ERILAISET KETTERÄT MENETELMÄT

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
syksy, 2022

Juha Puutula

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Tekijä Juha Puutula Vuosi 2022

Työn nimi Erilaiset ketterät menetelmät

Ohjaajat Lasse Seppänen

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tutkia, kuinka erilaiset ketterät ohjelmistokehityksen menetelmät eroavat toisistaan, mitä niissä on yhteistä ja voiko useampia menetelmiä käyttää yhdessä. Lisäksi selvitettiin menetelmien hyötyjä ja rajoitteita. Yrityshaastatteluiden avulla selvitettiin käytännönkokemuksia menetelmien käytöstä työelämässä. Opinnäytetyön toimeksiantaja oli HAMK.

Opinnäytetyön tietopohja koostuu erilaisten ketterien menetelmien teoriasta ja lisäksi mukana on perinteinen ohjelmistokehitysmenetelmä nimeltään vesiputousmalli. Opinnäytetyö on tutkimuksellinen. Tutkimusosuudessa hyödynnettiin teoriaosuuden tietopohjaa ja yrityshaastatteluista saatuja käytännönkokemuksia. Tutkimusaineistossa käytettiin strukturoitua laadullista haastattelua.

Johtopäätöksenä voidaan todeta, että kaikista tutkimuksessa mukana olleista ketteristä menetelmistä löytyy yhtäläisyyksiä, eroavaisuuksia ja monia menetelmiä voidaan käyttää yhtäaikaisesti. Lisäksi kaikista menetelmistä löytyy hyötyjä ja rajoituksia. Käytännönkokemukset antavat hyviä näkökulmia tilanteisiin, joissa pohditaan millaiset menetelmät sopivat erilaisissa tilanteissa ja projekteissa parhaiten.

Avainsanat XP, Scrum, Lean, Kanban, DevOps, SAFe, PRINCE2

Sivut 86 sivua ja liitteitä 2 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology

Author Juha Puutula

Subject Various Agile Methods

Supervisors Lasse Seppänen

Abstract

Year 2022

ABSTRACT

The aim of the thesis was to investigate how different agile software development methods differ from each other, what they have in common and whether several methods can be used together. In addition, the benefits and limitations of the methods were explored. Business interviews were conducted to find out about practical experiences of using the methods in working life. The client of the thesis was HAMK.

The knowledge base of the thesis consists of the theory of different agile methods and includes a traditional software development method called the waterfall model. The thesis is exploratory. In the research part, the knowledge base of the theoretical part and practical experiences from company interviews were used. A structured qualitative interview was used for the research data.

In conclusion, all the agile methods included in the study have similarities, differences and many methods can be used simultaneously. In addition, there are benefits and limitations to all methods. Practical experience provides good insights into situations where the question of which methods are best suited to different situations and projects is being considered.

Keywords XP, Scrum, Lean, Kanban, DevOps, SAFe, PRINCE2

Pages 86 pages and appendices 2 pages

Sanasto

Agile Release Train (ART)	Ketterä julkaisujuna SAFe:ssa
Business Case	Liiketoimintatarkastelu PRINCE2:ssa
CALMR-viitekehys	Kulttuuri, automaatio, Lean virtaus, mittaaminen ja palautuminen SAFe:ssa
CALMS-viitekehys	Kulttuuri, automaatio, Lean, mittaaminen ja jakaminen DevOpsissa
Inkrementti	Konkreettinen askel kohti tuotteen tavoitetta
Inkrementaaliset muutokset	Muutoksia tehdään pienissä osissa
Iteratiivinen	Asteittain tarkentuva
Product Owner	Tuoteomistaja, joka varmistaa arvon tuoton
Program Increment (PI)	Aikaväli, jolloin ketterä julkaisujuna tuottaa arvoa
Refaktorointi	Koodin uudelleenmuokkaus
Scrum Master	Johtaa, kouluttaa ja valmentaa Scrumin käytössä
Sprintti	Scrumin tapahtumajakso, joka sisältää kaikki muut tapahtumat
Work In Progress (WIP)	Käynnissä olevat tehtävät

Sisällys

Kuvat ja taulukot

Liitteet

1 Johdanto

Opinnäytetyön aiheena on vertailla erilaisia ketteriä ohjelmistokehityksen menetelmiä. Ketterien menetelmien tarkoituksena on varmistaa kehitettävien ohjelmistojen toimivuus, suora viestintä ja nopea muutoksiin reagointi. Ketteriä menetelmiä on useita ja tässä opinnäytetyössä niitä vertaillaan keskenään ja lisäksi teoriaosassa mukana on perinteinen ohjelmistokehitysmenetelmä nimeltään vesiputousmalli.

Ketteristä menetelmistä mukaan valikoitui XP, Scrum, Lean, DevOps, SAFe ja PRINCE2. Nämä menetelmät valikoituivat sen vuoksi, että ne ovat yleisesti käytössä olevia menetelmiä. Teoriaosuuden tietojen ja yrityshaastatteluiden perusteella selvitetään mitä yhtäläisyyksiä menetelmissä on ja miten ne eroavat toisistaan. Selvitetään lisäksi kustomoidut menetelmät ja eri menetelmien käyttö yhtäaikaisesti. Menetelmien hyödyt ja rajoitukset selvitetään.

Yrityshaastattelut toteutetaan laadullisella strukturoidulla haastattelulla. Kysymykset lähetetään yrityksille vastattavaksi ja ne anonymisoidaan. Haastattelukysymyksillä pyritään saamaan käytännön kokemuksia eri menetelmistä ja selvitetään millaisissa projekteissa eri menetelmät toimivat parhaiten.

Opinnäytetyön tutkimuskysymyksiä ovat:

- Mitä yhteistä ketterillä menetelmillä on?
- Miten ketterät menetelmät eroavat toisistaan?
- Voidaanko menetelmiä kustomoida tai käyttää yhtäaikaisesti?
- Millaisia hyötyjä ja rajoituksia eri menetelmissä on?

2 Erilaiset ohjelmistokehitysmenetelmät

Ketterät ohjelmistokehityksen menetelmät alkoivat yleistyä perinteisten ohjelmistokehitysmenetelmien rinnalle, kun haluttiin löytää vaihtoehtoja asiakirjoihin perustuvalle raskaalle ohjelmistokehitysprosessille. Vuonna 2001 eri ohjelmistokuntien edustajat kokoontuivat ja syntyi ketterän ohjelmistokehityksen julistus (Agilemanifesto 2001, n.d.).

Ketterän ohjelmistokehityksen julistukseen kirjattiin parempia tapoja tehdä ohjelmistokehitystä ohjelmistokuntien edustajien kokemusten perusteella. Siinä arvostetaan ensisijaisesti yksilöitä ja kanssakäymistä, toimivaa ohjelmistoa, asiakasyhteistyötä ja vastaamista muutokseen. Menetelmillä, työkaluilla, kattavalla dokumentaatiolla, sopimusneuvotteluilla ja suunnitelmassa pysymisellä on myös arvoa, mutta ensiksi mainitut ovat kaikkein tärkeimpiä (Agilemanifesto 2001, n.d.).

2.1 Vesiputousmalli

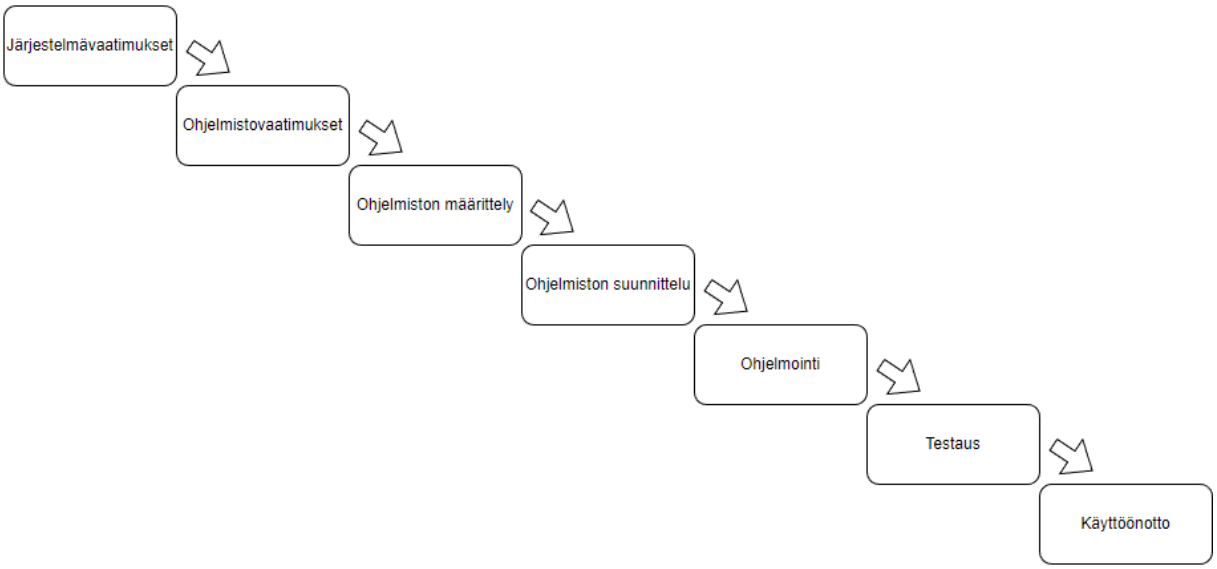
Vesiputousmallista on hyvä aloittaa erilaisiin menetelmiin tutustuminen, koska se on vanhin ohjelmistokehityksen menetelmä. Winston Royce julkaisi artikkelin Management of the development of Large Software jo vuonna 1970 (Royce, 1970). Artikkelin toisella sivulla esitettiin yksinkertainen prosessimalli, jota alettiin kutsumaan vesiputousmalliksi (Luukkainen & Ilves, 2021).

Nimi on erittäin osuva, koska vaiheet etenevät järjestyksessä, jossa määritellään järjestelmävaatimukset, ohjelmistovaatimukset, ohjelmiston määrittely, ohjelmiston suunnittelu, ohjelmointi, testaus ja lopuksi käyttöönotto. Ajatuksena on, että alussa käytetty aika järjestelmävaatimuksiin, ohjelmistovaatimuksiin, ohjelmiston määrittelyyn ja ohjelmiston suunnitteluun säästää aikaa loppupuolen ohjelmoinnissa, testauksessa ja käyttöönotossa (Juvonen, 2018, s 8-9).

Vesiputousmallin vahvuutena voi pitää sen selkeyttä ja yksinkertaisuutta. Sen heikkous on se, että käytännöntasolla pitää usein palata aikaisempiin vaiheisiin sekä muuttaa toteutusta tai suunnitelmia. Winston W. Royce tiedosti tämän heikkouden. Hän on määritellyt mallin niin, että myös edellisiin vaiheisiin voi palata tarvittaessa. Tämä määritelmä, joka on artikkelin

loppupuolella, jäi kuitenkin aikoinaan monilta huomaamatta ja mallia on laajalti käytetty ilman palaamista aiempiin vaiheisiin. Vesiputousmallia käytetään edelleen, mutta 2000-luvun alkupuolella ohjelmistokehitykseen alkoi tulla mukaan ketteriä menetelmiä (Juvonen, 2018, s 8-9). Kuvassa 1 on esitetty Roycen vesiputousmalli.

Kuva 1 Vesiputousmalli



2.2 Extreme Programming (XP)

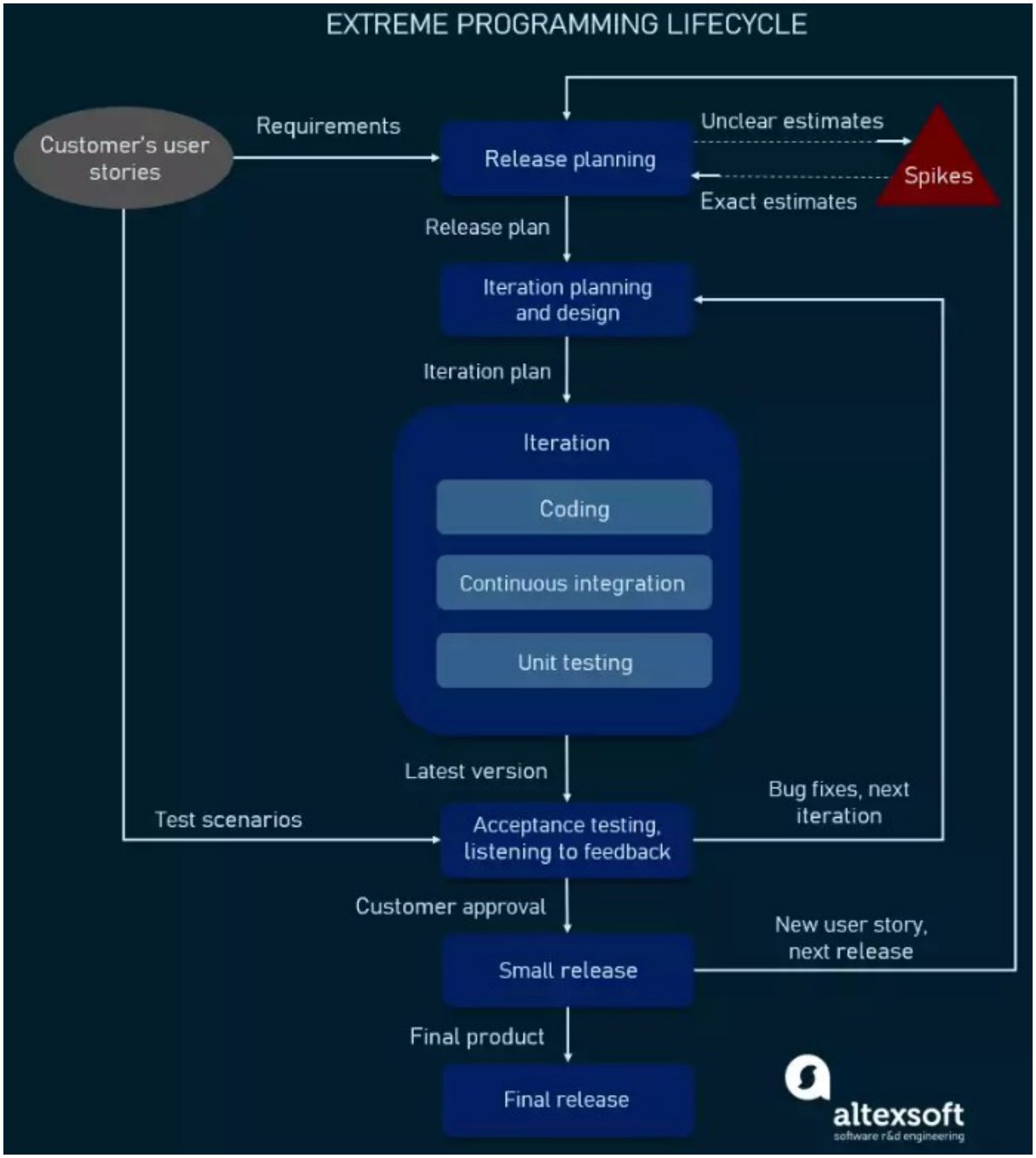
Extreme Programming eli XP on ohjelmistoinsinööri Ken Beckin 90-luvulla kehittämä menetelmä. Hänen tavoitteenaan oli löytää keinoja, kuinka kirjoittaa laadukkaita ohjelmistoja nopeasti ja mukautua asiakkaiden muuttuviin vaatimuksiin. Hän tarkensi XP:n lähestymistapoja kirjassa Extreme Programming Explained: Embrace Change vuonna 1999. XP on ketterä kehitysmenetelmä ja se korostaa ohjelmistokehityksen teknisiä näkökulmia. XP on joukko suunnittelukäytäntöjä. Nimessä oleva extreme johtuu siitä, että kehittäjät joutuvat ylittämään omat kykynsä toteuttaessaan suunnittelukäytäntöjä (Altexsoft 2021, n.d.).

XP:n kehitysprosessissa on viisi vaihetta, jotka toistuvat jatkuvasti. **Planning** eli **valmistelu** on vaihe, jossa kehitystiimi tapaa asiakkaan ja asiakas esittää kehitettävän ohjelmiston vaatimukset käyttäjätarinoiden muodossa, jotka kuvaavat millaiseksi ohjelmisto halutaan lopulta kehittää. Tiimi

arvioi käyttäjätarinat ja laatii julkaisusuunnitelman. Julkaisusuunnitelma jaetaan iteraatioihin, jotta vaaditut toiminnallisuudet voidaan toteuttaa osa kerrallaan. Jos yhtä tai useampaa käyttäjätarinaa ei voida arvioida, voi syntyä piikkejä ja silloin pitää tehdä lisää tutkimuksia (Altexsoft 2021, n.d.).

Designing eli **suunnittelu** on vaihe, joka liittyy edelliseen valmisteluvaiheeseen. Siinä korostuu XP:n arvo yksinkertaisuus. Hyvällä suunnittelulla järjestelmään saadaan logiikkaa ja rakennetta ja voidaan välttää tarpeettomat monimutkaisuudet ja turhat ominaisuudet. **Coding** eli **koodaus** on vaihe, jossa varsinainen koodi luodaan. Vaiheessa hyödynnetään XP:n käytäntöjä kuten koodausstandardit, pariohjelmointi, jatkuva integrointi ja koodin kollektiivinen omistajuus. **Listening** eli **kuunteleminen** tarkoittaa jatkuvaa viestintää ja palautetta. Asiakas ja projektipäällikkö ovat mukana kuvaamassa liiketoimintalogiikkaa ja odotettua arvoa. Kuvassa 2 on XP:n elinkaari. (Altexsoft 2021, n.d.).

Kuva 2 XP:n elinkaari (Altexsoft 2021, n.d.)



XP:n kehittämisprosessi vaatii useiden osallistujien yhteistyötä ja jokaisella on omat tehtävät ja vastuualueet. Sosiaaliset taidot kuten viestintä, yhteistyö, reagointikyky, palautteen arvo ja tärkeys korostuvat XP:n rooleissa**. Customers** eli **asiakkaat** ovat kehitysprosessissa mukana luomalla käyttäjätarinoita, antamalla jatkuvaa palautetta ja tekemällä projektiin liittyvät liiketoimintapäätökset (Altexsoft 2021, n.d.).

Programmers eli **kehittäjät** ovat tiimin jäseniä, jotka toteuttavat ohjelmiston kehittämisen. Heidän vastuulla on käyttäjätarinoiden toteuttaminen ja käyttäjätestien tekeminen, ellei testaajille ole erillistä roolia. XP:n tiimit ovat monialaisia, joten tiimissä on paljon erilaisia taitoja. **Trackers** tai **managers** eli **seuraajat** tai **johtajat** yhdistävät kehittäjät ja asiakkaat. Rooli ei ole pakollinen ja sen voi tehdä myös joku kehittäjistä. Heidän tehtäviään on tapaamisten järjestäminen, keskustelujen säätely ja tärkeiden edistymisen tunnuslukujen seuranta. **Coaches** eli **valmentajat** voidaan ottaa mentoreiksi tiimeihin opastamaan XP-käytäntöjen ymmärtämisessä. Kyseessä on yleensä ulkopuolinen avustaja tai konsultti. Hän ei osallistu varsinaiseen kehitysprosessiin, mutta hän on XP-asiantuntija ja hänen avullansa voidaan välttää virheiden syntymistä (Altexsoft 2021, n.d.).

XP:n arvot ovat yksinkertaisia sääntöjä, joiden avulla ohjataan tiimin työskentelyä. **Communication** eli **viestintä** tarkoittaa sitä, että tiimin kaikki jäsenet työskentelevät yhdessä projektin jokaisessa vaiheessa. **Simplicity** eli **yksinkertaisuus** tarkoittaa sitä, että kehittäjät pyrkivät pitämään koodin kirjoittamisen mahdollisimman yksinkertaisena, koska silloin voidaan tuottaa tuotteelle enemmän arvoa ja säästetään aikaa ja vaivaa. **Feedback** eli **palaute** tarkoittaa sitä, että ohjelmistoa toimitetaan jatkuvasti ja siitä saadun palautteen perusteella tuotetta voidaan parantaa uusien vaatimusten mukaisesti. **Respect** eli **kunnioitus** tarkoittaa sitä, että kaikki projektissa mukana olevat työntekijät edistävät yhteistä tavoitetta. **Courage** eli **rohkeus** tarkoittaa sitä, että kehittäjillä on rohkeutta arvioida omia töitään objektiivisesti ja rohkeutta reagoida muutostarpeisiin, kun niitä ilmenee. (Altexsoft 2021, n.d.)

Periaatteet yhdistävät XP:n arvot ja käytännöt. **Rapid feedback** eli **nopea palaute** tarkoittaa sitä, että tiimin jäsenet ymmärtävät annetun palautteen sisällön ja osaavat reagoida siihen välittömästi. **Assumed simplicity** eli **oletettu yksinkertaisuus** tarkoittaa sitä, että kehittäjät pystyvät keskittymään olennaiseen käynnissä olevissa töissään. He osaavat jättää turhat osat pois, eivätkä tee päällekkäisiä töitä. **Incremental changes** eli **inkrementaaliset muutokset** tarkoittavat sitä, että kehitettävään tuotteeseen tehdään muutoksia pienissä osissa, koska se toimii paremmin, kuin kerralla tehdyt isot muutokset. **Embracing change** eli **muutoksen hyväksyminen** tarkoittaa sitä, että asiakkaan muutospyynnöt kehitettävään tuotteeseen hyväksytään ja kehittäjät suunnittelevat ratkaisun, kuinka uudet vaatimukset toteutetaan. **Quality work** eli **laadukas työ** tarkoittaa sitä, että tiimi panostaa kehitettävän tuotteen laatuun ja arvoon ja voi tuntea ylpeyttä työnsä tuloksista (Altexsoft 2021, n.d.).

XP:n käytännöt vahvistavat toisiaan yhdessä käytettynä. Niiden avulla voidaan vähentää kehitysprosessin riskejä ja kehitettävästä ohjelmistosta saadaan laadukas tuote. XP:n 12 käytäntöä ovat neljässä eri ryhmässä. **Palauteryhmään** kuuluvat testauslähtöinen kehitys, suunnittelupeli, paikan päällä oleva asiakas ja pariohjelmointi. **Jatkuvan prosessin ryhmään** kuuluvat koodin uudelleenmuokkaus, jatkuva integrointi ja pienet julkaisut. **Koodin ymmärtämisen ryhmään** kuuluvat yksinkertainen suunnittelu, koodin kollektiivinen omistus, järjestelmämetafora ja koodausstandardit. **Työolosuhteiden ryhmässä** on vain 40-tuntinen viikko. Kuvassa 3 on XP:n käytännöt ryhmissä (Altexsoft 2021, n.d.).

Kuva 3 XP:n käytännöt ryhmissä (Altexsoft 2021, n.d.)

Test-Driven Development (TDD) eli **testauslähtöinen kehitys** tarkoittaa sitä, automaattinen yksikkötesti kirjoitetaan ennen kuin itse koodia kirjoitetaan. Jokaisen koodinpätkän on läpäistävä

testi ennen kuin julkaiseminen on mahdollista. Testauslähtöisen kehityksen ansiosta kehittäjät voivat hyödyntää välitöntä palautetta tuottaakseen luotettavia ohjelmistoja (Altexsoft 2021, n.d.).

The Planning Game eli **suunnittelupeli** tarkoittaa kokousta, joka pidetään iteraatiosyklin alussa. Siinä kehitystiimi ja asiakas keskustelevat tuotteen ominaisuuksista ja hyväksyvät ne. Suunnittelupelin jälkeen kehittäjät suunnittelevat tulevan iteraation ja julkaisun sekä tehtävät jaetaan tiimin kesken. **On-site Customer** eli **paikan päällä oleva asiakas** tarkoittaa sitä, että asiakas osallistuu kehitystyöhön ja hänellä on mahdollisuudet vastata kysymyksiin, asettaa prioriteetteja ja tarvittaessa ratkaista riitatilanteita (Altexsoft 2021, n.d.).

Pair programming eli **pariohjelmointi** tarkoittaa sitä, että kaksi ohjelmoijaa työskentelee yhdessä saman koodin kanssa. Samalla kun toinen kirjoittaa koodia, toinen tarkastelee sitä, ehdottaa parannuksia ja korjaa virheitä. Pariohjelmointi mahdollistaa laadukkaiden ohjelmistojen kehittämisen ja nopean tiedon jakamisen. Pariohjelmointi vie kuitenkin noin 15 prosenttia enemmän aikaa, joten sitä suositellaan enemmän pitkäkestoisissa projekteissa (Altexsoft 2021, n.d.).

Code Refactoring eli **koodin refaktorointi** tarkoittaa tarpeettomien toimintojen poistamista, koodin johdonmukaisuuden lisäämistä ja elementtien irrottamista toisistaan. Refaktoroinnin avulla voidaan tuottaa lisäarvoa jokaisessa lyhyessä iteraatiossa ja parantaa koodia jatkuvasti (Altexsoft 2021, n.d.).

Continuous integration eli **jatkuva integrointi** tarkoittaa sitä, että järjestelmä pidetään aina täysin integroituna. Iteratiivinen kehitys nousee uudelle tasolle, kun integrointi tapahtuu useita kertoja päivässä ja tätä kutsutaan myös jatkuvaksi toimitukseksi. Kehittäjät keskustelevat siitä, mitä osia koodista voidaan käyttää uudelleen tai jakaa, ja sen vuoksi he tietävät tarkalleen mitä toimintoja on kehitettävä. Koodin kollektiivinen omistus auttaa poistamaan integrointiongelmat ja automaattinen testaus auttaa havaitsemaan ja korjaamaan virheet ennen käyttöönottoa (Altexsoft 2021, n.d.).

Small releases eli **pienet julkaisut** tarkoittavat sitä, että **Minimum Viable Product (MVP)** eli pienin mahdollinen tuote julkaistaan pian ja sitä jatkokehitetään pienillä ja inkrementaalisilla päivityksillä.

Pienet julkaisut mahdollistavat jatkuvan palautteen, virheet huomataan aikaisessa vaiheessa ja voidaan seurata, kuinka tuote toimii tuotannossa (Altexsoft 2021, n.d.).

Simple Design eli **yksinkertainen suunnittelu** tarkoittaa sitä, että yksinkertaisin toimiva ratkaisu on paras ohjelmistosuunnittelussa ja monimutkaiset osuudet pitää poistaa. Kehitettävän ohjelmiston pitää läpäistä kaikki testit, siinä ei saa olla päällekkäistä koodia ja sen pitäisi sisältää mahdollisimman vähän metodeja ja luokkia. Suunnittelussa pitäisi myös selvästi näkyä kehittäjän aikomukset (Altexsoft 2021, n.d.).

Coding Standards eli **koodausstandardit** tarkoittavat sitä, että tiimin koodauskäytännöt ovat yhtenäisiä eli käytetään samanlaisia muotoja ja tyylejä koodin kirjoittamiseen. Koodausstandardit mahdollistavat sen, että tiimin jäsenien on helppo lukea, jakaa ja refaktoroida koodia. Koodausstandardien avulla nähdään, kuka koodia on kirjoittanut ja ne nopeuttavat kehittäjien oppimista. Kun koodi on kirjoitettu samojen sääntöjen mukaan, se kannustaa koodin kollektiiviseen omistajuuteen (Altexsoft 2021, n.d.).

Collective Code Ownership eli **koodin kollektiivinen omistus** tarkoittaa sitä, että koko tiimi on vastuussa järjestelmän suunnittelusta ja kaikilla on oikeus tarkistaa ja päivittää koodia. Kollektiivisen omistuksen ansiosta kehittäjät tietävät oikean paikan uuden koodin lisäämiseksi. Käytännön avulla voidaan välttää päällekkäisen koodin tekeminen ja se kannustaa tiimiä tekemään yhteistyötä sekä aktivoit tiimin tuomaan esiin uusia ideoita (Altexsoft 2021, n.d.).

System Metaphor eli **järjestelmämetafora** tarkoittaa yksinkertaista suunnittelua, jolla on joukko tiettyjä ominaisuuksia. Suunnittelu ja rakenne pitäisi olla sellainen, että myös uusi työntekijä ymmärtää ne. Luokkien ja metodien nimeäminen pitäisi olla johdonmukaista, esimerkiksi objektit tulisi nimetä niin kuin ne olisivat jo olemassa, jotta järjestelmän kokonaissuunnittelu olisi ymmärrettävää (Altexsoft 2021, n.d.).

40-Hour Week eli **40-tuntinen viikko** tarkoittaa sitä, että tarvitaan hyvin voivia ja levänneitä työntekijöitä, jotta nopea työskentely, tehokkuus ja tuotteen laadun ylläpitäminen olisivat mahdollisia. Kun työelämä ja yksityiselämä ovat tasapainossa, se estää työntekijöitä palamasta

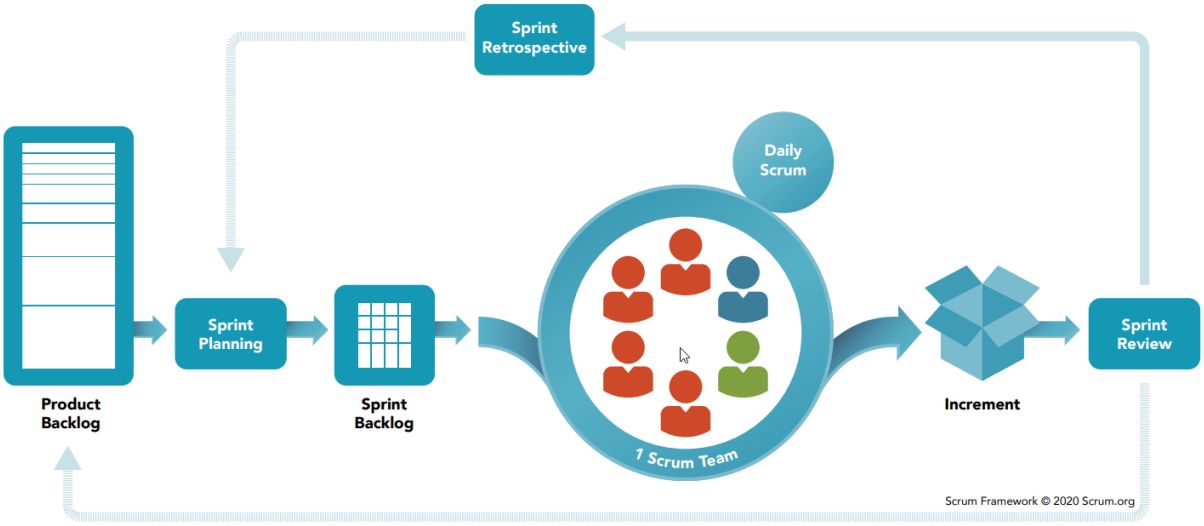
loppuun. Työtuntien määrä ei saisi ylittää yli 45 tuntia viikossa ja yksi ylityöpäivä viikossa on mahdollista, jos seuraavalla viikolla ei ole yhtään ylityöpäivää (Altexsoft 2021, n.d.).

2.3 Scrum

Vuonna 1986 julkaistussa artikkelissa ”The New New Product Development Game” Hirotaka Takeuchi ja Ikujiro Nonaka kertoivat ensimmäistä kertaa, kuinka voi saada huipputuloksia käyttämällä tiimipohjaista, skaalautuvaa ja kokonaisvaltaista menetelmää tuotekehityksessä. Artikkelilla on suuri merkitys Scrumiksi kutsutun menetelmän syntymiselle. Scrumin nimi on peräisin rugbysta, jossa vahingossa tapahtuneen rikkeen takia, aloitetaan peli uudelleen. Jeff Sutherland yhdessä tiiminsä kanssa kehittivät Scrum-prosessin vuonna 1993, jossa he yhdistivät vuoden 1986 artikkelin käsitteet ohjelmistokehitykseen. Ken Schwaber julkaisi ensimmäisen artikkelin Scrumista vuonna 1995. Tästä lähtien Schwaber ja Sutherland ovat yhdessä ja erikseen julkaisseet Scrumiin liittyviä julkaisuja mm. Agile Software Development with Scrum (Schwaber & Beedle 2001), Agile Project Management with Scrum (Schwaber 2004) ja “The Scrum Guide” (Schwaber & Sutherland 2011). Scrumia käytetään yleisimmin ohjelmistokehityksessä, mutta sen periaatteita ja arvoja voidaan hyödyntää mm. markkinoinnissa, myynnissä ja laitekehityksessä. (Rubin 2012, luku 1)

Scrum on kevyt, yksinkertainen ja ketterä ohjelmistokehityksen menetelmä. Se on viitekehys, jonka avulla tiimi pystyy joustavasti ratkomaan monimutkaisia ongelmia. Scrum-tiimissä on kolme erilaista roolia Product Owner eli tuoteomistaja, Scrum Master ja kehittäjät. Sprintti on kehitysjakso, jonka aikana tarvittavat tehtävät on tarkoitus tehdä. Se on johdonmukaisesti pituudeltaan vakio, korkeintaan kuitenkin vain kuukauden pituinen. Scrum Master luo ympäristön, jossa tuoteomistaja lisää tarvittavat tehtävät kehitysjonoon. Kehittäjät tekevät tarvittavat tehtävät sprintillä. Sprintin jälkeen arvioidaan tulokset ja niiden pohjalta päätetään seuraavan sprintin sisältö. Vaiheet toistuvat aina uuden sprintin alussa. Scrum korostaa vuorovaikutuksen ja yhteistyön merkitystä, eikä sen säännöissä anneta yksityiskohtaisia ohjeita, se on tarkoituksella epätäydellinen (Schwaber & Sutherland 2020, n.d.). Kuvassa 4 on Scrum-viitekehys (Scrumorg 2020, n.d.).

Kuva 4 Scrum-viitekehys (Scrumorg 2020, n.d.)



Scrum nojaa kahteen teoriaan, joista toinen on Lean-ajattelu ja toinen on empirismi. Lean-ajattelussa ideana on keskittyminen olennaiseen ja ylimääräisen vähentäminen. Empirismissä päätöksiä tehdään havaintojen perusteella ja tieto syntyy kokemuksesta. Scrumissa riskejä hallitaan ja ennustettavuutta optimoidaan inkrementaalisella (lisäävällä) ja iteratiivisella (asteittain tarkentuvalla) lähestymistavalla. Scrumin tarkoitus on koota asiantuntemus ja kaikki taidot samaan tiimiin, joka pystyy tekemään tarvittavat tehtävät ja hankkimaan ja jakamaan tarvittavaa tietoa (Schwaber & Sutherland 2020, n.d.).

Scrumin empiiriset peruspilarit ovat läpinäkyvyys, tarkastelu ja mukauttaminen. **Läpinäkyvyyden** ajatuksena on, että scrum-tiimissä kaikki ovat koko ajan tietoisia siitä, mitä muut tekevät. Riskit kasvavat ja voi syntyä arvoa heikentäviä päätöksiä, jos toiminta ei ole läpinäkyvää. Kun toiminta on läpinäkyvää, se mahdollistaa tarkastelun. Säännöllinen ja huolellinen **tarkastelu** on erittäin tärkeää, jotta voidaan havaita mahdollisia ongelmia ja poikkeamia prosessissa. Tarkastelun avulla on mahdollista löytää muutostoiminpiteet eli mukauttaminen. **Mukauttamisessa** on kyse siitä, että Scrum-tiimin tarkastelun avulla löydetyt muutostoiminpiteet on tehtävä mahdollisimman nopeasti, jotta voidaan minimoida mahdolliset lisäpoikkeamat. Scrum-tiimillä on oltava valtuudet tehdä muutostoiminpiteet (Schwaber & Sutherland 2020, n.d.).

Scrumin arvot ovat sitoutuminen, keskittyminen, avoimuus, kunnioitus ja rohkeus. **Sitoutumisella** tarkoitetaan sitä, että Scrum-tiimi tukee toisia jäseniään ja tekee parhaansa, että asetettuihin tavoitteisiin päästään. **Keskittyminen** tarkoittaa sitä, että Scrum-tiimi keskittyy sprintin tehtäviin

ensisijaisesti, jotta tavoitteisiin päästään. **Avoimuudella** tarkoitetaan sitä, että työn ja sen haasteiden suhteen ollaan avoimia Scrum-tiimissä ja siihen liittyvien sidosryhmien välillä. **Kunnioituksella** tarkoitetaan sitä, että Scrum-tiimin sisällä ja yhteistyökumppanien välillä kunnioitetaan yksilöiden osaamista ja kyvykkyyksiä ja sitä, että he suoriutuvat hyvin heidän omien osa-alueidensa tehtävistä. **Rohkeudella** tarkoitetaan sitä, että Scrum-tiimillä on rohkeus luottaa omaan ammattitaitoon, kun tehdään oikeita asioita. Scrum-tiimi uskaltaa tehdä rohkeita muutostoiminpiteitä, jos ollaan menossa väärään suuntaan tai ominaisuus ei toimi kuten sen pitäisi toimia (Schwaber & Sutherland 2020, n.d.).

Scrum-tiimin työn, toiminnan ja käyttäytymisen tulisi vahvistaa näitä arvoja. Arvojen kautta pääsee oikeuksiinsa myös Scrumin empiirinen perusta eli läpinäkyvyys, tarkastelu ja mukauttaminen. **Scrum-tiimi** on pieni ryhmä, johon kuuluu yksi Scrum Master, yksi tuoteomistaja ja kehittäjiä. Tiimissä on korkeintaan 10 jäsentä ja tiimin sisällä ei ole hierarkiaa tai alaryhmiä. Tiimin ollessa riittävän pieni, siinä pystytään toimimaan ketterästi ja kommunikaatio on parempaa. Tiimikoon kasvaessa liian suureksi se on syytä jakaa pienemmiksi sama tuotetta tekeviksi Scrum-tiimiksi. Tällaisessa tapauksessa usealla tiimillä on sama tuotteen tavoite, tuotteen kehitysjono ja tuoteomistaja. Tuotteen tavoite on tiimille tärkein ja sitä kohti mennään askel kerrallaan. Tiimi koostuu monialaisesta osaamisesta ja taidoista, joita tarvitaan arvon tuottamiseksi sprintin aikana. Heillä on valtuudet tehdä itsenäisiä ratkaisuja sprintin tehtävissään. Tiimin vastuulla on viestintä, laadun varmistus, ylläpito, tuotanto, kokeilu, tutkimus, kehittäminen ja kaikki mikä liittyy tuotteen prosessiin (Schwaber & Sutherland 2020, n.d.).

Scrum Master on vastuussa Scrum-ohjeiden teorian ja käytännön toteuttamisesta **Scrum-tiimiä** varten. Hän antaa tiimille mahdollisuudet kehittää käytäntöjä Scrum-viitekehyksessä.
Hän johtaa tiimiä esimerkillään ja opastaa tiimin jäseniä itseohjautuvuuteen ja monialaisuuteen. Hän auttaa tiimin jäseniä keskittymään arvokkaisiin ja olennaisiin tehtäviin. Hän varmistaa, että esteet tiimin etenemiseltä poistetaan. Hän varmistaa, että kaikki Scrumin vaiheet toteutetaan hyvässä hengessä, tuottavasti ja aikataulussa (Schwaber & Sutherland 2020, n.d.).

Scrum Master on monin tavoin tekemisissä **tuoteomistajan** kanssa. Hän avustaa tuoteomistajaa tuotteen kehitysjonon hallinnassa ja löytämään tehokkaita tapoja tuotteen tavoitteen määrittelyyn. Hän avustaa tiimiä löytämään johdonmukaisen ja selkeän sisällön tuotteen

kehitysjonoon. Hän avustaa kokeiluihin perustuvien tapojen luomisessa, kun tehdään suunnittelutyötä monimutkaisessa ympäristössä. Hän avustaa yhteistyötä sidosryhmien välillä (Schwaber & Sutherland 2020, n.d.).

Scrum master johtaa, kouluttaa ja valmentaa **organisaatiota** Scrumin käyttöönotossa. Hän suunnittelee ja neuvoo organisaatiota Scrumin toteuttamisessa. Hän avustaa havaintoihin perustuvan monimutkaisen lähestymistavan hyödyntämisessä ja ymmärtämisessä sekä sidosryhmiä, että työntekijöitä. Hänen tehtäviin kuuluu myös esteiden poistaminen Scrum-tiimien ja sidosryhmien väliltä (Schwaber & Sutherland 2020, n.d.).

Tuoteomistajan rooli Scrum-tiimissä on varmistaa, että tehty työ tuottaa mahdollisimman suurta arvoa. Hänen tehtävänään on asettaa tavoitteet tuotteelle ja niihin liittyvä täsmällinen viestintä. Hän vastaa kehitysjonon sisällön valmistelusta, järjestämisestä ja niihin liittyvästä viestinnästä. Hän varmistaa myös, että kehitysjono on läpinäkyvä, saatavilla ja ymmärrettävissä. Tuoteomistaja on vastuussa näistä tehtävistä, mutta hän voi halutessaan delegoida tehtäviä muille. Tuoteomistajan tekemät päätökset näkyvät kehitysjonon sisällössä ja järjestyksessä sekä sprintin lopussa tarkasteltavassa inkrementissä, joka on kooste sprintin aikana tehdyistä tehtävistä. Organisaation kunnioitus tuoteomistajan päätöksille on hänen onnistumisensa kannalta todella tärkeää. Vain yksi henkilö voi olla Scrum-tiimin tuoteomistaja ja tuotteen kehitysjonon avulla, hän voi edustaa useiden sidosryhmien tarpeita. Tuoteomistaja pitää ensin vakuuttaa muutosten tarpeellisuudesta, jos tuotteen kehitysjonoa halutaan muuttaa (Schwaber & Sutherland 2020, n.d.).

Kehittäjät sitoutuvat tekemään jokaisella sprintillä olevat tehtävänsä ja tuottavat käyttökelpoista lisäarvoa kehitettävälle tuotteelle. Kehittäjien joukossa tarvitaan monipuolista osaamista, jotta kehittäjät voivat keskittyä ydinosaamiseensa sprintillä olevissa tehtävissään. Heidän tehtäviinsä kuuluu sprintin kehitysjonossa olevat tehtävät ja niissä laatuun panostaminen. He mukauttavat päivittäin suunnitelmia, jotta sprintin tavoitteet saavutetaan ja arvostavat toistensa ammattitaitoa (Schwaber & Sutherland 2020, n.d.).

Scrumin tapahtumat on suunniteltu niin, että Scrumin empiirinen perusta, eli läpinäkyvyys, tarkastelu ja mukauttaminen ovat mahdollisia. Tapahtumat vähentävät ulkopuolisten palaverien

tarvetta ja luovat säännöllisyyttä. **Sprintti** on Scrumin tapahtumajakso, joka sisältää kaikki muut tapahtumat. Sprintin pituus pidetään loogisuuden vuoksi vakiona ja maksimissaan kuukauden kestoisena. Edellisen sprintin päättyessä aloitetaan heti seuraava sprintti. Sprintin aikana laatu ei saa heikentyä ja muutokset eivät saa vaarantaa sprintin tavoitteita. Tuotteen kehitysjonoa voidaan tarpeen mukaan jalostaa ja sprintin sisällön tarkennuksista voidaan neuvotella tuoteomistajan kanssa, jos sellaiselle on tarvetta. Tuotteen tavoitetta tarkastellaan ja mukautetaan vähintään kerran kuukaudessa, jonka ansiosta ennustettavuus paranee. Sprintin tavoite voi muuttua tarpeettomaksi, monimutkaisuus voi lisääntyä ja riskit kasvaa, jos sprintti on liian pitkä. Oppiminen on nopeampaa ja riskit sekä kustannukset ovat pienempiä, kun sprintin keston pitää lyhyenä. Edistymiskäyrät ja kertymäkuvaajat ovat hyödyllisiä, mutta ne eivät korvaa kokemuksien merkitystä, sillä monimutkaisessa ympäristössä on vaikea ennustaa tulevia tapahtumia. Vain tuoteomistajan päätöksellä on mahdollista keskeyttää sprintti, jos tavoite muuttuu tarpeettomaksi. Sprintin työvaiheet sisältävät sprintin suunnittelun, päivittäispalaverit, sprintin katselmoinnin ja sprintin retrospektiivin (Schwaber & Sutherland 2020, n.d.).

Sprintin suunnittelu käynnistää sprintin ja suunnitelma tehdään yhdessä koko Scrum-tiimin kanssa. Sprintin suunnitteluvaiheessa tuoteomistajan tehtävänä on saada osallistujat keskustelemaan tuotteen kehitysjonon tärkeimmistä kohdista ja niiden yhteyksistä tuotteen tavoitteeseen. Ulkopuolisia asiantuntijoita on mahdollista käyttää sprintin suunnitteluvaiheessa, jos se koetaan tarpeelliseksi. Sprintin suunnitteluvaiheessa etsitään vastauksia kolmeen kysymykseen. Miksi tämä sprintti on arvokas? Mitä tässä sprintissä voidaan saada valmiiksi? Miten valittu työ saadaan valmiiksi? (Schwaber & Sutherland 2020, n.d.).

Miksi tämä sprintti on arvokas? Pohditaan tuoteomistajan johdolla vastausta siihen, että millä keinoilla tuotteen hyödyllisyyttä ja arvoa voidaan kasvattaa sprintin aikana. Sprintille määritellään tavoite yhdessä koko Scrum-tiimin kanssa ja perustellaan sprintin arvokkuus sidosryhmille. Ennen kuin sprintin suunnittelu päättyy, tavoite viimeistellään. **Mitä tässä sprintissä voidaan saada valmiiksi?** Tuoteomistaja ja kehittäjät käyvät keskusteluja siitä, mitkä tehtävät valitaan mukaan alkavalle sprintille. Ymmärryksen ja ennustettavuuden lisäämiseksi Scrum-tiimi voi näiden keskusteluiden aikana tarkentaa tehtävien sisältöä. Uudessa Scrum-tiimissä on vaikea arvioida, kuinka paljon sprintillä on mahdollista saada aikaiseksi. Kokemuksen myötä Scrum-tiimi oppii tuntemaan suorituskykynsä, tulevan kapasiteettinsa ja valmiin määritelmän, jonka ansiosta

ennustettavuus paranee. **Miten valittu työ saadaan valmiiksi?** Kehittäjät saavat itsenäisesti päättää miten he sprintillä olevat tehtävänsä suunnittelevat ja toteuttavat. Tärkeitä on se, että askeleet kohti tavoitetta tuottavat arvoa ja vastaavat lopulta valmiin määritelmää. Kehitysjono muodostuu sprintin tavoitteesta, sprintille valituista tehtävistä ja suunnitelmasta niiden toteuttamiseksi (Schwaber & Sutherland 2020, n.d.).

Päivittäispalaverissa tarkastellaan, miten sprintin tavoitetta kohti on edistytty, kartoitetaan mahdollisia mukauttamistarpeita sprintin kehitysjonoon ja käydään läpi tulevia tehtäviä. Tämä palaveri on suunniteltu kehittäjiä varten, se on loogisesti samaan aikaan samassa paikassa ja sen kesto on vain 15 minuuttia. Scrum Master ja tuoteomistaja voivat myös osallistua palavereihin kehittäjän roolissa, jos he toteuttavat aktiivisesti jotain kehitysjonon tehtäviä. Päivittäispalaverin toteuttamistapa ja rakenne on itsenäisesti kehittäjien päätettävissä. Tärkeintä on, että tarkastellaan etenemistä kohti sprintin tavoitetta ja käyttökelpoinen suunnitelma päivän töihin löydetään. Päivittäispalaverit vähentävät tarvetta pitää muita kokouksia. Ne mahdollistavat paremman keskittymisen, itseohjautuvuuden, vuorovaikutuksen ja nopeat päätökset. Viestintä ei kuitenkaan rajoitu pelkästään päivittäispalavereihin. Kehittäjät kommunikoivat pitkin päivää sprintillä olevista tehtävistä ja niitä suunnitellaan ja mukautetaan aina tarpeen mukaan (Schwaber & Sutherland 2020, n.d.).

Sprintin katselmoinnissa Scrum-tiimi esittää työnsä tulokset sidosryhmille, siinä keskustellaan edistymisestä kohti tuotteen tavoitetta ja selvitetään mahdolliset muutostarpeet. Katselmoinnissa keskustellaan mahdollisista muutoksista tuotteen toimintaympäristössä, mitä sprintissä on saavutettu ja mitä kannattaa tehdä seuraavaksi. Katselmoinnissa voidaan myös muokata tuotteen kehitysjonoa vastaamaan paremmin uusiin mahdollisuuksiin. Katselmoinnin ei tulisi olla vain pelkkä demo, vaan ennemminkin työpaja. Katselmointi on sprintin toiseksi viimeinen tapahtuma ja sen kesto riippuu sprintin kestosta. Jos sprintin kestää kuukauden, niin katselmoinnin pituus on enintään neljä tuntia. Lyhyemmällä sprintillä katselmointikin voi olla lyhyempi (Schwaber & Sutherland 2020, n.d.).

Sprintin retrospektiivissä suunnitellaan keinoja tehokkuuden ja laadun parantamiseksi. Retrospektiivissä tarkastellaan sprintin sujuvuutta ihmisten, yhteistyön, prosessien, työkalujen ja valmiin määritelmän näkökulmista. Työn sisältö määrittelee, mitkä asiat ovat tarkastelussa.

Retrospektiivissä käydään läpi sprintin onnistumiset, ongelmat, harhaan johtaneet oletukset ja mitä ongelmia saatiin ratkaistua tai mitä jäi vielä ratkaisematta. Tehokkuuden parantamiseksi hyödyllisimmät muutokset tunnistetaan ja vaikuttavimmat parannukset toteutetaan parhaimmillaan seuraavalla sprintillä. Retrospektiivi on sprintin viimeinen vaihe ja se kestää korkeintaan kolme tuntia, jos sprintin kesto on kuukausi. Lyhyemmällä sprintillä myös retrospektiivi voi olla lyhyempi (Schwaber & Sutherland 2020, n.d.).

Scrumin tuotoksissa korostuu tiedon läpinäkyvyyden merkitys, koska silloin Scrum-tiimin kaikilla jäsenillä on samat lähtökohdat tuotosten mukauttamiseen. **Sidoksilla** varmistetaan läpinäkyvyys ja mitattavissa oleva edistyminen. Tuotteen kehitysjono on sidottu tuotteen tavoitteeseen. Sprintin kehitysjono on sidottu sprintin tavoitteeseen. Inkrementti on sidottu valmiin määritelmään. Nämä sidokset edistävät Scrumin arvojen toteutumista ja ne vahvistavat havaintoihin perustuvaa toimintaa (Schwaber & Sutherland 2020, n.d.).

Tuotteen kehitysjono on ainoa lähde Scrum-tiimissä tehtävälle työlle. Se on tuotteen kehitykseen tehty vähitellen syntyvä ja järjestetty lista. Kohtia, jotka tiimi voi saada valmiin määritelmän mukaan valmiiksi sprintin aikana, katsotaan valmistelluiksi. Sprintin suunnittelussa voidaan valmistelluista kohdista valita kohtia toteutettavaksi. Läpinäkyvyys saavutetaan jalostamalla tuotteen kehitysjonon kohtia. Jalostaminen tapahtuu pilkkomalla ja tarkentamalla kehitysjonon kohtia. Jalostuksessa tarkennetaan kuvausta, järjestystä ja kokoa, se on jatkuvaa toimintaa. Kehittäjät saavat itsenäisesti arvioida sprintillä tehtävän työmäärän. Tuoteomistaja voi auttaa kehittäjiä valitsemaan eri vaihtoehdoista sopivimmat (Schwaber & Sutherland 2020, n.d.).

Sidos tuotteen tavoitteeseen tarkoittaa tuotteen pitkän aikavälin tavoitetta. Tiimin pitää joko saavuttaa tai hylätä kukin tavoite, ennen kuin voidaan siirtyä seuraavaan. Tuotteen tavoite on kehitysjonossa ja kehitysjonon muut kohdat syntyvät matkan varrella ja niiden kautta pyritään kohti tuotteen tavoitetta (Schwaber & Sutherland 2020, n.d.).

Sprintin kehitysjono on suunnitelma, jonka kehittäjät tekevät itseään varten. Se koostuu sprintin tavoitteesta, kehitysjonon kohdista ja käytännön suunnitelmasta inkrementin toteuttamiseksi. Sprintin kehitysjono on reaaliaikainen ja läpinäkyvä kuva työstä, jonka kehittäjät ovat suunnitelleet tekevänsä sprintin aikana. Sprintin kehitysjono tarkentuu sprintin edetessä, kun tieto lisääntyy.

Yksityiskohtainen sprintin kehitysjono on hyödyllinen edistymisen tarkasteluun päivittäispalavereissa (Schwaber & Sutherland 2020, n.d.).

Sidos sprintin tavoitteeseen tarkoittaa sitä, että sprintin ainoa päämäärä on tavoitteen saavuttaminen. Kehittäjät sitoutuvat sprintin tavoitteeseen, mutta heillä on vapaus tehdä tarvittavat tehtävänsä joustavasti. Sprintin tavoite motivoi tiimiä yhteistyöhön ja parantaa sekä keskittymistä että johdonmukaisuutta. Sprintin suunnittelussa määritellään sprintin tavoite, jonka jälkeen se lisätään sprintin kehitysjonoon. Sprintin aikana työskennellessään, kehittäjät pitävät sprintin tavoitteen mielessään. Tuoteomistajan kanssa voidaan sopia kehitysjonon laajuudesta, jos työ osoittautuukin erilaiseksi, kuin odotettiin. Nämä muutokset eivät kuitenkaan saa vaikuttaa sprintin tavoitteeseen (Schwaber & Sutherland 2020, n.d.).

Inkrementti tarkoittaa konkreettista askelta kohti tuotteen tavoitetta. Inkrementtien pitää toimia keskenään yhdessä, koska jokainen inkrementti on lisäys aiempiin inkrementteihin. Inkrementin pitää olla myös käyttökelpoinen, jotta se tuottaa arvoa. Inkrementtejä voi olla useita yhden sprintin aikana ja niiden kokonaisuus esitellään sprintin katselmoinnissa. Havaintoihin perustuva empirismi toteutuu, kun toimitaan tällä tavalla. Inkrementti voidaan jakaa sidosryhmille myös ennen sprintin päättymistä, koska toiminnan läpinäkyvyyden vuoksi, arvokaan tiedon jakamista ei kannata odottaa. Valmiin määritelmän tulee täyttyä kaikessa työssä, joka liittyy inkrementtiin (Schwaber & Sutherland 2020, n.d.).

Sidos valmiin määritelmään tarkoittaa sitoutumista siihen inkrementin tilaan, jossa tuotteelle asetetut laatuvaatimukset täyttyvät. Kun tuotteen kehitysjonon kohta täyttää valmiin määritelmän, syntyy inkrementti. Valmiin määritelmän ansiosta löydetään ymmärrys inkrementtiin liittyvistä töistä ja läpinäkyvyys paranee. Jos valmiin määritelmä ei jossain kehitysjonon kohdassa täyty, niin sitä ei voida julkaista eikä esitellä sprintin katselmoinnissa. Siinä tapauksessa tämä kohta palautetaan tuotteen kehitysjonoon harkittavaksi. Kaikkien Scrum-tiimien tulee noudattaa valmiin määritelmää, jos se kuuluu organisaation standardeihin. Scrum-tiimin pitää itse luoda sopiva valmiin määritelmä, jos sitä ei ole organisaatiossa määritelty standardiksi. Tuotetta kehittävä Scrum-tiimi tai mahdollisesti useampi Scrum-tiimi, määrittelee ja sitoutuu yhdessä noudattamaan samaa valmiin määritelmää (Schwaber & Sutherland 2020, n.d.).

2.4 Lean

Lean termiä on käytetty valmistuksessa jo vuosikymmenien ajan. Lean on alun perin autovalmistaja Toyotalta lähtöisin, mutta Tom ja Mary Poppendieck sovittivat sen toimimaan ohjelmistokehityksessä 2000-luvun alussa. Lean on ajattelutapa ja termi Lean-ajattelu onkin yleisesti käytössä. Leanissa on arvoja ja niihin liittyviä periaatteita, joita kutsutaan ajattelutyökaluiksi (Stellman ym., 2014, s. 269).

Lean-ajattelu on hyvin osuva termi, koska menetelmän tehokas käyttöönotto edellyttää tietynlaista ajattelutapaa. Leanin ajattelutapa löydetään seitsemän arvon kautta. **Poista hukka** tarkoittaa sitä, että löydetään kehitettävästä ohjelmistosta elementit, jotka eivät tuota arvoa projektille. **Vahvista oppimista** tarkoittaa sitä, että projektista saatua palautetta käytetään ohjelmiston kehittämisessä. **Päätä mahdollisimman myöhään** tarkoittaa sitä, että projektin tärkeimmät päätökset tehdään silloin, kun tietoa on mahdollisimman paljon saatavilla. **Toimita mahdollisimman nopeasti** tarkoittaa sitä, että viiveen kustannukset ymmärretään ja pyritään minimoimaan jonon ja vetojärjestelmän avulla. Vetojärjestelmä tarkoittaa uuden työn aloittamista vasta silloin, kun sille on kysyntää (Stellman ym., 2014, s. 270–271).

Vahvista tiimiä tarkoittaa sitä, että rakennetaan tehokas ja hyvän keskittymisen mahdollistava työympäristö ja kootaan energinen tiimi. **Rakenna eheys** tarkoittaa sitä, että kehitettävä ohjelmisto on käyttäjien kannalta järkevä ja muodostaa yhtenäisen kokonaisuuden.
Näe kokonaisuus tarkoittaa sitä, että ymmärretään työt, joita projektissa tehdään. Varmistetaan oikeanlaisilla mittareilla, että nähdään kaikki projektin vahvuudet ja ongelmat.
Leanin arvojen avuksi on ajattelutyökaluja, joiden avulla tiimi pystyy soveltamaan arvoja oikeissa tilanteissa (Stellman ym., 2014, s. 270–271).

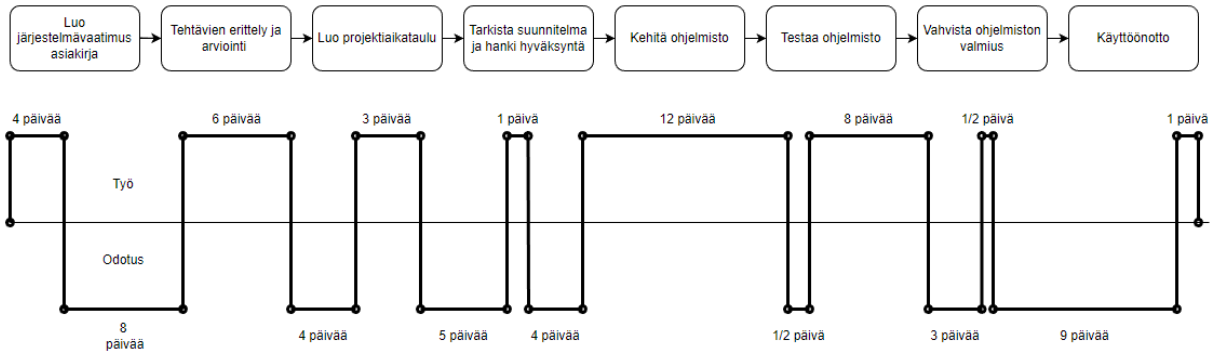
Poista hukka on Leanin ensimmäinen arvo ja sen ensimmäinen ajattelutyökalu on **näe hukka**. Hukkaa voi olla vaikea huomata, koska siitä on usein vastuussa tiimin ulkopuolinen henkilö, esimerkiksi projektipäällikkö tai ylin johto. Osan hukasta tiimi saattaa hyväksyä, koska he tietävät, että budjetointi on aikaa vievää eikä suoraan liity ohjelmointiin. Joskus hukka voi olla näkymätöntä, esimerkiksi jos työtila on suunniteltu niin, että tiimin jäsenet istuvat kaukana toisistaan ja aikaa kuluu kävelemiseen toisen työpisteelle keskustelemaan. Poppendieckit keksivät

Toyotalla kehitettyjen ideoiden pohjalta ajatuksen nimeltä **ohjelmistokehityksen seitsemän hukkaa** (Stellman ym., 2014, s. 284–285).

Osittain tehty työ tarkoittaa sitä, että työt on vain tehty, mutta lopputulos ei ole toimiva eikä valmis. Silloin toiminta ei ole tuottanut arvoa käyttäjille ja on hukkaa. **Ylimääräiset prosessit** tarkoittavat sitä, että työaikaa käytetään kohtuuttoman paljon esimerkiksi tilannepäivityksien ja seurantaraporttien tekemiseen, jotka eivät tuota mitään lisäarvoa. **Lisäominaisuudet** tarkoittavat sitä, että ohjelmistoon tehdään ominaisuuksia, joita käyttäjät eivät tosiasiassa tarvitse. **Tehtävien vaihdolla** tarkoitetaan sitä, että moniajaminen eri projektien välillä tai saman projektin toisiinsa liittymättömien tehtävien välillä, lisää odottamattomien viiveiden ja ongelmien määrää. **Odottamisella** tarkoitetaan sitä, että joudutaan odottamaan esimerkiksi tarkistuksia, käyttäjäoikeuksia, tietokoneongelmia ja lisenssejä. **Liikkeellä** tarkoitetaan sitä, että tiimin jäsenten istuessa kaukana toisistaan, heidän pitää kävellä pitkiä matkoja keskustellakseen keskenään. **Vioilla** tarkoitetaan sitä, että ohjelmiston varhaisella testauksella estetään paljon vikoja, joiden korjaaminen myöhemmin veisi paljon enemmän aikaa kuin itse testaaminen (Stellman ym., 2014, s. 285-286).

Leanissa työkaluna hukan löytämiseen käytetään arvovirtakuvausta eli **value stream mapping**. Siinä esitetään visuaalisesti kaikki prosessin vaiheet järjestelmävaatimuksista aina tuotteen toimitukseen saakka. Arvovirtakuvauksen luomisessa ei pitäisi kestää kuin puoli tuntia. Luomisessa käytetään **minimal marketable feature (MMF)** tekniikkaa, jossa aloitetaan pienimmästä osasta, jonka asiakkaat haluavat priorisoida ja edetään läpi kaikkien vaiheiden. Sitten arvioidaan, kauan aikaa kului ensimmäiseen työvaiheeseen ja kauan aikaa kului odottamiseen, ennen kuin seuraava vaihe aloitettiin. Tämä toistetaan jokaisen vaiheen kohdalla. Tiimi saa arvovirtakuvauksesta hyvää tietoa hukkaan menneestä ajasta. Visualisoinnin ansiosta tiimi löytää tapoja vähentää hukkaa ja lyhentää odotusaikoja seuraavien ominaisuuksien toimitusajoissa (Stellman ym., 2014, s. 287-288). Kuvassa 5 on esimerkki Lean arvovirtakuvauksesta.

Kuva 5 Lean arvovirtakuvaus



Ohjelmiston eheyteen liittyy **sisäinen eheys**, joka tarkoittaa ohjelmiston eheyttä kehittäjien näkökulmasta. Leanin arvo **rakenna eheys** sisältää kaksi ajatustyökalua sisäisen eheyden saavuttamiseksi, jotka ovat **refaktorointi** ja **testaus**. Refaktorointi on koodin uudelleenjärjestelyprosessi, jossa koodin alkuperäinen toimivuus ei muutu. Sen tavoite on parantaa koodin toimivuutta tekemällä pieniä muutoksia, jotka eivät kuitenkaan muuta koodin käyttäytymistä. Tehokas tapa rakentaa järjestelmä, jossa on korkea sisäinen eheysaste, on testauslähtöinen kehitys, refaktorointi ja inkrementaalinen suunnittelu. Inkrementaalisessa suunnittelussa ongelmat jaetaan pieniin osiin ja niitä työstetään yksitellen (Stellman ym., 2014, s. 288-289).

Ulkoinen eheys tarkoittaa ohjelmiston eheyttä käyttäjän näkökulmasta. Leanissa ulkoisen eheyden hallitsemiseen on kaksi ajatustyökalua, jotka ovat **koettu eheys** ja **käsitteellinen eheys**. **Koetulla eheydellä** tarkoitetaan sitä, kuinka hyvin tuote vastaa käyttäjien tarpeisiin ja kuinka käyttäjät välittömästi huomaavat tarpeidensa täyttyvän. **Käsitteellisellä eheydellä** tarkoitetaan sitä, kuinka hyvin ohjelmiston ominaisuudet toimivat yhdessä ja muodostavat yhtenäisen tuotteen (Stellman ym., 2014, s.289).

Leanin arvo **näe kokonaisuus** sisältää ajatustyökalun nimeltään **mittaus**. **Mittauksella** pyritään siihen, että kaikki tiimin jäsenet voisivat nähdä projektin samassa valossa. Tiimissä voidaan tehdä mittauksia monista eri asioista ja oikeiden mittausten valitseminen auttaa näkemään paremman kuvan projektista. Mittauksien valinta riippuu siitä, mikä ongelma halutaan ratkaista. Objektiiviset mittaukset saavat tiimin ajattelemaan projektia samasta näkökulmasta. Tiimi voi mitata esimerkiksi ominaisuuksien läpimenoaikaa, joka on keskimääräinen aika ominaisuuden pyynnöstä sen toimittamiseen saakka. Mittaustulokset ovat konkreettista faktaa, joilla voi osoittaa tiimille ja johdolle, jos jotain ongelmia havaitaan (Stellman ym., 2014, s.292-293).

Mittauksien lisäksi tarvitaan ymmärrystä löytää ongelmien perimmäinen syy, jotta kokonaisuus nähdään. Leanissa ongelmien perimmäisten syiden löytämiseen käytetään tekniikkaa nimeltä **Five Whys**. Tekniikassa kysytään, miksi ongelma tapahtui ja sitten vastataan kysymykseen. Tämä toistetaan yleensä noin viisi kertaa, kunnes ongelman perimmäinen syy löydetään (Stellman ym., 2014, s.293-294).

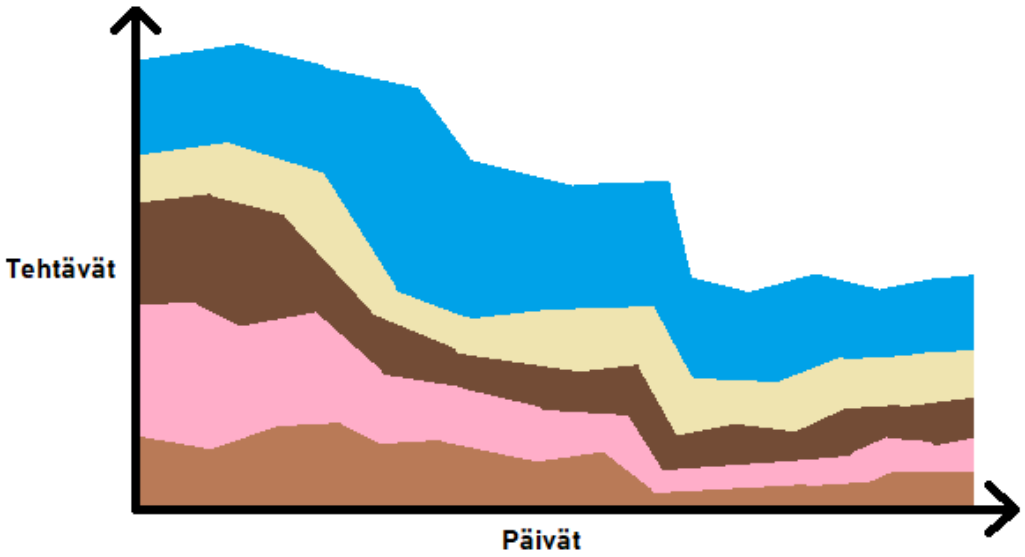
Leanin arvolla **toimita mahdollisimman nopeasti** on kolme ajattelutyökalua, jotka ovat **vetojärjestelmä**, **jonoteoria** ja **viivekustannus**. **Vetojärjestelmä** tarkoittaa tapaa, jossa uusi tehtävä aloitetaan vasta silloin, kun sille on kysyntää. Vetojärjestelmän avulla voidaan ajaa prosesseja, joissa käytetään jonoja tai puskureita rajoitusten poistamiseksi (Stellman ym., 2014, s.304).

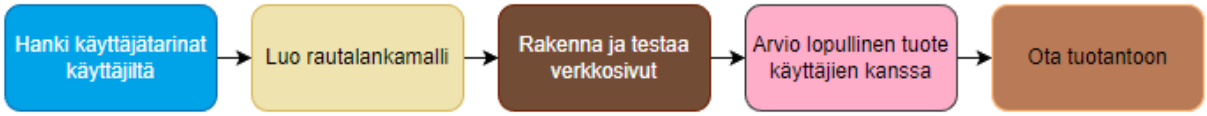
Jonoteoria on jonon matemaattista tutkimusta, johon kuuluu sen vaikutuksen ennustamista, kun jono otetaan käyttöön varhaisessa vaiheessa. Julkinen työjono auttaa tiimiä työskentelemään nopeammin. Jonoteorian ajatuksena on varmistaa, että työntekijät eivät ylikuormitu työstä ja heillä on riittävästi aikaa tehdä tehtävät oikein. Jono on tehtäväluettelo, jossa tehtävät tai ominaisuudet ovat listattuna joko yksittäiselle työntekijälle tai koko tiimille. Jonon tehtävät tehdään yleensä vanhimmasta tehtävästä aloittaen (Stellman ym., 2014, s.295-296).

Leanissa **viivekustannuksien** hallintaan on kehitetty **Toyota Production System** (**TPS**), jonka Poppendieckit mukauttivat ohjelmistokehitykseen sopivaksi. TPS:n ajatuksena on, että on olemassa kolme erilaista hukkaa nimeltään **Muda**, **Mura** ja **Muri**, jotka rajoittavat työnkulkua ja ne pitää poistaa. **Muda** tarkoittaa turhuutta, hyödytöntä, joutilaisuutta, liiallisuutta, hukkaa ja tuhlausta. **Mura** tarkoittaa epätasaisuutta, epäsäännöllisyyttä, yhtenäisyyden puutetta ja epätasa-arvoa. **Muri** tarkoittaa kohtuuttomuutta, mahdotonta, liian vaikeaa, väkisin tehtyä, pakolla tehtyä ja liiallisuutta (Stellman ym., 2014, s.304-305).

Leanissa toimituksen nopeuden mittaamiseen käytetään **work-in-progress (WIP)** aluekaaviota. Siinä näkyy kuinka ominaisuudet, tuotteet tai muut arvon mittarit kulkevat arvovirran jokaisen osan läpi. Sen tavoitteena on näyttää kaikki arvokkaat ominaisuudet, joiden parissa tiimi työskentelee (Stellman ym., 2014, s.296-304). Kuvassa 6 on esimerkki Lean WIP-aluekaaviosta.

Kuva 6 Lean WIP-aluekaavio





Lean tiimit sisältävät roolit Lean Master, Lean-projektinjohtaja ja Lean-tiimin jäsenet. **Lean Master** auttaa asiakasta valitsemalla tiimiin henkilöt, joilla on oikeat ja asiaankuuluvat taidot projektin läpiviemiseen. Hän ymmärtää Leanin työkalut ja tekniikat ja valmentaa, kouluttaa ja mentoroi tiimiä Leanin käytössä. Hän ylläpitää yleissuunnitelmaa ja vastaa muutostenhallinnasta. Hänellä on kokonaiskuva projektin vaiheista ja siitä mitä sen jälkeen tapahtuu. **Lean-projektinjohtaja** johtaa Lean-tiimejä ja projekteja. Hän raportoi edistymisestä ja poistaa esteitä. Hän vastaa viestinnästä, organisoinnista ja tiimin kehittämisestä. **Lean-tiimi** koostuu 6–9 jäsenestä. Heidän vastuullansa on prosessin kaikkien vaiheiden toteuttaminen. He ovat prosessin ja tehtäviensä asiantuntijoita esimerkiksi kehittäjiä tai testaajia. He työskentelevät yhteistyössä ratkaisujen suunnittelussa ja toteuttamisessa (Toolsqa 2019, n.d.)

2.5 Kanban

Nimi Kanban on japanin kieltä ja se tarkoittaa taulua. Kanban kehitettiin Japanissa Toyotan tehtaalla 40-luvun lopussa, kun Taiichi Onho ohjasi tuotantoa prosessien välillä toteuttaakseen Just in Time (JIT) tuotannon. Kuitenkin vasta 70-luvun laman aikana menetelmä nousi arvoonsa, koska Kanbanin avulla saatiin alennettua kustannuksia. Nykyään Kanbania käytetään kustannusten alentamisen ja virtauksen hallitsemisen lisäksi myös esteiden ja parantamismahdollisuuksien

tunnistamiseen. Kanbanin ansiosta Toyotan JIT-tuotannosta on tullut menestystarina (Gross ym., 2003, s.1-2).

Kanbanin sovitti ohjelmistokehitykseen David Anderson. Sitä käytetään prosessien optimointiin ja se auttaa näkemään selkeästi toimenpiteet, joita ohjelmiston rakentamiseen tarvitaan. Sen avulla voidaan parantaa vuorovaikutusta yrityksessä ja sitä käytetään epätasaisuuden, ylikuormituksen ja turhuuden hukan poistamiseen (Stellman ym., 2014, s.315-316).

Kanbanissa on yhdeksän arvoa, jotka motivoivat tiimityön ja tuotettujen palveluiden jatkuvaan kehittymiseen. **Avoimuus** tarkoittaa sitä, että tiedonvaihto on avointa, selkeää ja sanasto on yksiselitteistä. **Tasapaino** tarkoittaa sitä, että kaikkien osallistujien erilaiset kyvyt, näkemykset ja vaatimukset tasapainottamalla tehokkuus paranee. **Yhteistyö** on keskeinen osa Kanbania ja sen avulla yhteistyötä voidaan parantaa. **Asiakaskeskeisyys** tarkoittaa sitä, että yrityksen kaikilla osa-alueilla on luonnollinen kiinnostus asiakkaaseen ja heidän saamaansa arvoon (Meiling, 2018, n.d.). **Työnkulku** tarkoittaa sitä, että Kanbanin avulla voidaan tunnistaa, edustaako työ jatkuvaa tai satunnaista arvovirtaa ja tunnistamisen jälkeen sitä voidaan ylläpitää. **Johtaminen** tarkoittaa sitä, että arvon tuottamiseksi ja paremman tilan saavuttamiseksi, johtajuutta tarvitaan kaikilla tasoilla. **Ymmärtäminen** tarkoittaa sitä, että koko organisaatiolla ja yksittäisillä työntekijöillä on itsetuntemusta, jotta päästään eteenpäin. Kanbanin avulla kehitetään ja kehittäminen vaatii muutosta, johon vaaditaan ymmärrystä. Ongelma tunnistetaan ja sen syyt ja seuraukset määritellään. Työ ja siihen liittyvät prosessit on ensin ymmärrettävä, jotta niitä voidaan parantaa. Aloita siitä mitä teet ja ymmärrä miksi teet sitä (Meiling, 2018, n.d.).

Yhteisymmärrys tarkoittaa sitä, että osapuolet sopivat tavoittelevansa tavoitteita yhdessä. Kunnioitetaan erilaisia mielipiteitä ja lähestymistapoja ja näiden erilaisten näkökulmien tulisi lähentyä toisiaan. Lähentyminen on mahdollista silloin, kun kaikki osapuolet sitoutuvat yhdessä parantamaan kaikkia niitä prosesseja, jotka ovat välttämättömiä yhteisten tavoitteiden saavuttamiseksi. **Kunnioitus** on perusta kaikille muille Kanbanin arvoille. Kunnioitus ilmenee ihmisten arvostuksena, ymmärryksenä ja huomaavaisuutena. Lähde liikkeelle siitä mitä teet ja tarkastele miten se täyttää tai ei täytä ihmisten tarpeita organisaation sisällä ja sen ulkopuolella (Meiling, 2018, n.d.).

Kanbanissa on kolme muutosagendaa, joita organisaation menestyminen edellyttää. **Kestävyysagenda** koskee kestävän työtahtin löytämistä ja organisaation sisäisen keskittymisen optimointia. Tavoitteena on luoda palveluja, jotka tasapainottavat kysynnän ja olemassa olevan suorituskyvyn. Kysynnän ylittäessä suorituskyvyn, työntekijät kuormittuvat ylityöllä. Läpinäkyvyys työmäärässä otetaan käyttöön ja se asetetaan tasapainoon työntekijöiden suorituskyvyn kanssa. Työntekijöillä on siten valtuudet tehdä työtä säännöllisessä rytmissä ja he eivät ylikuormita itseään (Meiling, 2018, n.d.).

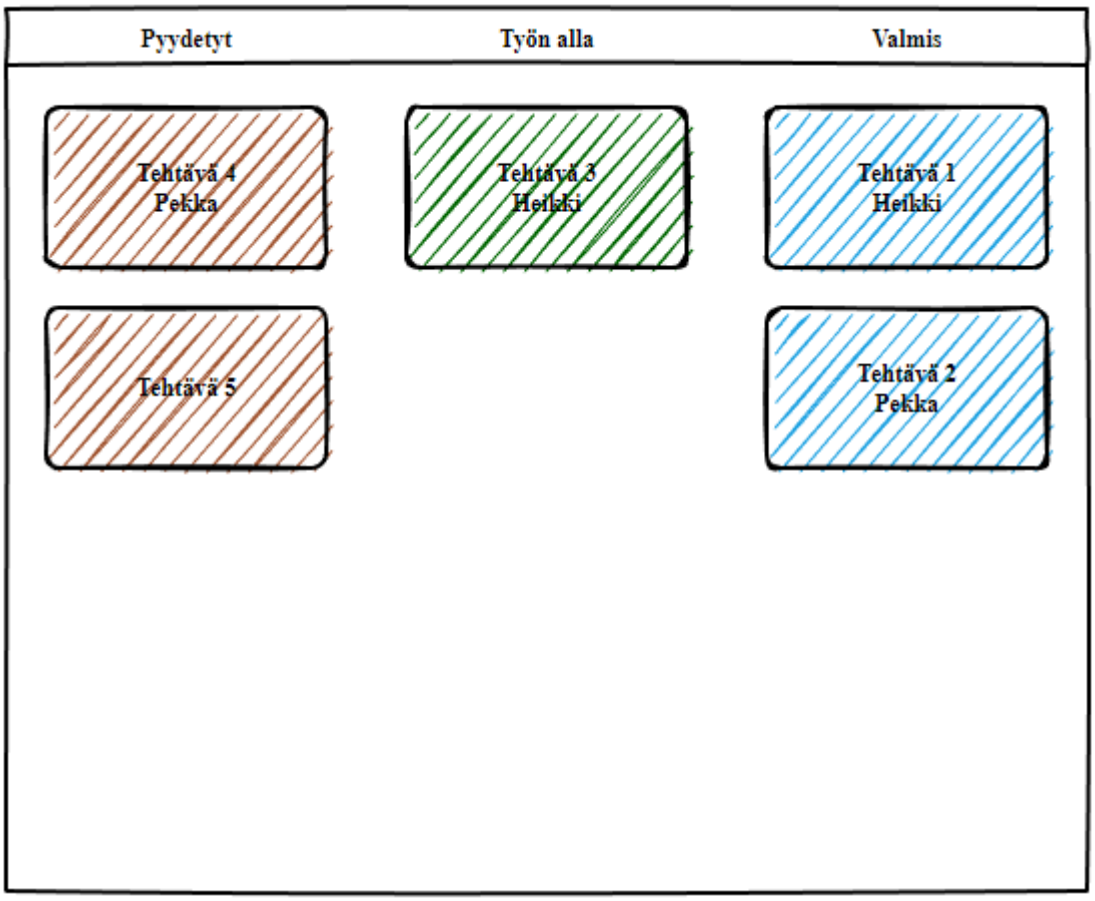
Palvelukeskeisyysagenda koskee yrityksen suorituskykyä ja asiakastyytyväisyyttä. Asiakkaisiin keskitytään ja aloitetaan yrityksen varsinaisesta tarkoituksesta. Palvelujen tarjoaminen ja niiden jatkuva kehittyminen on Kanbanin olennainen osa. Palvelukeskeisyysagendan tavoitteena on tarjota asiakkaille palveluja, jotka vastaavat yrityksen tarkoitusta. Toisaalta palveluiden pitää täyttää tai jopa ylittää asiakkaiden tarpeet ja odotukset. Saavutetaan erinomaisia tuloksia, kun kaikki työntekijät keskittyvät tähän. **Selviytymisagenda** koskee kilpailukykyä ja sopeutumiskyvyn säilyttämistä. Se keskittyy yrityksen tulevaisuuteen ja tavoitteena on varmistaa, että yritys jatkaa ja menestyy myös merkittävien muutosten aikana. Nykyhetkessä riittävät teknologiat ja prosessit on mukautettava jatkuvaan muutokseen. Kanbanin evolutiivinen lähestymistapa muutoksiin soveltuu erinomaisesti selviytymisagendan toteuttamiseen (Meiling, 2018, n.d.).

Kanbanissa on kahden tyyppisiä periaatteita, jotka ovat **muutoksenhallinnan periaatteet** ja **palveluntoimitusperiaatteet**. Muutoksenhallinnan periaatteista ensimmäinen on **aloita siitä mitä teet**. Sillä tarkoitetaan sitä, että Kanbania voidaan käyttää optimoimaan jo olemassa olevia työkulkuja ja prosesseja, koska ne ovat usein säilyttämisen arvoisia ja arvokkaita. Kanbanin avulla muutokset työkulkuihin ja prosesseihin voidaan toteuttaa ilman suurta häiriötä. **Suostu jatkamaan inkrementaalista ja evolutiivista muutosta** tarkoittaa sitä, että Kanban kannustaa tekemään jatkuvasti pieniä muutoksia nykyisessä prosessissa. Kanbanissa sitä toteutetaan yhteistyö- ja palautelomakkeiden avulla. **Rohkaise johtajuutta kaikilla tasoilla** tarkoittaa sitä, että jokainen työntekijä voi parantaa toimintatapoja omilla oivalluksillaan. Kaikki oivallukset tukevat jatkuvan kehittymisen ajattelutapaa ja niiden avulla voidaan saavuttaa optimaalinen suorituskyky tiimissä, osastolla tai koko yrityksen tasolla (Kanbanize, n.d.)

Palveluntoimitusperiaatteista ensimmäinen on **keskity asiakkaan tarpeisiin ja odotuksiin**. Sillä tarkoitetaan sitä, että organisaation keskeisimpiä tavoitteita tulisi olla arvon tuottaminen asiakkaille. Kun asiakkaiden tarpeet ja odotukset ymmärretään, heidän huomionsa kiinnittyy tuotteen tai palvelun luomaan arvoon ja laatuun. **Hallinnoi työtä** tarkoittaa sitä, että palveluverkoston työnjohto varmistaa, että työntekijät saavat keskittyä vahvuusalueidensa työtehtäviin. Siten voidaan keskittyä työn ja sen tuloksien hallinnointiin. **Palveluverkoston säännöllinen tarkastelu** tarkoittaa sitä, että palvelukeskeinen lähestymistapa vaatii jatkuvaa arviointia, jotta asiakaspalvelukulttuuria voidaan kehittää. Kanbanin avulla voidaan kehittää sovellettujen työkäytäntöjen arvioinnin ja palveluverkoston säännöllisen tarkastelun avulla saatuja tuloksia (Kanbanize, n.d.).

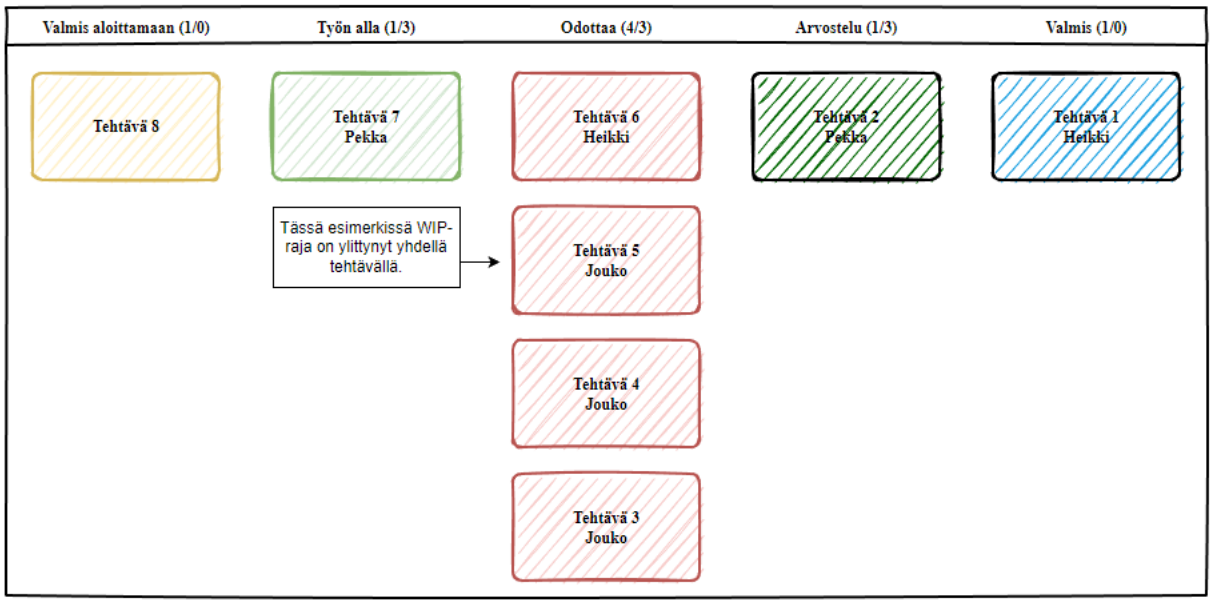
Kanbanissa on kuusi keskeistä käytäntöä. Käytännön vaiheiden kanssa pitää olla huolellinen ja niiden hallitseminen on kehittyvä prosessi. **Visualisoi työnkulku** tarkoittaa sitä, että työnkulun jokainen vaihe merkitään Kanban-tauluun, jossa on sarakkeita ja kortteja. Kanban-taulusta näkee työkulun todellisen tilan kaikkien riskien ja määrittelyiden kanssa. Sarakkeilta näkee työkulun vaiheet ja korteilta näkee työkohteet. Aluksi pitää ymmärtää, mitä tarvitaan, jotta tehtävä saadaan pyydetty sarakkeelta lopulta valmissarakkeelle. Kun tietää miten työ kulkee projektin aikana, pystyy havaitsemaan tarpeelliset muutokset ja parantamaan tuloksia jatkuvasti (Kanbanize, n.d.). Kuvassa 7 on esimerkki yksinkertaisesta Kanban-taulusta.

Kuva 7 Yksinkertainen Kanban-taulu



Limit Work in Progress (WIP) eli rajoita käynnissä olevia töitä tarkoittaa sitä, että käynnissä olevien tehtävien määrä pitää olla hallittavissa. Moniajaminen tai painopisteen vaihtaminen kesken prosessin johtaa tehottomuuteen ja syntyy hukkaa. Kanbanissa käynnissä olevia tehtäviä rajoitetaan vetojärjestelmän avulla joko joissain osissa työkulkua tai koko työkulussa. Käynnissä oleville tehtäville asetetaan maksimimäärä, kuinka monta tehtävää voi olla samaan aikaan käynnissä. Seuraavan tehtäväkortin voi vetää käynnissä olevien sarakkeeseen vasta, kun siellä on tilaa. Rajoituksien avulla ongelmakohdat virtauksessa (flow) löydetään ja ne voidaan tunnistaa ja ratkaista (Kanbanize, n.d.). Kuvassa 8 on Kanban-taulu WIP-rajoituksella.

Kuva 8 Kanban-taulun WIP-rajoituksella

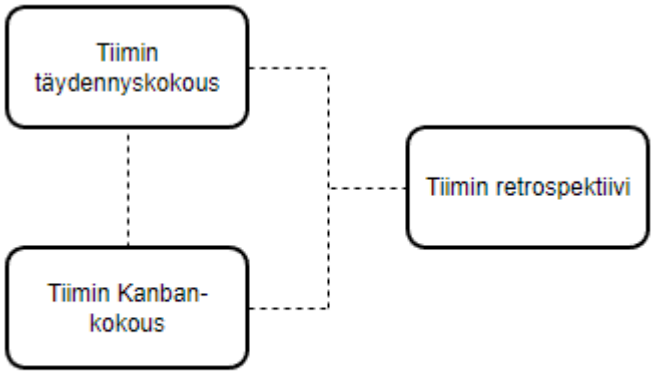


Hallinnoi virtausta (Flow) tarkoittaa sitä, että hallinnoidaan työtä, mutta ei työntekijöitä. Virtaus (flow) tarkoittaa sitä, kuinka tehtävät virtaavat tuotantoprosessin läpi kestävästi ja ennustettavasti. Kanbanissa työprosessien hallinnan avulla tehtävät tuottavat arvoa nopeammin ja ne saadaan tuotantoprosessin läpi mahdollisimman nopeasti (Kanbanize, n.d.).

Tee prosessikäytännöistä selkeitä tarkoittaa sitä, että prosessi määritellään selkeästi, jonka jälkeen se julkaistaan ja otetaan käyttöön. Näin voidaan keskittyä vain hyödyllisten ominaisuuksien kehittämiseen. Yhteisen tavoitteen ollessa kaikille selkeä, voidaan työskennellä ja tehdä päätöksiä, joilla on myönteinen vaikutus sitä kohden. Rajatut, läpinäkyvät, tarkasti määritellyt ja muutosvalmiit käytännöt lisäävät työntekijöiden mahdollisuuksia itsenäiseen työskentelyyn (Kanbanize, n.d.).

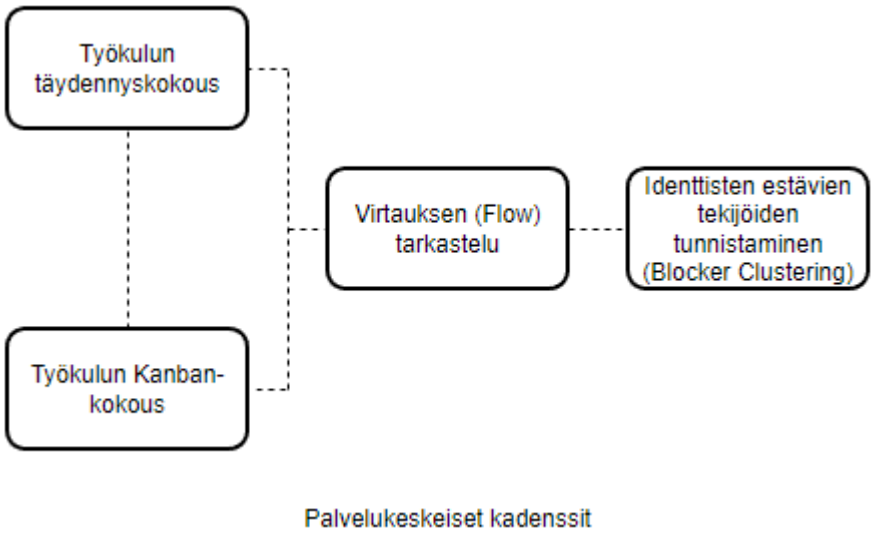
Toteuta palautesilmukat tarkoittaa sitä, että tieto kulkee sidosryhmien välillä ja muutoksiin reagoidaan asianmukaisesti. Palautesilmukat mahdollistavat ketterän toiminnan yrityksissä ja tiimeissä. Kanbanissa kadensseja (palautesilmukoita) on kahden tyyppisiä, **tiimitason kadenssit** ja **palvelukeskeiset kadenssit**. **Tiimitason kadenssissa** voidaan esimerkiksi seurata työn tilaa ja sen kulkua päivittäisessä Kanban kokouksessa. Siinä voidaan tunnistaa käytettävissä oleva kapasiteetti ja potentiaali, jotta toimitusvauhtia voidaan lisätä. Tiimin jäsenet kertovat mitä ovat tehneet eilen ja mitä he aikovat tehdä tänään (Kanbanize, n.d.). Kuvassa 9 on esimerkki tiimitason kadensseista.

Kuva 9 Kanbanin tiimitason kadenssit



Palvelukeskeisten kadenssien tarkoitus on, että toiminnot, palvelun toimittaminen, riskikokoukset voidaan synkronoida ja siten parantaa palvelua. Palveluverkostoa voidaan kehittää, kun ymmärretään, mikä estää tehokkaan palvelutarjonnan. Kadenssien kestoon vaikuttaa tiimin koko ja aiheet, mutta säännölliset kokoukset pienissä ryhmissä ovat hyväksi havaittuja (Kanbanize, n.d.). Kuvassa 10 on esimerkki palvelukeskeisistä kadensseista.

Kuva 10 Kanbanin palvelukeskeiset kadenssit



Kehitä yhteistyössä ja kokeellisesti (mallien ja tieteellisen menetelmän avulla) tarkoittaa sitä, että organisaatiossa jatkuva kehittyminen ja kestävä muutos saavutetaan toteuttamalla muutoksia yhteistyössä tieteellisesti todistettujen menetelmien, palautteen ja mittareiden kanssa. Kun

keskitytään evolutiivisen muutoksen avulla kehittymiseen, on tärkeää voida todistaa, että jokaisella hypoteesilla on myönteisiä tai kielteisiä tuloksia (Kanbanize, n.d.).

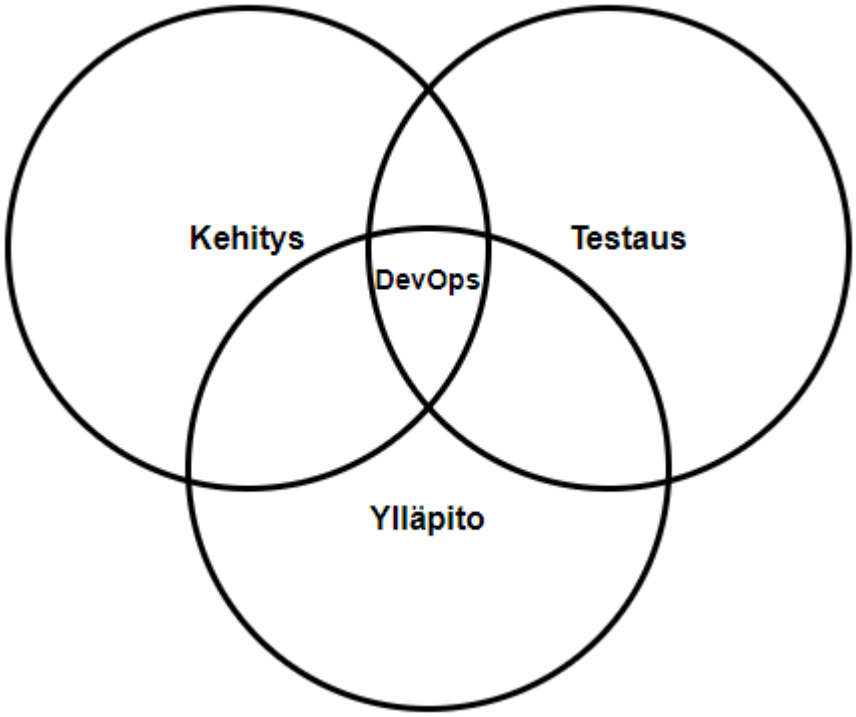
Kanbanissa on kaksi ensisijaista roolia, joita Kanban-tiimit voivat toteuttaa. **Service Delivery Manager (SDM)** eli palveluntoimituspäällikkö ja **Service Request Manager (SRM)** eli palvelupyyntöpäällikkö. **Service Delivery Manager (SDM)** eli **palveluntoimituspäällikön** tehtävä on parantaa työkulun tehokkuutta ja hänestä käytetään myös nimeä Flow manager. Roolissa on joitain yhtäläisyyksiä Scrum Masterin kanssa. SDM:n tehtäviä ovat: työkohteiden virtauksen varmistaminen, helpottaa muutosta ja jatkuvaa kehittymistä, Kanban-taulujen seuraaminen ja viivästyksistä ilmoittaminen, käytäntöjen noudattamisen varmistaminen, Kaizen-strategian toteuttaminen ja Lean sekä Kanban menetelmien perehdyttäminen. **Service Request Manager (SRM)** eli palvelupyyntöpäällikön roolissa on samanlaisia tehtäviä kuin Scrumin tuoteomistajalla. SRM:n ensisijainen tavoite on toimia riskienhallitsijana ja mahdollistajana. SRM:n tehtäviä ovat: työkohteiden järjestäminen backlogista ja priorisoiminen, yrityksen hallinnon kehittäminen, prosessien johdonmukaistaminen ja henkilöriskien vähentäminen. (Kanbanize, n.d.).

2.6 DevOps

DevOpsia ruvettiin kehittämään vuosien 2007-2008 aikana, kun ICT-asiantuntijoiden ja ohjelmistokehityksen yhteisössä ilmeni huolta, että alalla on kohtalokkaita toimintahäiriöitä. Haluttiin muuttaa perinteistä ohjelmointikehitysmenetelmää, jossa kehittäjät ovat erillään ICT-asiantuntijoista. Kehittäjillä ja ICT-asiantuntijoilla oli usein erilliset ja monesti kilpailevat tavoitteet, erillinen johto ja erilliset suorituskykymittarit. He saattoivat työskennellä eri kerroksissa tai jopa eri rakennuksissa (Atlassian, n.d.).

Nimi DevOps muodostuu sanoista Development ja Operations. Niillä tarkoitetaan toimintamallia, jossa ohjelmistokehittäjät ja ICT-asiantuntijat tekevät yhteistyötä ja kommunikoivat aktiivisesti. DevOpsissa integrointi eli ohjelmointipalikoiden yhteen liittäminen aloitetaan mahdollisimman aikaisin, kuten myös integrointitestaus. Näin integrointiin liittyvät ongelmat havaitaan ajoissa ja projektin etenemistä on helpompi arvioida (Juvonen, 2018, s.20). Kuvassa 11 on esimerkki kehityksen, testauksen ja ylläpidon yhteistyöstä.

Kuva 11 DevOpsin kehityksen, testauksen ja ylläpidon yhteistyö



DevOpsin tarkoituksena on tehdä ohjelmistonkehitysprosessi mahdollisimman suoraviivaisesti ja keinoina siihen ovat ohjelmiston testauksen ja julkaisun automatisointi. Ajatustapana on nopea ja jatkuva julkaiseminen ja DevOpsin avulla työvaiheita voidaan automatisoida yksi kerrallaan (Juvonen, 2018, s.20).

DevOpsissa on viisi periaatetta, joita noudattamalla tiimi saa täyden hyödyn DevOpsista. DevOps on kulttuurifilosofinen ajattelutapa, jonka avulla tiimi oppii uusia toimintatapoja. Periaatteiden kautta kehittäjät ymmärtävät paremmin käyttäjien vaatimuksia ja tarpeita. Ylläpitäjät lisäävät ylläpitovaatimuksia, asiakastarpeita ja osallistuvat kehitysprosessiin. Periaatteet perustuvat kuuteen pääideologiaan **jatkuvaan integrointiin**, **jatkuvaan toimitukseen**, **jatkuvaan testaukseen**, **jatkuvaan käyttöönottoon**, **jatkuvaan tuotantoon** ja **jatkuvaan yhteistyöhön**. Ensimmäinen periaate on yhteistyö. **Yhteistyö** tarkoittaa sitä, että kehittäjät, testaajat ja ylläpito työskentelevät, kommunikoivat ja jakavat palautetta keskenään koko sovelluksen elinkaaren ajan. Kehittäjät, testaajat ja ylläpito voivat tarvittaessa muodostaa yhteisen tiimin. DevOps-tiimi on kokonaisvaltaisesti vastuussa projektin laadusta sen kaikilla osa-alueilla. Tiimi omistaa sovelluksen tai ominaisuuden sen koko elinkaaren ajan ja se lisää tiimin sitoutumista ja panostusta tuottamaan laatua (Atlassian, n.d.).

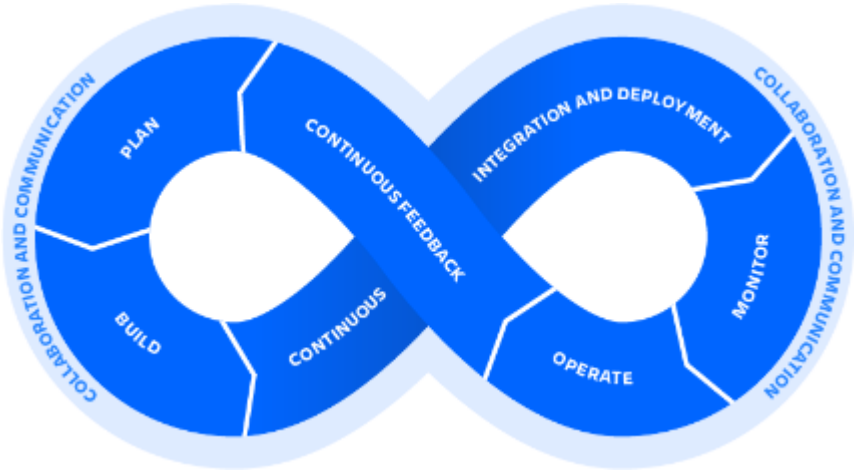
Automaatio tarkoittaa sitä, että mahdollisimman suuri osa ohjelmistokehityksen elinkaaresta automatisoidaan. Pitkälle viety automatisointi mahdollistaa sen, että kehittäjillä on enemmän aikaa uusien ominaisuuksien kehittämiseen ja ohjelmointiin. Automatisointi mahdollistaa jatkuvan integroinnin, jatkuvan toimituksen ja jatkuvan käyttöönoton. Sen avulla voidaan lisätä tuottavuutta ja vähentää inhimillisten virheiden määrää. Jatkuva kehittäminen on mahdollista, kun prosessit ovat automatisoituja ja iteraatioajat ovat lyhyitä, koska silloin voidaan reagoida nopeasti asiakaspalautteisiin. **Jatkuva kehittyminen** tarkoittaa sitä, että kokeillaan, minimoidaan hukkaa, optimoidaan nopeutta, kustannuksia ja toimituksen sujuvuutta. Jatkuvalla kehittymisellä on yhteys jatkuvaan toimittamiseen, koska tiimi voi tehdä jatkuvasti uusia päivityksiä ja näin kehittää järjestelmien tehokkuutta, poistaa hukkaa ja tuottaa arvoa asiakkaille (Atlassian, n.d.).

Asiakaskeskeinen toiminta tarkoittaa sitä, että käytetään lyhyitä palautesilmukoita asiakkaiden ja käyttäjien kanssa. Palautteen avulla tuotteita ja palveluita voidaan kehittää vastaamaan käyttäjien tarpeita. DevOpsin käytäntöjen avulla palaute voidaan kerätä reaaliaikaisesti ja siihen voidaan reagoida nopeasti ottamalla muutosehdotukset käyttöön. Vuorovaikutus käyttäjien kanssa auttaa tiimiä kehittämään tuotetta tai palvelua käyttäjäystävälliseen suuntaan (Atlassian, n.d.).

Luo lopputulos mielessä tarkoittaa sitä, että tuotteet ja palvelut, joita kehitetään ratkaisevat todellisia ongelmia ja vastaavat asiakkaiden tarpeita. Ohjelmistoa ei pitäisi kehittää pelkästään olettamuksien perusteella siitä, kuinka ohjelmistoa lopulta käytetään. Tiimissä tarvitaan kokonaisvaltainen käsitys kehitettävästä tuotteesta tai palvelusta koko sen elinkaaren ajan (Atlassian, n.d.).

DevOps työkaluja käytetään prosessien automatisointiin ja nopeuttamiseen. DevOps perustuu jatkuvalle integraatiolle, jatkuvalle toimitukselle, automaatiolle ja yhteistyölle. Työkaluketjun avulla tiimi pystyy toteuttamaan näitä perusteita. Kuvassa 12 on DevOps elinkaari, jossa työkaluketju näkyy (Atlassian, n.d.).

Kuva 12 DevOps elinkaari (Atlassian, n.d.)



Suunnitelma vaiheessa tulisi ottaa huomioon ketterät toimintamallit, kuten työn jakaminen pienempiin ja helpommin hallittaviin palasiin. Näin mahdollistetaan myös nopeampi palautteen saanti käyttäjiltä ja tuotetta voidaan optimoida sen perusteella. Jatkuva palautteen kerääminen käyttäjiltä ja sen järjestäminen sekä priorisoiminen auttaa kehittäjiä viemään ohjelmistoa oikeaan suuntaan. Työkalujen tulisi kannustaa avoimeen aivoriiheen ja niiden pitäisi mahdollistaa ideoiden, strategian, tavoitteiden, vaatimuksien, etenemissuunnitelmien ja dokumentaation jakaminen ja kommentointi kaikille. Suunnitelmassa on huomioitava myös integraatio ja ominaisuusliput. Ominaisuudet pitää muuntaa käyttäjätarinoiksi kehitysjonossa. Ominaisuusliput ovat if-lauseita koodissa, joiden avulla voidaan kytkeä ominaisuus päälle tai pois päältä. Sprinttien suunnittelussa, virheseurannassa, integraatiossa ja yhteistyön ylläpitämisessä hyödyllisiä työkaluja on esimerkiksi Jira, Confluence ja Slack (Atlassian, n.d.).

Rakennus vaiheessa tulisi ottaa huomioon tuotannon kanssa identtiset ympäristöt kehitystä varten. Työkaluja yksittäisten kehitysympäristöjen luomiseen ovat esimerkiksi avoimeen lähdekoodiin perustuvat Kubernates ja Docker. Virtuaalisessa suljetussa ja identtisessä ympäristössä ohjelmoimalla saadaan enemmän tuloksia aikaan, koska ympäristö on kaikille tiimissä samanlainen (Atlassian, n.d.).

Infrastruktuuri koodina tarkoittaa sitä, että uudelleen toimittaminen on nopeampaa, johdonmukaisempaa ja toistettavampaa kuin korjaaminen. Sillä tarkoitetaan edellisen lisäksi myös sitä, että kehitysympäristöstä voidaan luoda muunnelmia, joissa on samat määritykset kuin tuotantoympäristössä. Koodia voidaan soveltaa uudelleen, jotta palvelin saadaan tunnetulle

perustasolle. Koodia voidaan testata, vertaisarvioida, tallentaa versiohallintaan ja sisällyttää jatkuvaan integrointiin. Näin toimimalla prosesseista tulee toistettavia ja järjestelmistä luotettavia. Se vähentää myös käyttöoppaiden ja dokumentoinnin tarvetta. Työkaluja näihin toimiin ovat esimerkiksi Ansible, Chef, Docker, Puppet ja Terraform (Atlassian, n.d.).

Lähdekoodin hallinnan työkalut mahdollistavat koodin tallentamisen eri ketjuihin, jotta koodin muutokset nähdään ja ne on helposti jaettavissa koko tiimille. Ne mahdollistavat yhteistoiminnallisen koodauksen. Vetopyyntöjen avulla voidaan parantaa koodin läpimenoa ja laatua. Vetopyynnöt kertovat muutosehdotuksista ja tiimi voi tarkastella ja keskustella niistä ennen integrointia pääkoodiin. Vetopyyntöjen ansiosta virheet vähenevät ja se nopeuttaa kehitystä sekä vähentää käyttökustannuksia. Lähdekoodin hallinnan työkalut pitää integroida muiden työkalujen kanssa, jotta koodin kehittämisen ja toimittamisen eri osat voidaan yhdistää. Siten nähdään, onko jonkun ominaisuuden koodi jo käytössä tuotannossa ja jos virheitä havaitaan, ne löydetään ja voidaan korjata. Lähdekoodin hallinnan työkaluja ovat esimerkiksi Bitbucket, GitHub ja GitLab (Atlassian, n.d.).

Jatkuva integrointi ja toimitus on DevOps elinkaaren seuraava vaihe. Jatkuva integrointi tarkoittaa käytäntöä, jossa koodi tarkastetaan ja testataan monta kertaa päivässä. Tällä käytännöllä varmistetaan virheiden löytäminen ja korjaaminen ajoissa. DevOps North Star on työnkulku, jonka avulla koodin tarkastaminen vetokutsuilla voidaan haaroittaa nopeasti harvempiin haaroihin ilman, että testauksen laatu ja kehittämisen nopeus kärsii. Työkaluja jatkuvaan integrointiin ovat esimerkiksi Jenkins, AWS, Bitbucket, Circleci, Snyk ja Sonarsource (Atlassian, n.d.).

Testaustyökalujen avulla voidaan tehdä tutkivaa testausta, testauksen hallintaa ja organisointia. Automatisoitu testaus lisää tietoisuutta ja nopeuttaa kehitys- ja testaussykliä pitkässä juoksussa. Sen avulla myös ohjelmiston laatu paranee ja riskit vähenevät. Automatisoitua testausta voidaan käyttää käyttöliittymän, tietoturvan ja rasituksen testaukseen. Niiden avulla saa myös raportteja ja kaavioita, joiden avulla riskitekijät voidaan löytää. Testaustyökaluja ovat esimerkiksi Mabl, SauceLabs, XRay ja Zephyr (Atlassian, n.d.).

Käyttöönoton koontitaulu on työkalu, josta näkee kokonaiskuvan käyttöönoton kaikista toiminnoista sen eri vaiheissa. Sen tulisi olla yksi koontitaulu, joka on integroitu koodivarastoon ja

käyttöönottotyökaluihin. Siitä pitäisi näkyä julkaisun muutos-, testaus- ja käyttöönottotiedot samassa koontitaulussa. Koontitaulusta tulisi nähdä haarat, versiot, vetopyynnöt ja käyttöönottovaroitukset. Käyttöönoton koontitauluna voi käyttää esimerkiksi Jiraa (Atlassian, n.d.).

Automatisoitu käyttöönotto toteutetaan tapauskohtaisesti, koska sen toimivuuteen vaikuttaa aina sovellus ja sen toimintaympäristö. Usein aloitetaan toimintojen käyttöoppaan skriptaamisella Rubyn tai Bashin avulla. Päällekkäisen koodin välttämiseksi tarvitaan apumenetelmiä ja skriptejä. Käyttöönoton automatisointi on hyvä aloittaa alimmalta tasolta ja siirtyä vähitellen tuotantoympäristöön saakka. Näin saadaan luotua luettelo tehtävistä ja ne saadaan standardoitua. Työkaluja automatisoituun käyttöönottoon ovat esimerkiksi Bitbucket ja AWS CodePipeline (Atlassian, n.d.).

Toiminta on DevOpsin elinkaaren seuraava vaihe. Siihen liittyy palvelinten seuranta ja sovellusten suorituskyvyn seuranta ja ne molemmat pitäisi automatisoida. Tarvitaan työkalu, joka kuuntelee ja tallentaa tietoja ympäri vuorokauden, jotta voidaan ymmärtää sovelluksen yleistä tilaa ja suuntauksia. DevOpsissa jatkuva tarkkailtavuus on olennaista ja työkalun tulisi integroida asiakas ja tiimi siten, että tieto esimerkiksi häiriötilanteista kulkee sujuvasti koko ajan. Työkaluja palvelinten ja sovellusten suorituskyvyn seurantaan ovat esimerkiksi Appdynamics, Datadog, Slack, Splunk, New Relic, Opsgenie, Pingdom, Nagios, Dynatrace, Hosted Graphite ja Sumo Logic (Atlassian, n.d.).

DevOps-tiimin yhteistyön kannalta on tärkeää seurata tapahtumia, muutoksia ja ongelmia. Kaikkien tiimissä tulisi nähdä samat työt, tietää miten toimia häiriötilanteissa ja mihin mahdolliset muutokset liittyvät. Työkalussa olennaista on, että tapahtumat, häiriötilanteet, muutokset ja ongelmat ovat samalla alustalla, koska silloin ongelmat voidaan tunnistaa ja korjata nopeammin. Työkaluja tähän ovat esimerkiksi Jira Service Management, Jira Software, Opsgenie ja Statuspage (Atlassian, n.d.).

Jatkuva palaute on DevOps elinkaaren viimeinen vaihe. DevOpsissa kaikki tiimin jäsenet näkevät käyttäjien palautteen, koska se auttaa tiimiä jokaisessa kehitysvaiheessa viemään tuotetta tai palvelua oikeaan suuntaan. Jatkuvaan palautteeseen kuuluu asiakaskokemuksien mittaaminen

(Net Promoter Score), poistumiskyselyt, vikailmoitukset, tukipyynnöt ja Twitter ja Facebook julkaisujen kerääminen ja tarkastelu. Työkalujen avulla pikaviestimet voidaan integroida mittaamaan asiakaskokemuksia ja keräämään palautetta Twitteristä ja Facebookista. On myös olemassa sosiaalisen median hallinta-alustoja, joilla voidaan laatia raportteja historiatietojen avulla. Palautteen perusteella saadun tiedon analysointi vie ominaisuuksia käyttäjien haluamaan suuntaan ja se on pitkällä tähtäimellä tärkeämpää, kuin arvoa tuottamattomien ominaisuuksien kehittäminen. Jatkuvan palautteen työkaluja ovat esimerkiksi GetFeedback, Slack, Jira Service Management ja Pendo (Atlassian, n.d.).

DevOpsissa on viitekehyksiä, joiden avulla tiimit voivat toteuttaa käytäntöjä tehokkaasti ja tuloksellisesti. **CALMS-viitekehys** tulee sanoista Culture, Automation, Lean, Measurement ja Sharing eli kulttuuri, automatisointi, Lean-ajattelu, mittaaminen ja jakaminen. **Kulttuurilla** tarkoitetaan sitä, että yhteistyö kehittäjien ja ict-asiantuntijoiden kesken on olennaista, jotta DevOps käytännöt saadaan toimimaan oikein yrityksissä. **Automaatiolla** voidaan luoda luotettavia järjestelmiä, tuottaa toistettavia prosesseja ja vähentää manuaalisen työn määrää. **Lean-ajattelussa** olennaisinta DevOpsin kannalta on jatkuvan kehittymisen ja epäonnistumisen hyväksymisen periaatteet. **Mittaamisen** avulla voidaan todentaa, että jatkuvan kehittämisen toimet todella vievät projektia oikeaan suuntaan. **Jakamisen** merkitys on olennaista, koska sen avulla toiminta todella on läpinäkyvää ja kaikki tiimin jäsenet tietävät, mitä kukin tekee (Atlassian, n.d.).

Toinen DevOps-viitekehys on **tiimi-topologiat**. Niitä on neljä erilaista, jotka ovat virtaan suuntautunut tiimi, alustatiimi, monimutkaisen osajärjestelmän tiimi ja mahdollistava tiimi. **Virtaan suuntautunut tiimi** keskittyy yhteen vaikuttavaan työvirtaan. Se voi olla esimerkiksi tuote, palvelu, ominaisuus tai käyttäjätarina. **Alustatiimin** tehtävänä tarjota sisäisiä palveluja ja luoda valmiuksia, jotta itsenäinen työskentely olisi mahdollista virtaan suuntautuneille tiimille. **Monimutkaisen osajärjestelmän tiimi** on vastuussa erityistaitoja vaativien järjestelmän osien rakentamisesta ja ylläpidosta. **Mahdollistava tiimi** koostuu tietyn teknisen alan asiantuntijoista. He keskittyvät tutkimukseen ja kokeiluihin, jotta virtaan suuntautunut tiimi pystyy keskittymään ongelmiin ratkaisujen sijaan (Atlassian, n.d.).

Kolmas DevOps-viitekehys on **tiimirakenne**. Ensimmäinen tiimirakenne **Kehitys- ja toimintayhteistyö** on DevOpsin perusta, koska sen avulla tiimit pystyvät rakentamaan, testaamaan ja toimittamaan ohjelmistoja nopeammin ja luetettavammin. Tämän rakenteen toimivuuden kannalta olennaista on, että kehittäjät ja ICT-asiantuntijat ymmärtävät toistensa tarpeet. **Kehitys ja operaatiot yhdessä** on toinen tiimirakenne, jossa kehittäjät ja ICT-asiantuntijat ovat samassa tiimissä ja istuvat samassa tilassa. Tässä rakenteessa voi olla jopa niin, että samat henkilöt kehittävät ja käyttävät sovelluksia**.** Tätä tiimirakennetta käyttää esimerkiksi Facebook ja Netflix (Atlassian, n.d.).

DevOps/Site Reliability Engineering (SRE) on kolmas tiimirakenne, jossa kehitystiimi toimittaa SRE-tiimille lokitietoja ja SRE-tiimi arvioi täyttääkö ohjelmisto riittävät standardit. SRE tarkoittaa sivuston luotettavuustekniikkaa ja sen tavoitteena on luoda skaalautuvia ja luotettavia ohjelmistojärjestelmiä. Yhteistyötä tehdään toimintakriteereistä ja SRE-tiimillä on valtuudet pyytää kehittäjiltä korjauksia koodiin, ennen sen lanseeraamista. Tätä tiimirakennetta käyttää esimerkiksi Google. **Ops-alustana** on neljäs tiimirakenne ja siinä kehittäjätiimiin kuuluu ICT-asiantuntijoita, joiden vastuulla on kaikki toimintaan liittyvä ja suurin osa yhteydenpidosta IaaS-tiimin (Infrastucture as a Service) kanssa. IaaS-tiimi luo pilvipalveluun skaalautuvia virtuaalipalveluja, jossa olevat sovellukset ovat alustana tälle tiimirakenteelle. **DevOps ulkoisena osapuolena** on viides tiimirakenne. Siinä käytetään ulkopuolista DevOps-konsulttia tai DevOps-tiimiä vain sen aikaa, että kehitystiimi ja ICT-asiantuntijoiden tiimi pääsevät siirtymään joko **kehitys- ja toimintayhteistyö** tai **kehitys ja operaatiot yhdessä** rakenteeseen (Atlassian, n.d.).

Neljäs DevOps-viitekehys on **DevOps-mittarit**. Niiden avulla nähdään ohjelmistokehitysputken suorituskyky ja voidaan tunnistaa ja poistaa prosessissa olevat pullonkaulat nopeasti. DevOpsissa suorituskyvyn mittaamiseen on neljä keskeistä mittaria. Ensimmäinen niistä on **muutosten läpimenoaika**. Se tarkoittaa aikaa, joka kuluu koodimuutoksen siirtämisestä siihen, että se on käytettävissä. Esimerkiksi aika, joka kuluu siihen, että kaikki ennen julkaisua vaaditut testit on läpäisty. Toinen mittari on **muutosten epäonnistumisaste**. Sillä tarkoitetaan prosenttiosuutta koodimuutoksista, joiden osalta tarvitaan tuotannonjälkeisiä korjaustoimenpiteitä. Tällä mittarilla ei mitata ennen käyttöönottoa jo testauksessa havaittuja ja korjattuja virheitä. Kolmas mittari on **käyttöönottotiheys**. Sillä mitataan, kuinka usein uutta koodia otetaan käyttöön tuotannossa. Kehityksen aikana koodimuutoksia toimitetaan myös tuotantoa edeltävään testausympäristöön.

Tässä mitataan käyttöönottoon eli tuotantoon tulevia koodimuutoksia. Neljäs mittari on **Mean time to recovery (MTTR)** eli **keskimääräinen palautumisaika**. Sillä mitataan, kuinka kauan kestää palautua täydellisestä vikaantumisesta tai osittaisesta keskeytyksestä palvelussa (Atlassian, n.d.).

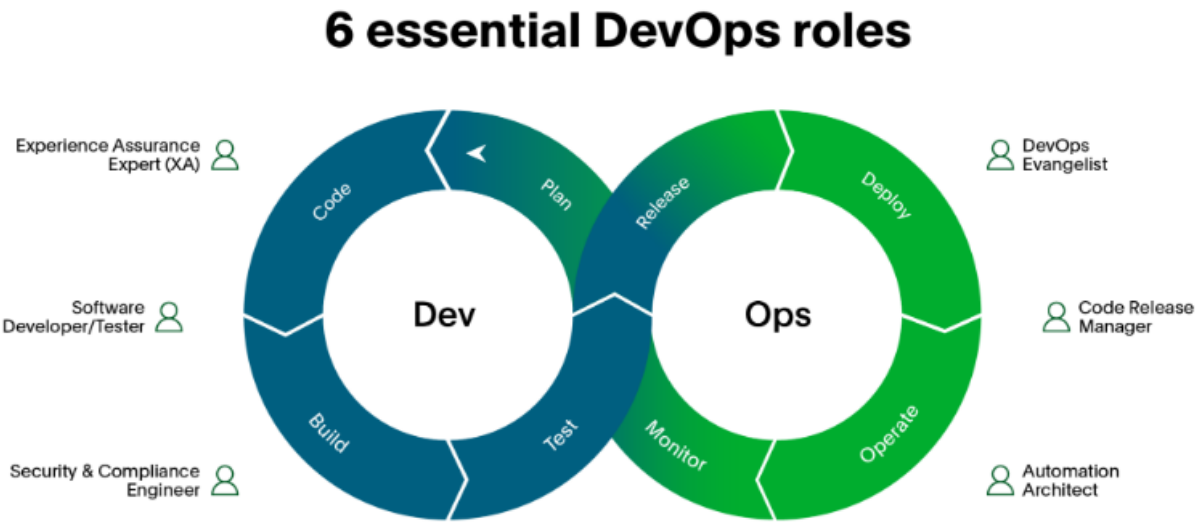
DevOps-tiimissä on kuusi keskeistä tehtävää. **The DevOps Evangelist** on muutosagentti, joka vastaa DevOps-kulttuurin muutoksesta ja toteuttamisesta. Hän vastaa prosessien ja tiimin identiteetin onnistumisesta ja täytäntöönpanon varmistamisesta. Hänen tehtävänsä ovat: hyötyjen edistäminen, kehitys- ja toimintatiimien hyväksymisen varmistaminen, keskeisten roolien määrittäminen ja tiimin jäsenten koulutuksen varmistaminen. **The Code Release Manager** on yleensä projektipäällikkö. Häneltä vaaditaan teknistä tietämystä ja asiantuntijuutta, jotta hän voi johtaa ja ylläpitää tuotteiden ja sovellusten kehitys- ja toimitusprosessia. Häneltä vaaditaan myös ketterien menetelmien ymmärrystä. Hänen tehtävänsä ovat tuotteiden ja sovellusten projektinhallinta kehityksestä käyttöönottoon saakka ja DevOpsin edistymisen seuraaminen vaikutusten ja muiden keskeisten mittareiden avulla (PagerDuty, n.d.).

The Automation Architect vastaa järjestelmien automatisoinnista. Hän luo prosesseja, joissa automaation avulla voidaan vähentää manuaalisia tehtäviä. Hän luo tehokkaamman prosessin ja löytää oikeat työkalut käytettäväksi ja integroitavaksi DevOps-malliin. Hänen tehtävänsä ovat strategioiden suunnittelu ja toteutus manuaalisten tehtävien automatisoimiseksi ja oikeiden DevOps-työkalujen löytäminen eri prosesseihin. **The Experience Assurance Expert (XA)** on samankaltainen kuin laadunvarmistaja, mutta hän on keskittynyt asiakaskokemukseen ja sovelluksen käytön yksinkertaisuuteen. XA vastaa lopputuotteen sujuvan käyttäjäkokemuksen luomisesta. Hän katsoo sovelluksen kehitystä asiakkaan näkökulmasta ja huolehtii, että lopputuote toimii oikein, siinä on oikeat ominaisuudet ja sitä on helppo käyttää. Hänen tehtävänsä ovat varmistaa, että lopputuotteessa on kaikki pyydetyt ominaisuudet ja käyttäjäkokemus on sujuva ja miellyttävä (PagerDuty, n.d.).

The Software Developer/Tester on tuotteen rakentaja. Hän vastaa koodin kirjoittamisesta ja DevOpsissa kehittäjä suorittaa myös yksikkötestauksen, käyttöönoton ja jatkuvan seurannan. Rooli on hieman laajempi, koska perinteisesti kehittäjä vastaa vain koodin kirjoittamisesta. Hänen tehtäviään ovat koodin kirjoittaminen uusia tuotteita, ominaisuuksia, tietoturvapäivityksiä ja virheiden korjauksia varten. Hän varmistaa, että koodi vastaa alkuperäisiä

liiketoimintavaatimuksia. Hän tekee yksikkötestauksen, käyttöönotot ja tuotteen suorituskyvyn seurannan. **The Security & Compliance Engineer (SCE)** vastaa järjestelmän yleisestä turvallisuudesta. SCE työskentelee suoraan kehitystyön rinnalla ja pystyy integroimaan turvallisuussuosituksensa jo tuotteen rakentamisen aikana eikä vasta jälkikäteen. Hän tekee tiivistä yhteistyötä kaikkien osastojen ja roolien kanssa varmistaakseen, että yritys on turvassa tietojensa kanssa ja noudattaa tarvittavia vaatimuksia. Hänen tehtävänsä ovat varmistaa, että tuotteet ovat kaikkien vakiintuneiden standardien ja määräysten mukaisia. Hän työskentelee kehitystyön rinnalla sen varmistamiseksi, että tuote on turvallinen ja suojattu mahdollisia hyökkäyksiä vastaan. Kuvassa 13 on DevOpsin 6 keskeistä roolia (PagerDuty, n.d.).

Kuva 13 DevOpsin 6 keskeistä roolia (PagerDuty, n.d.)



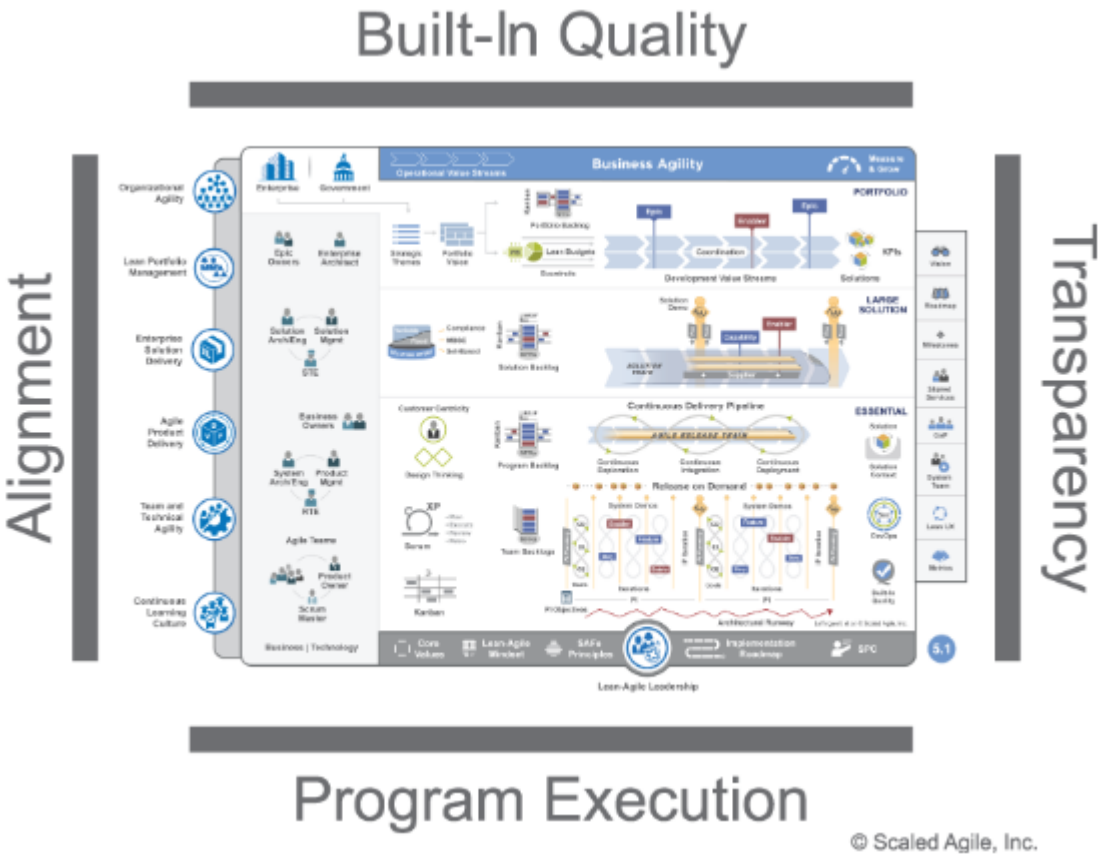
2.7 SAFe

SAFe muodostuu sanoista Scaled Agile Framework eli skaalautuva ketterä viitekehys. SAFe:n kehittivät Lean-Agile-asiantuntijat Dean Leffingwell ja Drew Jemilon ja se otettiin käyttöön vuonna 2011. SAFe:ssa mukautetaan ketterien menetelmien parhaimpia käytäntöjä siihen suuntaan, että ne toimivat myös suurissa organisaatioissa ja tiimeissä (Wrike, n.d.)

SAFe pohjautuu ydinarvoihin, periaatteisiin ja ydinkompetensseihin. Ydinarvoja ovat kohdistaminen, sisäänrakennettu laatu, läpinäkyvyys ja ohjelman toteuttaminen. **Ohjelman toteuttaminen** tarkoittaa sitä, että koko organisaatio ja asiakkaat osallistuvat tekemiseen ja keskittyvät arvoa tuottaviin asioihin. **Kohdistaminen** tarkoittaa sitä, että kaikilla on yhteinen

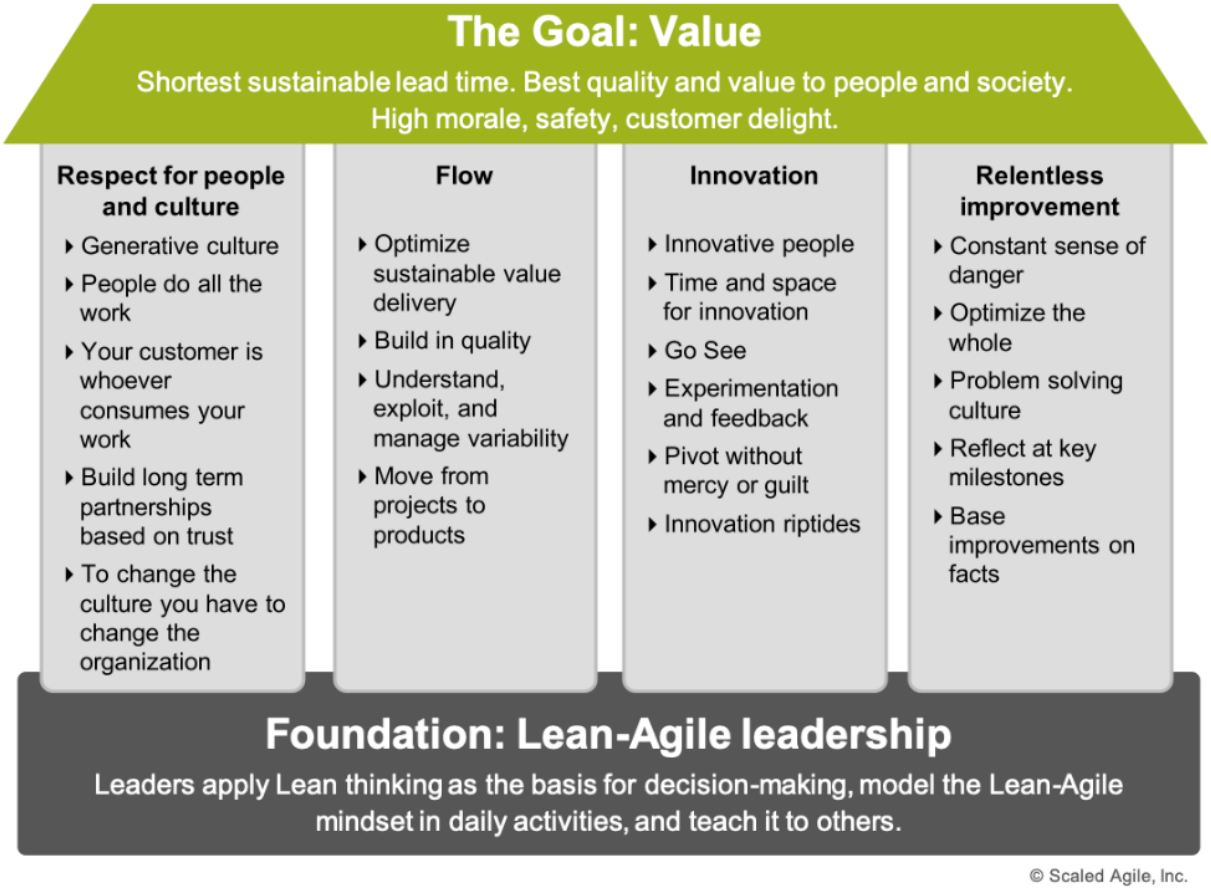
tavoite, jota kohti ollaan menossa. Yhdenmukaistaminen onnistuu, kun tiimeillä on selkeät roolit ja toimintoja synkronoidaan. **Sisäänrakennettu laatu** tarkoittaa sitä, että koko ajan pyritään rakentamaan korkealla laadulla valmiiksi saakka. Laatustandardit varmistetaan jatkuvalla testaamisella. Laatu toteutuu virtauksessa, arkkitehtuurissa, suunnittelussa, koodissa, järjestelmässä ja julkaisussa. **Läpinäkyvyys** tarkoittaa sitä, että kaikki työt ovat kaikkien työntekijöiden nähtävillä, tavoitteet ovat selkeitä ja tehtävät ovat pienissä palasissa, jotta virheet havaitaan nopeasti (Rusanen 2022, n.d.). Kuvassa 14 on SAFe:n ydinarvot (ScaledAgileFramework 2021, n.d.).

Kuva 14 SAFe:n ydinarvot (ScaledAgileFramework 2021, n.d.)



SAFe rakentuu Lean-ajattelulle ja sitä havainnollistamaan on tehty kuva, jota kutsutaan Lean-taloksi. Talon kattona on tavoite, joka on arvon tuottaminen. Kattoa tukee neljä pilaria, jotka ovat ihmisten ja kulttuurin kunnioittaminen, flow eli virtaus, innovaatio ja periksiantamaton kehittäminen. Lean-talon perustana on Lean-Agile-johtaminen (Rusanen 2022, n.d.). Kuvassa 15 on SAFe:n Lean-talo (ScaledAgileFramework 2021, n.d.).

Kuva 15 SAFe:n Lean-talo (ScaledAgileFramework 2021, n.d.)



SAFe:ssa on 10 periaatetta. **Ota taloudellinen näkökulma** tarkoittaa sitä, että viivästyksiä voidaan vähentää ja läpimenoaikaa voidaan lyhentää, kun sovelletaan taloudellista viitekehystä koko prosessin ajan. **Sovella systeemiajattelua** tarkoittaa sitä, että ratkaisuja kehitetään systeemiajattelun kolmen kesken käsitteen kautta, jotka ovat: ratkaisu on järjestelmä, järjestelmää rakentava yritys on myös järjestelmä ja koko arvovirran optimointi (Wrike, n.d.).

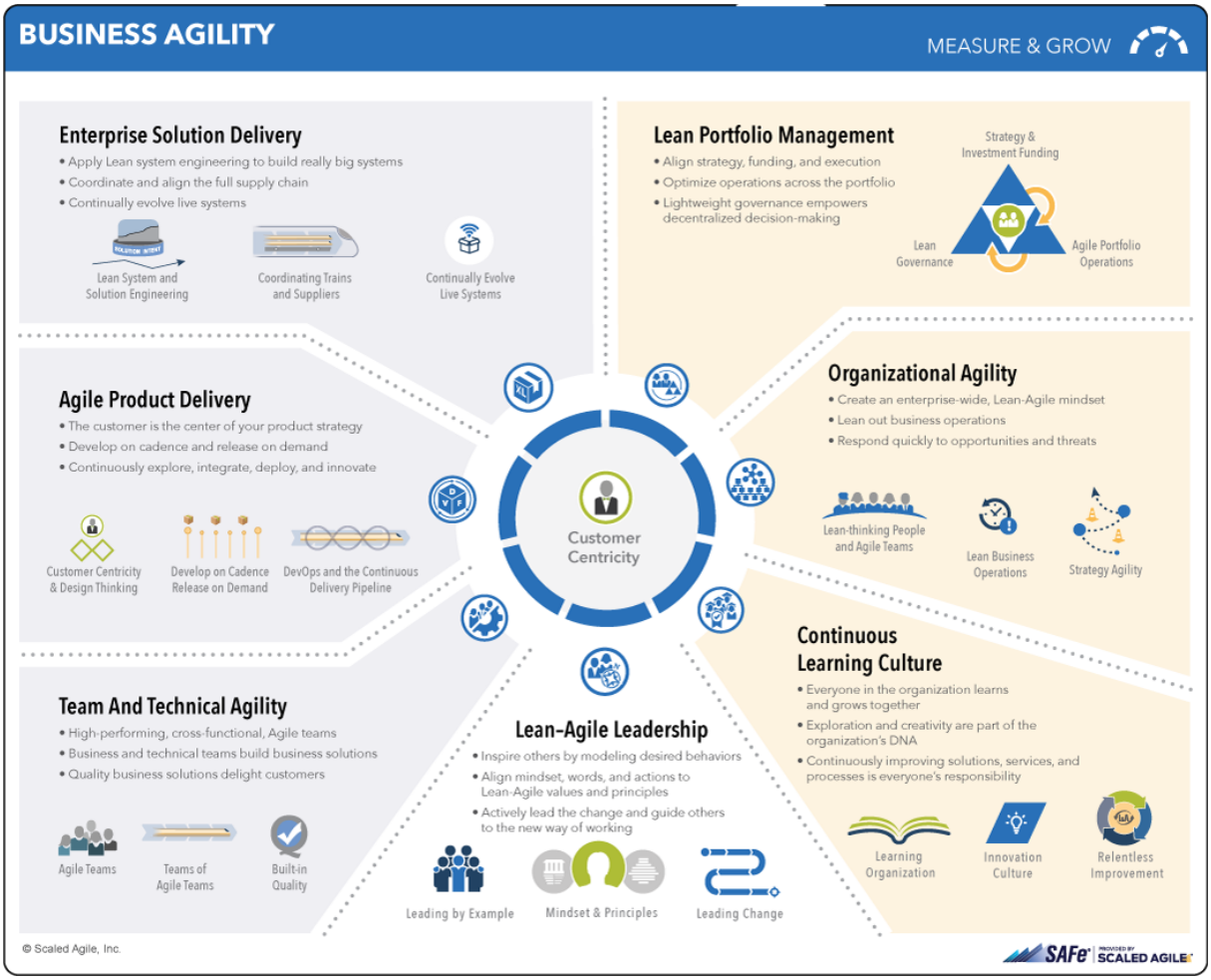
Oletetaan vaihtelevuus ja säilytetään vaihtoehdot tarkoittavat sitä, että ketterässä toiminnassa vaihtelevuus on väistämätöntä ja useita suunnitelmavaihtoehtoja voidaan hyödyntää joukkoihin perustuvassa suunnittelussa. **Rakenna inkrementaalisesti nopeiden integroitujen oppimissyklien avulla** tarkoittaa sitä, että riskejä voidaan vähentää, kun rakennetaan vaiheittain ja luodaan useita suunnitelma vaihtoehtoja**. Perusta välitavoitteet toimivien järjestelmien objektiiviseen arviointiin** tarkoittaa sitä, että käyttämällä toimivia järjestelmiä mallina suunnittelussa, kehittämisessä ja testauksessa, päästään parempiin tuloksiin (Wrike, n.d.).

Keskeneräisen työn visualisointi ja rajoittaminen, eräkokojen pienentäminen ja jonopituuksien hallinta tarkoittaa sitä, että käytetään Kanban-tauluja visualisointiin, kustannusten minimointiin ja odotusaikojen lyhentämiseen. **Sovella kadenssia ja synkronoi poikkihallinnollisen suunnittelun kanssa** tarkoittaa sitä, että tapahtumien rytmittäminen ja kadenssien synkronointi mahdollistaa useat tapahtumat samanaikaisesti, jonka ansiosta saadaan erilaisia näkökulmia päätöksien tueksi (Wrike, n.d.)**.**

Vapauta tietotyöntekijöiden sisäinen motivaatio tarkoittaa sitä, että työntekijöitä motivoidaan kannustinpalkalla, kuuntelemalla, säännöllisellä palautteella ja mahdollistamalla itsenäinen työskentely. **Hajauta päätöksenteko** tarkoittaa sitä, että tiimi saa tehdä usein toistuvia ja kiireellisiä päätöksiä ja johtajat voivat keskittyä strategisiin päätöksiin sekä isoon kuvaan. **Rakenna organisaatio arvon ympärille** tarkoittaa sitä, että organisaatiomallien avulla voidaan tehdä suoritteita ja reagoimaan nopeasti asiakkaiden muuttuviin vaatimuksiin (Wrike, n.d.).

SAFe:n seitsemän ydinkompetenssia ovat: yritysratkaisujen toimitus, ketterä tuotetoimitus, tiimin ja tekniikan ketteryys, Lean-salkunhallinta, organisaation ketteryys, jatkuvan oppimisen kulttuuri ja Lean-Agile johtaminen. Kaikkien näiden keskiössä on asiakaskeskeisyys (Rusanen 2022, n.d.). Kuvassa 16 on SAFe:n ydinkompetenssit (ScaledAgileFramework 2021, n.d.).

Kuva 16 SAFe:n ydinkompetenssit (ScaledAgileFramework 2021, n.d.)

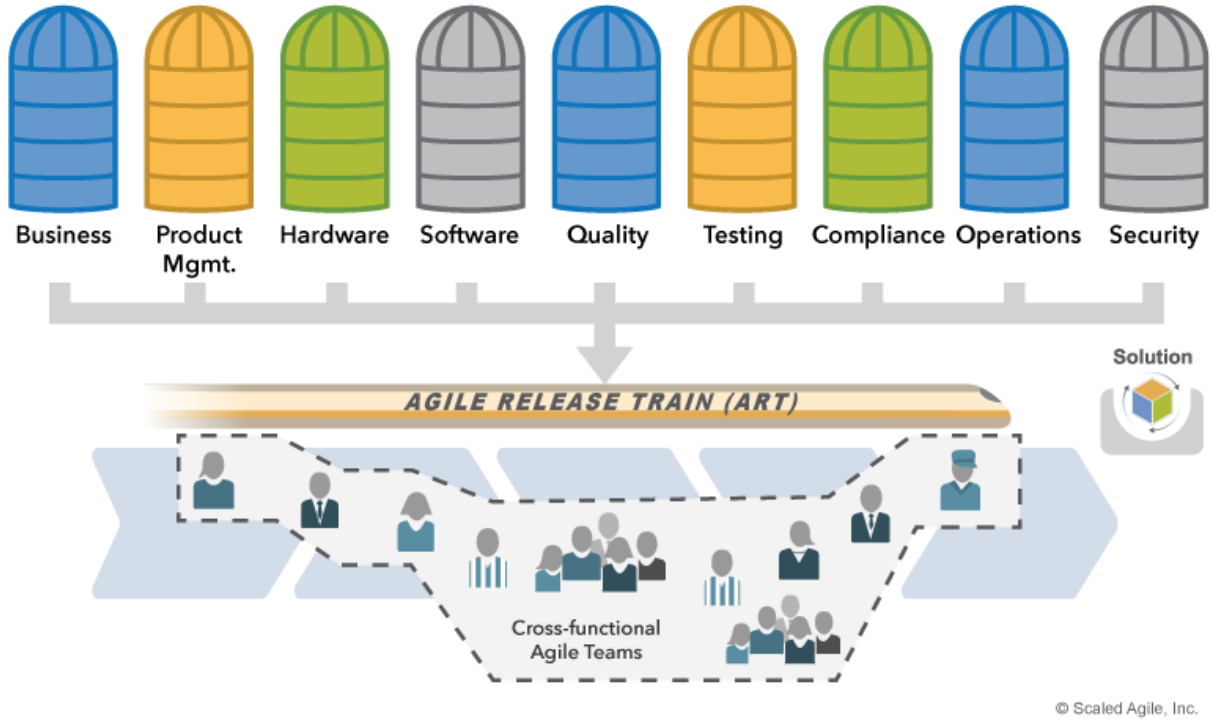


SAFe:n keskeisiä elementtejä ovat: koko liiketoiminnan ja organisaation ketteryys, asiakaslähtöisyys, arvovirrat ketteriksi julkaisujuniksi (Agile Release Trains), tiimien yhteenliittyminen ketteriksi julkaisujuniksi, Program Increment (PI) ja arvovirtojen budjetointi. **Koko liiketoiminnan ja organisaation ketteryys** tarkoittaa sitä, että kehitys, ICT-asiantuntijat, rahoitus, lakiasiantuntijat, markkinointi ja asiakas ovat kaikki mukana tekemisessä. Lisäksi tarvitaan Lean-ajattelua tiimeissä ja toiminnoissa sekä ketterää strategiaa. **Asiakaslähtöisyys** tarkoittaa sitä, että asiakas hyötyy tehdyn työn tuloksista. Asiakkaisiin keskitytään ja heidän tarpeensa ymmärretään. Ajatellaan ja tunnetaan kuin asiakas. Rakennetaan kokonaisratkaisuja ja ymmärretään asiakassuhteen elinkaaren arvo (Rusanen 2022, n.d.).

Arvovirrat ketteriksi julkaisujuniksi tarkoittaa sitä, että arvo muodostuu yli organisaatiorajojen ja ensin arvovirrat tunnistetaan ja sitten niihin muodostetaan yksi tai useampi ketterä julkaisujuna. **Tiimien yhteenliittyminen ketteriksi julkaisujuniksi** tarkoittaa sitä, että muodostetaan virtuaalinen organisaatio, jossa on 5–12 tiimiä ja työntekijöiden määrä voi olla 50–125 tai jopa enemmän (Rusanen 2022, n.d.).

Program Increment (PI) eli ohjelman inkrementti tarkoittaa sitä aikaväliä, jonka aikana ketterä julkaisujuna tuottaa inkrementaalista arvoa toimivien ja testattujen ohjelmistojen ja järjestelmien muodossa. PI:n kesto on yleensä 10 viikkoa, mutta se voi vaihdella 8–12 viikon välillä. Iteraatioita on tyypillisesti viisi kappaletta. PI System Demo sulkee palautesilmukan jokaisella PI:lla. Yhteinen tavoite ketterää julkaisujunaa varten tulee Program Backlogin kautta. Agile Release Train (ART) eli ketterä julkaisujuna on monitoiminnallinen, siinä on kyydissä osaamista liiketoiminnasta, tuotehallinnasta, laitteistosta, ohjelmistosta, laadunvalvonnasta, testaamisesta, lainsäädännöstä, toiminnasta ja tietoturvallisuudesta (Rusanen 2022, n.d.). Kuvassa 17 Agile Release Train (ART) eli ketterä julkaisujuna on monitoiminnallinen (ScaledAgileFramework 2021, n.d.).

Kuva 17 Agile Release Train (ART) eli ketterä julkaisujuna on monitoiminnallinen (ScaledAgileFramework 2021, n.d.)



Arvovirtojen budjetoinnilla tarkoitetaan sitä, että ei budjetoida koko projektia, vaan arvovirtoja. Kun budjetointi tehdään arvovirroille, niin kulujen hallinta on varmempaa. Silloin ei aiheudu viiveitä aiheuttavia ja kalliita kuluja projektikulujen analysoinnista. Resursseja ei tarvitse uudelleenallokoida eikä tarvitse etsiä syyllisiä projektien ongelmista. Arvovirtojen budjetoinnissa rajoitteista huolehditaan investointihorisonttien, kapasiteetin kohdennuksen, epic-aloitteiden hyväksymisen ja sidosryhmien jatkuvan osallistumisen kautta. Epic-aloitteet ovat niin huomattavan laajoja ja vaikuttavia, että ne vaativat pienimmän toteutuskelpoisen tuotteen määrittämisen ja portfolion johdon hyväksynnän ennen toteuttamista (Rusanen 2022, n.d.). Kuvassa 18 jokaisella arvovirralla on oma budjetti työntekijöille ja muille resursseille (ScaledAgileFramework 2021, n.d.).

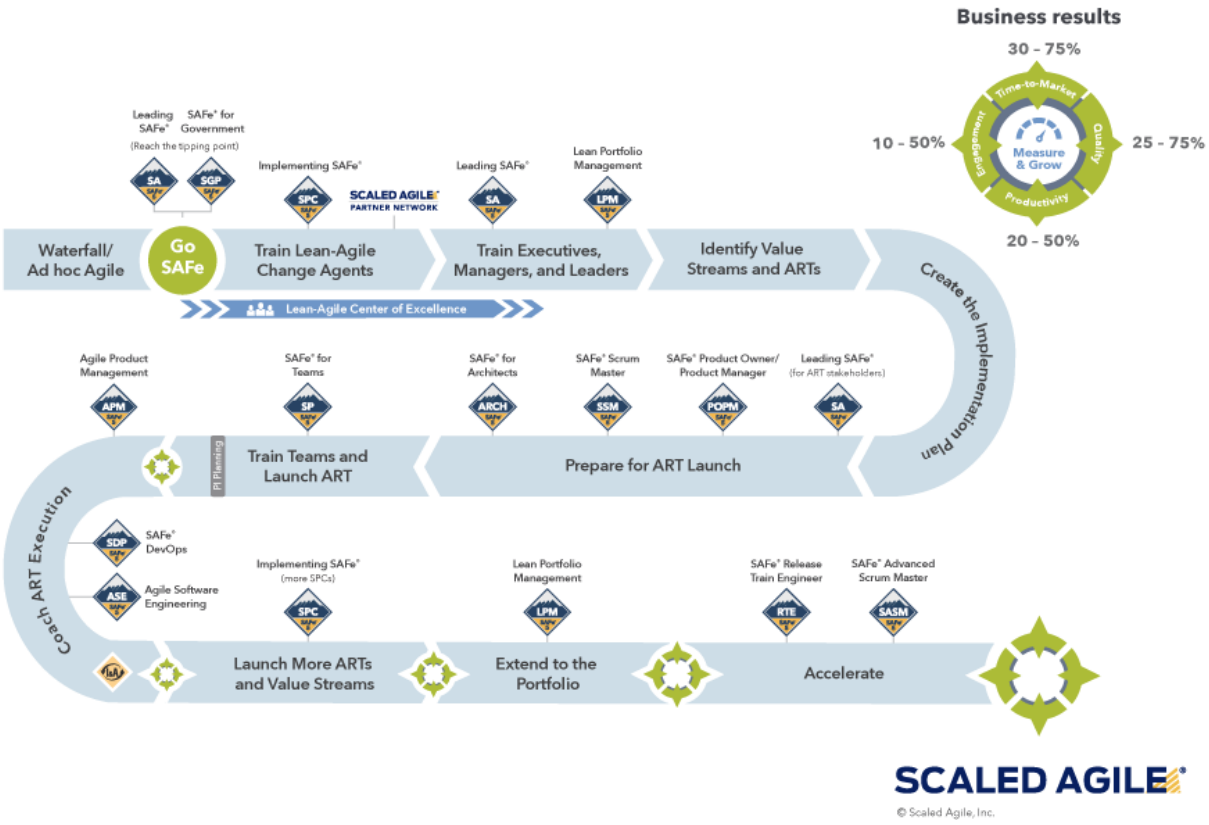
Kuva 18 Jokaisella arvovirralla on oma budjetti työntekijöille ja muille resursseille (ScaledAgileFramework 2021, n.d.)

SAFessa on ART-rooleja. **Product Management** edustaa asiakkaan toiveita, työskentelee asiakkaan ja tuoteomistajan kanssa ymmärtääkseen ja välittääkseen heidän tarpeensa, määritelläkseen järjestelmän ominaisuudet ja osallistuakseen tulosten validointiin. He ovat vastuussa Program Backlogista. **System Architect/Engineering** voi olla yksittäinen henkilö tai pieni poikkitieteellinen tiimi. He soveltavat systeemiajattelua, määrittelevät järjestelmän kokonaisarkkitehtuurin, auttavat tunnistamaan ei-toiminnalliset vaatimukset, määrittelevät merkittävät elementit ja osajärjestelmät sekä auttavat suunnittelemaan niiden väliset rajapinnat ja yhteistyön. **Release Train Engineer (RTE)** on palveleva johtaja ja junan Scrum Master. Hän helpottaa arvovirran optimointia varmistamalla, että ART-tapahtumat ja tehtävät toimivat oikein, mukaan lukien Program Kanban, Inspect & Adapt (I&A)-työpaja, ART Sync ja PI Planning. **Business Owners** on pieni joukko sidosryhmiä, joilla on liiketoiminnallinen ja tekninen vastuu ART:n kehittämän ratkaisun käyttökelpoisuudesta, hallinnoinnista ja sijoitetun pääoman tuotosta. He ovat ART:n ensisijaisia sidosryhmiä ja osallistuvat aktiivisesti ART-tapahtumiin (ScaledAgileFramework 2021, n.d.)

SAFe:ssa on lisäksi myös tiimin rooleja. **Agile Teams** ovat 5-11 henkilön monialaisia ryhmiä, jotka pystyvät määrittelemään, rakentamaan, testaamaan ja ottamaan käyttöön arvon lisäyksen lyhyessä ajassa. Kukin ART koostuu 5-12 ketterästä tiimistä ja se sisältää roolit ja infrastruktuurin, joita tarvitaan täysin toimivien ja testattujen liiketoimintaratkaisujen toimittamiseen. **Product Owner (PO)** eli tuoteomistaja on tiimin backlogin sisällöllinen auktoriteetti ja vastaa tarinoiden määrittelystä ja backlogin priorisoinnista. **Scrum Master** on palveleva johtaja ja ketterän tiimin valmentaja, joka auttaa tiimiä poistamaan esteitä, helpottaa tiimitapahtumia ja edistää ympäristöä, jossa tiimit toimivat (ScaledAgileFramework 2021, n.d.).

SAFe Implementation Roadmap on toteutuksen etenemissuunnitelma. Alkuvaiheessa kouluttaudutaan SAFe:n käyttöön ja melko pian sen jälkeen lähdetään laukaisemaan ketterä julkaisujuna. Sitten laukaistaan lisää junia liikkeelle ja otetaan portfolio käyttöön. Kun toiminta saadaan Lean-Agilen mukaiseksi, voidaan lähteä kiihdyttämään (Rusanen 2022, n.d.). Kuvassa 19 on SAFe:n etenemissuunnitelma (ScaledAgileFramework 2021, n.d.).

Kuva 19 SAFe:n etenemissuunnitelma (ScaledAgileFramework 2021, n.d.)



2.8 PRINCE2

PRINCE2:n ensimmäinen malliversio on laadittu tukemaan tietojärjestelmäprojekteja jo vuonna 1989. Mallia on päivitetty useita kertoja sen jälkeen. PRINCE2:n nimi tulee alun perin sanoista Projects IN Controlled Environments eli kontrolloitavat projektiympäristöt. Nykyisin PRINCE2 on kuitenkin kehittänyt projektimallia periaatteillaan jämäkäksi ja prosesseillaan joustavaksi (Reuter 2022, n.d.).

PRINCE2 on rakenteellinen projektinhallintamenetelmä, jonka periaatteet perustuvat aiemmista projekteista saatuihin hyviin ja huonoihin kokemuksiin. Menetelmä on prosessi- ja tuotepohjainen. Sen avulla voidaan saavuttaa määritellyt tavoitteet odotettujen ja realististen suorituskykytavoitteiden mukaisesti ajan, kustannusten, laadun, laajuuden, hyödyn ja riskien osalta, koska projektirakenteesta saadaan selkeä ja johdonmukainen (Reuter 2022, n.d.).

PRINCE2:ssa on seitsemän periaatetta. **Jatkuva liiketoiminnallinen perustelu** tarkoittaa sitä, että projektin täytyy olla liiketoiminnallisesti järkevä. Ajan ja resurssien käyttö pitää olla perusteltua ja sijoituksen tulee tuottaa arvoa. **Kokemuksesta oppiminen** tarkoittaa sitä, että otetaan huomioon aiemmista projekteista saadut kokemukset ja niistä pidetään ajantasaista oppimispäiväkirjaa. **Määrittele roolit ja vastuut** tarkoittavat sitä, että kaikki projektin työntekijät tietävät oman roolinsa, sen mitä muut tekevät ja ketkä tekevät päätöksiä**. Hallinnoi vaiheittain** tarkoittaa sitä, että tehtävät jaetaan hallinnointivaiheisiin eli helpommin hallittaviin palasiin. **Hallinnoi poikkeamien mukaan** tarkoittaa sitä, että projektin sujuessa suunnitelmien mukaisesti, ei hallinnointia juurikaan tarvita. Vain poikkeamat aktivoivat hallinnointitoimenpiteet. **Keskity tuotteisiin** tarkoittaa sitä, että tuotevaatimukset määrittelevät työtehtävät ja työntekijöiden pitää tietää etukäteen, mitä tuotteelta odotetaan. **Räätälöi ympäristöön sopivaksi** tarkoittaa sitä, että skaalaamalla ja räätälöimällä projekti voidaan sovittaa vastaamaan sen vaatimia tarpeita (PRINCE2, n.d.)

PRINCE2:ssa on seitsemän teemaa. **Business Case eli liiketoimintatarkastelu** tarkoittaa sitä, että projektin kannattavuus ja toteutettavuus pitää olla jatkuvasti liiketoiminnallisesti perusteltua. **Organisaatio** tarkoittaa sitä, että projektipäällikkö tietää projektin kaikkien työntekijöiden roolit ja vastuut. **Laatu** tarkoittaa sitä, että laadunmäärittely pitää tehdä heti projektin alkuvaiheessa, jotta aikataulussa pysytään. **Suunnitelma** tarkoittaa sitä, että kuvaillaan kuinka tavoitteisiin päästään.

Suunnitelmassa keskitytään tuotteeseen, aikatauluun, kustannuksiin, laatuun ja hyötyyn. **Riski** tarkoittaa sitä, että epävarmoja tapahtumia tunnistetaan, arvioidaan ja hallitaan projektin aikana ja sitten ne kirjataan riskipäiväkirjaan. Negatiiviset riskit ovat uhkia ja positiiviset riskit ovat mahdollisuuksia. **Muutos** tarkoittaa sitä, että sovitaan toimintatavoista, kun muutospyyntöjä tai ongelmia ilmenee projektin aikana. **Edistyminen** tarkoittaa sitä, että projektin seurannan avulla projektipäällikkö pystyy valvomaan ja tarkistamaan, onko projekti edistynyt suunnitelman mukaisesti (PRINCE2, n.d.).

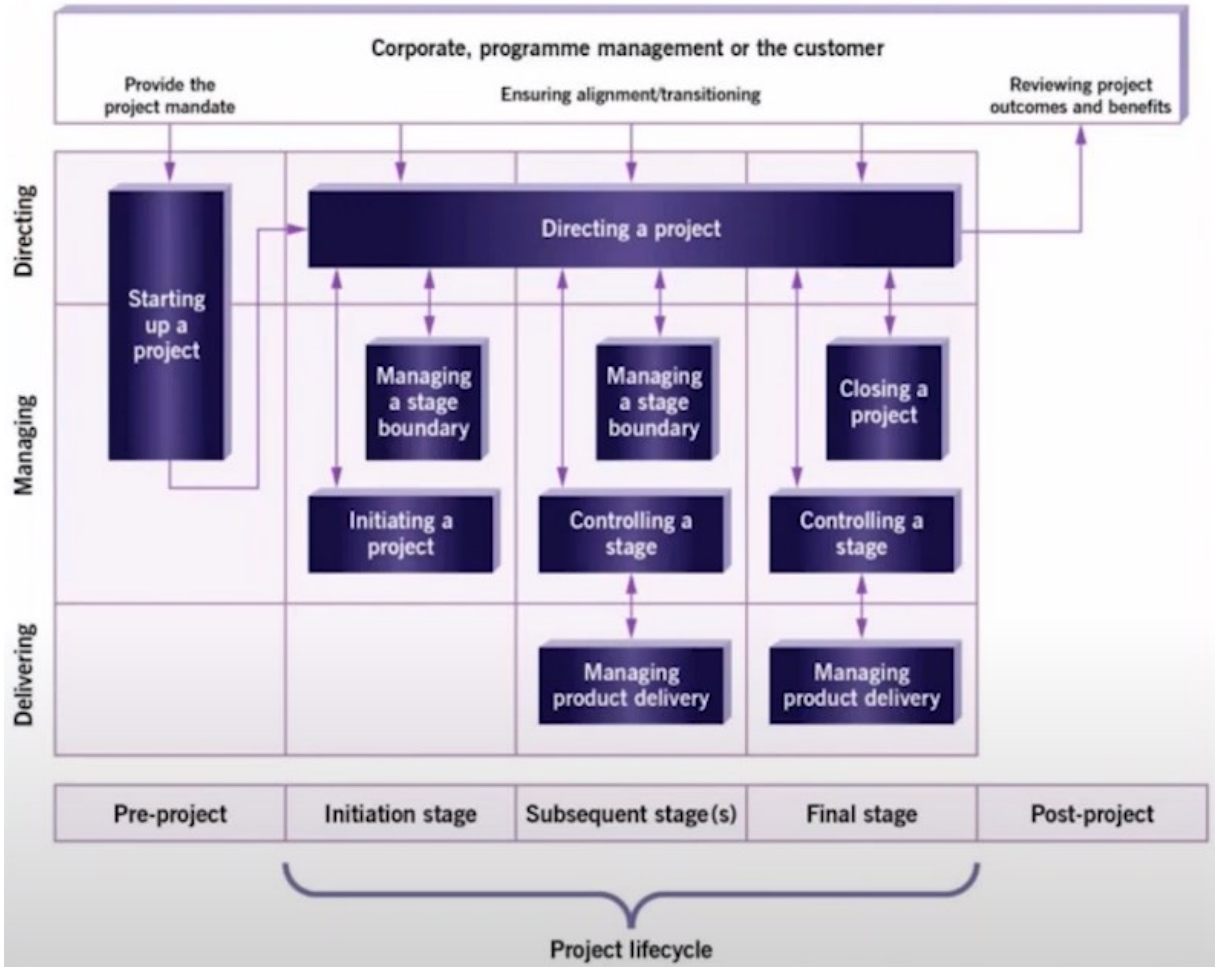
PRINCE2:ssa on seitsemän prosessia. **Projektin perustaminen** on prosessi, jossa luodaan projektimandaatti, jossa vastataan projektiin liittyviin logistisiin kysymyksiin. Projektimandaatissa avataan projektin tarkoitus, kuka sen toteuttaa ja miten se toteutetaan. Laaditaan lyhyt kuvaus projektista mandaatin, oppimispäiväkirjan ja projektissa käytyjen keskustelujen perusteella. Lyhyen kuvauksen jälkeen tiimille annetaan nimi ja sen jälkeen voidaan siirtyä seuraavaan prosessivaiheeseen. **Projektin käynnistäminen** on prosessi, jossa päätetään mitä tehtäviä tarvitaan, jotta projekti saadaan vietyä loppuun saakka. Projektipäällikön tehtävänä on päättää miten suoritustavoitteita kuten aika, kustannukset, laatu, laajuus, hyöty ja riski hallitaan (PRINCE2, n.d.).

Projektin johtaminen on jatkuva prosessi ja se kestää koko projektin ajan. Projektijohto hallinnoi projektin käynnistämistä, vaiheiden rajoja, ad hoc ohjausta ja projektin päättämistä. **Vaiheen hallinnointi** on prosessi, jossa projektipäällikkö hyväksyy työpaketit, jotka jaetaan hallittaviksi toiminnoiksi projektissa ja ne osoitetaan tiimeille ja niiden johtajille. Projektipäällikkö valvoo ja raportoi työpakettien edistymisestä ja reagoi ongelmien korjaamiseen. Tiimipäällikkö koordinoi päivittäiset työtehtävät ja huolehtii yhteydenpidosta tiimin ja projektipäällikön välillä. **Tuotteen toimituksen hallinnointi** on prosessi, jossa työpaketteja hyväksytään, toteutetaan ja toimitetaan (PRINCE2, n.d.).

Tämä viestintä toteutuu tiimipäällikön ja projektipäällikön välillä. **Vaiheen rajojen hallinnointi** on prosessi, jossa projektipäällikkö ja johtoryhmä tarkastavat projektin jokaisen vaiheen ja johtoryhmä päättää, jatketaanko projektia. Projektipäällikkö käy tiimin kanssa läpi, mitä on opittu tähän mennessä ja kirjaa tulokset seuraavaa vaihetta varten. Vaiheen rajojen hallinnointi sisältää toiminnot kuten seuraavan vaiheen suunnittelu, projektisuunnitelman päivitys,

liiketoimintasuunnitelman päivitys, vaiheen päättymisraportti tai poikkeussuunnitelman laatiminen. **Projektin päättäminen** on prosessi, jossa projekti poistetaan käytöstä, määritetään jatkotoimet, valmistellaan arviointikertomukset projektista ja sen hyödyistä, vapautetaan jäljelle jääneet resurssit ja luovutetaan tuote asiakkaalle (PRINCE2, n.d.). Kuvassa 20 on PRINCE2-prosessit projektin elinkaaren aikana (Reuter 2022, n.d.).

Kuva 20 PRINCE2-prosessit projektin elinkaaren aikana (Reuter 2022, n.d.)



PRINCE2:ssa on erilaisia rooleja. **Project Board eli Projektihallinto** on ryhmä, johon kuuluu **Executive, Senior User ja Senior Supplier.** Johtaja voi olla vain yksi henkilö, mutta Senior User ja Senior Supplier voi olla useampi henkilö. **Executive** omistaa Business Case:n eli liiketoimintatarkastelun ja on lopulta vastuussa projektista. Hänellä on viimeinen sana tehdyissä päätöksissä. Projektihallinto on vastuussa projektin onnistumisesta tai epäonnistumisesta. He antavat yhtenäisen suunnan projektille. He varaavat resurssit ja myöntävät määrärahat projektia

varten. He tukevat projektipäällikköä näkyvästi ja jatkuvasti. He varmistavat tehokkaan viestinnän projektin sisällä ja sidosryhmien välillä (Turley, n.d.).

Senior User määrittelee loppukäyttäjien vaatimukset. Hän pitää huolta viestinnästä projektihallinnon ja käyttäjien välillä. Hän varmistaa, että projektin ratkaisut vastaavat käyttäjien tarpeita ja vaatimuksia laadun ja helppokäyttöisyyden osalta. Hän toimittaa tietoa hyödyistä Benefits Management Approach hyötyjenhallintamenetelmää varten (Turley, n.d.).

Senior Supplier edustaa projektin suunnittelijoiden ja kehittäjien etuja. Hän pitää huolta projektin hankintaresursseista ja varmistaa, että oikeat ihmiset, työkalut, laitteet ja tietämys ovat käytettävissä. Hän varmistaa, että tuote täyttää odotetut kriteerit myös laadun osalta. **Project Assurance** varmistaa ensisijaisten sidosryhmien etuja. **Change Authority** päättää osan muutospyynnöistä projektihallinnon puolesta. **Project Manager** on vastuussa projektin päivittäisestä hallinnoinnista projektihallinnon puolesta. **Project Support** avustaa Project Manageria projektinhallintatoimissa. **Team Manager** voi olla yksi tai useampi henkilö ja hänen vastuullaan on tuotannon laadun ja muiden muuttujien varmistaminen tiimeissä (Turley, n.d.).

PRINCE2:ssa on **Management Products** eli **hallinnointituotteita**. Niitä ovat esimerkiksi projektisuunnitelma, projektin lyhyt kuvaus, päiväkirja ja projektin loppuraportti. Riippuu PRINCE2 projektin räätälöinnistä, mitä hallinnointituotteita otetaan käyttöön. Hallinnointituotteita on kolmessa eri kategoriassa, jotka ovat **baselines, records, reports** eli **peruslinjat, asiakirjat ja raportit. Peruslinjat** ovat tuotteita, joiden aikaisempia versioita voidaan tarkastella. Perustuotteet muuttuvat suurella todennäköisyydellä, joten niiden aikaisempia versioita halutaan tarkastella. **Asiakirjat** pidetään jatkuvasti ajan tasalla, joten niistä ei löydy aikaisempia versioita. Asiakirjoista esimerkiksi laaturekisteri sisältää luettelon laatutoimista, jotka pitää toteuttaa tietyn vaiheen aikana. Tallenteet ovat eräänlainen päivittyvä tarkistuslista. **Raportit** ovat tietyn ajankohdan tilannekuva. Se voi olla esimerkiksi tarkastuspistekertomus edellisen viikon edistymisestä (Prince2Primer, n.d.).

Hallinnointituotteisiin kuuluvat seuraavat peruslinjat:

- Benefits Management Approach eli hyödynhallinnan lähestymistapa
- Business Case eli liiketoimintatarkastelu
- Change Control Approach eli muutoksenhallinnan lähestymistapa
- Communication Management Approach eli viestinnän hallinnan lähestymistapa
- Plan eli suunnitelma
- Product Description eli tuotekuvaus
- Project Brief eli projektin lyhyt kuvaus
- Project Initiation Documentation eli projektin käynnistämistä koskeva dokumentaatio
- Project Product Description eli projektin tuotekuvaus
- Quality Management Approach eli laadunhallinnan lähestymistapa
- Risk Management Approach eli riskienhallinnan lähestymistapa
- Work Package eli työpaketti (Turley, n.d.).

Hallinnointituotteisiin kuuluvat seuraavat asiakirjat:

- Daily Log eli päiväkirja
- Issue Register eli ongelmarekisteri
- Lessons Log eli oppimispäiväkirja
- Quality Register eli laaturekisteri
- Risk Register eli riskirekisteri (Turley, n.d.).

Hallinnointituotteisiin kuuluvat seuraavat raportit:

- Checkpoint Report eli tarkistuspisteraportti
- End Project Report eli projektin loppuraportti
- End Stage Report eli vaiheen loppuraportti
- Exception Report eli poikkeusraportti
- Highlight Report eli kohokohtaraportti
- Issue Report eli ongelmaraportti
- Lessons Report eli oppituntiraportti (Turley, n.d.).

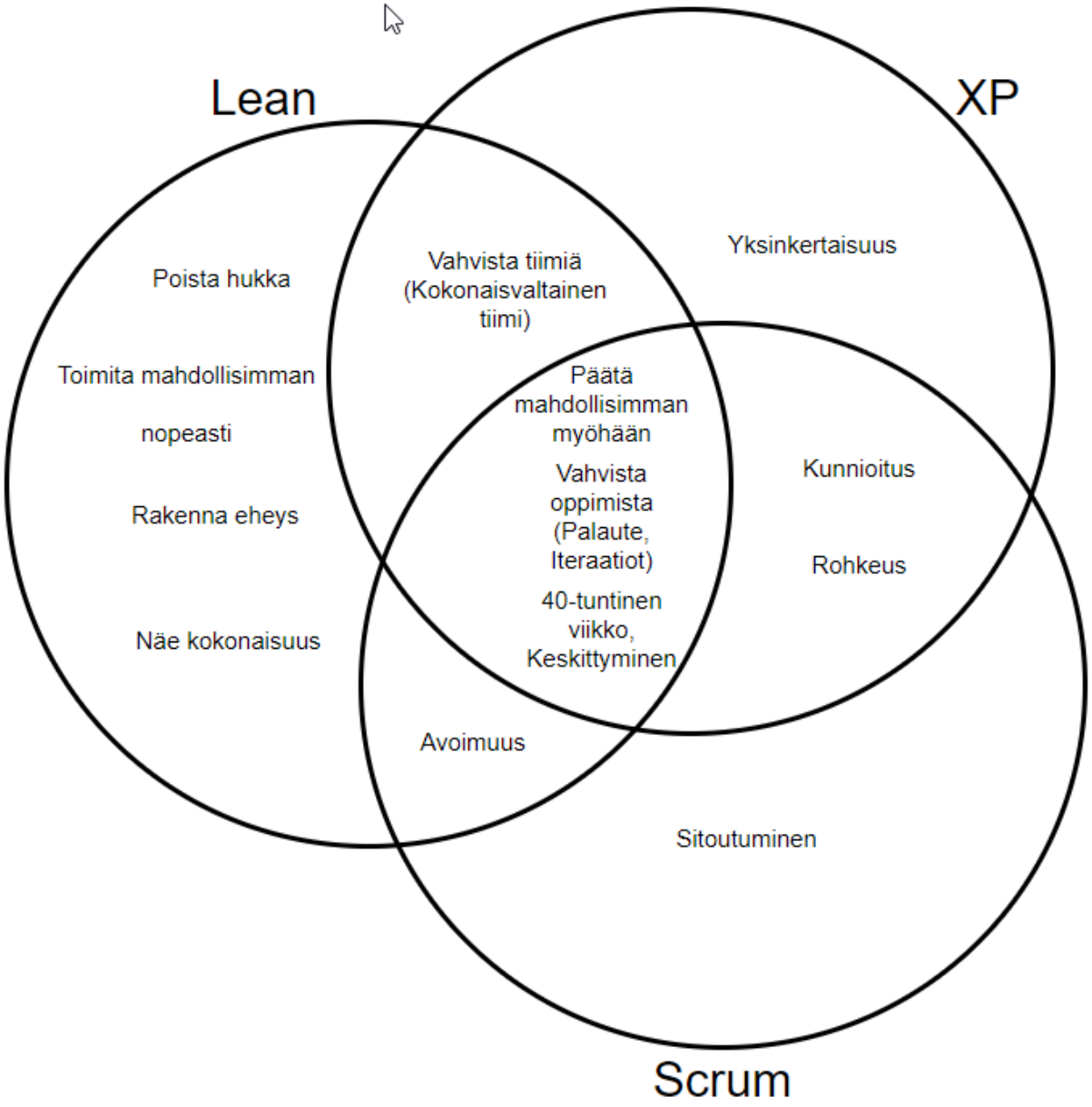
3 Menetelmien vertailu

Tutkimusosuudessa vertaillaan erilaisten ketterien menetelmien ominaisuuksia. Etsitään mitä samanlaista eri menetelmissä on keskenään. Selvitetään miten menetelmät eroavat toisistaan. Selvitään myös voiko menetelmiä käyttää yhtäaikaisesti ja voiko menetelmiä kustomoida. Lopuksi vielä etsitään menetelmien hyötyjä ja rajoituksia.

3.1 Ketterien menetelmien yhteiset tekijät

XP, Scrum, Lean, Kanban, DevOps, SAFe ja PRINCE2 ovat menetelmiä, joita kaikkia yhdistää keskittyminen muuttamaan tiimin ajattelutapaa ketteräksi. XP ja Scrum korostavat molemmat sitä, että tiimin pitää todella ymmärtää menetelmän arvot ja periaatteet, jotta menetelmä toimii. Arvot, periaatteet ja Lean-ajattelu yhdistää Leania muihin menetelmiin, koska kaikki menetelmät vaativat tietynlaista ajattelutapaa, jotta ne saadaan toimimaan oikein. Kun Lean sovitettiin ohjelmistokehitykseen sopivaksi, siihen lainattiin ketteriä osia mm. XP:sta. Leanin Toyotan aikaisia laadunhallinta ajatuksia on sen sijaan lainattu Scrumiin, kuten Daily Scrum, joka on muodollinen tarkastus. Kuvassa 21 nähdään kuinka paljon yhteistä XP:n, Scrumin ja Lean välillä on.

Kuva 21 Yhtäläisyydet XP:n, Scrumin ja Leanin välillä



Päätä mahdollisimman myöhään on arvo, joka on yhteistä XP:ssa, Scrumissa ja Leanissa. Päätöksiä tehdään viimeisellä vastuullisella hetkellä Scrumin suunnittelussa ja XP:ssa sitä sovelletaan suunnittelun lisäksi myös koodauksessa. Leanissa tehdään myös näin ja Leanissa on myös ajattelutyökalu eli periaate, jonka nimi on **viimeinen vastuullinen hetki**. **Vahvista oppimista** on myös arvo, joka on yhteinen kaikissa XP:ssa, Scrumissa ja Leanissa. Leanissa siihen liittyy ajatustyökalut **palaute** ja **iteraatiot** ja ne ovat myös yhteisiä XP:n ja Scrumin kanssa.

Leanissa **vahvista oppimista** arvolla on myös ajattelutyökalut **synkronointi** ja **joukkoihin perustuva kehittäminen**. Leanin **synkronointi** tarkoittaa melkein samaa kuin XP:n **jatkuva integrointi** ja **kollektiivinen omistajuus**. Leanin arvo **vahvista tiimiä** ja sen ajattelutyökalut **itsemääräämisoikeus, motivaatio, johtajuus ja asiantuntemus**, ovat melkein sama kuin XP:n käytäntö **40-tuntinen viikko** ja Scrumin arvo **keskittyminen**. Niissä pyritään välttämään pitkiä työpäiviä, joilla on erittäin kielteinen vaikutus työntekijöihin ja kehitettävään ohjelmistoon. Leanin arvo **näe kokonaisuus** on myös pitkälti sama ajatus, kuin Scrumin **avoimuus**, koska suuri osa kokonaisuuden näkemisessä on projektia koskevan tiedon jakaminen kaikille mukaan lukien johtajat.

Scrumin arvo **sitoutuminen** on myös tärkeä osa Leania ja XP:ta. Leanissa ajatus viedään hieman vielä pidemmälle ja siinä erotellaan se mihin on sitouduttu ja se mitä on oikeus tehdä sekä **joukkoihin perustuva kehittäminen**. **Joukkoihin perustuvassa kehittämisessä** projekti toteutetaan niin, että tiimi voi seurata useita polkuja samanaikaisesti ja sitten vaihtoehtoja voidaan vertailla. Leanin arvolla **poista hukka** on yhtäläisyyksiä XP:n käytönnön **koodin uudelleenmuokkaus** kanssa. Leanin **poista hukka** koskettaa koko projektin ylimääräisten osien poistamista, kun taas XP:n **koodin uudelleenmuokkaus** on keskittynyt koodin selkeyttämiseen. Scrumissa voidaan myös **poistaa hukkaa**, esimerkiksi korvaamalla hyödyttömän kokouksen **Daily Scrumilla**.

Leanin arvo **rakenna eheys** on mahdollista saavuttaa ajattelutyökalujen **koodin uudelleenmuokkauksen** ja **testauksen** kautta, jotka ovat aivan kuin XP:n käytännöt **testauslähtöinen kehittäminen**, **koodin uudelleenmuokkaus** ja **inkrementaalinen suunnittelu**. Leanin **Five Whys-tekniikkaa** käytetään myös XP:ssä. Leanin arvo **toimita mahdollisimman nopeasti** sisältää samankaltaisuuksia Scrumin arvon **keskittyminen** ja XP:n käytännön **40-tuntinen viikko** kanssa. Scrumissa ja XP:ssa optimaalinen toimitusvauhti voidaan saavuttaa **iteraatioiden** ja **flow:n** avulla. Leanissa ajatus viedään vielä pidemmälle ajatustyökalujen **vetojärjestelmä**, **jonoteoria** ja **viivytyksen kustannukset** avulla.

Kanbanin käyttö vaatii ajattelutapaa, kuten kaikki muutkin ketterät menetelmät, mutta Kanban perustuu erityisesti Lean-ajattelulle. Lean ja Kanban ovat molemmat lähtöisin Toyotan tehtaalta, joten sekin asia näitä menetelmiä yhdistää. Kuten XP:ssa, Scrumissa ja Leanissa myös Kanbanissa

ymmärretään, että työtehtäviä voi olla vain tietty määrä kerrallaan. Kanban hyödyntää Leanin tavoin **Limit Work In Progress (WIP)** menetelmää, jotta työn alla olevien tehtävien määrä pysyy kohtuuden rajoissa.

DevOpsin periaatteet ovat **yhteistyö**, **automaatio**, **jatkuva parantaminen**, **asiakaskeskeinen toiminta** ja **luo lopputulos mielessä**. Periaatteet perustuvat kuuteen pääideologiaan **jatkuvaan integrointiin**, **jatkuvaan toimitukseen**, **jatkuvaan testaukseen**, **jatkuvaan käyttöönottoon**, **jatkuvaan tuotantoon** ja **jatkuvaan yhteistyöhön**. DevOpsin CALMS-viitekehys muodostuu sanoista **Culture**, **Automation**, **Lean**, **Measuring** ja **Sharing**. **Culture**. **Lean** on yksi näistä ja DevOps on ottanut vaikutteita Leanin jatkuvasta kehittymisestä ja kokeilemisesta.

Measuring eli **mittaaminen** on myös Leanin ajatustyökalu. DevOpsin lisäarvo muihin ketteriin menetelmiin verrattuna on se, että sen avulla varmistetaan ohjelmiston turvallinen käyttöönotto sen jälkeen, kun se on kehitetty. DevOpsissa on yhtäläisyyksiä XP:n kanssa testauksen, sen automatisoinnin ja integroimisen kautta, joilla luodaan lisää joustavuutta ja turvallisuutta prosessiin. DevOps tavoitteena on arvon tuottaminen, joka on myös yhteistä muiden ketterien menetelmien kanssa, mm. SAFe ja PRINCE2 korostavat arvon tuottamisen tärkeyttä. DevOpsissa yhteistoiminnallisen työympäristön luomisella on suuri merkitys. Viestintä, tiedon jakaminen, ongelmien nopea havaitseminen ja reagointi ovat tuttuja ominaisuuksia muissakin ketterissä menetelmissä. DevOpsia ja muita ketteriä menetelmiä yhdistää etenkin Lean-ajattelu, joka on vahvasti mukana niissä kaikissa.

SAFe on vienyt ketterien menetelmien parhaiden ominaisuuksien yhdistämisen kaikkein pisimmälle. SAFesta löytyy XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin elementtejä. SAFe:n ScrumXP-tiimit yhdistävät Scrumin voiman ja XP:n käytännöt. ScrumXP-tiimeissä suunnitellaan, toteutetaan, tehdään retrospektiivejä ja tuotetaan arvoa asiakkaalle lyhyessä ajassa. Tiimeillä on SAFen periaatteen **hajauta päätöksenteko** mukaisesti valtuudet ja itsenäisyys suunnitella, toteuttaa ja hallita työtään. ScrumXP-tiimi voi käyttää XP:n ja Scrumin lisäksi Kanbania backlogien hallintaan ja muihin käytäntöihin ja toimintoihin, joita ei ole määritelty XP:ssa tai Scrumissa.

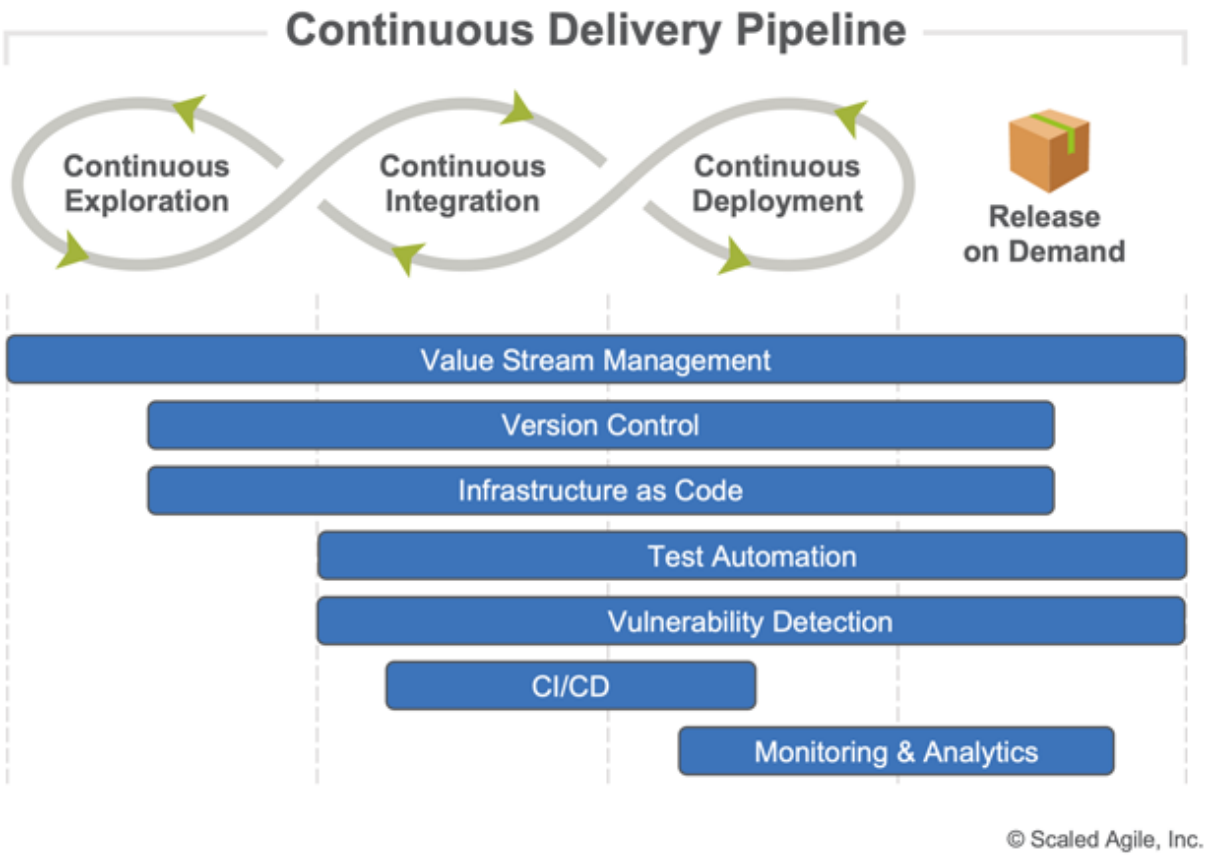
SAFe Team Kanban on toimintamalli, joka sopii hyvin esimerkiksi järjestelmätiimeille, tuotantoon, tukitoimintoihin, laitteistotiimeille, markkinointiin, myyntiin ja henkilöstöhallintoon. Kanbanin

avulla saadaan näkyvyys ja virtaus (flow) leviämään koko organisaatioon. Kanbania hyödynnetään myös Lean-Agile periaatteiden omaksumisessa kaikilla eri osa-alueilla organisaatiossa. Kanbania hyödynnetään backlogien luomisessa ja tarkentamisessa, jotta tiimin tavoitteet voidaan määritellä ja saavuttaa. Kanbania käytetään uusien toiminnallisuuksien rakentamisessa, integroimisessa, testaamisessa, validoinnissa ja käyttöönotossa varmistamaan sisäänrakennettua laatua.

DevOps näkyy SAFe:ssa koko organisaation mukaan ottamisessa projekteihin. SAFe:ssa kehitys, tuotanto, turvallisuus, infrastruktuuri, arkkitehtuuri ovat mukana koko arvovirran ajan mahdollistaakseen yhteistyön, nopeuden, laadun ja turvallisuuden. DevOps näkyy myös SAFe:n jatkuvan toimituksen putkessa (Continuous Delivery Pipeline), jonka DevOpsin ajattelutapa, käytännöt ja työkalut mahdollistavat. DevOpsin **jatkuva integrointi**, **jatkuva toimitus**, **jatkuva testaus**, **jatkuva käyttöönotto**, **jatkuva tuotanto** ja **jatkuva yhteistyö** toteutuvat SAFe:n jatkuvan toimituksen putkessa. DevSecOps on termi, jolla korostetaan tiukkojen tietoturvakäytäntöjen merkitystä jatkuvassa toimituksessa.

SAFe:ssa DevOps tarkoittaa samaa kuin DevSecOps ja tietoturva on ensisijaisen tärkeä osa SAFea. SAFe:ssa tietoturvan avulla suojataan asiakasta, työntekijöitä, kansalaisia, sotilaita, perheitä ja yritystä. SAFe:n nykyaikaiset turvallisuuskäytännöt näkyvät kokonaiskuva julkaisuissa, viitekehysohjeissa, kurssiohjelmissa, arvioinneissa ja artikkeleissa. SAFe on ottanut vaikutteita myös DevOpsin CALMS-viitekehyksestä. SAFe:n CALMR-viitekehys muodostuu sanoista Culture, Automation, Lean Flow, Measurement ja Recovery eli kulttuuri, automaatio, Lean virtaus, mittaus ja palautuminen. DevOpsin Lean on muokattu SAFe:ssa Lean virtaukseksi ja jakaminen on muokattu palautumiseksi. SAFe:ssa jakaminen sisällytetty kulttuuriin. SAFe:n Lean-Agile **kulttuurin** kaikki periaatteet ovat yhteensopivia DevOpsin kanssa. Kulttuurissa edellytetään asiakaskeskeisyyttä, yhteistyötä, riskiensietokykyä ja tiedon jakamista. SAFe:n **automaatiossa** DevOps näkyy jatkuvan toimituksen putkessa. Kuvassa 22 on DevOpsin vaikutukset SAFe:n jatkuvan toimituksen putkessa (ScaledAgileFramework 2021, n.d.).

Kuva 22 DevOpsin vaikutukset SAFe:n jatkuvan toimituksen putkessa (ScaledAgileFramework 2021, n.d.)



Jatkuvan toimituksen putkessa käytetään työkaluina arvovirran hallintaa, versionhallintaa, infrastuktuuria koodina, testauksen automatisointia, haavoittuvuuksien havaitsemista, jatkuvaa integrointia, jatkuvaa toimittamista, seurantaa ja analytiikkaa. **Lean Flow** näkyy ketterissä tiimeissä ja kouluttajissa, jotka pyrkivät saavuttamaan jatkuvan virtauksen. Jatkuvan toimituksen putken jatkuva optimointi tapahtuu visualisoinnin, käynnissä olevan työn rajoittamisen, eräkokojen pienentämisen ja jonojen pituuden hallinnoinnin avulla.

Mittaaminen DevOpsin avulla onnistuu, kun jatkuvan toimituksen putki on hyvin optimoitu. Mittaamisen ansiosta voidaan saavuttaa hyviä tuloksia. Hyödyllisiä virtausmittauksia voidaan tehdä virtausnopeudesta, virtaustehokkuudesta, virtausajasta, virtausjakaumasta ja virtauskuormasta. Laatua mitataan toiminnallisuuden, turvallisuuden ja vaatimuksien toteutumisen osalta. Arvoa mitataan taloudellisien tuloksien ja asiakastyytyväisyyden osalta.

Palautuminen liittyy myös DevOpsiin jatkuvan toimituksen putken kautta. Suunnittelun pitää mahdollistaa matalan riskin julkaisut ja nopea palautuminen toimintahäiriöistä. Palautumisen

nopeuttamiseen on kolme tekniikkaa. **Pysäytä linjasto mentaliteetti** tarkoittaa sitä, että ongelman vaarantaessa ratkaisun arvon, työ pysäytetään ja ongelma ratkaistaan. **Suunnittele ja harjoittele epäonnistumisia** tarkoittaa sitä, että DevOpsissa epäonnistuneet käyttöönotot ovat odotettuja ja niiden vaikutuksia voidaan minimoida palautussuunnitelmien ja harjoitusympäristössä testaamisen avulla. **Nopea korjaaminen ja palauttaminen** tarkoittaa sitä, että tuotantovirheet ovat väistämättömiä ja tiimeillä pitää olla valmiudet korjata virheet ja tarvittaessa palauttaa vakaa toimiva versio. Kuvassa 23 on DevOps CALMS-viitekehys ja SAFe CALMR-viitekehys.

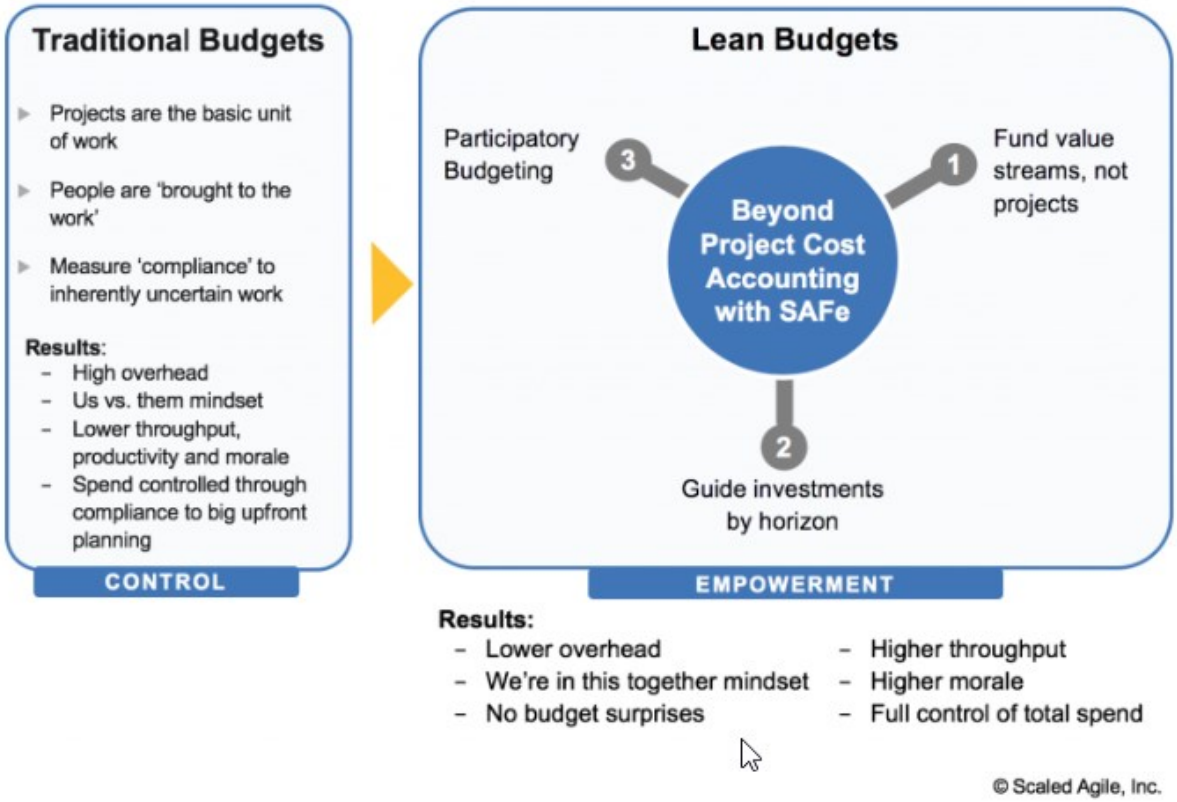
Kuva 23 DevOps CALMS-viitekehys ja SAFe CALMR-viitekehys

DevOps arvot (CALMS-viitekehys)	SAFe CALMR-viitekehys
Kulttuuri	Kulttuuri
Automaatio	Automaatio
Lean	Lean virtaus (flow)
Mittaaminen	Mittaaminen
Jakaminen	Palautuminen

Lean on vahvasti edustettuna SAFe:ssa Lean-Agile ajattelutavan, Lean-talon, Lean-budjetoinnin, Lean UX:n ja Lean Startupin muodossa. **Lean-Agile ajattelutavassa** yhdistyvät ketterän manifestin ja Lean-ajattelun käsitteet ja se on johtajuuden perusta, joka mahdollistaa ketteryyden. **Lean-talossa** Lean-ajattelu on sovellettu ohjelmistokehitykseen sopivaksi. Siinä arvo määritellään täsmällisesti tuotekohtaisesti, jokaisen tuotteen arvovirrat tunnistetaan, arvovirrat laaditaan ilman keskeytyksiä, asiakkaan annetaan ammentaa arvoa tuottajalta ja tavoitellaan täydellisyyttä.

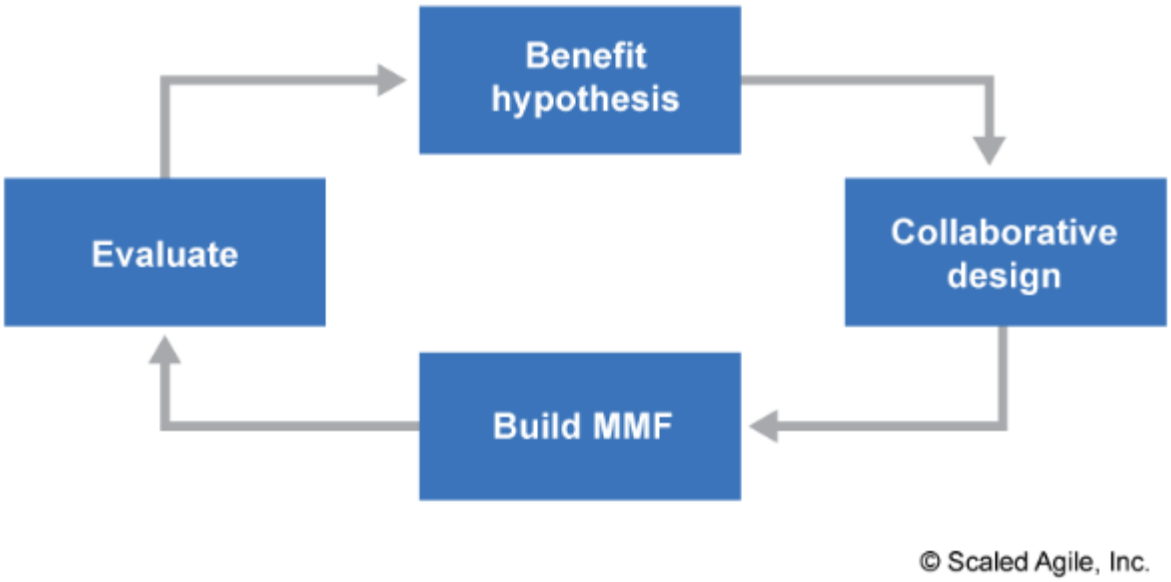
Lean budjetointi on lähestymistapa taloushallintoon, jonka avulla lisätään läpimenoa ja tuottavuutta, vähentämällä projektiin liittyviä kustannuksia. Lean budjetointi koostuu kolmesta keinosta. Rahoitetaan arvovirtoja sen sijaan, että rahoitettaisiin kokonaisia projekteja. Sijoituksia ohjataan horisontin mukaan tasapainottamalla niitä, joka auttaa arvovirtojen omistajia tekemään tietoon perustuvia investointipäätöksiä. Budjetointi on osallistuvaa eli kaikista arvovirroista on edustus budjetoinnissa, joka varmistaa, että arvovirrat saavat riittävän rahoituksen ratkaisujensa edistämiseen. Kuvassa 24 on Lean budjetointi verrattuna perinteiseen budjetointiin (ScaledAgileFramework 2021, n.d.).

Kuva 24 Lean budjetointi verrattuna perinteiseen budjetointiin (ScaledAgileFramework 2021, n.d.)



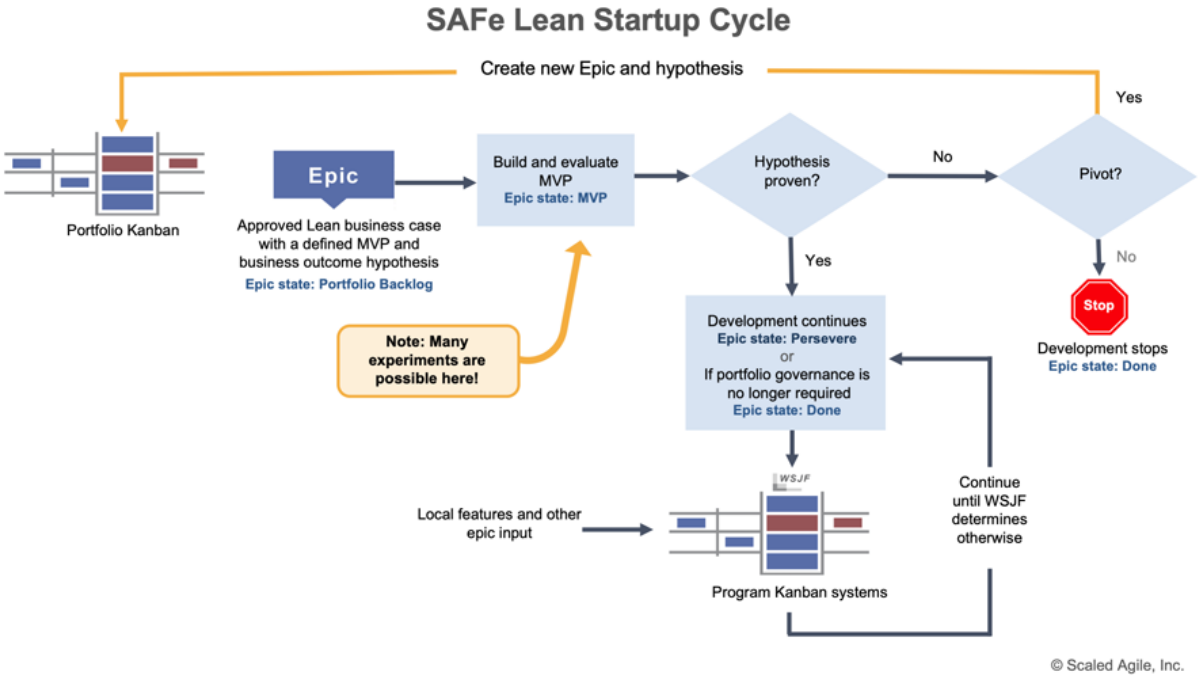
Lean User Experience (Lean UX) on käyttäjäkokemuksien parantamiseen suunniteltu ajattelutapa, kulttuuri ja prosessi, jossa hyödynnetään Lean-Agile menetelmiä. Lean UX-toiminnot toteutetaan pienimmissä mahdollisissa vaiheissa ja onnistuminen määritetään mittaamalla tuloksia hyötyyn perustuvaan hypoteesiin nähden. Lean UX edustaa käyttäjien käsityksiä järjestelmästä, helppokäyttöisyydestä, hyödyllisyydestä ja käyttöliittymän tehokkuudesta. Lean UX ottaa huomioon käyttäjien tarpeet, toiveet, kontekstin ja rajoitukset. Kuvassa 25 on Lean UX-prosessi (ScaledAgileFramework 2021, n.d.).

Kuva 25 Lean UX-prosessi (ScaledAgileFramework 2021, n.d.)



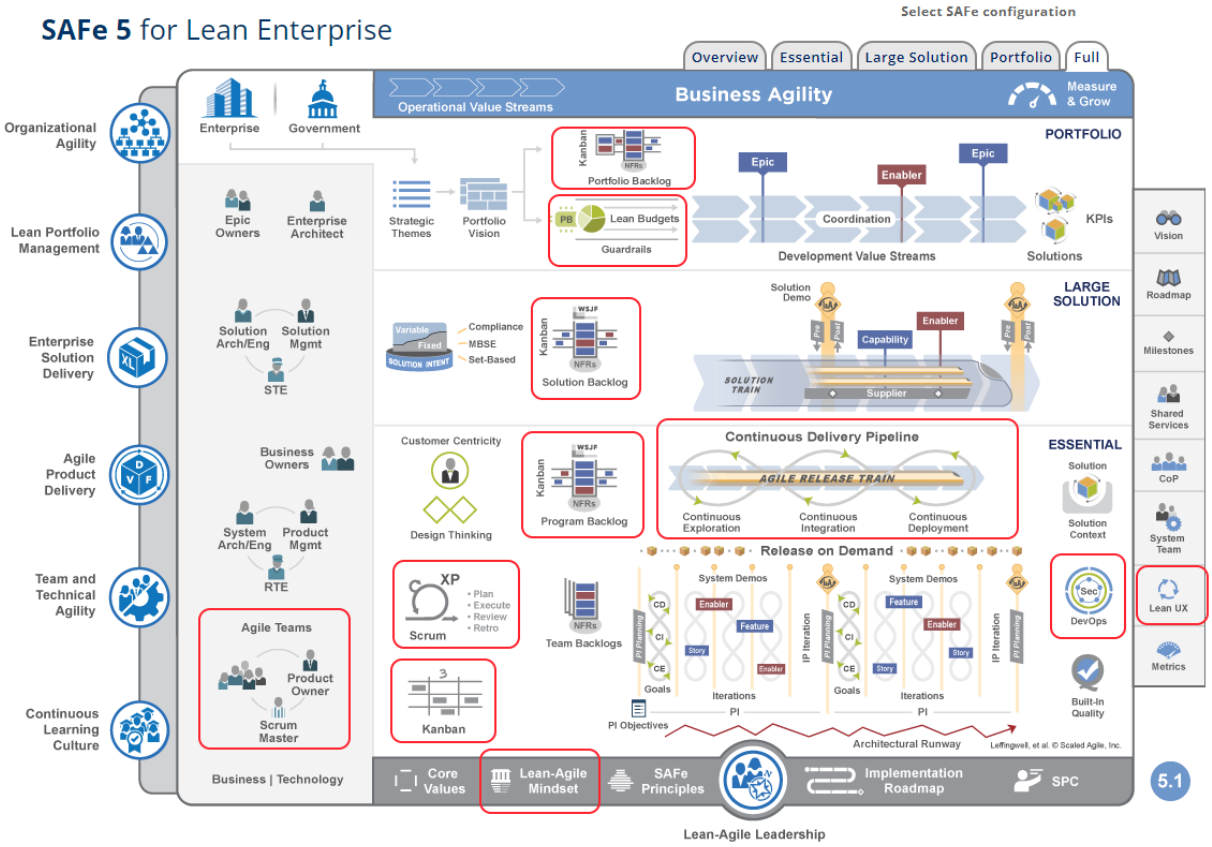
Lean Startup on strategia, jossa innovaatiot ja strategiset investoinnit toteutetaan iteratiivisen rakentamisen, mittaamisen ja oppimisen syklissä. Lean Startup strategiaa sovelletaan eepoksiin (epics) eli yrityksen merkittävimpiin investointeihin, joihin sidosryhmien pitää sopia niiden tarkoituksesta ja määritelmästä. Strategian avulla saavutetaan taloudellista ja strategista etua, kun investointeja ja riskejä hallitaan asteittain, samalla hyödyntäen virtauksen (flow) ja näkyvyyden hyötyjä. Kuvassa 26 on Lean Startup-sykli (ScaledAgileFramework 2021, n.d.).

Kuva 26 Lean Startup-sykli (ScaledAgileFramework 2021, n.d.)



SAFe:n arvo läpinäkyvyys on myös yhteinen tekijä Scrumin empiirisen peruspilarin läpinäkyvyys ja Kanbanin arvon avoimuus kanssa. SAFe:ssa läpinäkyvyys toteutuu erityisen hyvin myös yrityksen kotisivuilla, jossa koko menetelmää voi tarkastella ja jokainen osa-alue on selostettu tarkasti. Käyttöönottoa varten tehty Implemention Roadmap kuvailee tarkasti, kuinka SAFe otetaan käyttöön yrityksessä. Kuvassa 27 on SAFe:n kokonaiskuva, jossa nähdään suoraan missä osa-alueissa on yhtäläisyyksiä muihin ketteriin menetelmiin (ScaledAgileFramework 2021, n.d.).

Kuva 27 SAFe:n kokonaiskuva ja yhtäläisyydet ketteriin menetelmiin (ScaledAgileFramework 2021, n.d.)



PRINCE2 menetelmä pitää projektipäälliköt ajan tasalla ja varmistaa, että projektin kustannukset, aika, laatu, laajuus, hyödyt ja riskit ovat hallinnassa. Sen avulla parannetaan myös viestintää tiimeissä ja sidosryhmien välillä. Menetelmä on joustava, skaalautuva ja räätälöitävissä projekteihin. PRINCE2:sta löytyy yhtäläisyyksiä muihin ketteriin menetelmiin. PRINCE2:n periaate **jatkuva liiketoiminnallinen perustelu** ja teema **Business Case** eli **liiketoimintatarkastelu** ovat hyvin samanlaisia, kuin SAFe:n periaatteet **ota taloudellinen näkökulma** ja **rakenna organisaatio arvon ympärille**. PRINCE2:n teema **laatu** on samankaltainen kuin XP:n periaate **laadukas työ** ja SAFe:n arvo **sisäänrakennettu laatu**.

PRINCE2:n teema **suunnitelma** on samankaltainen kuin XP:n käytäntö **suunnitelmapeli** ja DevOpsin periaate **luo lopputulos mielessä**. PRINCE2:n teema **muutos** ja periaate **hallinnoi poikkeamien mukaan** ovat samankaltaisia kuin XP:n periaate **muutoksen hyväksyminen**, Scrumin empiirinen peruspilari **mukauttaminen** ja Kanbanin muutoksenhallintaperiaate **suostu jatkamaan inkrementaalista ja evolutiivista muutosta**. PRINCE2:n periaate **kokemuksesta oppiminen** on samankaltainen kuin SAFe:n ydinkompetenssi **jatkuvan oppimisen kulttuuri** ja Leanin arvo **vahvista oppimista**. PRINCE2:n periaate **määrittele roolit ja vastuut** ovat yhteistä kaikissa ketterissä menetelmissä. PRINCE2:n periaate **hallinnoi vaiheittain** on samankaltainen kuin XP:n käytäntö **pienet julkaisut**, Scrumin sprintit, Leanin periaate **vetojärjestelmä**, Kanbanin käytäntö **rajoita käynnissä olevia töitä** ja SAFe:n periaate **keskeneräisen työn visualisointi, rajoittaminen, eräkokojen pienentäminen ja jonopituuksien hallinta**.

PRINCE2:n periaate **keskity tuotteisiin** on samankaltainen kuin SAFe:n keskeinen elementti **asiakaslähtöisyys**, DevOpsin periaate **asiakaskeskeinen toiminta**, Kanbanin arvo **asiakaskeskeisyys**, Kanbanin palveluntoimitusperiaate **keskity asiakkaan tarpeisiin ja odotuksiin**, Kanbanin **palvelukeskeisyysagenda** ja XP:n käytäntö **paikan päällä oleva asiakas**.

3.2 Menetelmien eroavaisuudet toisistaan

XP:ssa keskitytään ohjelmistokehitykseen ja sen arvot ja käytännöt rakentuvat kehittämiselle suotuisan ympäristön luomisen ja ohjelmoijien tapojen kehittämisen ympärille. XP auttaa tiimiä suunnittelemaan ja rakentamaan koodia, joka on yksinkertaista ja helposti muutettavissa. XP:n tunnistettavia piirteitä ovat testauslähtöinen kehitys ja pariohjelmointi.
Scrumissa keskitytään ensisijaisesti projektinhallintaan, tehtävän työn laajuuteen, milloin työ toimitetaan ja vastaako työ käyttäjien ja sidosryhmien tarpeita. Scrumin tunnistettavia piirteitä on Daily Scrum, sprintit, Product Owner ja Scrum Master.

Lean on erilainen, eikä se sisällä käytäntöjä. Lean on ajattelutapa, joka sisältää arvoja ja periaatteita. Periaatteita Leanissa kutsutaan ajattelutyökaluiksi. Kanban-tiimit hyödyntävät Lean-ajattelua työskentelytapoihinsa. Kanbanin avulla poistetaan epätasaisuuden, ylikuormituksen ja turhuuden hukkaa Lean-ajattelun pohjalta. Kanban on monien mielestä tehokkain tapa tuoda

Lean-ajattelu organisaatioon. Kanban auttaa tiimiä parantamaan tapaa, jolla se rakentaa ohjelmistoja.

Kanbania käyttävällä tiimillä on selkeä kuva siitä, mitä toimia se käyttää ohjelmistojen rakentamiseen, miten se on vuorovaikutuksessa muun yrityksen kanssa, missä kohtaa se törmää tehottomuudesta ja epätasaisuudesta johtuvaan hukkaan ja miten se voi ajan mittaan parantaa poistamalla hukan perimmäisen syyn. Kanban keskittyy prosessien parantamiseen ja hyödyntää siihen ketteriä ajatuksia kuten **viimeistä vastuullista hetkeä**. DevOps eroaa muista ketteristä menetelmistä etenkin sillä, että se yhdistää kehityksen, testauksen ja ylläpidon toimimaan saumattomasti yhdessä. DevOpsissa dokumentointi on erityisen tärkeää. DevOpsissa testauksen ja prosessien automaatiolla on suuri merkitys. Pilvipalveluiden hyödyntäminen testiympäristöinä on myös ominaispiirre DevOpsille.

SAFe eroaa muista ketteristä menetelmistä erityisesti sillä, kuinka se ottaa organisaation kaikki osa-alueet mukaan projektien läpivientiin. Erityiseksi sen tekee myös se, kuinka se yhdistää XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin parhaimmat ominaisuudet viitekehykseensä. PRINCE2 eroaa muista ketteristä menetelmistä sen keskittymisellä liiketoiminnallisiin tavoitteisiin ja siihen millaiset tuotteet tukevat parhaiten liiketoiminnan tarpeita ja tuottavat arvoa. PRINCE2 edellyttää, että kaikkia sen periaatteita on noudatettava, jotta menetelmä toimii oikein. Prosessien kautta menetelmään saadaan ketteryyttä, koska PRINCE2 ei määrittele mitä menetelmiä kehityksessä käytetään tai miten tiimien pitäisi muodostua.

Ketterien menetelmien eroavaisuuksia voidaan löytää arvoista, periaatteista, käytännöistä, empiirisistä peruspilareista, agendoista, pääideologioista, CALMS-viitekehyksestä, CALMR-viitekehyksestä, ydinkompetensseista, keskeisistä elementeistä, teemoista ja prosesseista. Seuraavissa kuvissa 28, 29 ja 30 nämä ovat vertailussa.

Kuva 28 Vertailussa ketterien menetelmien arvot, CALMS-viitekehys, CALMR-viitekehys ja PRINCE2:n teema

XP arvot	Scrum arvot	Lean arvot	Kanban arvot
Viestintä	Sitoutuminen	Poista hukka	Avoimuus
Yksinkertaisuus	Keskittyminen	Vahvista oppimista	Tasapaino
Palaute	Avoimuus	Päätä mahdollisimman myöhään	Yhteistyö
Kunnioitus	Kunnioitus	Toimita mahdollisimman nopeasti	Asiakaskeskeisyys
Rohkeus	Rohkeus	Vahvista tiimiä	Työnkulku
		Rakenna eheys	Johtaminen
		Näe kokonaisuus	Ymmärtäminen
			Yhteisymmärrys
			Kunnioitus

DevOps arvot (CALMS-viitekehys)	SAFe CALMR-viitekehys	SAFe arvot	PRINCE2 teemat
Kulttuuri	Kulttuuri	Ohjelman toteuttaminen	Business Case eli liiketoimintatarkastelu
Automaatio	Automaatio	Kohdistaminen	Organisaatio
Lean	Lean virtaus (flow)	Sisäänrakennettu laatu	Laatu
Mittaaminen	Mittaaminen	Läpinäkyvyys	Suunnitelma
Jakaminen	Palautuminen		Riski
			Muutos
			Edistyminen

Kuvassa 28 nähdään, että DevOpsissa ei ole määritellä arvoja, mutta CALMS-viitekehystä voidaan pitää arvojen kaltaisena. SAFe:ssa on arvojen lisäksi DevOpsista muokattu CALMR-viitekehys ja se on sen vuoksi mukana tässä listassa. PRINCE2:ssa ei määritellä arvoja, mutta siinä on teemoja, jotka ovat arvojen kaltaisia.

Kuva 29 Vertailussa ketterien menetelmien periaatteet ja Scrumin empiiriset peruspilarit

XP periaatteet
Nopea palaute
Oletettu yksinkertaisuus
Inkrementaaliset muutokset
Muutoksen hyväksyminen
Laadukas työ

Scrum empiiriset peruspilarit
Läpinäkyvyys
Tarkastelu
Mukauttaminen

Lean periaatteet (ajatustyökalut)
Näe hukka
Refaktorointi
Testaus
Koettu eheys
Käsitteellinen eheys
Mittaus
Vetojärjestelmä
Jonoteoria
Viivekustannus

Kanban muutoksenhallintaperiaatteet
Aloita siitä, mitä teet
Suostu jatkamaan inkrementaalista ja evolutiivista muutosta
Rohkaise johtajuutta kaikilla tasoilla

Kanban palveluntoimitusperiaatteet
Keskity asiakkaan tarpeisiin ja odotuksiin
Hallinnoi työtä
Palveluverkoston säännöllinen tarkastelu

DevOps periaatteet
Yhteistyö
Automaatio
Jatkuva kehittyminen
Asiakaskeskeinen toiminta
Luo lopputulos mielessä

SAFe periaatteet
Ota taloudellinen näkökulma
Sovella systeemiajattelua
Oletetaan vaihtelevuus ja säilytetään vaihtoehdot
Rakenna inkrementaalisesti nopeiden integroitujen oppimissyklien avulla
Perusta välitavoitteet toimivien järjestelmien objektiiviseen arviointiin
Keskeneräisen työn visualisointi ja rajoittaminen, eräkokojen pienentäminen ja jonopituuksien hallinta
Sovella kadenssia ja synkronoi poikkihallinnollisen suunnittelun kanssa
Vapauta tietotyöntekijöiden sisäinen motivaatio
Hajauta päätöksenteko
Rakenna organisaatio arvon ympärille

PRINCE2 periaatteet
Jatkuva liiketoiminnallinen perustelu
Kokemuksesta oppiminen
Määrittele roolit ja vastuut
Hallinnoi vaiheittain
Hallinnoi poikkeamien mukaan
Keskity tuotteisiin
Räätälöi ympäristöön sopivaksi

Kuvassa 29 nähdään, että Scrumissa ei määritellä periaatteita, mutta siinä on empiiriset peruspilarit. Leanin periaatteita kutsutaan ajatustyökaluiksi, mutta ne ovat periaatteiden kaltaisia. Kanbanissa periaatteet on jaettu muutoksenhallintaperiaatteisiin ja palveluntoimitusperiaatteisiin.

Kuva 30 Ketterien menetelmien käytännöt, Kanbanin agendat, DevOpsin pääideologiat, SAFe:n ydinkompetenssit ja keskeiset elementit ja PRINCE2:n prosessit

XP käytännöt	Kanban käytännöt	DevOps pääideologiat
Testauslähtöinen kehitys	Visualisoi työkulku	Jatkuva integrointi
Suunnittelupeli	Rajoita käynnissä olevia töitä	Jatkuva toimitus
Paikan päällä oleva asiakas	Hallinnoi virtausta (Flow)	Jatkuva testaus
Pariohjelmointi	Tee prosessikäytännöistä selkeitä	Jatkuva käyttöönotto
Koodin refaktorointi	Toteuta palautesilmukat	Jatkuva tuotanto
Pienet julkaisut	Kehitä yhteistyössä ja kokeellisesti	Jatkuva yhteistyö
Yksinkertainen suunnittelu	Kanban agendat	
Koodausstandardit	Kestävyysagenda	
Koodin kollektiivinen omistus	Palvelukeskeisyysagenda	
Järjestelmämetafora	Selviytymisagenda	
40-tuntinen viikko		

SAFe ydinkompetenssit	SAFe keskeiset elementit	PRINCE2 prosessit
Yritysratkaisujen toimitus	Koko liiketoiminnan ja organisaation ketteryys	Projektin perustaminen
Ketterä tuotetoimitus	Asiakaslähtöisyys	Projektin käynnistäminen
Tiimin ja tekniikan ketteryys	Arvovirrat ketteriksi julkaisujuniksi	Projektin johtaminen
Lean-salkunhallinta	Tiimien yhteenliittyminen ketteriksi julkaisujuniksi	Vaiheen hallinnointi
Organisaation ketteryys	Program Increment (PI) eli ohjelman inkrementti	Tuotteen toimituksen hallinnointi
Jatkuvan oppimisen kulttuuri	Arvovirtojen budjetointi	Vaiheen rajojen hallinnointi
Lean-Agile johtaminen		Projektin päättäminen

Kuvassa 30 nähdään, että Scrumissa, Leanissa, DevOpsissa, SAFe:ssa ja PRINCE2:ssa ei ole määritelty käytäntöjä. Kanbanissa on käytäntöjen lisäksi agendat määriteltynä. DevOpsissa on pääideologiat määriteltynä. SAFessa on ydinkompetenssit ja keskeiset elementit määriteltynä. PRINCE2:ssa on prosessit määriteltynä.

Kuva 31 Ketterien menetelmien erilaiset roolit

XP roolit	Scrum roolit	Lean roolit	Kanban roolit
Asiakkaat	Scrum Master	Lean Master	Service Delivery Manager
Kehittäjät	Tuoteomistaja	Lean-projektinjohtaja	Service Request Manager
Seuraajat	Kehittäjät	Lean-tiimi	
Valmentajat			

DevOps roolit	SAFe ART-roolit	PRINCE2 roolit
The DevOps Evangelist	Product Management	Executive
The Code Release Manager	System Architect/Engineering	Senior User
The Automation Architect	Release Train Engineer (RTE)	Senior Supplier
The Experience Assurance Expert (XA)	Business Owners	Project Assurance
The Software Developer/Tester	SAFe tiimiroolit	Change Authority
The Security & Complience Engineer (SCE)	Scrum Master	Project Manager
	Tuoteomistaja	Project Support
	Kehittäjät	Team Manager

Kuvassa 31 nähdään, että roolit ovat varsin erilaisia, vaikka yhtäläisyyksiäkin löytyy. Asiakas on otettu rooliksi mukaan XP:ssa, mutta toisaalta Scrumissa ja SAFe:ssa tuoteomistaja edustaa myös asiakasta. DevOpsin The Experience Assurance Expert (XA) huolehtii myös, että tuotteessa on otettu huomioon asiakkaan toiveet ja vaatimukset. Scrumin roolit ja SAFe:n tiimiroolit ovat täysin samanlaiset, mutta SAFe:ssa tiimit hyödyntävät Scrumin lisäksi myös XP:ta ja Kanbania. XP:n, Leanin, Kanbanin ja DevOpsin roolit eivät ole SAFe:ssa käytössä, vaikka menetelmiä muilta osin SAFe:ssa hyödynnetäänkin. Leanin ja Kanbanin roolit ovatkin harvemmin käytössä, sillä niitä käytetään useimmiten ohjelmistokehityksessä muiden menetelmien kanssa yhdessä.

Kuva 32 Ketterien menetelmien tiimit

				Tiimit			
	XP	Scrum	Lean	Kanban	DevOps	SAFe	PRINCE2
Tiimin koko	2-12 henkilöä	3-9 henkilöä	3-5 henkilöä	4-12 henkilöä	5-10 henkilöä	2-12 henkilöä	Ei määritelty
Tiimien määrä	1 tiimi	1-4 tiimiä	Ei määritelty	Ei määritelty	Ei määritelty	5-12 tiimiä	Ei määritelty
Henkilöt yhteensä	Ei määritelty	Ei määritelty	Ei määritelty	Ei määritelty	Ei määritelty	Yht. 50-125 hlö	Ei määritelty

Kuvassa 32 nähdään ketterien menetelmien tiimikoko, tiimien määrä ja henkilömäärät yhteensä. Pienissä projekteissa Extreme Programming (XP) on havaittu erittäin tehokkaaksi, koska XP:n pariohjelmointi sekä koodaus- ja testauskäytännöt pääsevät pienissä projekteissa oikeuksiinsa. Extreme Programming onkin suunniteltu vain yhden tiimin ja 2-12 henkilön tiimeissä toimivaksi.

Scrumia käytettäessä on mahdollista tehdä hieman isompia projekteja, kuin Extreme Programmingilla. Tämä johtuu siitä, että Scrum-tiimejä voi olla 1-4. Isompiin projekteihin onkin hyvä ottaa useampi Scrum-tiimi, koska Scrumissa tiimikoon kasvaminen liian suureksi on ongelmallista. Scrumin ongelmia voidaan helpottaa ottamalla projektiin mukaan esimerkiksi Kanbanin ominaisuuksia, joilla on helpottava vaikutus varsinkin isommissa projekteissa.

Leania tai Lean-ajattelua on mahdollista tuoda mukaan myös kaikkin muihin ketteriin menetelmiin ja sitä onkin avoimesti tuotu kaikkiin muihin paitsi PRINCE2:seen. Sen vuoksi projektin koolla ei ole Leanin kohdalla yhtä suurta merkitystä, sillä se sopii mukaan niin pieniin kuin suuriinkin projekteihin. SAFe on ottanut Lean-ajattelua mukaan johtamiseen, budjetointiin ja useille muille osa-alueille, joten etenkin suurissa projekteissa Leanista on apua.

Kanban on Leanin tavoin mahdollista ottaa mukaan kaikkiin muihinkin menetelmiin. Kanban on monien mielestä paras tapa tuoda Lean-ajattelua organisaatioon ja Kanbanin avulla projekteihin saadaan siis molempia mukaan. Tämän vuoksi projektikoko voi olla pieni tai suuri, mutta etenkin suurissa projekteissa Kanbanista on apua hukan poistamisessa projekteista.

DevOps toimii erittäin hyvin myös isommissa projekteissa, koska siinä on yhdistetty kehittäminen, ylläpito ja testaus toimimaan saumattomasti yhteen. Myös DevOpsin pilvipalveluiden

hyödyntäminen testiympäristönä, pitkälle viety prosessien ja testauksen automatisointi tehostavat toimintaa niin paljon, että myös isommat projektit saadaan maaliin nopeammassa aikataulussa. DevOpsissa on myös hyvin pitkälle viety jatkuva integrointi ja jatkuva toimittaminen, joka tehostaa kehitysnopeutta ja varmuutta entisestään isoissakin projekteissa.

SAFe on suunniteltu nimenomaan todella suuriin projekteihin sopivaksi ja sitä voi olla todella haastavaa saada toimimaan pienissä projekteissa, joissa on alle 50 työntekijää. Suurissa projekteissa SAFe on erittäin hyvä vaihtoehto, sillä siinä tiimejä voi olla 5-12 ja työntekijöitäkin voi olla yli 125.

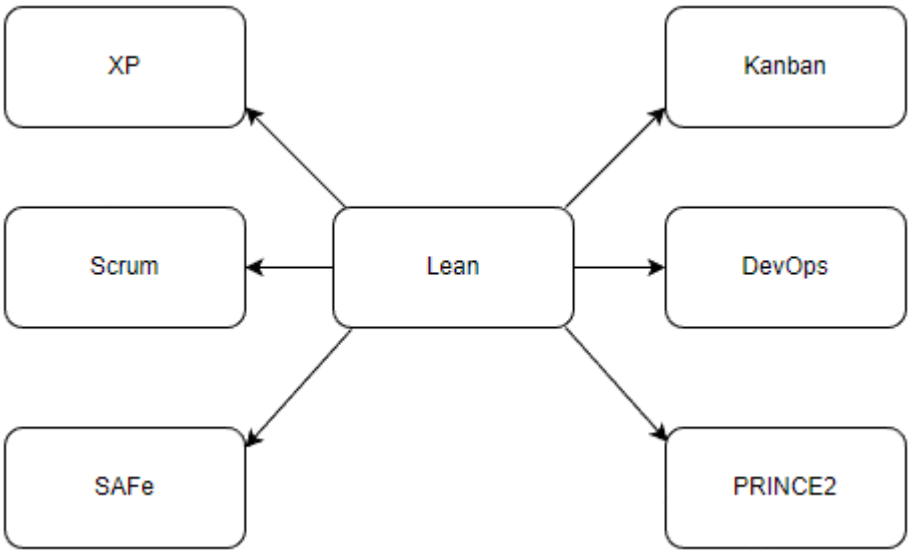
PRINCE2 pääsee SAFe:n tavoin oikeuksiinsa paremmin isommissa projekteissa. On sanottu, että PRINCE2:n avulla voisi toteuttaa jopa ydinvoimalan ja sillä tehdäänkin hävittäjälentokoneiden ohjelmointia, joten se kertoo varmasti siitä, että isoja projekteja todella voidaan sen avulla tehdä. Siinä tehdään tarkat suunnitelmat projekteihin etukäteen ja siksi sen kanssa tehdyt projektit ovat hyvin ennakoitavissa ja pysyvät helpommin aikataulussa. SAFe:n tavoin PRINCE2:ssa liiketoiminnalliset elementit otetaan hyvin huomioon, joka on varmasti isoissa projekteissa olennaista myös rahoittajien näkökulmasta.

3.3 Kustomoidut menetelmät ja useamman menetelmän käyttö yhtäaikaisesti

SAFe on yhdistänyt onnistuneesti XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin parhaimpia puolia toimimaan yhdessä viitekehyksessä ja se todistaa, että useamman menetelmän yhdistäminen todella on mahdollista. SAFe on yhdistänyt XP:n koodaus ja testauskäytännöt, Scrumin sprintit, päivittäiset Scrumit ja product backlogit ja Kanbanin prosessien parantamisen Scrumin rooleilla toimivaksi tiimiksi. On myös olemassa Scrum-ban niminen kustomoitu menetelmä, jossa Scrumin ja Kanbanin parhaat puolet ovat yhdistetty. Ajatus on samankaltainen SAFe:n kanssa, mutta SAFe:ssa mukaan on otettu vielä XP:n koodaus ja testaus käytännöt. Kanbanin mukana olo tuo automaattisesti mukanaan Lean-ajattelun, koska Kanban pohjautuu Lean-ajattelulle.

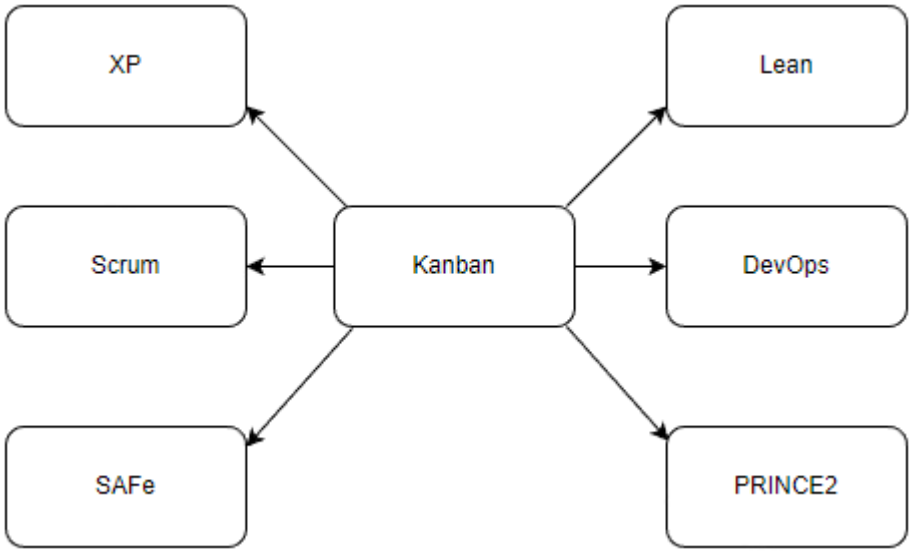
Lean-ajattelua voidaan hyödyntää kaikissa ketterissä menetelmissä. Kuvassa 33 on Leanin yhteensopivuus kaikkiin menetelmiin.

Kuva 33 Leanin yhteensopivuus kaikkiin menetelmiin



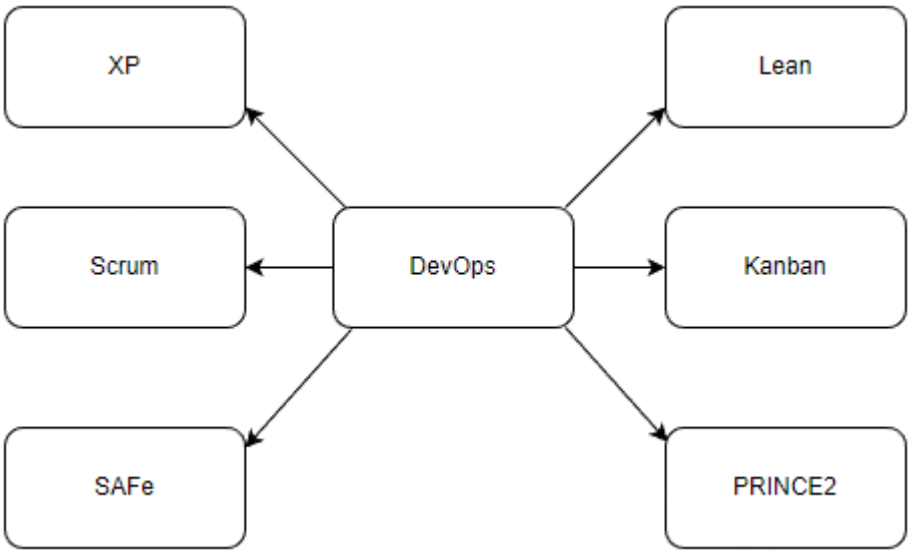
Kanban antaa käytäntöjä, joiden avulla projektia voidaan parantaa, mutta se ei kerro millä menetelmällä projektia pitää johtaa. Kanbania voi siis hyödyntää kaikissa ketterissä menetelmissä eli XP:ssa, Scrumissa, Leanissa, DevOpsissa, SAFe:ssa ja PRINCE2:ssa. Kuvassa 34 on Kanbanin soveltuvuus muihin ketteriin menetelmiin.

Kuva 34 Kanbanin soveltuvuus muihin ketteriin menetelmiin



DevOps on Leanin ja Kanbanin tavoin mahdollista ottaa käyttöön kaikissa muissakin ketterissä menetelmissä, esimerkiksi Scrum-tiimi tai XP-tiimi on mahdollista laajentaa ottamalla tuotanto mukaan kehittämiseen. Kuvassa 35 on DevOpsin soveltuvuus muihin ketteriin menetelmiin.

Kuva 35 DevOpsin soveltuvuus muihin menetelmiin



SAFe:ssa ketterien menetelmien yhdistäminen XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin osalta on sisäänrakennettuna SAFe:n viitekehykseen. PRINCE2:ssa ei määritellä menetelmiä, joita prosessissa käytetään, joten projektitiimin on mahdollista käyttää XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin ominaisuuksia projektin läpiviemiseksi.

3.4 Menetelmien hyödyt ja rajoitukset

XP:n hyödyt vähentävät kehitysaikaa ja kustannuksia. Jatkuva testaus ja refaktorointi mahdollistavat vakaiden järjestelmien kehittämisen, jotka toimivat hyvin eivätkä vaadi suuria virheenkorjauksia. XP:n arvo yksinkertaisuus ilmenee koodin selkeydessä, sillä koodia on helppo lukea ja se on helposti muutettavissa. Minimum Viable Product (MVP) eli pienin toimiva tuote saadaan toimitettua nopeasti, koska vain välttämättömät ominaisuudet rakennetaan. Dokumentoinnin tarve vähenee, koska tilaa vievät vaatimusasiakirjat on korvattu käyttäjätarinoilla. Ylitöitä ei pääsääntöisesti tehdä ollenkaan ja jos tehdään, niin se on erittäin vähäistä. Hyvä läpinäkyvyys on mahdollista jatkuvan viestinnän avulla ja tiimi on ajan tasalla projektin etenemisestä. Yhteistyö on mahdollista pariohjelmoinnin avulla tiimeissä ja asiakkaiden kanssa, koska he osallistuvat kehitys- ja testausprosessiin. Asiakastyytyväisyys toteutuu myös asiakkaiden osallistuvan roolin ansiosta.

XP:ssä on myös joitakin rajoituksia, joita kannattaa punnita silloin, kun valitsee mitä menetelmää aikoo käyttää projektissaan. Epäselvät estimaatit hankaloittavat projekteja, koska laajuutta,

kustannuksia ja aikaa ei mitata. Ajanhukkaa syntyy, koska säännölliset asiakastapaamiset vievät aikaa itse koodin kirjoittamisesta. Dokumentaation vähäisyys saattaa kostautua projektin laajenemisella, koska selkeät vaatimusmäärittelyt puuttuvat. Perinteisestä ohjelmistokehityksestä siirtyminen XP:n käyttöön vaatii suuria kulttuurisia ja rakenteellisia muutoksia. Pariohjelmointi vie enemmän aikaa ja se ei välttämättä onnistu tarkoituksenmukaisesti, jos ohjelmointipari ei ole yhteensopiva. XP:n paikalliset tiimit, jossa tiimit ja asiakkaat ovat henkilökohtaisesti läsnä, rajoittavat menetelmän käyttöä hajautetuissa tiimeissä. Asiakkaat eivät aina halua osallistua kehittämiseen tai heillä ei riitä aikaa siihen. Määräaikojen ollessa tiukkoja tämä saattaa lisätä stressiä, koska arvokasta palautetta ei saada kehittämisen tueksi. Koodiin keskittyminen suunnittelua enemmän ilmenee ongelmina laadunvarmistuksessa, koodin päällekkäisyydessä ja kokemattomien kehittäjien huonoina tuloksina.

Scrumin hyötyjä on useita. Sen avulla tiimit saavat projektin nopeasti ja tehokkaasti valmiiksi. Scrumin avulla rahaa ja aikaa käytetään tehokkaasti. Sprinttien ansiosta projektit ovat helpommin hallittavissa. Koodaus ja testaus tapahtuvat sprinttikatselmuksen aikana. Se sopii hyvin nopeasti eteneviin projekteihin. Päivittäispalaverit mahdollistavat läpinäkyvän toiminnan. Asiakkaiden ja sidosryhmien palaute huomioidaan projekteissa. Palautteen perusteella on helppo tehdä muutoksia, koska sprintit ovat lyhyitä. Päivittäispalaverien ansiosta työn etenemistä on helppo seurata.

Scrumissa on myös joitakin rajoituksia, joita voidaan kuitenkin helpottaa yhdistämällä siihen muita menetelmiä. Scope creep tarkoittaa jatkuvaa tai hallitsematonta kasvua projektin laajuudessa ja näin voi käydä Scrumia käytettäessä, koska lopetuspäivää ei ole määritelty. Jos Scrum-tiimi ei ole sitoutunut tai yhteistyökykyinen projektissa, niin epäonnistumisen mahdollisuus on suuri. Scrumin käyttö on haastavaa, jos tiimin koko on suuri. Scrumin käyttö vaatii oikein toimiakseen kokeneita työntekijöitä. Päivittäispalaverit voivat olla turhauttavia työntekijöille. Scrum vaatii toimiakseen Scrum Masterin, tuoteomistajan ja kehittäjät. Jonkin roolin puuttuessa tai poistuessa kesken projektin, se vaikuttaa erittäin negatiivisesti projektin läpiviemiseen. Laadun toteutuminen vaatii paljon testausta.

Leanissa on myös paljon hyötyjä. Leanin avulla voidaan vähentää projektin kustannuksia. Leanin hukan poiston ansiosta säästetään aikaa, rahaa ja inhimillisiä voimavaroja. Leanin panostus

laatuun vähentää myös tuki- ja ylläpitokustannuksia. Leanin avulla saadaan kehitettyä arvokas tuote, koska resurssit keskitetään arvon tuottamiseen asiakkaalle. Leanin avulla tuote vastaa kysyntään, koska jatkuvan oppimisen avulla saadaan palautetta läpi koko kehitysprosessin ajan ja tuote saadaan räätälöityä vastaamaan käyttäjien tarpeita. Leanin avulla tiimi on itseorganisoituva ja omavarainen, koska työntekijöillä on valtuudet tehdä päätöksiä itse ilman erillisiä määräyksiä tai lupia.

Leanissa on myös joitakin rajoituksia. Leanin autonomiset tiimit vaativat paljon kokemusta työntekijöiltä, koska päätöksien teko on mahdollista ilman erillisiä määräyksiä tai lupia. Tiimeissä vaaditaan ohjelmoinnin lisäksi kokemusta monelta eri osa-alueelta, jotta tuotekehitys on riittävän kokonaisvaltaista. Leanissa korostetaan päätöksen tekoa viimeisellä vastuullisella hetkellä ja se voi aiheuttaa huolta monille asianosaisille. Lean vaatii työskentelytapojen muutosta ja tiimin jäsenten tulee olla valmiita oppimaan uusia asioita jatkuvasti.

Kanbanissa on myös paljon hyötyjä. Kanban on helppokäyttöinen ja se on helposti ymmärrettävissä myös ensikertaa siihen tutustuvalle. Kanban on joustava, joka mahdollistaa reaaliaikaiset muutokset projektin aikana. Kanbanissa yhteistyö on tärkeää ja sen avulla tiimi saadaan työskentelemään yhdessä halutun lopputuloksen saavuttamiseksi. Kanban on prosessi, joka keskittyy jatkuvaan toimitukseen ja lisää tuottavuutta laajemmissa projektisykleissä.

Kanbanissa on myös joitakin rajoituksia. Kanbanissa ei aseteta tiukkoja vastuita ja sen vuoksi projekti on vaarassa harhautua. Kanban-tauluista saattaa tulla monimutkaisia, joka voi aiheuttaa hämmennystä. Kanbanista puuttuu ajoitusparametrit ja se voi olla projektipäällikön mielestä ongelmallista.

DevOpsissa on paljon hyötyjä. Pilvipalveluiden avulla DevOps työkalut ovat helposti kaikkien asianosaisten saatavilla. Sovellusten julkaisuaika on nopea, tuottavuus on korkeaa ja prosessit ovat tehokkaita. Tuotantosyklit ovat lyhyitä. Operatiivinen tuki on hyvä. Työntekijät ovat sitoutuneita ja motivoituneita. Asiakaskokemukset ovat hyvin hallinnoitavissa. Tiimillä on selkeä tuotevisio. Sovellusten käyttöönoton onnistumisprosentti on korkea. Kehitettävät tuotteet ovat laadukkaita. Tiimit ovat tehokkaita. Joustavuus ja käyttäjätuki ovat hyviä. Kehitettävien ohjelmistojen

epäonnistumisen mahdollisuudet ovat pieniä. Moniosaaminen ja kehittyminen ovat korkealla tasolla.

DevOpsissa on myös joitakin rajoitteita. DevOps vaatii ajattelutavan muutoksen koko yrityksessä. Liiketoiminnan tietoturva ulkoistetaan DevOps toiminnoille, joka saattaa aiheuttaa huolta yrityksessä. Vanhojen järjestelmien kanssa voi tulla teknisiä haasteita DevOpsin käyttöönotossa. Jatkuva integrointi ja jatkuva toimittaminen vaativat erillisen tietoturvan. DevOps asiantuntijoita voi olla vaikea löytää. Puutteet DevOps osaamisessa voivat aiheuttaa ongelmia automatisoinnissa ja jatkuvassa integroinnissa. DevOps työkalujen määrä ja työkalujen vaihtaminen voi aiheuttaa haasteita. DevOps ympäristön luomisessa on korkeat perustamiskustannukset.

SAFe:ssa on paljon hyötyjä. Sen avulla monipuoliset tiimit saadaan toimimaan yhdessä. Program Increment suunnittelun ansiosta kehitettävän ohjelmiston vaatimukset saadaan selvitettyä selkeästi. Liiketoiminnan tavoitteet saadaan yhdenmukaistettua. Arvon tuottaminen on nopeampaa, koska kaikki osa-alueet ovat mukana projektissa ja toimivat yhteistyössä noudattaen samoja prosesseja ja protokolleja. Toiminta on asiakaskeskeistä ja tutkimuslähtöistä, jonka vuoksi se palvelee erityisen hyvin loppukäyttäjää. Ketteryys on huomioitu erityisen tehokkaasti SAFe:n yhdistäessä useiden ketterien menetelmien parhaita puolia toimimaan yhdessä. Työntekijöillä on hyvä sitoutuminen projekteissa.

SAFe:ssa on myös joitakin rajoituksia. Paljon omaa teknistä terminologiaa, jonka sisäistämisessä voi kestää jonkin aikaa. Suunniteltu toimimaan suurissa yrityksissä ja sitä voi olla vaikeaa soveltaa pienissä projekteissa tai yrityksissä, joissa on alle 50 työntekijää. Päätöksien teko tapahtuu johdon ylimmällä tasolla, jonka vuoksi työntekijät eivät voi tehdä itsenäisiä ratkaisuja kehittämisessä. Pitkäkestoisten projektien määrittely vaatii sijoittajien hyväksynnän ja siinä on riski, että määrittely tehdään väärin.

PRINCE2:ssa on paljon hyötyjä. Ennakoitavuus on hyvää, koska projekti jaetaan vaiheisiin ja sen seuranta on mahdollista koko projektin ajan. Projekti pysyy hyvin hallinnassa ja kulkee oikeaan suuntaan. Laatuvaatimukset määritellään projektin alussa korkealle ja projektia voidaan pitää onnistuneena vasta, kun alkuperäiset laatuvaatimukset täyttyvät. Projektin osuudet standardisoidaan, jotta väärinkäsityksiä ei syntyisi. Työntekijät tietävät mitä tehdään ja milloin

standardoinnin, raportointiprosessin ja dokumentaation ansiosta. Riskienhallinta on kustannustehokasta, koska vain muutoksiin reagoidaan ja kokouksia pidetään vain, jos se on välttämätöntä. Menetelmässä on otettu huomioon projektinhallinnan asiantuntijoiden mielipiteet ja se on testattu ja todettu hyväksi menetelmäksi.

PRINCE2:ssa on myös joitakin rajoituksia. Dokumentoinnin määrä on erittäin suuri koko projektin ajan. Projektivaiheet tekevät menetelmästä jäykän ja sen vuoksi mukautuvuus on haastavaa. Menetelmän käyttäminen vaatii erillisen kouluttautumisen ja ilman koulutusta menetelmästä ei ole mitään hyötyä.

4 Kokemukset yrityksissä – haastattelut

Yrityshaastattelut toteutettiin laadullisena strukturoituna haastatteluna. Kysymykset oli laadittu etukäteen ja ne lähetettiin sähköpostilla yrityksille vastattavaksi. Seuraavana ovat koostetut vastaukset yrityshaastatteluiden kysymyksiin. Yrityksistä käytetään nimiä Yritys X ja Yritys Y.

1. Mistä seuraavista menetelmistä XP, Scrum, Lean, Kanban, DevOps, SAFe ja PRINCE2 teillä on kokemusta?

Yritys X:

Ketteristä menetelmistä käytössä on ollut XP, Scrum, Kanban ja DevOps.

Yritys Y:

Ketterien menetelmien toimintamalleissa on usein hyvin paljon päällekkäisyyksiä, joten siitä näkökulmasta yrityksemme sovelluskehityksessä käytetään menetelmiä useista eri ketteristä kehitystyökaluista, mutta pääsääntöisesti yleisimmin on käytössä Kanban, Scrum ja joltain osin DevOps.

2. Miten menetelmät ovat valikoituneet tiettyihin projekteihin?

Yritys X:

Menetelmien valikoitumiseen ovat vaikuttaneet ainakin projekteissa mukana olleiden henkilöiden aiemmat kokemukset, kyseisen yrityksen / organisaation käytännöt, ja henkilöstön läpikäymät koulutukset.

Yritys Y:

Yrityksemme on sovelluskehityksen osalta pääosin keskittynyt projektitoimituksiin (omien tuotteiden sijaan), jolloin on tärkeää, että asiakas pystyy mahdollisimman pienellä kynnyksellä omaksumaan käytetyn projektinhallintamenetelmän. Kanban ja nykyisin myös Scrum ovat Suomessa kohtuullisen hyvin tunnettuja menetelmiä ja usein siitä syystä mukana projekteissa.

Kanbania käytetään tyypillisesti ihan puhtaasti työnkulkuun liittyvässä projektiseurannassa, eli projektitehtävät (ja sovelluksiin kehitettävät toiminnallisuudet) kuvataan omiksi korteikseen ja niiden kulkua seurataan kanban-työpöydällä koreittain. Tyypillisesti korit on jaettu esimerkiksi: ”backlog”, ”kehityksessä”, ”testauksessa” ja ”valmis”. Kanban on myös usein käytetty menetelmä järjestelmien ylläpitopalveluissa, joissa vastaavalla tavalla avoimet palvelu- ja muutospyynnöt avataan kortteina kanban-työpöydälle ja kiinnitetään eri vastuuhenkilöille. Usein työpöytä on myös asiakkaan nähtävissä mikä tuo läpinäkyvyyttä kehitysprosessiin ja asiakas pystyy seuraamaan kehitysasioiden tilannetta. Tyypillisesti SLA-palvelusopimusten mittarit (ja sanktiot) ovat konfiguroitu kanban-työkaluihin, jolloin pystytään seuraamaan toimittajien reagoimista ja tehtävien ratkaisuaikoja.

Itse sovelluskehityksessä käytämme yleisimmin Scrumia jonka menetelmiä on hieman mukautettu yrityksemme tarpeisiin. Hyvin tyypillisesti (varsinkin projektimallisessa sovelluskehityksessä) käytettävissä olevat resurssit (budjetti tai henkilöresurssit) rajoittavat jonkin verran menetelmien käyttöä täysin mallin mukaisesti tai esimerkiksi asiakkaista riippuvista syistä mallia on tarve muokata. On esimerkiksi yleisestä, että asiakkaiden puolelta ei ole mahdollista kiinnittää projekteihin henkilöitä tuoteomistajiksi (tai Scrum Mastereiksi) tai heillä ei ole aikataulullisesti mahdollista osallistua kehitysvaiheeseen ketterän menetelmän vaatimalla tavalla. Toinen hyvin tyypillinen haaste projektimallisessa sovelluskehityksessä on, että liiketoiminnan tarpeet, sovellusten toiminnallisuudet (ja budjetti) on kuvattava etukäteen jo tarjousvaiheessa, kun taas ketterät menetelmät usein painottavat ratkaisujen (ja sovellustarpeiden) tuomista esiin löydöksinä vasta iteratiivisessa ja lyhytsyklisessä suunnitteluvaiheessa (kun projekti on aloitettu).

3. Käytetäänkö joskus useampaa menetelmää samanaikaisesti ja jos niin miten?

Yritys X:

DevOpsia on käytetty samaan aikaan Scrumin kanssa siten, että projektin henkilöstö on jakautunut operations- ja development-tiimeihin ja operations-tiimi on noudattanut DevOpsia ja development-tiimi Scrumia.

XP:tä ja Scrumia on käytetty samanaikaisesti siten, että kehitystiimissä oli pienempi porukka, joka toteutti yhtä erittäin liiketoimintakriittistä kokonaisuutta ja noudatti tekemisessään XP:tä, kun taas muu projektitiimi noudatti Scrumia.

Yritys Y:

Toiminnanohjausliiketoiminnassa (ja ERP-ratkaisujen käyttöönotoissa) projektikokonaisuudet saattavat olla valtavia, yhden tai useamman vuoden hankkeita. Nämä hankkeet usein toteutetaan vesiputousmallisella projektinhallintamenetelmällä, jonka sisällä saattaa olla useampia alaprojekteja. Esimerkiksi jos käyttöönotettavasta ERP-ratkaisusta ei löydy johonkin asiakkaan erityistarpeeseen soveltuvaa toiminnallisuutta, saatetaan se ratkaista omana alaprojektinaan hankkeen aikana kehittämällä tarpeeseen oma räätälöity sovellus. Näissä tapauksissa esimerkiksi hanketta johdetaan eri menetelmällä kuin hankkeen sisällä toteuttavia alaprojekteja, joissa saattaa usein olla ketterät menetelmät käytössä.

Toisaalta myös näissä alaprojekteissa (sovelluskehityksessä) käytetään hyvin yleisesti useampia menetelmiä samaan aikaan. Esimerkiksi DevOpsin ja Scrumin yhteydessä käytetään usein tukena tehtävien kuvaamista kanban-työpöydillä.

4. Onko menetelmiä sovellettu tai kustomoitu jollain tavalla projekteja varten ja jos on niin miten?

Yritys X:

Etenkin Scrumia on kustomoitu hyvin voimakkaasti siten, että joitain Scrumin keskeisiä osia (planning poker-estimoinnit, retrospectivet) on jätetty pois esim. aikataulusyihin vedoten. Lisäksi jos on ollut useita projektitiimejä, jotka ovat tehneet tiivistä yhteistyötä, niin näiden välillä on pidetty ns. scrum-of-scrums-kokouksia, joissa projektitiimit ovat jakaneet tietoa keskenään.

Yritys Y:

Ks. kohta 3. On tyypillisempää, että menetelmiä muokataan jonkin verran yrityksen tai asiakkaan tarpeisiin kuin, että noudetaan täysimääräisesti kyseisen ketterän menetelmän toimintamalleja. Projektit, niiden resurssit ja aikataulut vaihtelevat paljonkin, joten usein käytetään pohjana yhtä tai useampaa menetelmää ja mukautetaan ne projektin tarpeisiin.

5. Jos useampi menetelmä on ollut samanaikaisesti käytössä, niin miksi se on havaittu hyväksi?

Yritys X:

Käytettävä metodologia on riippunut hyvin paljon siitä, että missä elinkaaren vaiheessa toteutettava / kehitettävä kokonaisuus on ollut. Jo tuotannossa olevien järjestelmien osalta Scrum ei ole ollut sovelias, koska tuotannossa ilmeneviä mahdollisia ongelma- tai häiriötilanteita on vaikea estimoida etukäteen, joten DevOps on soveltunut paremmin projektinhallintametodologiaksi tällaisiin tilanteisiin. XP on sopinut pienille (3-4 henkilöä) ja tiiviille kehitystiimeille, kun niiden pitää saada nopeasti ja laadukkaasti aikaan asioita. XP on vähentänyt hallinnollista overheadia, jolloin kehitystiimi on voinut keskittyä itse tekemiseen. Scrum taas on parhaimmillaan, kun halutaan parantaa viestintää ja läpinäkyvyyttä kehitystiimin sisällä ja sopii tiimeille, jotka tekevät pitkän aikaa töitä yhdessä samalla porukalla ja saman palvelukokonaisuuden kehittämisen parissa. Tällöin esim. estimaattien laatu paranee ajan myötä ja tiimi toimii jatkuvan parantamisen mallin mukaisesti.

Yritys Y:

Usein, jos käytetään useampaa (ketterää) menetelmää samanaikaisesti, tarkoituksena on tuoda projektiin työkaluja mitkä saattavat puuttua jostain toisesta menetelmästä. Osa malleista keskittyvät puhtaasti tehtävien ratkaisemiseen mahdollisimman tehokkaasti, kun taas toisissa pyritään projektin toimintamalleja tehostamalla mahdollistamaan projektitiimille mahdollisimman sujuva ja keskeytyksen työskentely. Näitä yhdistelemällä pystytään usein sujuvoittamaan projektin hallintaa sekä itse sovelluskehitystöitä samanaikaisesti.

6. Minkälaisia kokemuksia käytetyistä menetelmistä on? (Onko esimerkiksi jotain onnistumisia tai ongelmia havaittu menetelmien käytössä?)

Yritys X:

Scrum on toiminut hyvin vain niissä tilanteissa, kun projektissa on ollut hyvä product owner, hyvä scrum master, ja työkalut jotka tukevat scrum-metodologiaa. Jos yksikin näistä on puuttunut, niin scrum ei ole juurikaan tuonut lisäarvoa, vaan backlogit ovat täyttyneet merkityksettömillä backlog-itemeilla, sprinteissä ei olla saatu toteutettua niitä asioita mitä ollaan alun perin tavoiteltu, ja scrum on kokonaisuudessa typistynyt vain joka-aamuiseksi 15-30min pakolliseksi status-checkiksi.

Yritys Y:

Jokaisessa ketterässä menetelmässä (kuten myös projektinhallinnassa yleisestikin) on tärkeää, että kaikki kehitystöihin osallistuvat perehdytetään valittuun malliin ennen kehitysprojektin aloittamista. Osittain juuri siitä syystä, että vaikka hallitsisikin menetelmän itsessään, usein sitä on mukautettu projektia varten joltain osin. Ketterät menetelmät toimivatkin juuri niin hyvin kuin niitä johdetaan – pääkehittäjien ja projektipäälliköiden toimesta. Koska ketterät menetelmät perustuvat puhtaasti informaation ja kommunikaation läpinäkyvyyteen (kaikki tietävät mitä ja milloin ollaan tekemässä), sekä häiriötekijöiden minimointiin, on projektin onnistumisen kannalta tärkeää, että kaikki projektitiimissä ovat motivoituneita tavoitteiden saavuttamiseksi ja noudattavat sovittuja menetelmiä ja työskentelytapoja kehitysprojektin aikana.

5 Johtopäätökset ja pohdinta

Opinnäytetyön teoriaosuudessa selvitettiin perusteellisesti millaisia ketterät menetelmät Extreme Programming (XP), Scrum, Lean, Kanban, DevOps, SAFe ja PRINCE2 ovat. Näiden lisäksi mukana oli perinteinen ohjelmistokehityksen menetelmä nimeltään vesiputousmalli. Nämä menetelmät valikoituivat mukaan vertailuun, koska ne ovat yleisimmin käytössä olevia menetelmiä.

Opinnäytetyön tutkimusosuudessa vertailtiin ketteriä menetelmiä Extreme Programming (XP), Scrum, Lean, Kanban, DevOps, SAFe ja PRINCE2. Tutkimuksessa käytettiin teoriaosuudesta saatuja tietoja ketteristä menetelmistä sekä yrityshaastatteluista saatuja käyttäjäkokemuksia menetelmien käytöstä. Tutkimuskysymyksillä etsittiin vastauksia siihen, mitä yhteistä ketterillä menetelmillä on verrattuna toisiinsa, miten ketterät menetelmät eroavat toisistaan, voiko menetelmiä kustomoida tai käyttää useampaa menetelmää samanaikaisesti ja mitä hyötyjä ja rajoituksia eri menetelmillä on.

Ensimmäiseen tutkimuskysymykseen, jossa selviteltiin menetelmien yhtäläisyyksiä, löytyi paljon vastauksia. XP:n, Scrumin ja Leanin arvoissa ja periaatteissa on paljon samaa. Kanban perustuu Lean-ajattelulle ja se yhdistää näitä menetelmiä. DevOpsissa on yhtäläisyyksiä XP:n kanssa, koska molemmissa testaus, automatisointi ja integrointi ovat tärkeitä. DevOpsissa käytetään myös Lean-ajattelua ja korostetaan arvon tuottamisen tärkeyttä kuten SAFe:ssa ja PRINCE2:ssa. DevOpsissa viestinnän merkitys on tärkeää kuten monissa muissakin ketterissä menetelmissä. SAFe:ssa menetelmien yhtäläisyydet on viety avoimesti kaikkein pisimmälle. SAFe:ssa onkin yhdistetty XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin parhaita ominaisuuksia samaan viitekehykseen.

PRINCE2:ssa on yhtäläisyyksiä SAFe:n kanssa liiketoiminnallisen näkökulman kautta. PRINCE2:n periaate jatkuva liiketoiminnallinen perustelu ja teema Business Case ovat hyvin samankaltaisia kuin SAFe:n periaatteet ota taloudellinen näkökulma ja rakenna organisaatio arvon ympärille. PRINCE2:n teema laatu on myös samankaltainen XP:n periaatteen laadukas työ ja SAFe:n arvon sisäänrakennettu laatu kanssa. PRINCE2:n yhtäläisyyksiä löytyy myös Scrumin, Leanin, Kanbanin ja DevOpsin kanssa.

Toisessa tutkimuskysymyksessä etsittiin vastauksia menetelmien eroavaisuuksista toisiinsa verrattuna. XP:ssa keskitytään erityisesti ohjelmoinnin tapojen kehittämiseen ja suotuisan ympäristön luomiseen sen ympärille. Sen tunnistettavin piirre on pariohjelmointi. Scrumissa keskitytään projektinhallintaan, tehtävän työn laajuuteen, milloin työ toimitetaan ja vastaako työ käyttäjien ja sidosryhmien tarpeita. Scrumin tunnistettavia piirteitä ovat päivittäispalaverit, sprintit ja roolit tuoteomistaja ja Scrum Master.

Lean eroaa näistä, eikä se sisällä käytäntöjä. Lean on ajattelutapa, joka sisältää arvoja ja periaatteita. Leanin periaatteita kutsutaan ajattelutyökaluiksi. Kanbania pidetään monien mielestä parhaana tapana tuoda Lean-ajattelu organisaatioon. Kanbanin avulla poistetaan epätasaisuuden, ylikuormituksen ja turhuuden hukkaa Lean-ajattelun pohjalta. Kanban auttaa tiimiä parantamaan tapaa, jolla se rakentaa ohjelmistoja. Kanbanissa keskitytään prosessien parantamiseen.

DevOpsin erottaa muista se, kuinka se yhdistää kehityksen, testauksen ja ylläpidon toimimaan saumattomasti yhdessä. DevOpsissa dokumentointi on erityisen tärkeää. DevOpsissa testauksen ja prosessien automaatiolla on suuri merkitys. Pilvipalveluiden hyödyntäminen testiympäristönä on myös ominaispiirre DevOpsissa.

SAFe on lainannut DevOpsilta eri osastojen yhdistämisen, mutta vienyt sen hieman vielä pidemmälle ottamalla organisaation kaikki osa-alueet mukaan projektien läpivientiin. SAFe:n erottuu silläkin muista, kuinka se on yhdistänyt XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin parhaimmat ominaisuudet viitekehykseensä.

PRINCE2 eroaa muista menetelmistä sen erityisellä keskittymisellä liiketoiminnallisiin tavoitteisiin. PRINCE2 edellyttää, että kaikkia sen periaatteita on noudatettava, jotta menetelmä toimii oikein. Prosessien kautta menetelmään saadaan ketteryyttä, koska PRINCE2 ei määrittele, mitä menetelmiä kehityksessä käytetään tai miten tiimien pitäisi muodostua. Menetelmistä löytyi myös eroavaisuuksia niiden arvoista, periaatteista, käytännöistä, empiirisistä peruspilareista, agendoista, CALMS-viitekehyksestä, CALMR-viitekehyksestä, ydinkompetensseista, keskeisistä elementeistä, teemoista ja prosesseista. Näiden lisäksi eroavaisuuksia löytyi menetelmien rooleista, tiimien koosta, tiimien määrästä ja henkilömääristä yhteensä.

Projektin laajuudella on suuri merkitys valittaessa sopivaa ketterää menetelmää projektiin. XP sopii hyvin pieniin projekteihin. Scrumilla on mahdollista tehdä isompia projekteja, koska tiimejä voi olla jopa neljä. Leania tai Kanbania voi yhdistää muihin menetelmiin, jolloin isompien projektien hallinta paranee. DevOps, SAFe ja PRINCE2 soveltuvat paremmin isommissa projekteissa.

Kolmannessa tutkimuskysymyksessä etsittiin vastauksia menetelmien kustomoinnista ja useamman menetelmän käytöstä yhtäaikaisesti. SAFe nousi tässä esiin päällimmäisenä, koska siinä on avoimesti yhdistetty XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin parhaimmat puolet toimimaan yhdessä samassa viitekehyksessä. Lean-ajattelua, Kanbania ja DevOpsia voidaan kaikkia yhdistää muiden menetelmien kanssa. Kustomoitu menetelmä nimeltään Scrum-ban yhdistää Scrumin ja Kanbanin parhaat ominaisuudet. Kanban tuo mukanaan automaattisesti Lean-ajattelua, joten siinä nämä kaksi aina yhdistyvät. PRINCE2:ssa ei määritellä menetelmiä, joita prosessissa käytetään, joten projektitiimin on mahdollista käyttää XP:n, Scrumin, Leanin, Kanbanin ja DevOpsin ominaisuuksia projektin läpiviemiseksi.

Neljännessä tutkimuskysymyksessä etsittiin vastauksia menetelmien hyödyistä ja rajoituksista. Hyötyjä ja rajoituksia löytyi jokaisesta menetelmästä. XP:n hyöty on mm. jatkuva testaus ja refaktorointi. Rajoitus on mm. dokumentaation vähäisyys. Scrumin hyöty on mm. projektien helppo hallittavuus sprinttien ansiosta. Rajoitus on mm. projektien hallitsematon kasvu, koska lopetuspäivää ei ole määritelty. Leanin hyöty on mm. projektin kustannusten vähentäminen. Rajoitus on mm. Leanin vaatima kokemus, jotta autonomiset tiimit voivat toimia tarkoituksenmukaisesti. Kanbanin hyöty on mm. helppokäyttöisyys. Rajoitus on mm. projektin harhautumisvaara, koska Kanbanissa ei aseteta tiukkoja vastuita. DevOpsin hyöty on mm. asiakaskokemuksien hyvä hallittavuus. Rajoitus on mm. tekniset haasteet käyttöönotossa vanhojen järjestelmien kanssa. SAFe:n hyöty on mm. monipuolisten tiimien toimiminen yhdessä. SAFe:n rajoitus on mm. oman teknisen terminologian opiskeluun kuluva aika. PRINCE2 hyöty on mm. hyvä ennakoitavuus. Rajoitus on mm. dokumentoinnin erittäin suuri määrä.

Haastattelujen vastauksista saatiin hyviä käytännönkokemuksia, jotka tukevat ja vahvistavat tutkimusosuuden antia. Eri menetelmien valikoitumiseen projekteissa vaikuttaa mm. työntekijöiden kokemukset, organisaation käytännöt, henkilökunnan koulutukset ja projektin

koko. Scrumia käytetään yleisimmin itse sovelluskehityksessä ja Kanbania käytetään yleisimmin työkulun seurannassa ja järjestelmien ylläpitopalveluissa. DevOpsia on käytetty samanaikaisesti Scrumin kanssa siten, että operations-tiimi on noudattanut DevOpsia ja development-tiimi Scrumia. XP ja Scrum ovat myös olleet samanaikaisesti käytössä niin, että pienempi porukka toteutti XP:tä noudattaen liiketoimintakriittisen kokonaisuuden ja muu projektitiimi noudatti Scrumia. DevOpsin ja Scrumin yhteydessä käytetään usein tukena tehtävien kuvaamista Kanbanin avulla.

Menetelmiä on myös kustomoitu projektiin, resursseihin, aikatauluun ja asiakkaiden tarpeisiin sopiviksi. Esimerkiksi Scrumista on joskus jätetty pois retrospektiivi aikataulusyihin vedoten. DevOps on ollut hyvä vaihtoehto tilanteissa, joissa tuotannossa ilmeneviä ongelma- tai häiriötilanteita on vaikea estimoida etukäteen. XP on sopinut parhaiten pieniin projekteihin. Scrum on sopinut hyvin silloin, kun halutaan parantaa viestintää ja läpinäkyvyyttä kehitystiimin sisällä ja kun tehdään pitkään töitä yhdessä samalla porukalla. Usean ketterän menetelmän käyttöä yhtäaikaisesti hyödynnetään, koska halutaan tuoda projektiin mukaan työkaluja, jotka mahdollisesti puuttuvat jostain menetelmästä. Scrum vaatii toimiakseen hyvän Scrum Masterin, tuoteomistajan ja työkalut, jotka tukevat scrum-metologiaa, muuten se ei tuota juurikaan lisäarvoa projektissa.

6 Yhteenveto

Opinnäytetyön tavoite on vertailla ohjelmistokehityksen erilaisia ketteriä menetelmiä. Vertailussa pyrittiin selvittämään mitä yhteistä ja mitä eroavaisuuksia menetelmissä on. Sen lisäksi selvitettiin voiko menetelmiä käyttää yhtäaikaisesti tai voiko menetelmiä kustomoida. Näiden lisäksi selvitettiin vielä mitä hyötyjä ja rajoituksia eri menetelmissä on. Yrityshaastatteluista pyrittiin löytämään käytännönkokemuksia menetelmien käytöstä, jotka voisivat auttaa muita löytämään oikeat menetelmät omiin projekteihinsa.

Tutkimusosuuden tiedot saatiin teoriaosuudesta sekä yrityshaastatteluiden käyttäjäkokemuksista. Yrityshaastattelut toteutettiin laadullisella strukturoidulla haastattelulla, jossa kysymykset olivat ennakkoon muotoiltuja.

Tutkimusosuuden vertailulla ja yrityshaastatteluilla saatiin hyvä yleiskuva menetelmien eroavaisuuksista, yhtäläisyyksistä ja menetelmien kustomoinnista sekä menetelmien käytöstä yhtäaikaisesti. Lisäksi selvitettiin menetelmien hyödyt ja rajoitukset. Yrityshaastatteluiden avulla saatiin hyviä käytännönkokemuksia, joiden avulla voidaan pohtia, mikä menetelmä voisi sopia omaan projektiin.

Opinnäytetyön tavoitteet onnistuivat hyvin ja uskon, että näistä tiedoista on hyötyä ketteristä menetelmistä kiinnostuneille opiskelijoille sekä työelämässä niille, jotka pohtivat millä menetelmällä kannattaisi ruveta tekemään ohjelmistokehityksen projekteja erilaisissa tilanteissa.

Lähteet

Agilemanifesto (2001). *Ketterän ohjelmistokehityksen julistus.*
https://agilemanifesto.org/iso/fi/manifesto.html

Altexsoft (2021). *Extreme Programming: Values, Principles, and Practices.*
https://www.altexsoft.com/blog/business/extreme-programming-values-principles-and-practices/

Atlassian, (n.d.). *What is DevOps?*
https://www.atlassian.com/devops

Gross, John M., McInnis, Kenneth R. (2003). *Kanban Made Simple.*
https://hamk.finna.fi/Record/nelli19.111086906305782

Juvonen Rami (2018). *Ohjelmistoprojektin sudenkuopat ja miten ne vältetään.*
https://hamk.finna.fi/Record/vanaicat.132081

Kanbanize, (n.d.). *What is Kanban?*
https://kanbanize.com/kanban-resources/getting-started/what-is-kanban

Luukkainen, M. & Ilves, K. (2021). *Ohjelmistotuotanto 2021, Osa 1 Ohjelmistotuotanto ja sen osa-alueet.* Helsingin yliopisto. https://ohjelmistotuotanto-hy.github.io/osa1/

Meiling Julia (2018). *Kanban in Organizations Part I.*
https://www.projectwizards.net/en/blog/2018/11/kanban-organizations

PagerDuty, (n.d.). *6 Essential DevOps Roles You Need on Your Team.*
https://www.pagerduty.com/resources/learn/essential-devops-roles/

PRINCE2 (2016). *The 7 Principles, Themes and Processes of PRINCE2.*
https://www.prince2.com/eur/blog/the-7-principles-themes-and-processes-of-prince2

PRINCE2 Primer, (n.d.). *PRINCE2 Management Products.*
https://www.prince2primer.com/prince2-management-products/

Reuter Niklas (2022). *PRINCE2 projektimalli luo periaatteilla jämäkkyyttä ja prosesseilla joustoa.* https://www.tieturi.fi/webinaari/webinaari-prince2-projektimalli-luo-periaatteilla-jamakkyytta-ja-prosesseilla-joustoa/

Rubin Kenneth S. (2012) *Essential Scrum, A Practical Guide to the Most Popular Agile Process.* https://www.oreilly.com/library/view/essential-scrum-a/9780321700407/cover.html

Rusanen Antti (2022) *Mikä on SAFe (Scaled Agile Framework)?* https://www.tieturi.fi/webinaari/mika-on-safe-scaled-agile-framework/

Scaled Agile Framework (2021) *Essential SAFe.* https://www.scaledagileframework.com/essential-safe/

Scrumorg (2020) *Scrum Framework.* https://scrumorg-website-prod.s3.amazonaws.com/drupal/2021-01/Scrumorg-Scrum-Framework-tabloid.pdf

Stellman, Andrew, Greene, Jennifer (2014). *Learning agile: understanding Scrum, XP, Lean, and Kanban.* https://hamk.finna.fi/Record/vanaicat.126301

Schwaber Ken, Sutherland Jeff (2020). *The 2020 Scrum Guide.* https://scrumguides.org/scrum-guide.html#scrum-theory

ToolsQA (2019). *Lean Software Development: Comprehensive Guide*. https://www.toolsqa.com/agile/lean-software-development/

Turley Frank, (n.d.). *PRINCE2 wiki.* https://prince2.wiki/

Wrike, (n.d.). *A Guide to the Scaled Agile Framework (SAFe).* https://www.wrike.com/agile-guide/what-is-safe/

Liite 1: Aineistonhallintasuunnitelma

Opinnäytetyötä varten tehdään oma kansio verkkoselaimen kirjanmerkkeihin, josta löytyy jokaiselle eri ketterälle menetelmälle oma kansio. Näin lähteet on helppo löytää omien aihepiirien kansioista opinnäytetyötä tehdessä ja lopulta viimeistelyvaiheessa. Opinnäytetyö tallennetaan omalle koneelle ja sen lisäksi työ tallentuu automaattisesti myös OneDrive-pilvipalveluun. Varmuuskopioita otetaan sähköpostiin ja omalle koneelle. Tallennan itselleni myös lukukappaleen, jonka avulla voi helpottaa viimeistelyvaihetta, kun esimerkiksi lähdeluetteloa tarkastetaan kuntoon.

Opinnäytetyöhön liittyvät muistiinpanot ja itse tehdyt kuvat ovat myös OneDrive-pilvipalvelussa oman tietokoneeni lisäksi. Yrityshaastattelujen vastaukset tallennetaan omalle koneelle ja nekin menevät automaattisesti pilvipalveluun ja ovat myös sähköpostissani. Opinnäytetyötä ja siihen liittyvää materiaalia säilytetään vähintään vuosi opinnäytetyön sisään jättämisestä. Yrityshaastattelut käsitellään anonyymisti eikä opinnäytetyöhön liity muitakaan henkilötietoja tai arkaluonteista tietoa. Yrityshaastattelut toteutetaan laadullisena strukturoituna kyselynä, jossa valmiiksi laadittu kysymyspohja lähetetään yrityksille vastattavaksi.

Liite 2: Yrityshaastatteluiden kysymyspohja

1. Mistä seuraavista menetelmistä XP, Scrum, Lean, Kanban, DevOps, SAFe ja PRINCE2 teillä on kokemusta?

2. Miten menetelmät ovat valikoituneet tiettyihin projekteihin?

3. Käytetäänkö joskus useampaa menetelmää samanaikaisesti ja jos niin miten?

4. Onko menetelmiä sovellettu tai kustomoitu jollain tavalla projekteja varten ja jos on niin miten?

5. Jos useampi menetelmä on ollut samanaikaisesti käytössä, niin miksi se on havaittu hyväksi?

6. Minkälaisia kokemuksia käytetyistä menetelmistä on? (Onko esimerkiksi jotain onnistumisia tai ongelmia havaittu menetelmien käytössä?)