Sami Peltola

JULKAISUNSUUNNITTELUPROSESSIT KETTERÄSSÄ KEHITTÄMISESSÄ

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2012

TIIVISTELMÄ

Peltola, Sami
Julkaisunsuunnitteluprosessit ketterässä kehittämisessä
Jyväskylä: Jyväskylän yliopisto, 2012, 41 s.
Tietojärjestelmätiede, kandidaatin tutkielma
Ohjaaja: Leppänen, Mauri

Ketterässä kehittämisessä ohjelmistoja kehitetään lyhyissä iteraatioissa. Tällä pyritään mm. siihen, että käyttäjät saavat toimivan ohjelmiston käyttöönsä mahdollisemman nopeasti ja muuttuvat vaatimukset pystytään joustavasti huomioimaan. Ohjelmistoja ei voida kuitenkaan aina toimittaa käyttäjille jokaisen iteraation jälkeen vaan käyttäjille jaettava ohjelmistokokonaisuus, julkaisu, koostuu yhden tai useamman iteraation tuloksesta. Asiakkaiden tarpeiden ja toimittajan resurssien käytön yhteensovittamiseksi ohjelmistokehityksessä tarvitaan huolellista julkaisunsuunnittelua.

Tässä tutkielmassa tarkastellaan julkaisunsuunnitteluprosesseja ketterässä kehittämisessä. Tutkimuksen tavoitteena on selvittää kirjallisuuteen perustuen, millaisia prosessimalleja ketterän julkaisunsuunnittelun tueksi on olemassa, millaisia ominaispiirteitä näillä prosessimalleilla on ja miten mallit soveltuvat käytettäväksi Scrumin yhteydessä. Toisena tutkimuksen kohteena on selvittää, miten julkaisunsuunnittelu liittyy projekti- ja tiimitasoiseen ohjelmiston kehittämiseen.

Tutkimuksessa tarkastellaan neljää ketterän kehittämiseen tarkoitettua julkaisunsuunnitteluprosessia, joista kolme on sovellettavissa Scrumiin. Yleisimpinä piirteinä tarkastelluissa prosessimalleissa nousevat esiin vaatimusten priorisoinnin, julkaisun määrittelyn ja erilaisten arviointien aktiviteetit. Malleista puuttuvat jakeluun valmistaminen ja julkaistavan version vahvistaminen -aktiviteetit. Toiseksi tutkimuksessa esitetään kaksi ketterää tuotehallintaa jäsentävää mallia, joiden avulla julkaisunsuunnittelun ja projekti- ja tiimimuotoisen kehittämisen välillä havaitaan kaksi yhteyttä. Julkaisunsuunnittelu ohjaa kehittämistä ja kehittäminen tarjoaa palautetta julkaisunsuunnittelulle. Julkaisunsuunnittelussa ja tiimitasoisessa kehittämisessä käsitellyt vaatimukset ovat myös yhteydessä ja jäljitettävissä toisiinsa.

Avainsanat: julkaisu, ketterät menetelmät, tuotehallinta

ABSTRACT

Peltola, Sami
Release Planning Processes in Agile Development
Jyväskylä: University of Jyväskylä, 2012, 41p.
Information Systems, Bachelor's Thesis
Supervisor: Mauri Leppänen

In agile development software is engineered in short iterations. This enables the delivery of working software to users as soon as possible and maintains the ability to react flexibly to changing user requirements. Software cannot always be delivered to users after each iteration. Instead, software is delivered in a release that combines the results of one or many iterations. To reconcile the client's needs and the supplier's resources use, careful release planning is needed.

In this study we focus on agile release planning process models. Our purpose is to find out, based on the literature, what kind of process models have been suggested for agile release planning, what kind of characteristics these models have and whether these models are compatible with Scrum method. Secondly, the study aims to consider how release planning is linked with project and team-level development in these suggestions.

In the study, four release planning process models are selected for a closer consideration. These are described and compared to each other. Three of these are compatible with Scrum. The most common characteristics within the models are requirement prioritization, release definition and different estimation activities. They commonly lack launch preparation and build validation activities. In addition, the study describe two models of agile software product management and finds two links between release planning and project/team-level development. The first link is steering and feedback and the second link concerns requirements in release planning and development levels.

Keywords: release, agile methods, product management

KUVIOT

KUVIO 1 Suunnitteluasteikko (planning spectrum) (Boehm 2002, 65) 9
KUVIO 2 Scrumin prosessi (Abrahamsson ym., 2002, 28) 12
KUVIO 3 Cohnin (2006, 135) prosessimalli..... 17
KUVIO 4 Julkaisun yhteissuunnittelutapahtuma (Heikkilä, 2010, 175)..... 19
KUVIO 5 Loguen ja McDaidin (2008, 439) prosessimalli..... 21
KUVIO 6 Tuotehallinnan kompetenssimalli (mukaillen Bekkers ym., 2010, 4) . 29
KUVIO 7 Big picture -malli (Leffingwell, 2009, 4) 30
KUVIO 8 ATMAN-viitekehys (Vähäniitty, 2010a, 124) 31
KUVIO 9 Kontrollisyklimalli aikajanalla kuvattuna (Rautiainen ym., 2006, 7) . 32

TAULUKOT

TAULUKKO 1 Prosessimallien vertailu..... 24
TAULUKKO 2 Tuotehallintaa jäsentävien mallien vertailu 34

SISÄLLYS

TIIVISTELMÄ 2

ABSTRACT 3

KUVIOT 4

TAULUKOT 4

SISÄLLYS 5

1 JOHDANTO 6

2 KETTERÄ LÄHESTYMISTAPA 8
2.1 Ketterän lähestymistavan tausta 8
2.2 Ketterän kehittämisen pääpiirteitä 10
2.3 Scrum–menetelmä 11
2.3.1 Prosessi 11
2.3.2 Roolit 12
2.3.3 Työlistat 13

3 JULKAISUNSUUNNITTELUPROSESSI 14
3.1 Julkaisu 14
3.2 Julkaisunsuunnittelu 15
3.3 Prosessimalleja 16
3.3.1 Bekkersin ym. prosessimalli 16
3.3.2 Cohnin prosessimalli 16
3.3.3 Julkaisun yhteissuunnittelu -menetelmä 18
3.3.4 Loguen ja McDaidin prosessimalli 20
3.3.5 SCERP-prosessimalli 22
3.4 Prosessimallien vertailu 23
3.5 Yhteenveto 26

4 JULKAISUNSUUNNITTELU OSANA TUOTEHALLINTAA 28
4.1 Tuotehallinnan kompetenssimalli 28
4.2 Leffingwellin malli 29
4.3 ATMAN-malli 30
4.4 Mallien vertailu 33

5 YHTEENVETO 37

LÄHTEET 39

1 Johdanto

Ketterässä kehittämisessä (agile development) ohjelmistoja tuotetaan lyhyissä noin 2–4 viikon mittaisissa sykleissä, joita kutsutaan iteraatioiksi. Tarkoituksena on tuottaa ohjelmistoa mahdollisimman nopeasti siten, että asiakas saa käyttöönsä ensin kaikkein eniten tarvitsemansa ohjelman osat. Lyhyissä iteraatioissa tapahtuva ohjelmistonkehitys mahdollistaa sen, että kehitettävän ohjelmiston suuntaa voidaan muuttaa joustavasti asiakkaiden muuttuvien tai tarkentuvien tarpeiden mukaan (Agile Manifesto, 2001).

Nopealla tahdilla ja joustavasti tapahtuvasta ohjelmistonkehityksestä huolimatta ohjelmiston tulevista toiminnallisuuksista tarvitaan iteraatiota pidemmän aikavälin yleiskuva. Tällainen yleiskuva mahdollistaa ohjelmistoprojektien tehokkaan resursoinnin ja antaa ohjelmiston kehittäjille tavoitteet, joita kohti ohjelmistoa kehitetään. Se toimii myös kehitystyön onnistumisen ja aikataulutuksen mittarina. Myös asiakkaiden ja muiden sidosryhmien edustajien on hyödyllistä tietää, milloin jokin tietty ohjelmiston osa valmistuu. (vrt. Cohn 2006.) Lisäksi syy iteraatiota pidemmän aikavälin suunnitelman tarpeellisuudella on se, että usein yhden 2–4 viikon mittaisen kehitysjakson aikana ei ole aina mahdollista saada valmiiksi asiakasta tyydyttävää ohjelmistokokonaisuutta. Asiakkaan kannalta ei ole myöskään aina käytännöllistä, että uusi ohjelmisto otetaan käyttöön jokaisen iteraation jälkeen. Näistä syistä johtuen asiakkaan käyttöön tuotava ohjelmisto koostuu usein osista, jotka tuotetaan usealla iteraatiolla. Tällaista kokonaisuutta kutsutaan julkaisuksi. (esim. Cohn, 2006; Leffingwell, 2009.)

Julkaisujen toteutuksen lähtökohtana toimii julkaisusuunnitelma ja julkaisusuunnitelman luontiprosessia kutsutaan julkaisunsuunnitteluksi. Ketterän julkaisunsuunnittelun (agile release planning) tueksi on kehitetty erilaisia suunnitteluprosessimalleja (esim. Cohn, 2006; Logue & McDaid, 2008). Näissä on usein hyödynnetty perinteisen julkaisunsuunnittelun käytäntöjä. On kehitetty myös erilaisia tuotehallinnan malleja, jotka jäsentävät julkaisunsuunnittelua osana ketterää kehitystoimintaa ja tuotehallintaa (esim. Leffingwell, 2009; Vähäniitty, 2010a).

Aiemmasta tutkimuksesta huolimatta kaivataan kokonaisesitystä ketterän julkaisunsuunnittelun prosesseista ja siitä, miten julkaisunsuunnitteluprosessit nähdään osana yritysten ohjelmistonkehitystyötä ja tuotehallinnan kenttää ketterässä kehittämisessä. Esimerkiksi Vähäniitty (2010a) pitää tärkeänä, että ymmärretään, miten julkaisunsuunnitteluprosessit ja pidemmän aikavälin suunnitelmat ilmenevät ketterässä kehittämisessä.

Tutkimuksen tarkoituksena on luoda kirjallisuuteen perustuen yleiskuva siitä, millaisia prosessimalleja julkaisunsuunnitteluun ketterän kehitystavan mukaisessa kehittämisessä on olemassa ja miten julkaisunsuunnittelua voidaan tehdä ketterässä lähestymistavassa saumattomasti yhdessä kehitystyön kanssa. Tutkielman aihetta on rajattu siten, että ketterien menetelmien tarkastelu on rajattu vain yleisimmin käytössä olevaan Scrum-menetelmään. Tutkielman tutkimuskysymykset ovat:

- Millaisia prosessimalleja julkaisunsuunnittelun tueksi on kehitetty ja millaisia ominaispiirteitä niillä on? Miten nämä prosessimallit soveltuvat käytettäväksi Scrumin yhteydessä?

- Millä tavoin julkaisunsuunnittelu liittyy projekti- tai tiimitasoiseen ohjelmistonkehittämiseen ja miten julkaisunsuunnittelua voidaan tehdä saumattomasti ketterän kehittämisen kanssa?

Tutkielman ensimmäisessä osassa (Luku 2) kuvataan ketterän kehittämisen taustaa, pääpiirteitä ja erästä ketterää menetelmää, Scrumia. Tätä seuraavassa luvussa (Luku 3) määritellään ensin julkaisu- ja julkaisunsuunnittelukäsitteet. Sen jälkeen tarkastellaan erilaisia ketterän kehittämisen kontekstiin tarkoitettuja julkaisunsuunnitteluprosessimalleja ja verrataan niitä keskenään. Luvussa 4 tarkastellaan julkaisunsuunnittelua osana tuotehallintaa. Tässä yhteydessä esitetään kaksi mallia, jotka jäsentävät tuotehallinnan kenttää ketterässä kehittämisessä ja julkaisunsuunnittelun ja ketterän kehittämisen välistä yhteyttä. Tutkielman viimeisessä osassa (Luku 5) vedetään lyhyesti yhteen tutkielman aiempia osia, kootaan vastaukset tutkielman tutkimuskysymyksiin ja esitetään tutkimuksen pohjalta jatkotutkimusaiheita.

2 Ketterä lähestymistapa

Tässä luvussa esitellään ensin lyhyesti ketterän kehittämisen taustaa ja sen keskeisiä piirteitä. Tämän jälkeen kuvataan yksityiskohtaisemmin käytetyintä ketteristä menetelmistä, eli Scrumia.

2.1 Ketterän lähestymistavan tausta

Ketterien menetelmien synty voidaan nähdä reaktiona perinteisten ohjelmistonkehitystapojen jäykkyyteen sekä voimakkaassa muutoksessa olevaan liiketoimiympäristöön (Abbas, Gravell & Wills, 2008). Perinteisille vesiputousmallin omaisille kehitysmenetelmille on ominaista, että vaatimukset lyödään lukkoon kehityksen alussa ennen suunnittelu- ja toteutusvaiheita (McCauley, 2001). Niissä oletetaan, että riittävällä panostuksella ohjelmiston vaatimukset on mahdollista tunnistaa etukäteen ja näin pystytään ennaltaehkäisemään vaatimuksien muuttuminen ohjelmiston kehityksen elinkaaren aikana (Highsmith & Cockburn, 2001). Boehmin (2002) mukaan perinteiset menetelmät ovatkin omimmillaan ympäristössä, jossa ohjelmiston vaatimukset ovat ennakkoon selvitettävissä ja suhteellisen muuttumattomat. Muuttuvassa ympäristössä tai tilanteessa, jossa vaatimukset eivät ole aluksi selvitettävissä, tämän mukainen toiminta on altis viivästyksille ja sille, että aiemmin tehtyihin kehitysvaiheisiin joudutaan muutoksien tapahtuessa uudelleen palaamaan

Ohjelmistojen vaatimukset muuttuivat yhä vaikeammin ennakoitavaksi, erityisesti internet-sovellusten aikakautena. Tuolloin myllerryksessä oleva liiketoimintaympäristö aiheutti sen, että vaatimukset muuttuivat yhä nopeammin. Syntyi tarve aiempaa joustavammille kehitysmenetelmille (Baskerville, Ramesh, Levine, Pries-Heje & Slaughter, 2003). 1990–luvulla useat tahot pyrkivät vastaamaan tähän tarpeeseen. Syntyi joukko menetelmiä, joita kutsuttiin aluksi kevyiksi menetelmiksi (light-weight methods). Vuonna 2001 joukko kevyiden menetelmien kehittäjiä ja asiantuntijoita kokoontui keskustelemaan yhteisien näkemyksien löytämiseksi. Kokousta voidaan pitää onnistuneena, sillä osallis-

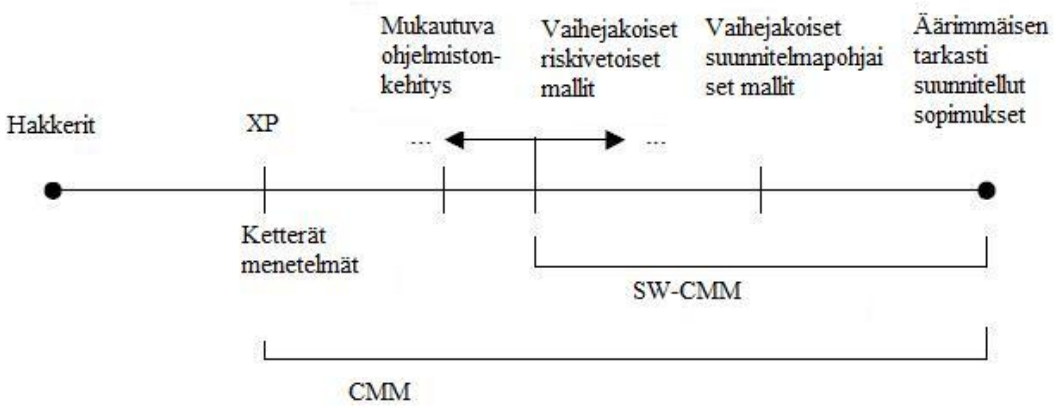
tujat laativat ja allekirjoittivat yhteisen ketterän julistuksen (agile-manifest), jossa määriteltiin ketterän kehityksen neljä arvoa ja 12 periaatetta. Julistuksessa kevyet menetelmät saivat nimen ketterät menetelmät. (Agile Manifesto, 2001).

Ketterän julistuksen sisältämät neljä arvoa ovat:

- *Yksilöt ja heidän välinen vuorovaikutus on tärkeämpää kuin* prosessit ja työkalut.
- *Toimiva ohjelmisto on tärkeämpi kuin* kattava dokumentaatio.
- *Asiakasyhteistyö on tärkeämpää kuin* sopimusneuvottelu.
- *Muutokseen vastaaminen on tärkeämpää kuin* pitäytyminen suunnitelmassa.

Arvot on esitetty siten, että lauseiden vasemmalla olevat asiat ovat tärkeämmällä sijalla kuin oikealla olevat. Tämä ei tarkoita sitä, että oikealla olevat olisivat arvottomia. (Agile Manifesto, 2001.)

Ohjelmistonkehityksen historiassa kehittämisen lähestymistavat ovat vaihdelleet voimakkaammin tai kevyemmin kehittämisprosessia kontrolloivien menetelmien välillä (Shalloway, Beaver & Trott, 2010). Boehm (2002) vertaili erilaisia menetelmiä keskenään (ks. Kuvio 1) sen mukaan, miten suunnittelupohjaisia ja kontrolloituja menetelmät ovat. Boehm sijoittaa ketterät menetelmät lähelle ns. ”hakkerimenetelmiä”, joissa ei ole minkäänlaista suunnitelmallisuutta tai prosessia. Perinteiset menetelmät on sijoitettu lähemmäs äärimmäisen tarkasti suunniteltuja sopimuksia. Boehm pitää sekä ketteriä että perinteisiä menetelmiä suunnitelmallisuuden ja kontrollin suhteen lähempänä vertailuasteikon keskikohtaa kuin radikaaleja ääripäitä. Hän toteaa ketterien menetelmien sisältävän suunnitelmallisuutta, joka painottuu kuitenkin enemmän itse suunnitteluprosessiin kuin suunnitelmien dokumentointiin.



KUVIO 1 Suunnitteluasteikko (planning spectrum) (Boehm 2002,65)

Ketterä julistus voidaan nähdä yrityksenä luoda tasapaino keveän prosessin ja riittävän kontrollin välillä ohjelmiston kehityksessä (Agile Manifesto, 2001). Tätä tasapainon hakemista ilmentävät osaltaan ketterän julistuksen arvot.

2.2 Ketterän kehittämisen pääpiirteitä

Abbas ym. (2008) määrittelevät historiallisia taustoja käsittelevässä julkaisussaan ketterän kehittämisen olevan

1. mukautuvaa
2. iteratiivista ja inkrementaalista
3. henkilökeskeistä.

Mukautuvuudella tarkoitetaan ketterien menetelmien tapaa suhtautua myönteisesti muutoksiin niin vaatimuksien, käytetyn teknologian kuin käytettävän kehitysmenetelmän itsensäkin suhteen. Menetelmissä huomioidaan myös aiemmista työvaiheista saatu palaute. *Iteratiivisuudella* tarkoitetaan sitä, että ohjelmistoja tuotetaan osa kerrallaan iteraatioissa. Jokaisessa iteraatiossa tehdään tietty ohjelmiston osa alusta loppuun saakka valmiiksi, aina suunnittelusta testaukseen, ja/tai parannetaan aiemmin tehtyä ohjelmiston osaa. Ohjelmiston toiminnallisuudet kasvavat *inkrementaalisesti*, kun uudet kehitetyt toiminnallisuudet integroidaan toimivaan ohjelmistoon. Yhden tai useamman iteraatiovaiheen jälkeen ohjelmisto julkaistaan asiakkaalle, jolloin saadaan taas uuttaa palautetta tuotteen käyttäjiltä. *Henkilökeskeisyys* liittyy siihen, miten ketterässä kehittämisessä yksilöt nähdään tärkeimpinä ohjelmistoprojektien onnistumiseen vaikuttavana tekijöinä. Prosessien tehtävä on tukea yksilöitä heidän työssään. Ketterissä menetelmissä korostuu myös kasvokkain kommunikointi niin kehitystiimin sisällä kuin ohjelmiston kehittäjien ja asiakkaankin välillä. Asiakkaan ja kehitystiimien välinen yhteistyö nähdään tärkeäksi.

Yhteenvetona edellä esitettyihin ominaisuuksiin Abbas ym. (2008, 96) toteavat: ”Ohjelmiston kehitys on vaikeasti ennakoitavaa toimintaa. Siksi tarvitaan mukautuva prosessi epävarmuuden kontrolloimiseksi. Iteratiivinen ja inkrementaalinen kehittäminen on tämän prosessin paras kontrolloija. Lisäksi prosessissa pitää olla mukana luovia ja kyvykkäitä yksilöitä.”

Ohjelmistoja on kehitetty aiemminkin inkrementaalisesti ja iteratiivisesti (Larman & Basili, 2003), mutta ketterät menetelmät ovat tuoneet ohjelmiston kehitykseen uutena piirteenä näkökulman, jossa yksilöt tunnustetaan tärkeimmiksi tekijöiksi ohjelmistoprojektien onnistumisessa, yhdessä tehokkuuden ja liikkuvuuden kanssa (Highsmith ym., 2001).

Toisin kuin perinteisissä menetelmissä, ketterässä lähestymistavassa muutoksia ei pyritä estämään vaan hyväksytään se, että kaikkia muutoksia ei ole mahdollista etukäteen ennakoida. Muutoksien välttämisen sijaan ketterissä menetelmissä pyritäänkin minimoimaan tulevista muutoksista johtuvat kustannukset (Highsmith ym., 2001).

Ketteriä menetelmiä on useita, esimerkiksi Scrum, Extreme programming (XP) ja Kanban. Menetelmistä tällä hetkellä yleisin käytössä oleva on Scrum, johon tutustutaan seuraavaksi. (VersionOne, 2011.)

2.3 Scrum–menetelmä

Ensimmäisenä Scrum-termin käytön ja Scrumin työmuodon idean esittivät Takeuchin ja Nonaka (1986) artikkelissaan *The New Product Development Game*. Artikkelissa esitetään tuotekehityksen malli, jossa itseorganisoituvat moniosaajatiimit kehittävät tuotteen alusta loppuun sen sijaan, että tuotetta kehitetään perinteisesti työvaiheittain (Takeuchi ja Nonaka, 1986). Scrum-menetelmä syntyi varsinaisesti vuonna 1994, kun Jeff Sutherland ryhtyi soveltamaan Takeuchin ja Nonakan ajatuksia ohjelmiston kehitykseen. Myöhemmin Ken Schwaber liittyi mukaan kehittämään menetelmää (Schwaberin & Beedle, 2002, 11).

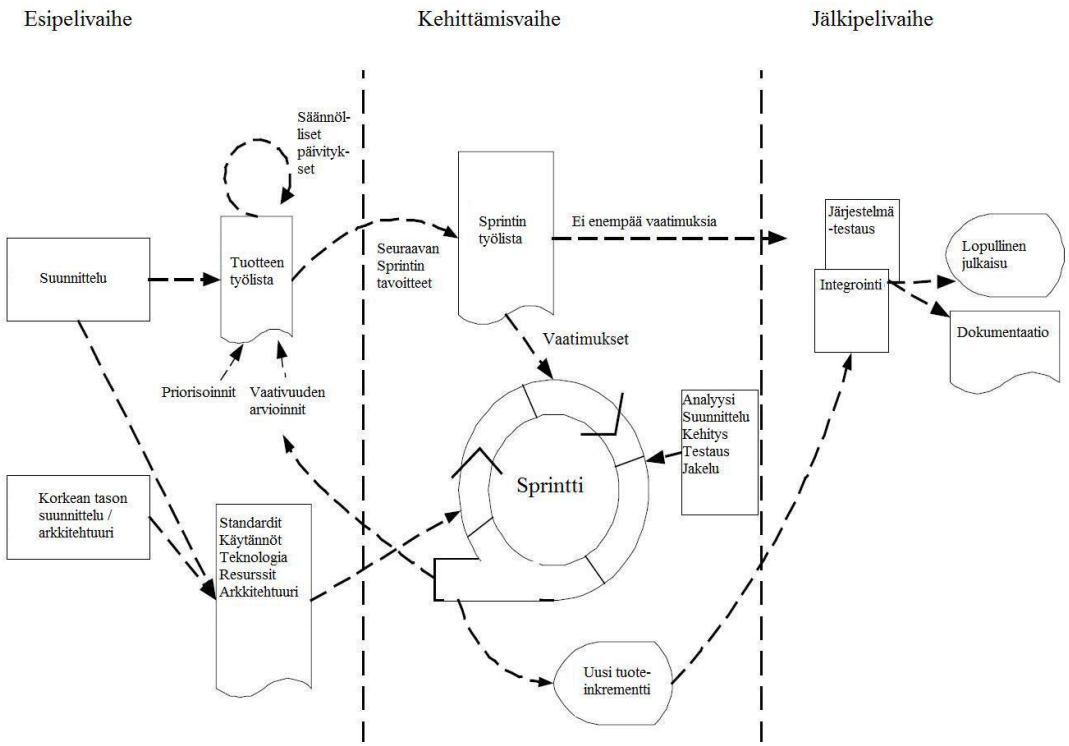
Vlaanderenin, Jansenin, Brinkkemperin ja Jaspersin (2009, 2) mukaan Scrumin keskeisenä ajatuksena on, että monia ohjelmistonkehityksen prosesseja on vaikea kontrolloida ja että Scrum pyrkii vastaamaan tähän ongelmaan joustavalla tavalla. Scrumissa vain prosessin ensimmäinen ja viimeinen vaihe on tarkkaan määritelty. Näiden välissä on peräkkäisiä kehittämisjaksoja joissa ohjelmistoja toteutetaan joustavasti. Kehittämisjaksot ovat lukittuja eli niiden aikana uusia vaatimuksia ei voi tuoda mukaan kehitystyöhön. Näin ohjelmistoa voidaan luoda kontrolloidusti myös muuttuvassa ympäristössä. (Vlaanderen ym. 2009, 2.)

Scrum-menetelmä auttaa ohjelmistonkehityksen projektinhallinnassa (Dybå & Dingsøyr, 2009, 6). Se ei niinkään tarjoa käytännön ohjeita ohjelmiston kehitykseen (Abrahamsson, Salo, Roinkainen & Warsta 2002, 27).

2.3.1 Prosessi

Scrumin prosessi voidaan jakaa kolmeen vaiheeseen, esipeliin, kehittämiseen ja jälkipeliin. Prosessin vaiheet on esitetty kuviossa 2. *Esipelivaiheessa* suunnitellaan tulevan ohjelmiston sisältö ja laaditaan sen arkkitehtuuri sekä järjestelmän korkean tason suunnitelma (Schwaber, 1995). Tämän vaiheen aikana luodaan myös tuotteen työlista.

Kehittämisvaiheessa tapahtuu varsinainen ohjelmiston kehitys lyhyissä 1–4 viikon mittaisissa kehitysjaksoissa, joita kutsutaan *sprinteiksi.* Sprintin aikana järjestetään päivittäisiä Scrum-tapaamisia (daily scrum-meeting), joissa ohjelmistoa kehittävän Scrum-tiimin (ks. luku 2.3.2) jäsenet kertovat kukin vuorollaan, missä vaiheessa he ovat kehitystyössään ja millaisia ongelmia he ovat kohdanneet. (Schwaber, 1995; Abrahamsson ym., 2002, 33.)



KUVIO 2 Scrumin prosessi (Abrahamsson ym., 2002, 28)

Jälkipelivaiheeseen siirrytään, kun todetaan, että erilaiset ajalliset, kustannukselliset, kilpailulliset, laadulliset ja vaatimuksiin liittyvät tekijät ovat kypsiä ohjelmiston julkaisuun. Tässä vaiheessa tehdään mm. järjestelmätestausta, integrointia ja muita ohjelmiston julkaisun valmisteluun liittyviä tehtäviä. (Schwaber, 1995.)

2.3.2 Roolit

Sprintin aikana varsinaisesta kehitystyöstä vastaa *Scrum-tiimi,* joka kehittää sprinttiin valituista toiminnallisuuksista valmiin ohjelmistokokonaisuuden. Tiimillä on vapaus ja vastuu päättää, millä tavoin tuote kehitetään. Suositeltu Scrum-tiimin koko on 5–9 henkilöä. (Schwaberin ja Beedlen, 2002, 36.)

Scrum-tiimin työtä tukee *Scrum-mestari*, joka huolehtii Scrumin periaatteiden ja arvojen toteutumisesta tiimin työssä sekä siitä, että tiimi pysyy tavoitteessaan ja aikataulustaan. Hänen tehtävänään on myös tukea Scrum-tiimin työtä raivaamalla pois tiimin työtä haittaavia esteitä (Schwaber & Beedle, 2002, 31).

Tuotteen omistaja (product owner) on vastuussa ohjelmistoprojektin onnistumisesta ja tuotteen työlistan ylläpitämisestä. Hänellä on lopullinen päätösvalta työlistan sisällöstä ja priorisoinnista. (Schwaber & Beedle, 2002, 34.)

2.3.3 Työlistat

Scrum sisältää erilaisia työlistoja jotka ovat tuotteen, julkaisun ja sprintin työlistat. *Tuotteen työlistassa* (product backlog) on kuvattu kaikki se tekemätön työ, jota ohjelmiston tekemisen on suunniteltu edellyttävän (esim. Abrahamsson ym., 2002, 32). Siihen on kerätty tietoa useasta lähteestä, esimerkiksi myynnistä, käyttäjätuesta ja kehitystiimeiltä. Listan kohdat on järjestetty tärkeysjärjestykseen (Schwaber & Beedle, 2002, 33). Tuotteen työlistan kohdat on kuvattu tyypillisesti toiminnallisuuksina (features) tai vaihtoehtoisesti käyttäjätarinoina (user stories) (Highsmith ym., 2001; Cohn, 2006). *Käyttäjätarina* on Cohnin (2006) mukaan ”lyhyt kuvaus toiminnallisuudesta järjestelmän käyttäjän tai asiakkaan kuvaamana”. *Toiminnallisuus* taas on muutaman lauseen pituinen kuvaus tuoteominaisuudesta, mikä on sidosryhmien edustajien helppo ymmärtää (esim. Leffingwell, 2009, 11).

Julkaisun työlista on osa tuotteen työlistaa, jonka tuotteen omistaja on määritellyt olevan mukana tulevassa julkaisussa (Schwaber & Beedle, 2002, 71).

Sprintin työlista (Sprint backlog) koostuu sprinttiin kehitettäviksi valituista toiminnallisuuksista. Se luodaan *sprintin suunnittelupalaverissa* Scrum-mestarin, tuotteen omistajan ja Scrum-tiimin yhteistyönä. Sprintin työlista jäädytetään sprintin ajaksi, jolloin sitä ei voi muuttaa (Abrahamsson ym., 2002, 33). Sprintin työlistalla olevat kohdat on kuvattu tehtävinä. *Tehtävät* ovat käyttäjätarinoita tarkemman tason kuvauksia, joiden avulla kehittäjät voivat kehittää käyttäjätarinoiden mukaiset toiminnot (esim. Leffingwell, 2009, 9).

3 Julkaisunsuunnitteluprosessi

Tässä luvussa määritellään julkaisukäsite ja tarkastellaan, millä tavoin iteraatiot liittyvät julkaisuihin. Tämän jälkeen kerrotaan, mitä tutkielmassa tarkoitetaan julkaisunsuunnittelulla. Käsitteitä tarkastellaan ketterän kehittämisen kontekstissa. Luvun lopussa tarkastellaan erilaisia ketterän julkaisunsuunnittelun prosesseja ja verrataan niitä keskenään.

3.1 Julkaisu

Ketterässä kehittämisessä ohjelmistoja tuotetaan lyhyissä iteraatioissa. Lähtökohtana on, että iteraatiot tuottavat toimivia ohjelmisto-osia (esim. Leffingwell, 2009, 10). Pyrkimyksenä on näin tuottaa asiakkaalle mahdollisimman nopeasti arvoa ja toisaalta mahdollistaa nopea palautteen saanti ohjelmiston käyttäjiltä (Agile Manifesto, 2001).

Käyttäjien kannalta ei ole aina järkevää tai edes mahdollistakaan ottaa uusi ohjelmisto-osa käyttöön jokaisen iteraation jälkeen. Käyttäjän liiketoiminta voi kokea häiriötä jatkuvasta päivitystahdista ja siitä aiheutuvasta tarpeesta kouluttaa ohjelman käyttäjiä. (Leffingwell, 2009, 10). Ohjelmiston kehittäjän näkökulmasta ei ole myöskään aina tarkoituksenmukaista julkaista ohjelmistoa iteraation päätyttyä. Ohjelmakoodiin on voinut jäädä kehitystyön aikana ns. ”teknistä velkaa”, jolla tarkoitetaan mm. pieniä virheitä tai sitä, että ohjelmistoa on systeemitestattava ennen ohjelman jakelua. Ennen julkaisua ohjelmistoon jäänyt tekninen velka voidaan poistaa seuraavan tai viimeisen iteraation yhteydessä. (Leffingwell, 2009, 10.)

Edellä mainitut ovat esimerkkejä syistä, joiden takia asiakkaalle toimitettava ohjelmistokokonaisuus muodostuu usein usean iteraation tuloksesta. Tällaista kokonaisuutta kutsutaan *julkaisuksi (release)*. Sommervillen (2007, 699) mukaan julkaisu on ohjelmistoversio, joka tulee käyttäjien saataville. Ohjelmistoversio taas on ohjelman ilmentymä, joka eroaa jollain tavoin aiemmista ohjelman ilmentymistä, esimerkiksi parannetun toiminnallisuuden tai korjattujen

virheiden muodossa (Sommerville, 2007, 699). Edellisestä määritelmästä poiketen julkaisulla voidaan tarkoittaa sekä käyttäjille jaettavia ulkoisia ohjelmakokonaisuuksia tai sisäisiä kokonaisuuksia, joita ei jaeta suoraan loppukäyttäjille. Leffingwell (2011) kutsuu jakeluun päätymättömiä julkaisuja potentiaalisesti julkaistavaksi inkrementiksi (potentially shippable increment, PSI).

Termiä julkaisu käytetään joissakin yhteyksissä tarkoittamaan myös sitä kehittämisprosessin osaa, joka tuottaa julkaisun. Tämän mukaisen määritelmän on esittänyt muiden muassa Cohn (2006, 25): ”Julkaisu muodostuu yhdestä tai useammasta iteraatiosta, jotka rakentuvat toistensa päälle toteuttaakseen joukon toisiinsa liittyviä toiminnallisuuksia”.

Tässä tutkielmassa julkaisulla tarkoitetaan Leffingwellin (2011) määrityksen mukaisesti ohjelmakokonaisuutta. Julkaisu koostuu yhden tai useamman iteraation tulosta, ja se voi olla joko sisäinen tai ulkoinen. Julkaisun aikaan saamiseksi tarvittavaa kehittämisjaksoa kutsutaan julkaisuprosessiksi.

3.2 Julkaisunsuunnittelu

Edellä todettiin, että eri syistä johtuen ohjelmistoja ei julkaista asiakkaalle useinkaan iteraatio kerrallaan. Miten sitten päätetään, miten pitkään julkaisua kehitetään, monenko iteraation tuloksesta julkaisu koostetaan ja mitä ominaisuuksia julkaisuun sisällytetään? Näitä kysymyksiä pyritään ratkaisemaan julkaisunsuunnittelulla.

Julkaisunsuunnittelulle on esitetty hieman erilaisia määritelmiä. Shallowayn ym. (2010, 11) mukaan julkaisunsuunnittelu on ”prosessi tuotteen vision siirtämiseksi tuotteen työlistaan”. Logue ja McDaid (2008, 437) taas määrittelevät julkaisunsuunnittelun ”prosessiksi, jossa syntyy korkean tason suunnitelma, josta tunnistetaan milloin julkaisu valmistuu ja mitkä toiminnallisuudet siihen sisältyvät”. Myös Cohn (2006) kuvaa julkaisunsuunnittelua prosessiksi, jossa luodaan korkean tason suunnitelma. Prosessissa syntyvä suunnitelma kattaa yhtä iteraatiota pidemmän ajanjakson ja se tarjoaa kontekstin, jonka puitteissa iteraatiot on mahdollista rakentaa tyydyttäväksi kokonaisuudeksi (Cohn, 2006, 133).

Tässä tutkielmassa julkaisunsuunnittelu nähdään kaikkien edellä mainittujen määritelmiä mukailevalla tavalla. Cohnia (2006) mukaillen julkaisunsuunnittelu nähdään prosessina, jonka lopputuloksena syntyy korkean tason suunnitelma, joka viitoittaa iteraatiotason suunnittelua. Loguen ja McDaidin (2008) mukaisesti julkaisunsuunnittelun tuloksena syntyy julkaisusuunnitelma, josta ilmenee todennäköinen julkaisun valmistumisajankohta ja julkaisun sisältö. Julkaisunsuunnittelun tehtävänä on myös Shallowaytä ym. (2010) mukaillen muuttaa organisaation visiota ja strategisia suunnitelmia konkreettisempaan ja helpommin käsiteltävään muotoon.

3.3 Prosessimalleja

Tässä alaluvussa kuvataan muutamia kirjallisuudessa julkaisunsuunnitteluun esitettyjä prosessimalleja. Nämä prosessimallit ovat Bekkersin, Weerdin, Spruitin ja Brinkkemperin (2010) malli, Cohnin (2006) malli, julkaisun yhteissuunnittelun menetelmä (Heikkilä, 2010), Loguen ja McDaidin (2008) malli, ja SCERP-malli (Heikkilä, Jadallah, Rautiainen & Ruhe, 2010). Alaluvun jälkeen Bekkersin ym. (2010) mallia käytetään osittain muiden mallien vertailun pohjana.

3.3.1 Bekkersin ym. prosessimalli

Bekkersin ym. julkaisunsuunnittelun prosessimalli on osa ohjelmistotuotehallinnan kompetenssimallia (Bekkers ym. 2010). Prosessi on kuvattu hyvin yleisellä tasolla; prosessin vaiheiden kuvaukset sisältävät tiedon vaiheiden tehtävistä, mutta ei kuvausta miten vaiheiden tehtävät toteutetaan. Prosessia ei ole tarkoitettu tarkaksi ohjeeksi julkaisusuunnitelman kehittämiseen. Prosessissa on esitetty seuraavat vaiheet: vaatimusten priorisointi, julkaisun määrittely, julkaisun määrittelyn vahvistaminen, laajuuden muutosten hallinta, julkaistavan version vahvistaminen ja jakeluun valmistautuminen. Seuraavaksi kuvataan prosessin vaiheita Bekkersin ym. (2010) mukaan.

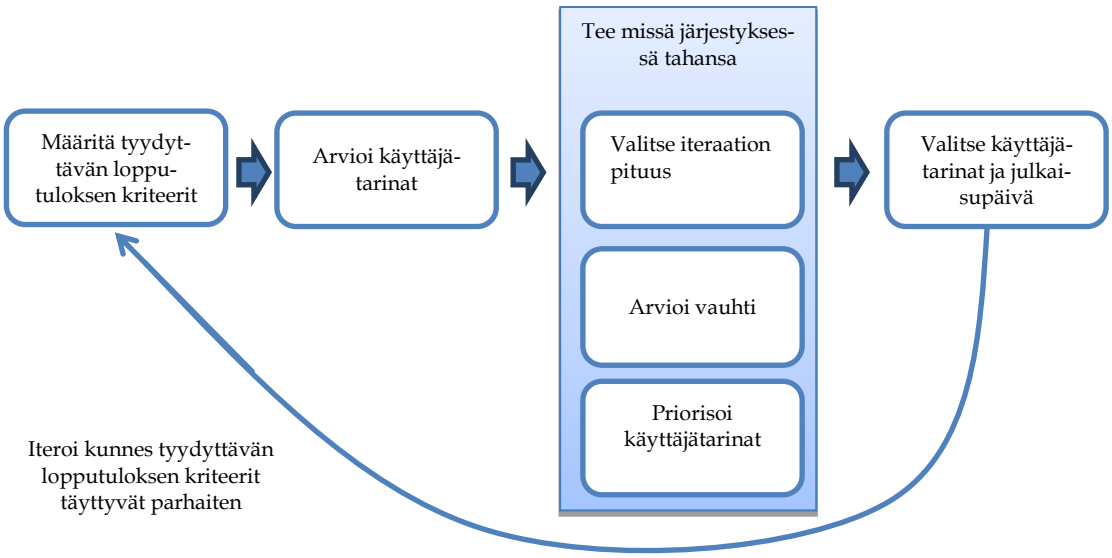
Vaatimusten priorisointi (requirements prioritization) -vaiheessa tunnistetut ja järjestetyt vaatimukset priorisoidaan. *Julkaisun määrittely* (release definition) -vaihe sisältää priorisoitujen vaatimusten valitsemisen tulevaan julkaisuun. *Julkaisun määrittelyn vahvistaminen* (release definition validation) -vaiheessa julkaisuun valittujen vaatimukset vahvistetaan eri sidosryhmillä. *Laajuuden muutoksen hallinta* (scope change management) -vaihe sisältää kaikkien julkaisun kehityksen aikana tehtävien muutoksien hallinnan. *Julkaistavan version vahvistaminen* (build validation) –vaihe tehdään, kun ohjelmiston kehittäjät ovat saaneet julkaisun valmiiksi. Sen tarkoituksena on vahvistaa julkaisu ennen kuin se asetetaan jakeluun. *Jakeluun valmistautuminen* (launch preparation) -vaiheen tarkoituksena on valmistaa sisäisten ja ulkoisten sidosryhmien edustajat uuteen julkaisuun. Siihen kuuluu tehtäviä liittyen viestintään, dokumentaatioon, kouluttamiseen ja itse julkaisun implementointiin.

3.3.2 Cohnin prosessimalli

Cohn kuvaa julkaisunsuunnittelua iteratiiviseksi prosessiksi, jossa suunnitelmaa päivitetään säännöllisesti (Cohn, 2006, 143). Suunnitteluprosessin tuloksena syntyy käyttäjätarinoiden tarkkuudella kuvattu korkean tason suunnitelma, joka kattaa iteraatiota pidemmän ajanjakson eli tyypillisesti 3–6 seuraavan kuukauden tai 3–12 iteraation ajan (Cohn, 2006, 133). Prosessissa työmäärään ja kehitysvauhdin arvioinneissa käytetään yksikkönä joko tarinapisteitä tai ideaalipäiviä. *Tarinapisteet* (story points) ovat tarinoiden vaativuutta kuvaavia suhteellisia numeroarvoja. Jos jokin tarina arvioidaan vaativuudella 1, tätä tarinaa kak-

si kertaa vaativammaksi arvioidulle tarinalle annetaan arvo 2. Tarinapisteiden mittakaavalla ei ole merkitystä, mutta Cohn suosittelee käyttämään 1–10 pisteen asteikkoa. (Cohn, 2006, 36.) *Ideaalipäivä* (ideal days) kuvaa miten monta henkilötyöpäivää tarinan toteuttamiseen kuluu (Cohn, 2006, 43).

Cohnin prosessissa julkaisunsuunnittelu tapahtuu kuvion 3 mukaisissa vaiheissa. *Tyydyttävän lopputuloksen kriteerien määrittely* –vaiheessa tuotteen omistaja määrittelee ensin hyväksyttävän suunnitelman kriteerit. Kriteerit liittyvät tavallisesti määräaikaan (schedule), toiminnallisuuksien määrään eli laajuuteen (scope) tai käytössä oleviin resursseihin. (Cohn, 2006, 135).



KUVIO 3 Cohnin (2006, 135) prosessimalli

Julkaisujen tavoitteet perustuvat tyypillisesti joko aikaan tai julkaisun toiminnallisuuksiin. Määräaikalähtöisessä (date-driven) julkaisuprojektissa täytyy olla julkaistu tiettyyn ajankohtaan mennessä, mutta sen toiminnallisuuksista voidaan olla valmiita tarvittaessa joustamaan. *Toiminnallisuuslähtöisessä* (feature-driven) projektissa on tärkeää, että valmis julkaisu sisältää halutut toiminnallisuudet. (Cohn, 2006, 136.)

Käyttäjätarinat arvioidaan kehittäjien toimista niiden toteuttamisen vaativuuden perusteella. Vaativuuden arvioinnissa käytetään mittarina joko tarinapisteitä tai ideaalipäiviä. Arviointi tehdään vain niiden käyttäjätarinoiden osalta, joilla on kohtuullinen todennäköisyys olla mukana tulevassa julkaisussa. Joskus voidaan arvioida myös muutamien seuraavien julkaisujen toivelistalla olevia tarinoita. (Cohn, 2006, 136.)

Iteraation pituuden valitseminen -vaiheessa päätetään, minkä pituisissa iteraatioissa julkaisu kehitetään. Tavallisesti iteraation pituus on 2–4 viikkoa, mutta pituus voi tarvittaessa poiketa tästä ajasta. Sopiva iteraation pituus riippuu ympäristöstä, jossa ohjelmistoa kehitetään. Valintaan vaikuttavia tekijöitä on useita, joista esimerkkinä ovat julkaisun pituus ja kehitystyöhön liittyvä epävarmuus. Cohnin (2006) mukaan on tärkeää, että julkaisuprosessin aikana on

riittävän monta välietappia, joissa kehityksen suuntaa voidaan tarkastella. Lyhyissä julkaisuprojekteissa voi olla hyödyllistä käyttää lyhyempiä iteraatioita kuin pidemmissä projekteissa. Toisaalta mitä enemmän epävarmuutta ohjelmistonkehitykseen liittyy, sitä lyhempiä iteraatioita kannattaa käyttää. (Cohn, 2006, 167.)

Käyttäjätarinoiden priorisointi -vaiheessa käyttäjätarinat asetetaan haluttuun tärkeysjärjestykseen tuotteen omistajan toimesta. Priorisointiin vaikuttavia tekijöitä ovat esimerkiksi toiminnallisuudesta saatava liiketoiminta-arvo, kehittämisen kustannus, se syntyykö toiminnallisuuden kehittämisestä merkittävää tietoa kehitystiimille ja mitkä riskit vähenevät, jos toiminto kehitetään. (Cohn, 2006, 80).

Vauhdin arviointi -vaiheessa arvioidaan tiimin vauhtia (velocity) toiminnallisuuksien kehittämisessä. Jos tiimi on tottunut työskentelemään keskenään, arvioinnin lähtökohtana kannattaa käyttää tiimin viimeaikaista nopeutta. Jos taas esimerkiksi liiketoimintaympäristössä tai käytetyssä teknologiassa on tapahtunut muutoksia, Cohn suosittelee käyttämään tiimin vanhempia nopeuksia. (Cohn, 2006, 136.)

Käyttäjätarinoiden ja julkaisupäivän valitseminen –vaiheeseen vaikuttaa se, onko julkaisun tavoite toiminnallisuus- vai määräaikalähtöinen. Jos julkaisun tavoite on toiminnallisuuslähtöinen, lasketaan yhteen kaikkien haluttujen käyttäjätarinoiden pisteet ja jaetaan saatu summa tiimin nopeudella. Tulokseen saadaan iteraatioiden määrä, joka toiminnallisuuksien toteuttamiseen tarvitaan. Mikäli projekti on määräaikaan sidottu, iteraatioiden määrä voidaan laskea suoraan projektiin käytettävissä olevasta ajasta. Kertomalla iteraatioiden määrän tiimin arvioidulla nopeudella, saadaan tieto sitä, kuinka monta tarinapistettä näiden iteraatioiden aikana on mahdollista toteuttaa. Tämän jälkeen valitaan julkaisuun mahtuvat käyttäjätarinat. (Cohn, 2006, 137.)

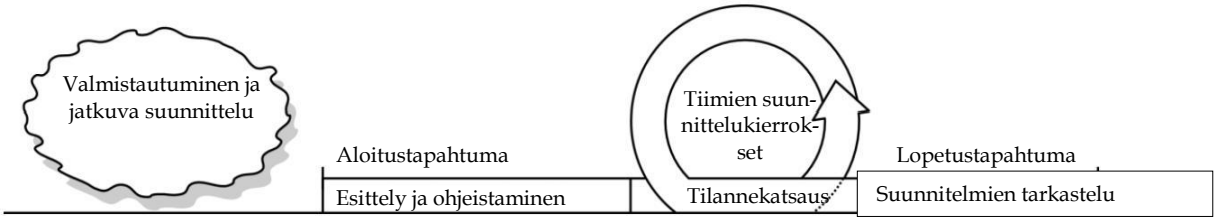
Käyttäjätarinat voidaan sijoittaa tarkasti haluttuihin iteraatioihin julkaisussa tai yleisemmin niin, että valmista tarinoiden iteraatioihin sijoittamista ei tehdä. Tarkemmasta määrittelytavasta voi olla hyötyä, jos työtä jaetaan useamman tiimin kesken. Haittapuolena on tarkempaan suunnittelutyöhön kuluva pidempi aika. (Cohn, 2006, 137.)

Kun käyttäjätarinat ja julkaisupäivä on valittu, tarkastellaan miten hyvin suunnitelman lopputulos täyttää suunnitelmalle asetetut kriteerit (vrt. kuvio 3). Tarvittaessa suunnitteluprosessi voidaan toistaa. Cohn (2006, 139) suosittelee myös päivittämään suunnitelmaa säännöllisesti, mieluiten joka iteraation alussa.

3.3.3 Julkaisun yhteissuunnittelu -menetelmä

Julkaisun yhteissuunnittelu -menetelmässä (joint release planning method) (Heikkilä, 2010) kuvataan suunnittelutapahtuma, jossa usea tiimi voi suunnitella yhdessä tulevan julkaisun sisältöä niin, että eri tiimien työlistoissa olevien käyttäjätarinoiden väliset yhteydet tulevat huomioitua. Menetelmän kuvauksessa on esitetty sekä suunnittelutapahtumaa edeltäviä ja tapahtumien aikaisia

ja niiden jälkeen tehtäviä aktiviteetteja. Tapahtuman rakennetta on esitetty kuviossa 4.



KUVIO 4 Julkaisun yhteissuunnittelutapahtuma (Heikkilä, 2010, 175)

Suunnittelutapahtumaa edeltää *valmistautuminen ja jatkuva suunnittelu* -vaihe. Vaiheessa tehdään mm. van de Weerdin, Brinkkemperin, Nieuwenhuisin, Versendaalin ja Bijlsman ym. (2006) esittämiä ohjelmistotuotehallinnan osa-alueiden, kuten vaatimusten hallinnan tehtäviä (Heikkilä, 2010, 175) ja valmistellaan seuraavan julkaisun yhteissuunnittelun järjestämistä (Heikkilä, 2010, 181). Tuotteen omistajat ja tuotepäälliköt myös valitsevat suunnittelutapahtumassa käsiteltävät toiminnallisuudet siten, että tulevassa suunnittelutapahtumassa on mukana riittävän kattava joukko toiminnallisuuksia. Heikkilä (2010) suosittelee rajoittamaan suunnittelutilaisuuteen valittujen toiminnallisuuksien määrää niin, että mukana on vain toiminnallisuuksia, joilla on realistinen mahdollisuus tulla mukaan seuraavaan julkaisuun. Toiminnallisuudet jaetaan alustavasti tiimeille, mutta tiimit voivat myöhemmin vapaasti vaihtaa toiminnallisuuksia keskenään. Esijako tehdään siksi, että usein tiimeille on muodostunut omia vahvuusalueitaan ja ei ole käytännöllistä, että useat tiimit keskustelisivat toiminnallisuuksien jaosta aluksi keskenään. Esijaetut toiminnallisuudet myös priorisoidaan tiimikohtaisesti toiminnallisuuksien ajoittamisen helpottamiseksi. Näin tiimi voi esimerkiksi valita jonkin vähemmän tärkeän toiminnallisuuden tehtäväksi ennen tärkeää toiminnallisuutta, jos huomataan, että vähemmän tärkeä toiminnallisuus on sidoksissa toisen tiimin erittäin tärkeään toiminnallisuuteen. Vaiheen aikana asiantuntijat (ml. järjestelmäarkkitehdit) valmistelevat omat materiaalinsa. Kehitystiimien vastuulla on varmistaa työnsä edistymistä kuvaavien mittarien ajantasaisuus ja arvioida tiimiensä työskentelyvauhdit. Tapahtumaa varten on varattava riittävän suuri yhteinen tila niin, että tiimeille, sidosryhmien edustajille, esiintymistilaisuuksille ja tauoille on omat alueensa. (Heikkilä, 2010, 175-178.)

Varsinainen suunnittelutapahtuma alkaa *esittely ja ohjeistaminen* –vaiheella, jossa tiimejä opastetaan tarpeen mukaan julkaisunsuunnittelun arviointi- ja suunnittelutekniikoiden käytössä. Tiimeille painotetaan myös, että he ovat vastuussa toiminnallisuuksien välisten riippuvuuksien selvittämisestä ja siitä, että tiimien tulisi aktiivisesti olla yhteydessä muihin tiimeihin ja sidosryhmiin avun saamiseksi. Esittelyn jälkeen tiimeille esitetään julkaisuprojektin visio, julkaisun tavoitteen informoimiseksi. Samassa yhteydessä tiimeille jaetaan alustavat vaatimukset, jotka esitetään yleisesti konseptitasolla ja tarkemmin toiminnallisuuksina. Konseptitason esitys motivoi tiimejä ja toiminnalli-

suuksina kuvatut vaatimukset antavat suuntaa kehitystyölle. Eri asiantuntijatahot, kuten arkkitehdit, esittävät tuotteesta oman näkökulmansa mukaisen esityksen. Toisessa ja myöhemmissä julkaisun yhteissuunnittelun tapahtumissa käydään myös läpi ohjelmistoprojektin edistymisaste. (Heikkilä, 2010, 178.)

Suunnittelutapahtuman toisessa, *tiimien suunnittelukierrokset ja tilannekatsaukset,* vaiheessa tiimit jakautuvat omille aluilleen ja aloittavat oman tiiminsä julkaisuaikataulun iteratiivisen suunnittelun. Tässä yhteydessä tiimit tarkentavat toiminnallisuudet tarkemmiksi käyttäjätarinoiksi. Laadittujen käyttäjätarinoiden sisältö ja toteutusjärjestys päätetään sen mukaan, millaisia teknisiä vaatimuksia käyttäjätarinoiden toteutuksesta syntyy ja mikä on niiden liiketoiminta-arvo. Käyttäjätarinoiden väliset riippuvuudet pyritään selvittämään välittömästi. Käyttäjätarinat myös jaetaan yksittäisiin iteraatioihin ottamalla samalla huomioon tiimien iteraatiokohtaiset kehittämisnopeudet. Suunnitelmia päivitetään sitä mukaan, kun tiimien toiminnallisuuksien välillä löytyy uusia riippuvuuksia. Toiminnallisuuksien riippuvuuksissa havaitut konfliktit pyritään ratkaisemaan arvioimalla, mikä on hyödyllinen toiminnallisuuksien toteuttamisjärjestys koko julkaisun kannalta. Vaiheen aikana on tärkeää, että tiimit saavat eri sidosryhmien edustajilta apua sidosryhmien asiantuntija-alueelle liittyvien ongelmien ratkaisemiseen. Vaiheen onnistumisen kannalta on myös tärkeää, että tiimit ymmärtävät tiimien välisen viestinnän tärkeyden, ja viestintätavat on etukäteen sovittu. Vaiheen aikana järjestetään lisäksi säännöllisesti tilannekatsauksia joihin osallistuu kustakin tiimistä yksi edustaja kerrallaan. (Heikkilä, 2010, 178-180.)

Tapahtuman viimeisessä vaiheessa, *suunnitelmien tarkastelussa* esitetään jokaisen tiimin yleisellä tasolla kuvaama suunnitelma ja esiin nousseet tiimin työhön vaikuttavat riskit (Heikkilä, 2010, 181).

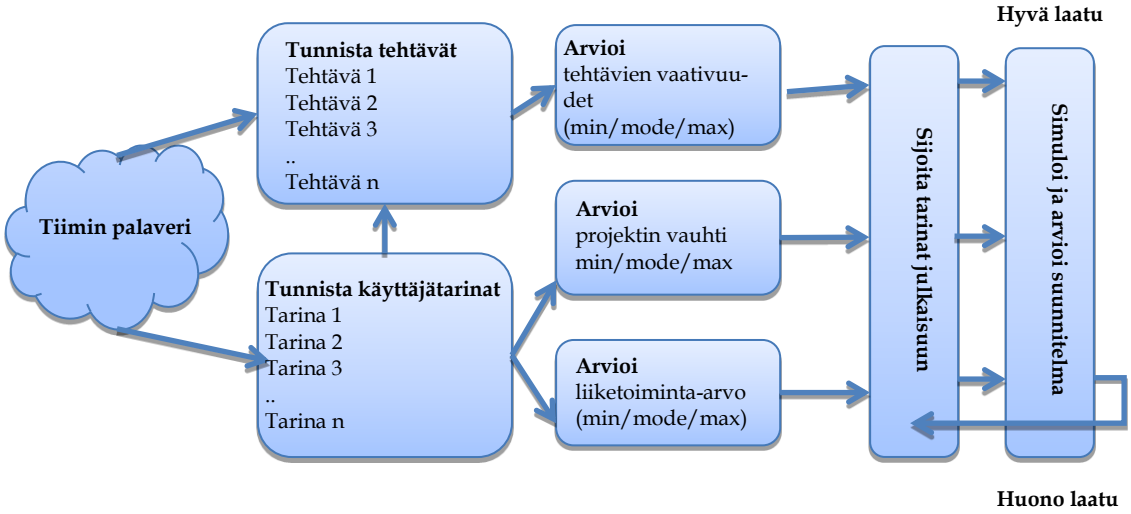
Suunnittelutapahtuman jälkeen, kun kehitystiimit ovat aloittaneet kehitystyön, *projektia seurataan ja ohjataan.* Tuotepäälliköt ja tuotteen omistajat aloittavat välittömästi seuraavan suunnittelutapahtuman valmistelun. Kehittämistyön aikana tiimit ovat vastuussa siitä, että kehitystyön edistymisestä on ajantasaista seurantatietoa. Mikäli kehitystyö näyttää viivästyvän, tuotteen johto (product management) on vastuussa toiminnallisuuksien toteutuksen laajuuden rajoittamisesta tai aikataulujen muuttamisesta. Lisäksi tiimien edistymisestä ja tiimien kehittämien käyttäjätarinoiden välisestä riippuvuudesta kertovat tiedot asetetaan kehitysorganisaatiossa keskeiselle paikalle esille tai saataville. (Heikkilä, 2010, 181.)

3.3.4 Loguen ja McDaidin prosessimalli

Loguen ja McDaidin (2008) esittävät prosessimallin julkaisunsuunnittelulle, jonka tarkoituksena on tuottaa tilastollista menetelmää hyödyntäen joukko suunnitelmia, joihin käyttäjätarinat on sijoitettu optimaalisella tai lähes optimaalisella tavalla. Prosessin etuna on, että se sallii epävarmuuden käyttäjätarinoiden vaativuuden ja niiden tuottaman liiketoiminta-arvon sekä projektin

vauhdin suhteen. (Logue & McDaid, 2008, 437.) Kuviossa 5 on kuvattu prosessin vaiheet. Vaiheet on selitetty jäljempänä.

Loguen ja McDaidin (2008) prosessissa käyttäjätarinoiden koko arvioidaan joko tarinapisteiden tai ideaalisten päivien mukaan. Loguen ja McDaidin (2008) artikkeli ei selitä tarkemmin, soveltuuko prosessi pidemmän ajan kuin seuraavaan julkaisun suunnitteluun.



KUVIO 5 Loguen ja McDaidin (2008, 439) prosessimalli

Prosessi alkaa tuleviin julkaisuihin sisällytettävien *käyttäjätarinoiden tunnistamisella ja valinnalla*. Tämän jälkeen käyttäjätarinoiden vaativuus arvioidaan. Arviointi tehdään jakamalla käyttäjätarinat ensin tehtäviin, jonka jälkeen *tehtävien vaativuudet arvioidaan*. Arviointi tehdään antamalla pessimistinen, todennäköinen ja lopuksi optimistinen arvio tehtävien vaativuudesta. Seuraavaksi samaa kolmitasoista arviointitapaa käytetään *käyttäjätarinoiden liiketoiminta-arvon* ja *projektin vauhdin arviointiin*. (Logue & McDaid, 2008, 439-440.)

Kun edellä mainitut arvioinnit on tehty, tuotteen omistaja *sijoittaa käyttäjätarinat julkaisuun*. Tämän jälkeen tehtävien aika-arvioiden perusteella suoritetaan erityinen Monte Carlo -simulaatio, jonka tuloksena saadaan jakauma, joka osoittaa minä ajankohtana julkaisu saadaan kaikkein todennäköisimmin kehitettyä. Käytettyä menetelmää on tarkasteltu hieman tarkemmin Loguen ja McDaidin (2008) artikkelissa, mutta menetelmän yksityiskohdat sivuutetaan tässä esityksessä. Simulointimenetelmää käytetään myös projektin työskentelyvauhdin arviointiin, jolloin saadaan laskettua todennäköinen aika, jolloin valittu julkaisu valmistuu kalenteripäivissä. Lopuksi simulointimenetelmää käytetään julkaisuun valittujen käyttäjätarinoiden todennäköisen liiketoiminta-arvon laskemiseen (Logue & McDaid, 2008, 439-441.)

Edellä esitetty prosessi ei tähtää suoraan optimaalisen julkaisusuunnitelman tekemiseen, vaan prosessin avulla tuotteen omistaja voi verrata keskenään erilaisten käyttäjätarinavalintojen liiketoiminta-arvoa ja niiden todennäköistä

kehitysaikaa keskenään. Näin ollen esitetty menetelmä voidaan nähdä julkaisunsuunnitteluprosessin lisäksi päätöksenteon tekniikkana.

3.3.5 SCERP-prosessimalli

Heikkilä, Jadallah, Rautiainen ja Ruhe (2010) esittävät *sidosryhmäkeskeisen julkaisunsuunnittelun* (Stakeholder-CEntric Release Planning method, SCRERP) prosessimallin. Prosessi tarjoaa tukea päätöksenteolle ja on yhteensopiva ReleasePlanner™ -työkalun kanssa. Prosessi mahdollistaa usean sidosryhmän edustajan mielipiteen huomioonottamisen julkaisujen suunnittelussa.

Julkaisunsuunnitteluun osallistuu tuotepäällikkö (product manager) sekä kriittisten sidosryhmien edustajat. Heikkilän ym. (2010) mukaan kriittisten sidosryhmien valintaan käytetään Sharpin, Finkelsteinin ja Galalin (1999) esittämää menetelmää. Jokaiselle valitulle sidosryhmän edustajalle määritellään myös suhteellinen painoarvo asteikolla 1–9 niin, että arvo 1 merkitsee sidosryhmän edustajan vähäistä merkitystä ja arvo 9 suurta merkitystä.

SCERP-prosessin vaiheet ovat:

1. kriittisten sidosryhmien ja alustavien toiminnallisuuksien valinta
2. toiminnallisuuksien priorisointi
3. vaativuuden arviointi
4. optimaalisten suunnitelmavaihtoehtojen laskeminen
5. julkaisusuunnitelmien priorisointi.

Kriittisten sidosryhmien ja alustavien toiminnallisuuksien valinta -vaiheessa tunnistetaan tärkeät sidosryhmän edustajat ja kutsutaan heidät osallistumaan suunnitteluprosessiin. Tuotepäällikkö ja sidosryhmien edustajat tuovat päätöksentekoon omat ideansa toiminnallisuuksista, ja näistä ideoista valitaan alustava toiminnallisuuksien joukko.

Kun alustavat toiminnallisuudet on valittu, *toiminnallisuudet priorisoidaan* valittujen arviointikriteerien mukaisesti. Arviointikriteereinä voidaan käyttää esimerkiksi toiminnallisuudesta saatava liiketoiminta-arvoa tai sen kiireellisyyttä, tai sitä miten suuri haitta koituu, jos toiminnallisuuden toteutus viivästyy. Priorisointiin osallistuvat sidosryhmän edustajat arvioivat toiminnallisuuksia antamalla kriteereille arvon yhdestä yhdeksään. Järjestykseen asettaminen voi olla kumulatiivista, jolloin toiminnallisuudet ovat keskenään priorisoituja ja sama arvo ei voi olla usealla toiminnallisuudella, tai vapaata, jolloin toiminnallisuuksilla voi olla myös samat arvot. Vaiheen tavoitteena on saada tietoa siitä, miten eri sidosryhmä arvostavat eri toiminnallisuuksia.

Vaativuuden arviointi -vaiheessa sidosryhmien edustajat, tyypillisesti kehittäjät, arvioivat kunkin toiminnallisuuden vaativuuden. Arvioinnissa voidaan käyttää jotain tunnettua arviointitekniikkaa, kuten suunnittelupokeria (esim. Moløkken-Østvold, Haugen & Benestad, 2008).

Optimaalisten suunnitelmavaihtoehtojen laskeminen -vaiheessa tuotepäällikkö käyttää Ngo-Then ja Ruhen (2007) esittämää optimointimenetelmää, jonka syöt-

teenä käytetään sidosryhmän edustajien tietoja ja heille asetettuja painoarvoja, alustavien toiminnallisuuksien joukkoa, arvioituja toiminnallisuuksien vaativuuksia, tietoa siitä kuinka nopeasti ohjelmistoja voidaan kehittää yhden julkaisun aikana ja sidosryhmien kriteeripainotuksia. Tuloksena saadaan viisi optimaalista julkaisusuunnitelmavaihtoehtoa.

Julkaisusuunnitelmien priorisointi -vaiheessa jokaisen sidosryhmän edustajia pyydetään arvioimaan edellisessä vaiheessa tuotettuja viittä suunnitelmaa ja priorisoimaan suunnitelmat samalla tavoin kuin yksittäiset toiminnallisuudet arvioitiin toiminnallisuuksien priorisointi –vaiheessa. Tuloksena saadaan sidosryhmien edustajien suositukset toteutettavasta suunnitelmasta.

3.4 Prosessimallien vertailu

Tässä alaluvussa verrataan edellä kuvattuja prosessimalleja viiden kriteerin mukaisesti. Ensimmäisenä vertailukriteerinä käytetään *prosessimallien kattavuutta.* Arvioinnissa tutkitaan, mitkä Bekkersin ym. (2010) mallin vaiheet sisältyvät vertailtaviin prosessimalleihin. Toisena kriteerinä verrataan *mitä aktiviteetteja prosesseissa on kuvattu, jota Bekkersin ym. (2010) malli ei sisällä*. Kolmantena vertailukriteerinä käytetään prosessien *käyttötapaa*. Tämän jälkeen verrataan prosessien *roolituksia*. Viidenneksi tutkitaan, miten eri *prosessit soveltuvat käytettäväksi Scrumin yhteydessä*. Yhteenveto vertailun tuloksista on esitetty taulukossa 1.

Ensiksi verrataan mitä julkaisunsuunnitteluun liittyviä aktiviteetteja kukin malli kattaa käyttämällä vertailupohjana Bekkersin ym. (2010) mallissa kuvattuja julkaisunsuunnittelun vaiheita, jotka ovat:

- vaatimusten priorisointi
- julkaisun määrittelyn
- julkaisun määrittelyn vahvistaminen
- laajuuden muutosten hallinta
- julkaistavan version vahvistamisen
- jakeluun valmistautuminen.

Jokaiseen vertailtavaan prosessimalliin sisältyy vaatimusten priorisointia. Cohnin mallissa priorisoinnissa otetaan huomioon useita eri kriteerejä, kuten käyttäjätarinoista saavutettava liiketoimintahyöty ja niistä aiheutuvat kustannukset. Heikkilän (2010) kuvaamassa julkaisun yhteissuunnittelun menetelmässä priorisointi tapahtuu tuotteen omistajan ja tuotepäällikön yhteistoiminnalla sekä tiimien yhteistyössä. Loguen ja McDaidin (2008) mallissa ja Heikkilän ym. (2010) SCERP-mallissa vaatimuksia priorisoidaan useiden eri kriteerien perusteella. Näistä kriteereistä esimerkkeinä ovat liiketoiminta-arvo ja vaatimuksien toteuttamisen vaativuus. Heikkilän ym. (2010) SCERP-malli huomioi lisäksi usean eri sidosryhmän edustajan mielipiteen priorisoinnissa.

TAULUKKO 1 Prosessimallien vertailu

Mallit	**Bekkers ym. malli**	**Cohnin malli**	**Julkaisun yhteissuunnittelu**	**Loguen & McDaidin malli**	**SCERP-malli**
Julkaisunsuunnittelun tehtävät					
Vaatimusten priorisointi	-	Kattaa	Kattaa	Kattaa	Kattaa
Julkaisun määrittely	-	Kattaa	Kattaa	Kattaa	Kattaa
Julkaisun määrittelyn vahvistaminen	-	Kattaa	Ei kata	Osittain kattaa	Osittain kattaa
Laajuuden muutostenhallinta	-	Osittain kattaa	Osittain kattaa	Ei kata	Ei kata
Julkaistavan version vahvistaminen	-	Ei kata	Ei kata	Ei kata	Ei kata
Jakeluun valmistautuminen	-	Ei kata	Ei kata	Ei kata	Ei kata
Aktiviteetit joita Bekkersin ym. malliin ei sisälly	-	Arvioinnit	Esittely- ja ohjeistaminenaktiviteetti	Arvioinnit ja optimoinnit	Arvioinnit, sidosryhmien valinta ja suunnitelmien priorisointi
Käyttötapa	Voi toimia vertailupohjana muille prosesseille	Julkaisusuunnitelman luonti	Usean tiimin yhteisen julkaisusuunnitelman luonti	Suunnitelman arvioiminen	Eri sidosryhmän mielipiteen huomioiminen julkaisusuunnittelussa
Roolitus	Ei kuvattu	Tuotteen omistaja ja kehittäjät	Tuotteen omistaja, tuotepäällikkö ja sidosryhmät, asiantuntijat ja kehitystiimit	Tuotteen omistaja	Tuotepäällikkö, kehittäjät ja eri sidosryhmät
Soveltuvuus Scrumin yhteyteen	Ei sovellu	Soveltuu	Soveltuu	Ei suoraan sovellettavissa	Soveltuu

Kaikki kuvatut prosessimallit sisältävät myös julkaisun määrittelyn eli julkaisuun sisältyvien vaatimusten valinnan. Osassa prosesseja (Cohn, 2006; Heikkilä, 2010; Heikkilä ym., 2010) käytetään vaatimuksien esivalintaa, jolloin suunnitte-

luprosessiin otetaan mukaan vain ne vaatimukset, joilla on realistinen mahdollisuus olla mukana tulevissa julkaisuissa.

Julkaisun määrittelyn vahvistamiseen liittyen prosessikuvauksista Heikkilän ym. (2010) prosessimalli pyrkii ottamaan huomioon eri sidosryhmän edustajien erilaiset mielipiteet julkaisuun valittujen vaatimusten tärkeydestä sekä eri suunnitelmavaihtoehtojen paremmuudesta. Malli ei ota kantaa siihen kuka lopullisen vahvistamisen tekee. Cohnin (2006) sekä Loguen ja McDaidin (2008) malleissa prosessia jatketaan niin kauan kuin tyydyttävän lopputuloksen mukainen tai tarpeeksi hyvän laatuinen suunnitelma on valmis. Cohnin (2006) prosessin kuvauksen mukaan tuotteen omistaja hyväksyy julkaisun määrittelyn, mutta Loguen ja McDaid (2008) malli ei ota tähän kantaa. Heikkilän (2010) mallissa tiimit suunnittelevat yhdessä julkaisusuunnitelman, mutta prosessin kuvauksessa ei mainita, miten lopullinen suunnitelma vahvistetaan.

Laajuuden muutostenhallintaan liittyen julkaisun yhteissuunnittelumenetelmän (Heikkilä, 2010) kuvauksessa ohjeistetaan, että tuotteen johto on vastuussa suunnittelutapahtumien jälkeen kehitettävien toiminnallisuuksien laajuuden rajaamisesta, jos kehitystyö ei edisty ennakoidulla tavalla. Cohnin (2006) prosessikuvauksessa kerrotaan, että suunnitteluprojektit ovat usein määräaika- tai toiminnallisuuslähtöisiä, jolloin viivästyksien tullessa joko julkaisun ajankohdasta tai toiminnallisuuksien laajuudesta voidaan olla valmiita tinkimään. Cohn (2006) suosittelee päivittämään suunnitelmaa myös säännöllisesti jokaisen iteraation jälkeen. Muut tarkastellut prosessit eivät ota kantaa kattavuuden muutostenhallintaan.

Mikään vertailussa mukana ollut prosessimalli ei ota kantaa siihen, millä tavoin julkaistava ohjelmistoversio vahvistetaan ennen sen jakelua tai valmistellaan käyttäjille.

Seuraavaksi tarkastellaan, mitä sellaisia aktiviteetteja tarkasteltavat mallit sisältävät, joita ei ole kuvattu Bekkersin ym. (2010) mallissa. Cohnin (2006) malli sisältää vaiheita, joissa arvioidaan käyttäjätarinoiden vaativuutta ja työskentelynopeutta. Julkaisun yhteissuunnittelu –menetelmässä (Heikkilä, 2010) on kuvattu esittely- ja ohjeistamisvaihe. Loguen ja McDaidin (2008) malli sisältää arviointivaiheita, joissa arvioidaan käyttäjätarinoiden toteutuksen vaativuutta ja niistä saavutettavaa liiketoiminta-arvoa. Lisäksi mallissa arvioidaan tehtävien vaativuutta ja valmiita suunnitelmavaihtoehtoja. Loguen ja McDaidin (2008) malli sisältää myös tilastollisen optimointimenetelmän käytön. SCERP-malli (Heikkilä ym., 2010) sisältää suunnitteluun osallistuvien sidosryhmän edustajien valinnan, vaatimusten arvioinnin ja valmiiden julkaisusuunnitelmien priorisoinnin.

Kolmanneksi tarkastellaan prosessimallien käyttötarkoituksia. Bekkersin ym. (2010) prosessimalli on tarkoitettu viitekehykseksi, jota voidaan käyttää esimerkiksi muiden julkaisunsuunnitteluprosessimallien vertailuun tai kehittämiseen. Cohnin (2006) mallia voidaan käyttää julkaisusuunnitelman laatimiseen. Yhdistetyn julkaisun yhteissuunnittelun (Heikkilä, 2010) malli tarjoaa menetelmän, jonka avulla julkaisunsuunnittelu voidaan järjestää niin, että useat tiimit tekevät suunnitelmia yhtäaikaisesti ja tiimien kehittämien toiminnalli-

suuksien väliset riippuvuudet tulevat huomioitua. Malli siis kuvaa suunnittelua usean tiimin yhtäaikaisen suunnittelun perspektiivistä. Yksittäisen tiimin suunnitteluperiaatteisiin menetelmä ei ota merkittävästi kantaa. Loguen ja McDaidin (2008) prosessimallia voidaan käyttää varsinkin tilanteissa, joissa epävarmuus käyttäjätarinoiden vaativuudesta, kehitystiimin vauhdista ja käyttäjätarinoista saatavista liiketoiminta-arvoista on suuri. Malli tarjoaa työkalun erilaisten suunnitelmien hyötyjen arvioimiseen, ei niinkään yhden optimaalisen julkaisunsuunnitelman tekoon (Logue & McDaid, 2008). SCERP–mallin (Heikkilä ym., 2010) avulla on mahdollista luoda viisi erilaista julkaisunsuunnitelmaa, joissa on huomioitu eri sidosryhmien mielipide. Kukin sidosryhmän edustaja voi myös antaa suosituksensa lopullisesta suunnitelmasta. Malli toimii näin myös päätöksenteon tukivälineenä.

Neljäntenä kriteerinä tarkastellaan, miten roolien osallistuminen on kuvattu prosesseissa. Bekkersin ym. (2010) mallissa ei ole otettu kantaa prosessin roolitukseen. Cohnin (2006) prosessiin osallistuvat tuotteen omistaja ja kehittäjät. Julkaisun yhteissuunnittelun menetelmässä (Heikkilä, 2010) prosessiin osallistuvat tuotepäällikkö, tuotteen omistaja, kehitystiimien jäsenet, erilaiset asiantuntijat ja sidosryhmien edustajia. Loguen ja McDaidin (2008) prosessiin osallistuu ainakin tuotteen omistaja. SCERP-mallissa (Heikkilä ym., 2010) prosessiin osallistuvat tuotepäällikkö ja erilaiset sidosryhmät mukaan lukien kehitystiimin jäsenet.

Lopuksi pohditaan lyhyesti tutkielmassa kuvattujen prosessimallien yhteensopivuutta Scrum–menetelmän kanssa. Tarkastelussa käytetään lähteenä kunkin prosessimallin yhteydessä esitettyä lähdemateriaalia. Logue ja McDaid (2008, 439) kertovat, että prosessi on yhteensopiva Extreme Programming (XP) –menetelmän kanssa. Prosessi ei ole välttämättä sellaisenaan yhteensopiva Scrumin kanssa, koska siinä käyttäjätarinat on jaettava tehtäviksi ennen kuin julkaisunsuunnitelmia laaditaan. Mikäli käyttäjätarinat määritellään varhaisessa vaiheessa, prosessi sopii myös Scrumin yhteyteen. Heikkilän ym. (2010) artikkelin mukaan heidän SCERP-malliaan on testattu Scrum–ympäristössä, joten prosessi sopii todennäköisesti hyvin Scrumiin. Julkaisun yhteissuunnittelun (Heikkilä, 2010) kuvauksen perusteella malli on yhteensopiva erityisesti Scrum–menetelmän kanssa. Malli ei ota kantaa tarkasti yksittäisen julkaisun suunnitteluun, vaan se keskittyy usean tiimin suunnittelutapahtumaan järjestämiseen. Cohnin (2006) prosessimallin kuvauksen yhteydessä ei oteta kantaa siihen, mihin menetelmään suunnitelma perustuu. Mallin ollessa kuvaukseltaan hyvin yleinen ja siinä esitetyt roolit ovat yhdenmukaisia Scrumin kanssa, sitä on mahdollista käyttää Scrumin yhteydessä.

3.5 Yhteenveto

Julkaisulla tarkoitetaan tässä tutkielmassa ohjelmistokokonaisuutta, joka koostuu yhden tai useamman iteraation tuloksesta ja joka voi olla joko käyttäjille ulkoiseen tai toimittajayrityksen sisäiseen käyttöön jaettava (Leffingwell, 2011).

Tutkielmassa julkaisunsuunnittelu nähdään prosessina, jonka lopputuloksena syntyy suunnitelma, joka antaa puitteet iteraatioissa tapahtuvaan kehittämisen suunnitteluun (Cohn, 2006). Prosessin lopputuloksena syntyvästä suunnitelmasta ilmenee lisäksi julkaisun valmistumisajankohta ja julkaisun sisältö (Loguen & McDaid, 2008). Julkaisunsuunnittelu on myös prosessi, joka muuttaa organisaation visiota konkreettisempaan muotoon (Shalloway ym., 2010).

Luvussa esiteltiin myös erilaisia julkaisunsuunnittelun prosessimalleja, jotka olivat Bekkersin ym. (2010) malli, Cohnin (2006) malli, julkaisun yhteissuunnittelun menetelmä (Heikkilä, 2010), Loguen ja McDaidin (2008) malli ja SCERP-malli (Heikkilä ym., 2010).

Mallien esittelyn jälkeen luvussa verrattiin prosessimallien kattavuutta käyttämällä vertailupohjana Bekkersin ym. (2010) mallia. Kaikki tarkastelussa olleet mallit sisälsivät vaatimusten priorisointiin ja valintoihin liittyviä aktiviteetteja. Yksikään malleista ei sisältänyt aktiviteetteja liittyen julkaistavan version vahvistamiseen tai jakeluun valmistautumiseen. Vain osa malleista sisälsi julkaisun määrittelyn vahvistamisen ja laajuuden muutostenhallinnan tehtäviä.

Toisena kriteerinä verrattiin, mitä sellaisia aktiviteetteja eri mallit sisältävät, jotka eivät sisälly Bekkersin ym. (2010) malliin. Yleisimpinä yhteisinä piirteinä esiin nousivat erilaiset arviointiaktiviteetit, joita esiintyi kaikissa verratuissa malleissa yhtä mallia (Heikkilä, 2010) lukuun ottamatta.

Lopuksi prosessimalleja verrattiin keskenään käyttämällä vertailukriteereinä mallien käyttötapaa, malleissa esiteltyjä rooleja ja mallien soveltuvuutta Scrumin yhteyteen. Erilaisia käyttötapoja olivat yhden tiimin julkaisunsuunnittelu, usean tiimin yhteisen suunnittelutapahtuman järjestäminen, prosessien kehittäminen, suunnitelmien arvioiminen ja sidosryhmien mielipiteen huomioiminen suunnittelussa. Yleisimpinä rooleina esiin nousivat tuotteenomistajan ja kehittäjien roolit ja vähemmissä määrin tuotepäällikön, sidosryhmien ja asiantuntijoiden roolit. Kaikki tarkastellut menetelmät olivat Bekkersin ym. (2010) ja Loguen ja McDaidin (2008) malleja lukuun ottamatta soveltuvia käytettäväksi Scrumin yhteydessä.

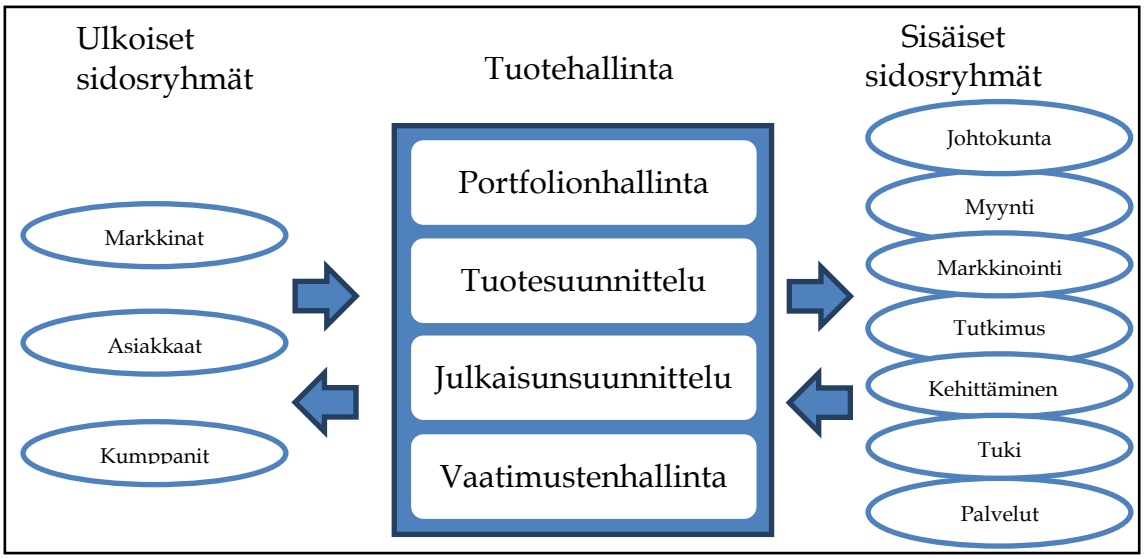
4 Julkaisunsuunnittelu osana tuotehallintaa

Julkaisunsuunnittelu on osa laajempaa kehitystoimintaa, jota kutsutaan tuotehallinnaksi (Bekkers ym., 2010) tai tuoteportfolion hallinnaksi (Vähäniitty, 2010b). Tämän luvun tarkoituksena on kuvata, miten julkaisunsuunnittelu nähdään osana tuotehallintaa ja millainen on liittymäpinta projekti- tai tiimitasoiseen ohjelmistokehittämiseen. Esitys on jaettu neljään osaan. Ensimmäiseksi esitetään tuotehallinnan kompetenssimalli (Bekkers ym., 2010), joka jäsentää tuotehallinnan neljään tasoon. Seuraavaksi kuvataan aihetta kahden ketterää tuotehallintaa jäsentävän mallin avulla (Leffingwell, 2009; Heikkilä, 2010). Viimeisessä alaluvussa käytetään kompetenssimallia viimeksi mainittujen mallien vertailuun.

4.1 Tuotehallinnan kompetenssimalli

Bekkers ym. (2010) esittävät tuotehallinnan kompetenssimallin, joka kuvaa tuotehallinnan ydinalueita, tuotehallinnan sidosryhmiä, ydinalueiden sisäisiä prosesseja ja edellä mainittujen välistä vuorovaikutusta. Tuotepäälliköt voivat käyttää tätä viitekehystä vertailupohjana, jonka avulla olemassa olevia ohjelmistotuotehallinnan käytänteitä on mahdollista kehittää (Bekkers, 2010, 1). Tässä yhteydessä aihetta ei tarkastella niin yksityiskohtaisella tavalla. Tästä syystä kuviossa 6 on esitetty yksinkertaistettu kuvaus mallista.

Ylimpänä tasona on *portfolionhallinta*. Tämä taso sisältää tuoteportfolion strategisen tason tiedonkeruun ja päätöksenteon tehtäviä. Seuraava taso on *tuotesuunnittelu*, joka sisältää ensinnäkin tuotteisiin, tuoteryhmiin ja tuotteiden ydinalueisiin liittyvää tiedonkeruuta, ja toiseksi edellä mainittuihin asioihin liittyvää pitkän aikavälin suunnittelua. Kolmantena tasona on *julkaisunsuunnittelu,* johon kuuluu edellisessä luvussa esillä olleita, julkaisun luontiin ja jakeluun liittyviä tehtäviä. Mallissa alimpana tasona on *vaatimustenhallinta*. Taso sisältää julkaisunhallinnan jatkuvana toimintona tapahtuvan vaatimusten keräämisen, tarkentamisen ja organisoinnin. (Bekkers ym., 2010, 4-5.)



KUVIO 6 Tuotehallinnan kompetenssimalli (mukaillen Bekkers ym., 2010, 4)

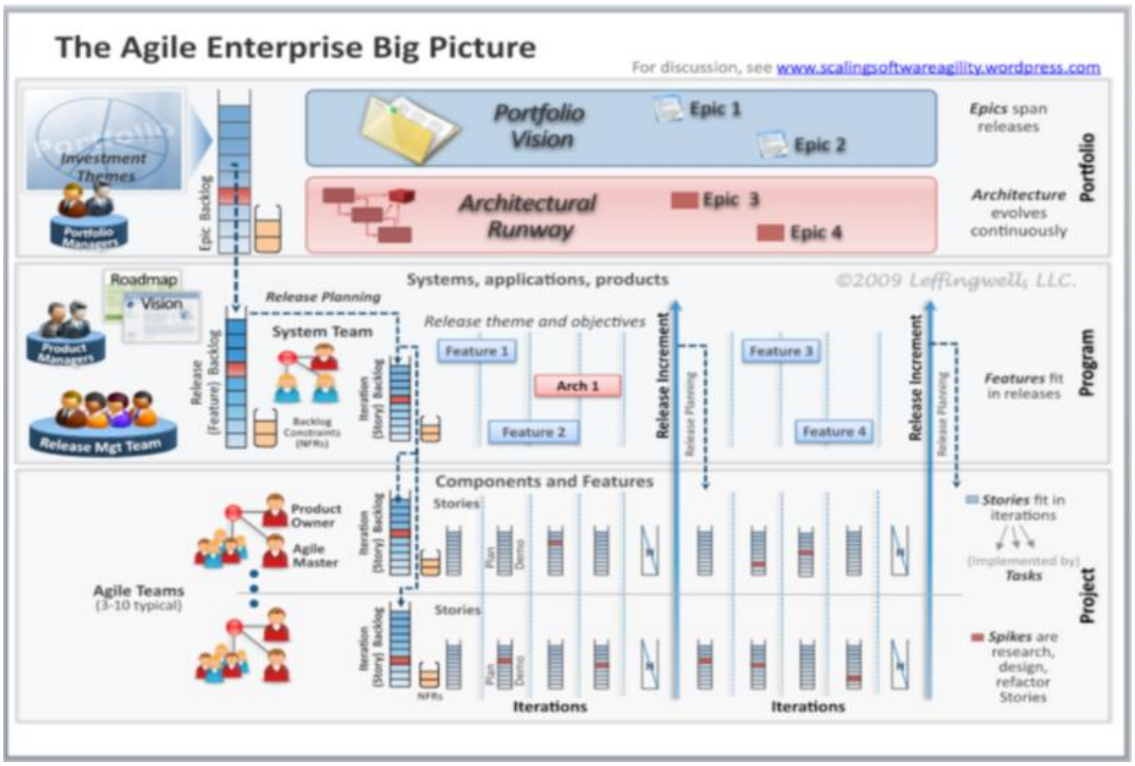
Mallissa tuotteenhallinta on kytketty nuolin monenlaisiin ulkoisiin sidosryhmiin, kuten markkinat, asiakkaat ja kumppanit, ja sisäisiin sidosryhmiin kuten yrityksen johtokunta, myynti, markkinointi ja tuotekehitys & tutkimus. Tuotehallinnan alla kuviossa ovat, kehittäminen (development), tukipalvelut (support) ja palvelut (services) (Bekkers ym., 2010). Kehittäminen vastaa projektityötä ohjelmistotuotteiden suunnittelemiseksi ja toteuttamiseksi.

Samaa mallia on esitelty myös van de Weerdin ym. (2006) sekä Vlaanderenin (2010) artikkeleissa. Esitykset poikkeavat hieman toisistaan.

4.2 Leffingwellin malli

Leffingwellin (2009) on esittänyt laajan mallin, jolla pyritään jäsentämään ketterää vaatimusmäärittelyä organisaation ja prosessien osalta. Mallista käytetään nimitystä Agile Enterprise Big Picture: Scaled Agile Delivery Model eli lyhyemmin Big Picture (Leffingwell (2011, 32). Malli rakentuu kolmesta tasosta (Kuvio 7), jotka ovat portfolio-, ohjelma- ja projektitaso. Projektitasosta voidaan vaihtoehtoisesti käyttää nimeä tiimitaso (Leffingwell, 2011, 300).

Ylimmällä *portfoliotasolla* esitetään yleisiä strategisia investointiteemoja ja epiikkejä, jotka tarkentuvat ja jakautuvat *ohjelmatasolla* konkreettisimmiksi toiminnallisuuskuvauksiksi. Nämä on osana julkaisun työlistaa (release backlog) ja ne jaetaan ohjelmatason julkaisunsuunnitteluprosessissa toteutettaviksi yksittäisissä julkaisuissa. Julkaisut puolestaan koostuvat useiden iteraatioiden tuloksista. Ohjelmatasolla voi toimia erityisiä järjestelmätiimejä (system team), jotka vastaavat laaja-alaisten järjestelmätoiminnallisuuksien kehittämisestä. Tähän tasoon kuuluvat myös tuotteen visio ja tiekartta.



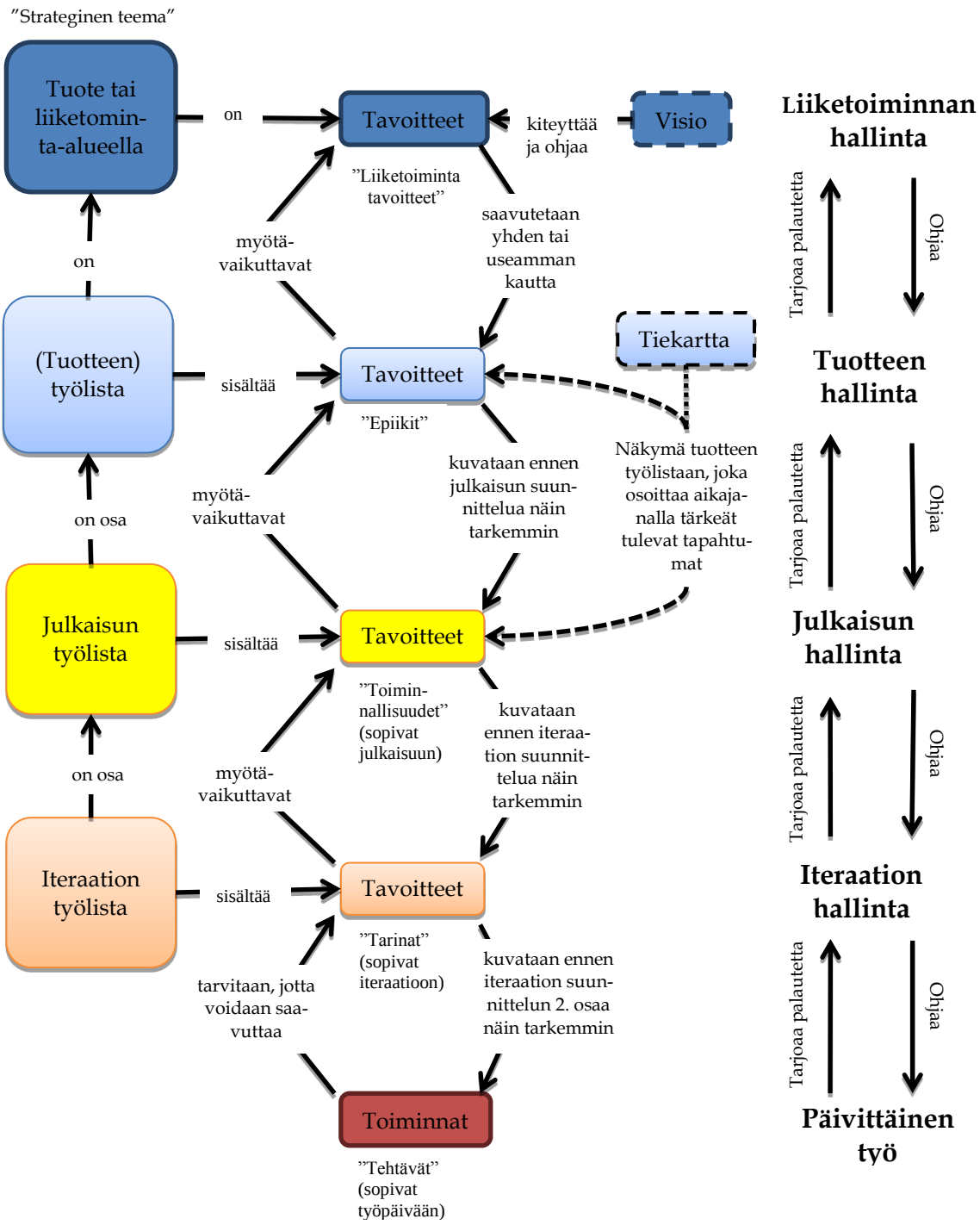
KUVIO 7 Big picture –malli (Leffingwell, 2009, 4)

Tiekartta on hyvin korkealla tasolla oleva kuvaus tulevien julkaisujen teemoista, sisällöistä ja julkaisuaikatauluista (Leffingwell, 2009, 12). *Projektitasolla* yksi tai useampi tiimi kehittää ohjelmiston toiminnallisuuksia iteraatio kerrallaan. Kullakin tiimillä on oma iteraation työlista, jossa toiminnallisuudet on kuvattu tarkemmin käyttäjätarinoina. Kullakin mallin organisaatiotasolla on kuvattu keskeiset roolit, jotka osallistuvat mallissa kuvattuun vaatimusten arvoketjuun. Arvoketjussa korkean tason ideat ja vaatimukset jakautuvat ja tarkentuvat ja lopulta toteutuvat tiimitasolla asiakkaalle arvoa tuottavina julkaisuina. (Leffingwell, 2009.)

4.3 ATMAN-malli

ATMAN-malli (Approach and Tool support for development portfolio MANagement) (Vähäniitty, 2010a, 124) kuvaa ohjelmistotuotteen kehitystyön jakaantumista viiteen tasoon, jotka ovat: liiketoiminnan hallinta, tuotteen hallinta, julkaisun hallinta, iteraation hallinta ja päivittäinen työ (Kuvio 8). Lisäksi malli osoittaa, millä tavoin vaatimukset on eri tasoilla määritelty ja miten eri tasojen vaatimukset liittyvät toisiinsa.

Seuraavaksi tarkastellaan ensin ATMAN-mallin vasemmalla reunalla kuvattua vaatimushierarkiaa tasoittain Vähäniityn (2010a) mukaan. Tämän jälkeen tarkastellaan viitekehyksen oikealla puolella olevaa kontrollisyklit -mallia.



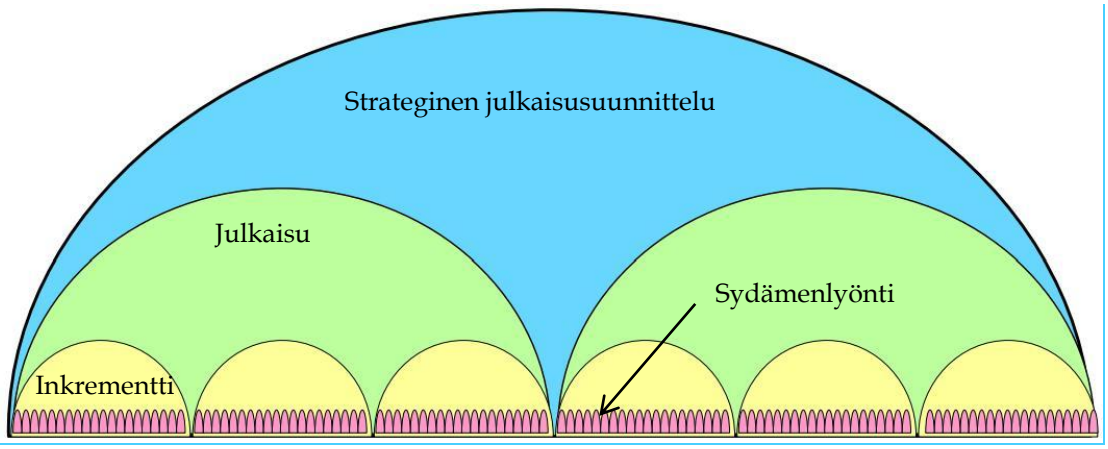
KUVIO 8 ATMAN–viitekehys (Vähäniitty, 2010a, 124)

Liiketoiminnan hallinnan tasolla yritysten tuote- ja liiketoiminta-alueilla on erilaisia strategisia teemoja, joihin liittyy liiketoimintatavoitteita (business goals). Liiketoimintatavoitteet kiteyttävät yrityksen vision. *Tuotteen hallinnan tasolla* tuotteilla on omat työlistansa, joissa listataan epiikit (epics). Leffingwellin (2009, 14) mukaan *epiikit* ovat korkeimman abstraktiotason kuvauksia käyttäjien tarpeista. *Julkaisun hallinnan tasolla* tuotteen työlistan osa, *julkaisun työlista*, sisältää tulevaan julkaisuun listattuja vaatimuksia. Käyttäjien tarpeet on esitetty tällä

tasolla toiminnallisuuksina (features). Toiminnallisuudet ovat epiikkien tarkempia kuvauksia. *Iteraation hallinnan tasolla* vaatimukset ovat listattu *iteraation työlistaan* (vrt. sprintin työlista) käyttäjätarinoina. Käyttäjätarinat ovat tarkempia kuvauksia toiminnallisuuksista. *Päivittäisen työn tasolla* käyttäjätarinat jaetaan yksityiskohtaisemmiksi tehtäviksi, joita yksittäiset kehittäjät suorittavat ja joiden lopputuloksesta käyttäjätarinan kuvaama toiminnallisuus toteutetaan (esim. Leffingwell, 2009, 9).

Edellä esitetyssä kuvauksessa eri tasojen vaatimukset muodostavat hierarkian. Heikkilä pitää hyödyllisenä sitä, että eri abstraktiotasojen vaatimuksien liittyminen toisiinsa säilytetään, kun vaatimuksia tarkennetaan yleisemmän tason kuvauksista tarkempiin kuvauksiin. Näin voidaan esimerkiksi jäljittää, mitkä iteraation hallinnantason käyttäjätarinat liittyvät tuotteen hallinnan tason epiikkeihin ja päinvastoin. Heikkilän mukaan jäljitettävyydestä voi olla hyötyä esimerkiksi tilanteessa, jossa organisaatio muuttaa strategista teemaa. Tällöin pystytään muuttamaan myös ne eri tasoilla oleva vaatimukset, jotka ovat tarkennettuja kuvauksia kyseisestä teemasta. Vastaavasti, kun vaatimuksia jaetaan pienemmiksi osiksi, voi syntyä tietoa, joka vaatii ylemmän tason vaatimusten muuttamista (Heikkilä, 2010, 123-125).

Viitekehyksen oikealla reunalla on kuvattu, miten alemmilta tasoilta saadaan palautetta ylemmille tasoille ja vastaavasti, miten ylemmän tason suunnittelu ohjaa alempien tasojen toimintaa. Tämä osa ATMAN-viitekehyksestä vastaa Rautiaisen ym. (2006) kontrollisykli (cycles of control)–mallia. *Kontrollisykli*mallissa on kuvattu liiketoiminnan, tuotteen ja iteraation hallinnan sekä päivittäisen työn tasot. Kontrollin malli pohjautuu *aikatahditus* (time pacing) -käsitteeseen, jolla tarkoitetaan sitä, että tietty ajanjakso jaetaan useisiin kiinteisiin ajanjaksoihin, joiden lopussa on kontrollipiste. Kontrollipisteissä arvioidaan saavutettuja tuloksia, tehdään päätöksiä ja päivitetään suunnitelmia. Kuviossa 9 on esitetty kontrollisyklimalli havainnollisemmin aikajanalla. (Rautiainen ym., 2006.)



KUVIO 9 Kontrollisyklimalli aikajanalla kuvattuna (Rautiainen ym., 2006, 7)

Kuviossa 9 nähdään laajimpana kaarena strateginen suunnittelu, joka kuviossa 8 vastaa tuotetason suunnittelua. Strategisen suunnittelun ajanjaksoon sisältyy

yllä esitetyssä kuviossa kaksi julkaisuprosessia, jotka kuviossa 8 vastaavat julkaisunsuunnittelun tasoa. Vastaavasti jokaisen julkaisun ajanjakson sisällä on useita inkrementtejä (näitä voidaan kutsua mallissa myös iteraatioksi). Jokainen iteraatio koostuu ”sydämenlyönneistä” (heartbeats), joilla tarkoitetaan yksittäisiä ohjelmistonkehityspäiviä. Jokaisen kuvion 9 sydämenlyönnin väli voidaan nähdä päivittäisenä kontrollipisteenä, jolloin arvioidaan mitä edellisinä päivinä on saatu aikaan ja muutetaan toimintaa aikaan saadun työn ja opitun uuden tiedon mukaan. Vastaavasti inkrementtien tai iteraatioiden välit ovat kontrollipisteitä, joissa arvioidaan miten edellinen iteraatio onnistui ja päätetään opitun uuden tiedon ja työnedistymisen perusteella, miten tulevien iteraatioiden suunnitelmia voidaan päivittää. Iteraatioiden välissä myös korkeamman tason suunnitelmia voidaan päivittää vastaamaan nykyistä tilannetta. Kontrollipisteiden aikana organisaatio ottaa huomioon aiemmista työvaiheista saadun palautteen ja päivittää suunnitelmiaan uuden tiedon valossa. (Rautiainen ym., 2006.)

4.4 Mallien vertailu

Tässä alaluvussa verrataan Leffingwellin (2009) mallia ja ATMAN-mallia (Vähäniitty, 2010a) toisiinsa käyttäen perustana Bekkersin ym. (2010) kompetenssimallia. Vertailu tehdään siten, että ensin verrataan kumpaakin mallia erikseen Bekkersin ym. (2010) malliin. Tämän jälkeen Leffingwellin mallia ja ATMAN-mallia verrataan toisiinsa taso kerrallaan käyttäen vertailukriteereinä sitä, millä tavoin vaatimukset on kuvattu eri tasoilla ja mitä keskeisiä piirteitä tasot sisältävät. Yhteenveto tästä vertailusta esitetään taulukossa 2. Jälkimmäisessä osassa vertailua tarkastellaan, miten eri mallit kuvaavat varsinaisen ohjelmistokehityksen ja julkaisunsuunnittelun välisen yhteyden.

Bekkersin ym. (2010) malli sisältää portfolionhallinnan, tuotesuunnittelun, julkaisunsuunnittelun ja vaatimustenhallinnan tasot. Lisäksi malli sisältää projekteina toteuttavan kehittämisen.

Leffingwellin malli sisältää Bekkersin ym. malliin verrattuna portfolionhallinnan tasoa vastaavan portfoliotason, julkaisunsuunnittelutasoa vastaavan ohjelmatason sekä projektitason, joka Bekkersin ym. viitekehyksessä vastaa projektitasoista kehittämistä. Malli ei sisällä tuotesuunnittelun tasoa, mutta sen ohjelmataso sisältää piirteitä, kuten tiekartan, jotka sisältyvät Bekkersin ym. tuotesuunnittelutasoon. Malli ei sisällä vaatimustenhallinnan tasoa mutta mallin eri tasoilla kuvataan vaatimusten organisointia ja eri tason vaatimusten keskinäistä suhdetta.

Leffingwellin malli ja ATMAN-malli sisältävät kumpikin portfoliotason jossa on kuvattu mm. strategisia teemoja. Leffingwellin mallissa portfoliotasolla kuvataan epiikit ja portfoliotason visio, joita ATMAN-mallin taso ei sisällä. ATMAN-mallissa kyseisellä tasolla on sen sijaan kuvattu liiketoimintatavoitteet ja visio. Leffingwellin mallissa tasolla on kuvattu portfoliopäällikkö-rooli

TAULUKKO 2 Tuotehallintaa jäsentävien mallien vertailu

Bekkersin ym. malli	Leffingwellin malli	ATMAN-malli
Portfolionhallinta Vaatimukset Muita piirteitä	**Portfoliotaso** - Strategiset investointiteemat ja epiikit - Portfoliotason visio ja arkkitehtuuri - Roolit: Portfoliopäälliköt (portfolio managers)	**Liiketoiminnan hallinta** - Strategiset teemat, liiketoimintatavoitteet ja visio - Taso ohjaa alemman tason toimintaa
Tuotesuunnittelu Vaatimukset Muita piirteitä		**Tuotteen hallinta** - Epiikit tuotteen työlistassa, tiekartta (roadmap) - Taso ohjaa alempaa tasoa ja tarjoaa palautetta ylemmälle tasolle
Julkaisunsuunnittelu Vaatimukset Muita piirteitä	**Ohjelmataso** - Toiminnallisuudet julkaisun työlistassa ja käyttäjätarinat iteraation työlistassa. Tiekartta ja visio - Kuvattu miten toiminnallisuudet sopivat julkaisuihin - Roolit: Järjestelmä- ja julkaisunhallintatiimit sekä tuotepäälliköt	**Julkaisun hallinta** - Toiminnallisuudet julkaisun työlistassa - Kuvattu miten toiminnallisuudet sopivat julkaisuihin - Taso ohjaa alempaa tasoa ja tarjoaa palautetta ylemmälle tasolle
Vaatimustenhallinta	Malli kuvaa miten korkean tason vaatimukset jakautuvat tarkemmiksi vaatimuskuvauksiksi	Malli kuvaa miten korkean tason vaatimukset jakautuvat tarkemmiksi vaatimuksiksi ja miten eri tason vaatimukset ovat sidoksissa toisiinsa
Kehittäminen -sidosryhmä Vaatimukset Muita piirteitä	**Projektitaso** - Käyttäjätarinat iteraation työlistassa ja tehtävät - Kuvattu miten käyttäjätarinat sopivat iteraatioihin ja tarinat toteutetaan tehtävinä - Roolit: kehitystiimit	**Iteraation hallinta** - Käyttäjätarinat iteraation työlistassa - Kuvattu miten käyttäjätarinat sopivat iteraatioihin - Taso ohjaa alempaa tasoa ja tarjoaa palautetta ylemmälle tasolle
Kehittäminen (viitekehyksen ulkoinen sidosryhmä) Vaatimukset Muita piirteitä		**Päivittäisen työn taso** - Tehtävät - Kuvattu miten tehtävät sopivat työpäivään ja ovat tarkempia kuvauksia käyttäjätarinoista - Taso tarjoaa palautetta ylemmälle tasolle

(portfolio managers). ATMAN-malli ei oteta kantaa rooleihin, mutta mallissa kuvataan, miten portfolionhallintataso ohjaa ja saa palautetta alemmalta tasolta.

Leffingwellin malli ei sisällä tuotehallinnan tasoa. ATMAN-mallissa tällä tasolla on kuvattu epiikit, tuotteen työlista, tiekartta ja se miten taso tarjoaa palautetta ylemmälle liiketoiminnan hallinnan tasolle ja ohjaa alemman julkaisun hallinnan tason toimintaa.

Kumpikin malli sisältää julkaisunsuunnittelutasoa vastaavan tason, joissa vaatimukset on kuvattu toiminnallisuuksina, jotka ovat osa julkaisun työlistaa. Leffingwellin mallissa tasolla on lisäksi esitetty tiekartta ja visio. Kummankin mallin kohdalla tasolla on esitetty, miten toiminnallisuudet sopivat aina tiettyyn julkaisuun. Leffingwellin mallissa tasolla on kuvattu myös käyttäjätarinat, iteraation työlista ja roolit, jotka ovat järjestelmä- ja julkaisunhallintatiimit sekä tuotepäälliköt. ATMAN-mallissa kuvataan, miten taso tarjoaa palautetta ylemmälle tasolle ja miten se ohjaa alemman tason toimintaa.

Kumpikaan tarkasteltavista malleista ei sisällä vaatimustenhallintasoa, mutta vaatimuksien organisoimista on kuvattu kunkin mallin jokaisella tasolla. ATMAN-malli kuvaa organisoimista hieman Leffingwellin mallia tarkemmin, sillä malli esittää, miten eri tasojen vaatimukset ovat jäljitettävissä toisiinsa.

Sekä Leffingwellin malli että ATMAN-malli sisältävät iteraatio- tai tiimitason, jolla on esitetty käyttäjätarinat ja iteraation työlistat ja joilla on kuvattu, miten käyttäjätarinat sopivat yksittäisiin iteraatioihin. Leffingwellin mallissa tällä tasolla on lisäksi kuvattu tehtävät. Roolien osalta Leffingwellin mallissa on kuvattu kehitystiimit. ATMAN-mallissa on esitetty, miten taso tarjoaa palautetta ylemmälle tasolle ja ohjaa alemman tason toimintaa. ATMAN-mallissa on kuvattu vielä päivittäisen työn taso, jota ei ole Leffingwellin mallissa. Tällä tasolla on esitetty tehtävät ja se miten tehtävät sopivan työpäiviin. Taso tarjoaa palautetta ylemmille tasoille.

Seuraavaksi tarkastellaan lyhyesti edellä mainittuja malleja ja viitekehyksiä erityisesti siitä näkökulmasta, miten ne kuvaavat varsinaisen ohjelmistokehityksen ja julkaisunsuunnittelun välisen yhteyden. Bekkersin ym. (2010) viitekehys kuvaa ohjelmistokehityksen tuotehallinnan ulkopuolella olevana osana (development) eikä se selitä tarkasti, millä tavoin julkaisunsuunnittelu liittyy ohjelmiston kehitystyöhön. Leffingwellin malli (2009) kuvaa erinomaisesti, miten organisaation tuotetasolla tehtävä julkaisunsuunnittelu liittyy projektitason ohjelmistokehitykseen ja miten epiikit jakautuvat tuotetason toiminnallisuuksiin ja toiminnallisuudet edelleen tiimitason käyttäjätarinoihin. Mallissa on selitetty myös eri organisaatiotasojen suunnitteluun osallistuvat roolit. Esitetyistä malleista ATMAN–viitekehys (Vähäniitty, 2010a) vastaa tutkielman kirjoittajan mielestä parhaiten siihen, miten julkaisunsuunnittelun taso liittyy ohjelmistokehitykseen ja siksi tätä mallia on tarkasteltu tutkielmassa perusteellisimmin. ATMAN-mallissa on kuvattu eri organisaatiotasojen vaatimusten muodostama hierarkia ja vaatimusten välinen jäljitettävyys. Viitekehyksen osana on kontrollisyklit–malli, joka kuvaa miten säännöllisin väliajoin ylemmät organisaatiotasot vastaanottavat palautetta alemmilta tasoilta ja miten ylemmät tasot ohjaavat alempien tasojen toimintaa.

Edellä esitetyt mallit tarjoavat vastauksia kysymykseen, miten julkaisunsuunnittelu tapahtuu saumattomasti ketterän kehittämisen kanssa. Yhteistyö ei ole sanatarkasti täysin saumatonta, vaan ”saumoina” toimivat osittain kontrollisykleissä esitetyt kontrollipisteet. Kuten tutkielman alaluvussa 2.3 todettiin, esimerkiksi Scrum–menetelmässä iteraatiot (sprintit) ovat rauhoitettu kehitystyön ajaksi ja muita kehitettäväksi valittuja toiminnallisuuksia muokataan vasta iteraation jälkeen. Näin toimitaan riittävän kurinalaisuuden ja mukautuvuuden tasapainon säilyttämiseksi kehitystyössä.

5 Yhteenveto

Ketterän kehittämisen eräinä keskeisinä piirteinä ovat inkrementaalisesti ja lyhyissä iteraatioissa tapahtuva ohjelmistonkehitys ja kehittämisen kyky mukautua muuttuviin vaatimuksiin (esim. Abbas ym., 2008). Ohjelmistoja ei ole kuitenkaan aina tehokasta toimittaa asiakkaille joka iteraation jälkeen, vaan asiakkaalle toimitettava julkaisu koostu tyypillisesti usean iteraation tuloksesta (Leffingwell, 2009). Jotta asiakkaiden vaatimukset ja ohjelmistontoimittajan tehokas resurssien käyttö pystytään sovittamaan yhteen, tarvitaan huolellista julkaisunsuunnittelua.

Tämän tutkimuksen tarkoituksena oli luoda kirjallisuuteen perustuen yleiskuva siitä, millaisia prosessimalleja julkaisunsuunnitteluun ketterän lähestymistavan mukaisessa kehittämisessä on olemassa ja miten julkaisunsuunnittelua voidaan tehdä ketterässä kehittämisessä saumattomasti yhdessä kehitystyön kanssa. Tutkimus oli rajattu koskemaan ketterien menetelmien osalta Scrum-menetelmää.

Tutkimuksen alussa kuvattiin lyhyesti ketterän lähestymistavan taustaa, pääpiirteitä ja Scrum-menetelmää. Scrumista esiteltiin sen syntyä, prosessia, rooleja ja työlistoja. Toiseksi tutkimuksessa määritettiin julkaisun ja julkaisunsuunnittelun käsitteet sekä kuvattiin viisi erilaista julkaisunsuunnitteluun tarkoitettua prosessimallia. Näistä yhtä, Bekkersin ym. (2010) mallia, käytettiin muiden mallien vertailun pääasiallisena perustana. Vertailun tarkoituksena oli selvittää, millaisia yhteisiä ominaispiirteitä tarkastelluilla malleilla on ja miten mallit ovat yhteensopivia Scrumin kanssa. Tarkastelluista malleista kolme (Cohn, 2006; Heikkilä, 2010; Heikkilä ym., 2010) sopii Scrumin yhteydessä käytettäväksi ja neljäs (Logue & McDaid, 2008) on yhteensopiva XP (Extreme programming) –menetelmän kanssa. Malleille on yhteistä julkaisun määrittelyn ja vaatimusten priorisoinnin aktiviteetit sekä se etteivät mallit ota kantaa julkaisujen vahvistamiseen ja jakeluun valmistautumiseen. Erilaiset arviointitehtävät olivat myös malleissa yleisiä. Yleisimpinä rooleina esiin nousivat tuotteen omistajat ja kehittäjät. Kolmanneksi tutkielmassa tarkasteltiin, miten julkaisunsuunnittelu liittyy projekti- ja tiimitasoiseen ohjelmistonkehittämiseen ja miten julkaisunsuunnittelua ja kehittämistä voidaan tehdä yhdessä saumattomasti. Tätä

tarkastelua varten kuvattiin kahta kokonaisvaltaista tuotehallintaa esittävää mallia; Leffingwellin (2009) mallia ja ATMAN-mallia (Vähäniitty, 2010a). Näitä vertailtiin keskenään käyttäen Bekkersin ym. (2010) mallia viitekehyksenä. ATMAN- malli toi esille, miten julkaisunsuunnittelu ohjaa iteraatiotason ja päivittäin tehtävän työn tason kehittämistä ja miten päivittäin tehdyssä työssä opitut asiat heijastuvat palautteena iteraatio- ja julkaisusuunnitteluun. Toisena yhteytenä havaittiin eri suunnittelutasojen vaatimusten välinen yhteys ja jäljitettävyys. Myös Leffingwellin malli (2009) toi esille, miten julkaisunsuunnittelu ohjaa iteraatiotason kehittämistä ja miten eri tasojen vaatimukset liittyvät toisiinsa. ATMAN-malli (Vähäniitty, 2010a) kuvasi kuitenkin yhteyksiä malleista monipuolisimmin. Julkaisunsuunnittelun ja kehitystyön välinen yhteys ei ole tutkielman havaintojen valossa täysin saumatonta. Ketterässä kehittämisessä suunnitelmia päivitetään säännöllisin väliajoin iteraatioiden välissä, ei kesken iteraatioissa tapahtuvan kehitystyön. Tällä pyritään siihen, että kehitystoiminta on mukautuvaa, mutta myös riittävän kurinalaista.

Tässä tutkimuksessa esitetyt tulokset voivat toimia lähtökohtana aihepiirin systemaattisemmalle tutkimukselle. Henkilöt, jotka tekevät työssään julkaisunsuunnittelun tehtäviä, voivat tutkielman avulla tutustua eri prosessimalleihin ja niiden eroihin sekä poimia malleja mukaan omaan työhönsä.

Mallien tarkastelu oli tutkielman luonteesta johtuen suppeaa eikä se ollut kovin syvällistä. Tarkasteluun olisi voitu ottaa myös muita malleja. Joitakin vertailun päätteeksi esitettyjä johtopäätöksiä on myös varaa tarkentaa.

Tutkimus tarjoaa useampia mahdollisuuksia jatkotutkimuksiin. Tarkastellut prosessimallit eivät ota kantaa siihen, miten julkaisujen jakeluun valmistaudutaan. Tämän havainnon pohjalta olisi mielenkiintoista selvittää, onko jakeluun valmistautumisessa erityispiirteitä, jotka on otettava huomioon erityisesti ketterän kehittämisen kontekstissa. Tutkielman kirjoittajan käsityksen mukaan ketterän kehittämisen mukaisessa kehitystyössä pyritään usein tiheään ohjelmistojen jakeluun, vaikka ohjelmistoja ei asetettaisikaan jakeluun jokaisen iteraation päätteeksi. Yksi tutkimusaihe voisi siis olla, miten julkaisujen jakeluun valmistaudutaan ketterässä kehittämisessä ja millä tavoin usein tapahtuvaa jakelua voidaan käyttäjien puolesta helpottaa.

Tutkielman pohjalta on myös mahdollista selvittää, millaisia tekniikoita tässäkin tutkielmassa läpikäytyjen julkaisunsuunnitteluprosessien vaiheiden tueksi on olemassa. Tekniikoista yhtenä esimerkkinä voidaan mainita suunnittelupokeri (planning poker). Tutkielman pohjalta voisi esimerkiksi tarkastella, millaisia arviointiin, vaatimuksien priorisointiin ja valitsemiseen liittyviä tekniikoita ketterän kehittämisen tueksi on kehitetty. Esimerkiksi Cohn (2006) on käsitellyt erilaisia tekniikoita, mutta tutkielman kirjoittaja ei ollut löytänyt tekniikoita jäsentävää kokonaisesitystä.

Kolmanneksi olisi mielenkiintoista laajentaa tutkielman näkökulmaa tuotehallinnan mallien muille osa-aluille, kuten siihen, miten vaatimustenhallinta jäsentyy ketterän lähestymistavan kehittämiseen. Tässäkin tutkielmassa käsitellyissä kahdessa ketterää tuotehallintaa jäsentävässä mallissa vaatimukset näyttivät olevan ikään kuin sisäänrakennettuja mallien joka tasolle.

LÄHTEET

Abbas, N., Gravell A. M. & Wills, G. B. (2008). Historical roots of agile methods: where did "agile thinking" come from? Teoksessa P. Abrahamsson, R. Baskerville, K. Conboy, B. Fitzgerald, L. Morgan & X. Wang (toim.), *Proceedings of the 9th International Confrence on Agile Processes in Software Engineering and Extreme Programming* (s. 94-103). Berlin: Springer-Verlag.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002). *Agile software development methods: review and analysis* (VTT:n julkaisuja 478). Helsinki: VTT. Haettu 9.2.2012 osoitteesta http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

Agile Manifesto (2001). *Agile manifesto*. Haettu 14.1.2012 osoitteesta http://agilemanifesto.org

Baskerville, R., Ramesh, B., Levine, L., Pries-Heje, J., & Slaughter, S. (2003). Is internet-speed software development different? *IEEE Software, 20*(6), 70-77.

Bekkers, W., Weerd, I., Spruit, M. & Brinkkemper, S. (2010). A framework for process improvement in software product management. Teoksessa A. Riel, R. O'Connor, S. Tichkiewitch & R. Messnarz (toim.), *Systems, software and Services Process Improvement* (s. 1-12). Berlin: Springer.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35*(1), 64-69.

Cohn, M. (2006). *Agile estimating and planning*, Upper Saddle River, NJ: Prentice Hall.

Dybå, T. & Dingsøyr, T. (2009). What do we know about agile software development? *IEEE Software, 26*(5), 6-9.

Heikkilä, V. (2010), Scaling up agile release planning. Teoksessa V. Heikkilä, K. Rautiainen & J. Vähäniitty (toim.), *Towards Agile Product and Portfolio Management* (s. 170-183). Espoo: Aalto University.

Heikkilä, V., Jadallah, A., Rautiainen, K. & Ruhe, G. (2010). Rigorous support for flexible planning of product releases - A stakeholder-centric approach and its initial evaluation. Teoksessa *Proceedings of the 43rd Hawaii International Conference on System Sciences (HICSS-2010)* (s. 1-10).

Highsmith, J. & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer, 34*(9), 120-127.

Larman, C. & Basili, V. (2003). Interactive and incremental development: a brief history. *IEEE Computer Society, 36*(6), 47-56.

Leffingwell, D. (2009). *The big picture of enterprise agility (rev. 2)*. Haettu 29.3.2012 osoitteesta http://scalingsoftwareagility.files.wordpress.com/2009/08/the-agile-enterprise-big-picture-2.pdf

Leffingwell, D. (2011). *Agile software requirements – Lean requirements practices for teams, programs and the enterprise*. Addisson-Wesley.

Logue, K. & McDaid, K. (2008). Agile release planning: dealing with uncertainty in development time and business value. Teoksessa *Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems. ECBS 2008* (s. 437-442).

McCauley, R. (2001). Agile development methods poised to upset status quo. *SIGCSE Bulletin 33* (4), 14-15.

Moløkken-Østvold, K., Haugen, N. C. & Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software, 81*(12), 2106-2117.

Ngo-The, A. & Ruhe, G. (2007). A systematic approach for solving the wicked problem of software release planning. *Soft Computing, 12*(1), 95-108.

Rautiainen, K., Lassenius, C., Vähäniitty, J., Itkonen, J., Mäntylä, M., Rusama, M. & Vanhanen, J. (2006), *Pacing software product development: a framework and practical implementation guidelines* (Teknillisen korkeakoulun ohjelmistoliiketoiminnan ja –tuotannon laboratorion tekninen raportti 3). Espoo: Teknillinen korkeakoulu, Tietotekniikan osasto. Haettu 28.3.2012 osoitteesta http://www.soberit.hut.fi/sems/pacingsoftwareproductdevelopment.pdf

ReleasePlanner, Haettu 27.2.2012 osoitteesta http://www.releaseplanner.com

Schwaber, K. (1995). SCRUM development process. Teoksessa *Proceedings of OOPSLA'95 Workshop on Business Object Design and Implementation*. Springer-Verlag.

Schwaber K. & Beedle M. (2002). *Agile software development with Scrum*, Upper Saddle River, NJ: Prentice Hall.

Shalloway, A., Beaver, G. & Trott, J. (2010). *Lean-Agile software development: achiving enterprise agility*. Adisson-Wesley.

Sharp, H., Finkelstein, A. & Galal, G. (1999). Stakeholder identification in the requirements engineering process. Teoksessa *Proceedings of the Tenth International Workshop on Database and Expert Systems Applications* (s. 387-391).

Sommerville, I. (2007). *Software engineering (8th edition),* Adisson-Wesley.

Takeuchi H. & Nonaka I. (1986). The new product development game. *Harvard Business Review* (January-February), 137-146.

VersionOne (2011). 6th Annual survey "The state of agile survey". Haettu 8.2.2012 osoitteesta http://www.versionone.com/state_of_agile_development_survey/11/

Vlaanderen, K. (2010). Software product management competence model. Haettu 12.3.2012 osoitteesta http://www.cs.uu.nl/wiki/bin/view/Spm/SpmCompetenceModel

Vlaanderen, K., Brinkkemper, S., Jansen, S., & Jaspers, E. (2009). The agile requirements refinery: applying SCRUM principles to software product management. Teoksessa *Proceedings of 3rd International Workshop on Software Product Management (IWSPM 2009)* (s. 1-10).

Vähäniitty, J. (2010a), Agile product management. Teoksessa V. Heikkilä, K. Rautiainen & J. Vähäniitty (toim.), *Towards Agile Product and Portfolio Management* (s. 115-125). Espoo: Aalto University.

Vähäniitty, J. (2010b), Portfolio management and agile software development. Teoksessa V. Heikkilä, K. Rautiainen & J. Vähäniitty (toim.), *Towards agile product and portfolio management* (s. 126-148). Espoo: Aalto University.

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J. & Bijlsma, L. (2006). Towards a reference framework for software product management. Teoksessa *Proceedings of the 14th IEEE International Conference on the Requirements Engineering* (s. 319-322).