevaisuudessa, mikäli yritys haluaa laajentaa sovellusliiketoimintaansa, Scrum skaalautuu helposti myös suuremmalle organisaatiolle. Scrum on tänä päivänä eniten käytetty projektinhallintamenetelmä ohjelmistotuotannossa. Projektinhallintamenetelmä kannattaa kuitenkin valita yrityksen projektien mukaan. Lisäksi Scrumia, kuten mitään muutakaan ohjelmistotuotannon projektinhallintamenetelmää, ei yleensä käytetä tai kuulukaan käyttää yrityksissä sellaisenaan ohjeita kirjaimellisesti seuraten. Ketterän kehityksen tavoite on mukautua muuttuviin vaatimuksiin ja asiakastarpeisiin mahdollisimman hyvin.

Organisaatiolle on hyödyllistä valita menetelmiä, joita voidaan soveltaa sekä innovaatioprosessin alkuvaiheessa että tuotekehitysvaiheessa. Näin organisaation oppimiskynnys madaltuu ja menetelmien käytön tuntemus laajenee organisaatiossa. Tässä ehdotuksessa sekä innovaatioprosessin alkuvaihe että tuotekehitys toimivat ketterän kehityksen periaatteilla, ja niitä yhdistävät muun muassa Scrum backlogit. Mikäli yrityksessä on henkilöitä, joilla on rooli sekä innovaatioprosessin alkuvaiheessa että tuotekehityksessä, tiedonkulku prosessin eri vaiheiden välillä helpottuu. Scrum-mallissa esimerkiksi tuotteen omistaja voi olla tällainen henkilö.

6 Johtopäätökset

Työn tavoitteena oli tutkia ohjelmistotuotannon alalle laajentamista ja tarjota kattava esittely erilaisista ohjelmistotuotannon innovaatioprosessia tukevista menetelmistä sekä ehdotus Yritys X:lle ohjelmistotuotantoon laajentamiseen. Tämä luku tarjoaa yhteenvedon tutkimuksen tuloksista niin yleisellä tasolla kuin kohdeyrityksen Yritys X näkökulmasta.

Mitä ohjelmistokehitykseen laajentaminen vaatii yritykseltä?

Ohjelmistokehitykseen laajentaminen vaatii yritykseltä investointeja muun muassa henkilöstöön, työkaluihin ja koulutuksiin. Yrityksen työntekijöiden tulee ymmärtää käyttöön otettavat menetelmät. Ohjelmistoalalla aloittavassa tai alalle laajentavassa yrityksessä tämä voidaan taata kokeneen ja motivoituneen henkilökunnan palkkaamisella, henkilökunnan kouluttamisella sekä mahdollisesti resurssivuokrauksella alihankintana.

Ohjelmistokehityksessä on monia erityispiirteitä, jotka yrityksen tulee ottaa huomioon. Esimerkkejä tästä ovat prototyyppien helppo toteuttaminen sekä uudelleenkäytettävien ohjelman osien käyttäminen.

Asiakkaan osallistaminen on viime aikoina noussut entistäkin tärkeämpään rooliin ohjelmistokehityksen alalla. Asiakasvaatimukset muuttuvat usein projektin kuluessa, ja asiakkaan osallistaminen projektin eri vaiheissa ohjaa tuotteen kehittämistä mahdollisimman käyttökelpoiseksi ja asiakasta miellyttäväksi.

Millaisia menetelmiä innovaatioprosessin alku- ja tuotekehitysvaiheiden hallintaan on olemassa?

Sekä innovaatioprosessin alkuvaiheeseen että tuotekehitykseen on olemassa paljon erilaisia menetelmiä ja työkaluja. Innovaatioprosessin alkuvaiheen menetelmistä tässä kandidaatintyössä käsiteltiin erityisesti ideointiin liittyviä menetelmiä, kuten esimerkiksi brainstorming, ja ideoiden valintaan liittyviä menetelmiä, kuten esimerkiksi ABC-analyysi ja NUF-mittari. Lisäksi esimerkiksi tuplatiimi liittyy niin ideoiden luomiseen kuin ideoiden valintaankin.

Nykypäivänä ketterät kehitysmenetelmät ja etenkin Scrum ovat suosittuja tuotekehityksen projektinhallintamenetelmiä. Sekä käytännössä että tutkimuksessa ketterät menetelmät ovat saaneet paljon huomiota ja kehuja. Silti edelleen perinteinen vesiputousmalli on

suosituimpien tuotekehitysmenetelmien joukossa ja menetelmä sopii hyvin tietynlaisiin projekteihin, kuten esimerkiksi pieniin sisäisiin projekteihin, joiden vaatimukset voidaan määritellä tarkasti jo projektin alussa.

Mitä yrityksen tulee ottaa huomioon menetelmien valinnassa?

Mikään menetelmä ei sovi jokaiselle yritykselle, vaan menetelmät on aina valittava tapauskohtaisesti. Valittavaan tuotekehitysprojektin hallintamenetelmään vaikuttaa muun muassa yrityksen projektien luonne, organisaation koko sekä mahdolliset asiakkaiden toiveet. Asiantuntijoiden mukaan yrityksen ei kannata seurata mitään valmista menetelmää liian tarkasti vaan käyttää niitä pohjana ja luoda itselleen sopiva tapa toimia.

Millaisia menetelmiä Yritys X:n kannattaa ottaa käyttöön laajentuessaan ohjelmistotuotantoon?

Yritys X on laajentamassa liiketoimintaansa ohjelmistokehitykseen. Tavoitteena on aloittaa toiminta mahdollisimman pienellä organisaatiolla. Innovaation alkuvaiheen ideoinnissa kannattaa tutkimusten perusteella hyödyntää vaihtelevasti yksilö- ja ryhmätyötä. Esimerkki yksilö-, pari- ja ryhmätyöskentelyä hyödyntävästä menetelmästä on tuplatiimi.

Yrityksen on projektinhallintamenetelmää valitessaan huomioitava, että osa menetelmistä soveltuu paremmin pienellä organisaatiolla toteutettaviksi, kun taas toisiin vaaditaan suurempi organisaatio. Yrityksen tavoitteena on tulevaisuudessa kasvattaa liiketoimintaa ohjelmistoalalla. Valittavien menetelmien hyvä skaalautuvuus mahdollistaa käytön sujuvuuden organisaation kasvaessa. Tutkituista tuotekehitystä tukevista projektinhallintamenetelmistä Yritys X:n tarpeisiin sopii parhaiten Scrum, sillä se täyttää parhaiten edellä kuvatut vaatimukset.

Tutkimuksen rajoitteet ja jatkotutkimuskohteet

Tämä kandidaatintyö keskittyi tutkimaan liiketoiminnan laajentamista ohjelmistokehitykseen ja sitä tukevia menetelmiä. Tuloksia ei voi yleistää kaikkiin yrityksiin. Erityisesti viides luku vastaa Yritys X:n tarpeisiin eikä sitä voi sellaisenaan soveltaa muissa organisaatioissa. Siihen, mitkä projektinhallintamenetelmät sopivat parhaiten, vaikuttavat muun muassa organisaation koko, projektin tyyppi sekä mahdollisesti myös asiakkaan toiveet ja vaatimukset projektille. Tutkimuksen tulokset eivät ole yleistettävissä ohjelmistokehityksen ulkopuolelle.

Innovaatioprosessi jaetaan usein kolmeen osaan, jotka ovat innovaatioprosessin alkuvaihe, tuotekehitys ja kaupallistaminen. Tässä työssä käsiteltiin näistä vain kahta ensimmäistä ja kaupallistaminen jätettiin rajauksen ulkopuolelle. Eräs jatkotutkimuskohde voisi olla ohjelmistotuotteiden kaupallistaminen. Lisäksi mielenkiintoista olisi tutkia haastattelututkimuksella yrityksissä jo toteutettua ohjelmistoalalle laajentumista sekä siihen liittyviä haasteita ja menestystekijöitä.

Lähteet

Annosi, M.C., Magnusson, M., Martini, A., Appio, F.P. 2016. Social Conduct, Learning and Innovation: An Abductive Study of the Dark Side of Agile Software Development. Creativity & Innovation Management. Vol. 25, nro. 4. s. 515-535.

Baldassarre, M.T., Caivano, D., Fucci, D., Juristo, N., Romano, S., Scanniello, G., Turhan, B. 2021. Studying test-driven development and its retainment over a six-month time span. Journal of Systems and Software. Vol. 176. s. 110937-.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. 2001. Manifesto for agile software development. [WWW-dokumentti]. [viitattu 8.10.2022]. Saatavilla: https://agilemanifesto.org/.

Bissi, W., Serra Seca Neto, A.G., Emer, M.C.F.P. 2016. The effects of test driven development on internal quality, external quality and productivity: A systematic review. Information and Software Technology. Vol. 74. s. 45–54.

Bosch, J., Olsson, H., Björk, J., Ljungblad, J. 2013. The Early Stage Software Startup Development Model: A Framework for Operationalizing Lean Principles in Software Startups. In International Conference on Lean Enterprise Software and Systems. s. 1–15. Berliini: Springer.

Braglia, M., Gabbrielli, R., Marrazzini, L. 2020. Rolling Kanban: a new visual tool to schedule family batch manufacturing processes with kanban. International Journal of Production Research. Vol. 58, nro. 13. s. 3998–4014.

Broy, M. (2018). Yesterday, Today, and Tomorrow: 50 Years of Software Engineering. IEEE Software. Vol. 35, nro. 5. s. 38–43.

Causevic, A., Sundmark, D., Punnekkat, S. 2011. Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review. Teoksessa: Verification and Validation 2011 Fourth IEEE International Conference on Software Testing. s. 337–346.

Chao, R.O., Kavadias, S. 2008. A Theoretical Framework for Managing the New Product Development Portfolio: When and How to Use Strategic Buckets. Management Science. Vol. 54, nro. 5. s. 907–921.

Cooper, R., Edgett, S., Kleinschmidt, E. 2001. Portfolio management for new product development: results of an industry practices study. R&D Management. Vol. 31, nro. 4. s. 361-380.

Cui, Z., Kumar PM, S., Gonçalves, D. 2019. Scoring vs. Ranking: An Experimental Study of Idea Evaluation Processes. Production and Operations Management. Vol. 28, nro. 1. s. 176–188.

Delbecq, A.L., Van de Ven, A.H. 1971. A Group Process Model for Problem Identification and Program Planning. The Journal of Applied Behavioral Science. Vol. 7. s. 466–492.

Díaz, J., López-Fernández, D., Pérez, J., González-Prieto, Á. 2021. Why are many businesses instilling a DevOps culture into their organization? Empirical software engineering: an international journal. Vol. 26, nro. 2. s. 25.

digital.ai, 2021. 15th Annual State Of Agile Report [WWW-dokumentti]. [viitattu 25.11.2022]. Saatavilla: https://digital.ai/resource-center/analyst-reports/state-of-agile-report/.

Dikert, K., Paasivaara, M., Lassenius, C. 2016. Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software. Vol. 119. s. 87–108.

Duc, A.N., Abrahamsson, P. 2016. Minimum Viable Product or Multiple Facet Product? The Role of MVP in Software Startups. Teoksessa: Sharp, H., Hall, T. (Toim.), Agile Processes, in Software Engineering, and Extreme Programming, Lecture Notes in Business Information Processing. Cham: Springer International Publishing. s. 118–130.

Ebert, C., Paasivaara, M. 2017. Scaling Agile. IEEE Software. Vol. 34, nro. 6. s. 98–103.

Fagarasan, C. et al. 2021. Agile, waterfall and iterative approach in information technology projects. IOP conference series. Materials Science and Engineering. Vol. 1169, nro. 1. s. 12025.

Flores, B.E., Whybark, D.C., 1986. Multiple Criteria ABC Analysis. International Journal of Operations and Production Management. Vol. 6, s. 38–46.

França, B.B.N. de, Jeronimo, H., Travassos, G.H. 2016. Characterizing DevOps by Hearing Multiple Voices. Teoksessa: Proceedings of the 30th Brazilian Symposium on Software Engineering. Maringá Brazil. s. 53–62.

Gaete, J., Villarroel, R., Figueroa, I., Cornide-Reyes, H., Muñoz, R. 2021. Enfoque de aplicación ágil con Scrum, Lean y Kanban. Ingeniare : Revista Chilena de Ingeniería. Vol. 29, nro. 1. s. 141–157.

Gloger, B. 2010. Scrum: Der Pradigmenwechsel im Projekt- und Produktmanagement – Eine Einführung. Informatik-Spektrum. Vol. 33, nro. 2. s. 195–200.

Haikala, I., Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt, 12. painos. Helsinki: Talentum.

Helin, K. 1987. Kehitämme innovoimaa. 2. painos. Imatra: Innotiimi.

Hohl, P. et al. 2018. Back to the future: origins and directions of the ‘Agile Manifesto’ – views of the originators. Journal of software engineering research and development. Vol. 6, nro. 1. s. 1–27.

Janzen, D., Saiedian, H. 2005. Test-driven development concepts, taxonomy, and future direction. Computer (Long Beach, Calif.). Vol. 38, nro. 9. s. 43–50.

Johnson, B.R., D’Lauro, C.J. 2018. After Brainstorming, Groups Select an Early Generated Idea as Their Best Idea. Small Group Research. Vol. 49, nro. 2. s. 177–194.

Kalargiros, M., Geng, X., Pittz, T.G. 2019. A Revival of Osborn’s Original Propositions: The Role of Inspirational Facilitation in Divergent Thinking Effectiveness. Journal of Managerial Issues: JMI. Vol. 31, nro. 2. s. 151.

Karac, I., Turhan, B. 2018. What Do We (Really) Know about Test-Driven Development? IEEE Software. Vol. 35, nro. 4. s. 81–85.

Kasteleiner, B., Schwartz, A. 2019. DevOps: Schnell, zuverlässig und sicher von der Idee zur Realisierung. Informatik-Spektrum. Vol. 42, nro. 3. s. 211–214.

Kim, J., Wilemon, D. 2002. Strategic issues in managing innovation’s fuzzy front-end. European Journal of Innovation Management. Vol. 5, nro. 1. s. 27–39.

Kirovska, N., Koceski, S. 2015. Usage of Kanban methology at software development teams. Journal of Applied Economics and Business. Vol. 3, nro. 3. s. 25-34.

Kock, A., Heising, W., Gemünden, H.G. 2015. How Ideation Portfolio Management Influences Front-End Success. Journal of Product Innovation Management. Vol. 32, nro. 4. s. 539–555.

Koen, P., Ajamian, G., Burkart, R., Clamen, A., Davidson, J., D'Amore, R., Elkins, C., Herald, K., Incorvia, M., Johnson, A., Karol, R., Seibert, R., Slavejkov, A., Wagner, K. 2001. Providing Clarity and a Common Language to the "Fuzzy Front End." Research Technology Management. Vol. 44, nro. 2. s. 46-55.

König, G., Kugel, R. 2019. DevOps—Welcome to the Jungle. HMD Praxis der Wirtschaftsinformatik. Vol. 56, nro. 2. s. 289–300.

Korde, R., Paulus, P.B. 2017. Alternating individual and group idea generation: Finding the elusive synergy. Journal of Experimental Social Psychology. Vol. 70. s. 177–190.

Kornish, L.J., Hutchison-Krupat, J. 2017. Research on Idea Generation and Selection: Implications for Management of Technology. Production and Operations Management Vol. 26. s. 633–651.

Kudrowitz, B., Wallace, D. 2013. Assessing the quality of ideas from prolific, early-stage product ideation. Journal of Engineering Design. Vol. 24, nro. 2. s. 120–139.

Kuhrmann, M., Diebold, P., Munch, J., Tell, P., Trektere, K., McCaffery, F., Garousi, V., Felderer, M., Linssen, O., Hanser, E., Prause, C.R. 2019. Hybrid Software Development Approaches in Practice: A European Perspective. IEEE Software. Vol. 36, nro. 4. s. 20–31.

Lee, C., Jeon, D., Ahn, J.M., Kwon, O. 2020. Navigating a product landscape for technology opportunity analysis: A word2vec approach using an integrated patent-product database. Technovation. Vol. 96. s. 102140-.

Lei, H., Ganjeizadeh, F., Jayachandran, P.K., Ozcan, P. 2017. A statistical analysis of the effects of Scrum and Kanban on software development projects. Robotics and Computer-Integrated Manufacturing. Vol. 43, s. 59–67.

Martinsuo, M., Poskela, J. 2011. Use of Evaluation Criteria and Innovation Performance in the Front End of Innovation*. Journal of Product Innovation Management. Vol. 28, nro. 6. s. 896–914.

Mehboob, B., Chong, C.Y., Lee, S.P., Lim, J.M.Y. 2021. Reusability affecting factors and software metrics for reusability: A systematic literature review. Software, practice & experience. Vol. 51, nro. 6. s. 1416–1458.

Michinov, N., 2012. Is Electronic Brainstorming or Brainwriting the Best Way to Improve Creative Performance in Groups? An Overlooked Comparison of Two Idea-Generation Techniques: Electronic Brainstorming and Brainwriting. Journal of Applied Social Psychology. Vol. 42, nro. 1. s. E222–E243.

Munir, H., Moayyed, M., Petersen, K. 2014. Considering rigor and relevance when evaluating test driven development: A systematic review. Information and Software Technology. Vol. 56, nro. 4. s. 375–394.

Osborn, A., 1953. Applied Imagination: Principles and Procedures of Creative Thinking. New York: Scribner.

Pugh, S., 1991. Total design: integrated methods for successful product engineering. Workingham: Addison-Wesley.

Rebernik, M., Bradač, B. 2008. Module 4: Idea evaluation methods and techniques. Ekonomska-poslovna fakulteta : Institute for Entrepreneurship and Small Business Management. [WWW-dokumentti]. [viitattu 26.11.2022]. Saatavilla: https://www.rockypeaklc.com/ideaevaluation.pdf.

Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L.E., Tiihonen, J., Männistö, T., 2016. DevOps Adoption Benefits and Challenges in Practice: A Case Study. Teoksessa: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., Mikkonen, T.(Toim.): Product-Focused Software Process Improvement, Lecture Notes in Computer Science. Cham: Springer International Publishing. s. 590–597.

Royce, W.W. 1987. Managing the development of large software systems: concepts and techniques. International Conference on Software Engineering: Proceedings of the 9th international conference on Software Engineering. s. 328–338.

Salminen, A., 2011. Mikä kirjallisuuskatsaus? johdatus kirjallisuuskatsauksen tyyppeihin ja hallintotieteellisiin sovelluksiin. Vaasa: Vaasan yliopisto.

Sandhu, P.S., Aashima, Kakkar, P., Sharma, S. 2010. A survey on Software Reusability. Teoksessa: 2010 International Conference on Mechanical and Electrical Technology. IEEE. s. 769–773.

Santiago, L.P., Soares, V.M.O. 2020. Strategic Alignment of an R&D Portfolio by Crafting the Set of Buckets. IEEE Transactions on Engineering Management. Vol. 67, nro. 2. s. 309–321.

Santos, A., Vegas, S., Dieste, O., Uyaguari, F., Tosun, A., Fucci, D., Turhan, B., Scanniello, G., Romano, S., Karac, I., Kuhrmann, M., Mandić, V., Ramač, R., Pfahl, D., Engblom, C., Kyykkä, J., Rungi, K., Palomeque, C., Spisak, J., Oivo, M., Juristo, N. 2021. A family of experiments on test-driven development. Empirical Software Engineering. Vol. 26, nro. 3. s. 1-53.

Schwaber, K. 1997. Scrum development process. Teoksessa: Business object design and implementation. s. 117-134. Lontoo: Springer.

Sommerville, I. 2016. Software engineering. 10. painos. Boston: Pearson.

Sugimori, Y., Kusunoki, K., Cho, F., Uchikawa, S. 1977. Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system. International Journal of Production Research. Vol. 15, nro. 6. s. 553–564.

Tiwana, A., Konsynski, B., Bush, A.A., 2010. Research Commentary-Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. Information Systems Research. Vol. 21, nro. 4. s. 675–687.

Ulonska, S., Welo, T. 2014. Product portfolio map: a visual tool for supporting product variant discovery and structuring. Advances in manufacturing. Vol. 2, nro. 2. s. 179–191.

Unger-Windeler, C., Klünder, J.A.-C., Reuscher, T., Schneider, K. 2021. Are Product Owners communicators? A multi-method research approach to provide a more comprehensive picture of Product Owners in practice. Journal of Software: Evolution and Process. Vol. 33, nro. 1.

Vijayasarathy, L. R. & Butler, C. W. 2016 Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter? IEEE software. Vol. 33, nro. 5. s. 86–94.

Wilkerson, J.W., Nunamaker, J.F., Mercer, R. 2012. Comparing the Defect Reduction Benefits of Code Inspection and Test-Driven Development. IEEE Transactions on Software Engineering. Vol. 38. s. 547–560.