Sami Peltola

JULKAISUNSUUNNITTELU KETTERÄSSÄ KEHITTÄMISESSÄ

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2013

TIIVISTELMÄ

Peltola, Sami
Julkaisunsuunnittelu ketterässä kehittämisessä
Jyväskylä: Jyväskylän yliopisto, 2013, 102 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Leppänen, Mauri

Ketterässä kehittämisessä ohjelmistoja kehitetään lyhyissä iteraatioissa. Tällä pyritään siihen, että muuttuvat vaatimukset pystytään joustavasti huomioimaan. Käyttäjille ohjelmistot jaetaan yhden tai useamman iteraation tuloksista koostuvana julkaisuna. Asiakkaiden tarpeiden ja toimittajan resurssien käytön yhteensovittamiseksi ohjelmistokehityksessä tarvitaan huolellista julkaisunsuunnittelua. Kirjallisuudessa on ehdotettu monenlaisia julkaisunsuunnittelun prosesseja, menetelmiä ja tekniikoita.

Tämän tutkimuksen tavoitteena on selvittää, millaista tukea löytyy kirjallisuudesta ketterään julkaisusuunnitteluun. Tätä varten tutkielmassa tarkastellaan ketterästä tuotehallinnasta tehtyjä viitekehyksiä ja julkaisunsuunnitteluun ehdotettuja prosesseja, menetelmiä ja tekniikoita sekä arvioidaan niiden soveltuvuutta ketterän ohjelmistokehityksen yhteyteen.

Tutkimuksessa kuvataan ja arvioidaan kahdeksaa julkaisunsuunnittelun prosessia ja menetelmää. Ne opastavat tekemään julkaisunsuunnittelua järjestelmällisesti, arviointiin perustuen tai hybridi-suunnitteluna. Yleisimmät aktiviteetit ovat vaatimusten priorisointi, julkaisun määrittely, laajuuden muutostenhallinta sekä koon ja työmäärän arviointi. Ehdotuksista kolme sopii Scrumin mukaiseen kehittämiseen ja neljä vaihtelevin rajoituksin. Kuudessa järjestelmällistä suunnittelua sisältävässä ehdotuksessa yleisimmät vaatimusten valintatekijät ovat arvotekijät, työmäärärajoitteet ja vaatimusten riippuvuudet. Tutkimuksessa tarkastellaan lisäksi kahta ketterää tuotehallintaa jäsentävää mallia. Ohjelmistokehitys on kiinteä osa malleja, ja vaatimustenhallinta on jakautunut mallien jokaiselle tasolle. Mallit osoittavat julkaisunsuunnittelun ja tiimi- ja projektimuotoisen kehittämisen välille kaksi yhteyttä, jotka ovat tasojen välinen ohjaus- ja palautesuhde sekä vaatimusten välinen yhteys. Lisäksi tutkimuksessa tarkastellaan viittä priorisointitekniikkaa ja kahta koon arviointitekniikkaa. Priorisointitekniikat ovat pääosin helppokäyttöisiä ja niistä kolme arvioitiin sopivan hyvin ketterään kehittämiseen.

Avainsanat: julkaisu, julkaisunsuunnittelu, ketterät menetelmät, Scrum, ohjelmistotuotehallinta

ABSTRACT

Peltola, Sami
Release Planning in Agile Software Development
Jyväskylä: University of Jyväskylä, 2013, 102p.
Information Systems, Master's Thesis
Supervisor: Mauri Leppänen

Agile software development is accomplished in short iterations. This enables flexible reactions to changes in user requirements. Software is delivered to customers in releases each of which combines the outcomes of one or more iterations. To reconcile the client's needs and the supplier's resources use, careful release planning is needed. The literature provides many kinds of processes, methods and techniques to support release planning.

The purpose of this study is to find out which kind of support to agile release planning can be found in the literature. For this purpose, the thesis considers frameworks of agile software product management, and processes, methods and techniques of release planning, as well as assesses their suitability to agile software development.

The thesis describes and compares eight release planning processes and methods suggested in the literature. They guide to conduct release planning in a systematic, judgement-based or mixed manner. The most common activities are requirement prioritization, release definition, scope change management and size/effort estimation. Three of the suggestions are compatible with Scrum method and four are compatible with variable restrictions. In six systematic and hybrid suggestions the most common requirement selection factors are value, effort and requirements dependencies. In addition, the study describes and compares two frameworks of agile software product management. The development level is an integral part of the frameworks and requirement management is divided on every level of these frameworks. The frameworks show 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. The study also describes five prioritization techniques and two size estimation techniques. The prioritization techniques are mostly easy to use and three of them are judged to be suitable for agile development

Keywords: release, release planning, agile methods, Scrum, software product management

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Ketterässä kehittämisessä ohjelmistoja tuotetaan lyhyissä, tyypillisesti 2–4 viikon pituisissa sykleissä, joita kutsutaan iteraatioiksi. Tavoitteena on tuottaa käyttäjille aikaisessa vaiheessa toimitetulla ohjelmistolla arvoa ja samalla taata palautteen vastaanottaminen. Lyhyissä iteraatioissa tapahtuva ohjelmistonkehitys mahdollistaa sen, että kehitettävän ohjelmiston suuntaa pystytään joustavasti muuttamaa käyttäjien muuttuvien tai tarkentuvien tarpeiden mukaan. (esim. Agile Manifesto, 2001.)

Joustavasti lyhyissä jaksoissa tehtävän kehittämisen lisäksi ohjelmistonkehityksessä tarvitaan pidemmän aikavälin tavoitteita. Ne mahdollistavat ohjelmistoprojektien tehokkaan resursoinnin ja antavat ohjelmiston kehittäjille suunnan, jota kohti ohjelmistoa kehitetään. Ohjelmistoa ei aina pystytä kehittämään tyydyttäväksi kokonaisuudeksi yhdessä iteraatiossa ja asiakkaat eivät välttämättä kykene tai halua ottaa ohjelmistoa näin säännöllisesti vastaan. Tällöinkin tarvitaan suunnitelma ohjelmiston julkaisuajasta ja sen sisältämistä ominaisuuksista, sillä asiakkaiden ja kehittäjien lisäksi eri sidosryhmät tarvitsevat tietoa toiminnassaan. (esim. Cohn, 2006; Leffingwell, 2009.)

Suunnitelmallisuus on erityisen tärkeää, kun kehitetään ohjelmistotuotetta. Tuotteella ei ole vain yhtä tilaajaa, joka määrittelee sen ohjelmiston ominaisuudet vaan kehitysorganisaation on suunniteltava tuotteen ominaisuudet markkinoiden vaatimusten perusteella ja päätettävä otollinen aika ohjelmiston julkaisuun. (Xu & Brinkkemper, 2007).

Edellä kuvattujen tavoitteiden asettamiseen ja suunnitelmien laatimiseen tarvitaan huolellista julkaisunsuunnittelua. Julkaisunsuunnittelu (release planning) on määritelty mm. ”prosessiksi, jossa syntyy korkean tason suunnitelma, josta tunnistetaan, milloin julkaisu valmistuu ja mitkä toiminnallisuudet siihen sisältyvät” (Logue & McDaid, 2008a, 437), tai ”prosessiksi tuotteen vision siirtämiseksi tuotteen työlistaan” (Shalloway, Beaver & Trott, 2010, 11). Cohn (2006) kuvaa julkaisunsuunnittelua prosessiksi, jossa luodaan korkean tason suunnitelma. Suunnitelma kattaa yhtä iteraatiota pidemmän ajanjakson ja tarjoaa kontekstin, jonka puitteissa iteraatiot on mahdollista rakentaa tyydyttäväksi kokonaisuudeksi (Cohn, 2006, 133). Myös julkaisukäsitteelle on useita määri-

telmiä. Sillä voidaan tarkoittaa aiemmista ohjelmistoversioista poikkeavaa versiota, joka tuodaan käyttäjien saataville (Sommerville, 2007, 699) tai myös kehittämisorganisaation sisäistä versiota, jota ei jaeta suoraan loppukäyttäjille (esim. Leffingwell, 2011).

Tässä tutkimuksessa *julkaisunsuunnittelu* nähdään Cohnin (2006) ja Loguen ja McDaidin (2008a) määritelmien mukaisella tavalla prosessina, jonka lopputuloksena syntyy korkean tason suunnitelma, joka viitoittaa iteraatioiden suunnittelua. Syntyneestä suunnitelmasta ilmenee arvioitu julkaisun valmistumisajankohta ja sisältö. *Julkaisulla* tarkoitetaan ohjelmakokonaisuutta, joka koostuu yhden tai useamman iteraation tuloksesta, ja se voi olla joko sisäinen tai ulkoinen.

Julkaisunsuunnittelu on hyvin haastava tehtävä erityisesti laajemmille markkinoille tehtävässä tuotekehityksessä. On vaikeaa yhdistää lukuisten vaatimusten joukosta mahdollisimman aikaisin markkinoille tuotava, määrätyn asiakasjoukon tarpeet tyydyttävä kokonaisuus ja ottaa samalla huomioon erilaiset kehittämisen rajoitteet (esim. Jantunen, Lehtola, Gause, Dumdum & Barnes, 2011). Ketterän julkaisunsuunnittelun (agile release planning) tueksi on kehitetty erilaisia ehdotuksia, joita on kutsuttu prosesseiksi (esim. Cohn, 2006), malleiksi (esim. Szoke, 2011b), menetelmiksi (esim. Li, Huang, Shu & Li, 2006) ja tekniikoiksi (esim. Grenning, 2002). Näissä on usein hyödynnetty perinteisen julkaisunsuunnittelun käytäntöjä. Ketterän kehittämisen yhdistämiseksi osaksi laajempaa tuotehallinnan kenttää on kehitetty myös erilaisia viitekehyksiä (esim. Leffingwell, 2011; Vähäniitty, 2010b). Näissä malleissa julkaisunsuunnittelu on osana.

Julkaisunsuunnittelua on tutkittu varsin vähän ketterän kehittämisen kontekstista. Esimerkiksi Vähäniitty (2010b, 116) pitää tärkeänä, että ymmärrettäisiin paremmin, miten julkaisunsuunnittelu ja pidemmän aikavälin suunnittelu ilmenevät ketterässä kehittämisessä. Perinteisen julkaisunsuunnittelun ja ketterän lähestymistavan käytäntöjen ei ole myöskään nähty kaikilta osin sopivan yhteen (esim. Heikkilä, 2010, 172).

Ymmärryksen lisäämiseksi aiheesta on hyödyllistä luoda kokonaisesitys ketterään lähestymistapaan ehdotettujen julkaisunsuunnittelun prosessien, menetelmien ja tekniikoiden piirteistä. Tällaista kokonaisesitystä ei tiettävästi ole tehty. Aihepiiriä sisältyviä osa-alueita on sen sijaan tutkittu. Esimerkiksi Racheva, Daneva ja Buglione (2008) ovat tutkineet ketterään kehittämiseen sopivia priorisointitekniikoita. Svahnberg, Gorschek, Feldt, Torkar, Saleem ja Shafique (2010) ovat vertailleet julkaisunsuunnitteluun ehdotettuja formaaleja suunnittelumalleja, mutta tätä tarkastelua ei ole tehty ketterän lähestymistavan näkökulmasta.

Tutkimuksen tavoitteena on muodostaa kokonaiskuva ketterän julkaisunsuunnittelun tueksi kehitetyistä prosesseista, menetelmistä ja tekniikoista sekä siitä, miten ketterä julkaisunsuunnittelu liittyy laajempaan tuotehallinnan kokonaisuuteen. Tutkimusongelma voidaan kiteyttää siihen, millaista tukea löytyy kirjallisuudesta ketterään julkaisunsuunnitteluun. Tutkimusongelma on jaettu seuraaviin tutkimuskysymyksiin:

- Mitä tarkoitetaan tuotehallinnalla ja millainen on ketterä tuotehallinta?
- Mitä tarkoitetaan julkaisunsuunnittelulla, millaisia julkaisunsuunnittelun prosesseja, menetelmiä ja tekniikoita on olemassa ja mitkä näistä soveltuvat ketterän ohjelmistokehityksen yhteyteen?
- Miten julkaisunsuunnittelu on yhteydessä ketterään tiimi- tai projektimuotoiseen ohjelmistonkehitykseen?

Tutkimus on luonteeltaan käsitteellis-teoreettinen, ja se perustuu alan kirjallisuuteen. Tutkimuksen keskeisenä osana on alan kirjallisuudesta löydettyjen, ketterän julkaisunsuunnittelun tueksi esitettyjen ehdotusten vertailu. Kirjallisuushaut on kuvattu liitteessä 1. Kirjallisuushaulla pyrittiin tunnistamaan tieteellisistä tietokannoista mahdollisimman kattavasti ketterään julkaisunsuunnitteluun ja tuotehallintaan ehdotettuja ehdotuksia ja toiseksi keräämään muuta taustamateriaalia tutkimuksen aihepiiristä. Tutkimuksen yhtenä keskeisenä vertailun viitekehyksenä käytetään Bekkersin, van de Weerdin, Spruitin ja Brinkkemperin (2010) tuotehallinnan viitekehystä.

Tutkimus rajataan koskemaan ketteristä menetelmistä Scrum-menetelmää. Vaikka tutkimuksessa tarkastellaan julkaisunsuunnittelua tuotehallinnan osana, tutkimuksessa ei ole rajauduttu käsittelemään vain laajoille markkinoille tarkoitettujen ohjelmistotuotteiden kehitykseen sopivia ehdotuksia.

Tutkimus on jäsennetty kahdeksaan lukuun. Luvussa 2 kuvataan ketterän ohjelmistokehityksen taustaa, pääpiirteitä ja erästä ketterää menetelmää, Scrumia. Luvussa 3 tarkastellaan ohjelmistotuotehallintaa ja tuotehallinnan viitekehyksiä sekä perustellaan tutkimuksessa keskeinen viitekehyksen valinta. Luvussa 4 tuotehallintaa ja sen tutkimusta tarkastellaan ketterän lähestymistavan näkökulmasta. Luvussa esitellään vertailujen perustana toimiva viitekehys ja kaksi ketterää tuotehallintaa jäsentävää mallia. Luvussa 5 kuvataan ensin julkaisunsuunnittelua yleisesti ja sen jälkeen ketterän kehittämisen näkökulmasta erityisesti. Luvussa kuvataan kahdeksan julkaisunsuunnittelun prosessia ja menetelmää ja verrataan niitä keskenään. Luvussa 6 määritellään julkaisunsuunnittelun tekniikka -käsite ja annetaan esimerkkejä tekniikoista. Lähemmin tarkastellaan priorisointi- ja vaatimusten koon arviointitekniikoita. Luvussa 7 kootaan edeltävien lukujen tulokset ketterää julkaisunsuunnittelua koskevaksi kokonaiskuvaksi. Tutkimuksen viimeisessä luvussa tiivistetään tutkimuksen tulokset, kerrotaan niiden hyödyntämismahdollisuuksista, tarkastellaan niitä kriittisesti sekä esitetään jatkotutkimusaiheita.

2 KETTERÄ LÄHESTYMISTAPA

Tässä luvussa esitellään ensin 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 vaatimusten 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 yhteisten näkemysten löytämiseksi. Kokousta voidaan pitää onnistuneena, sillä osallistu-

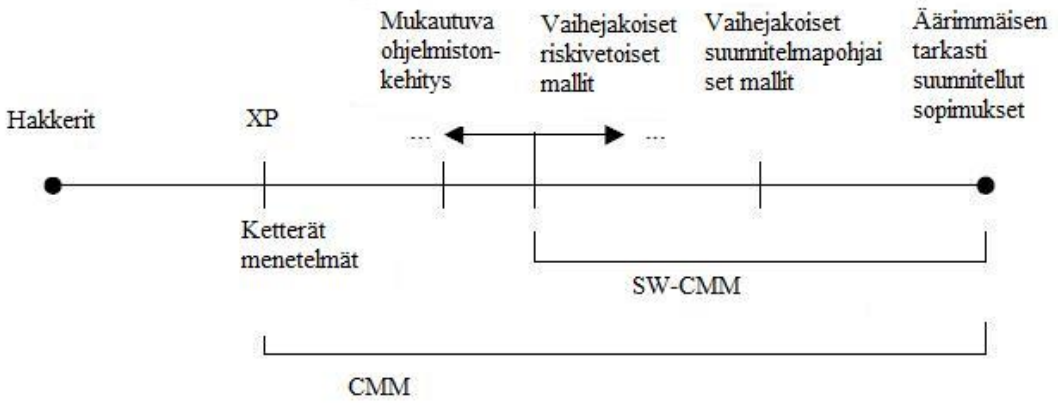
jat laativat ja allekirjoittivat yhteisen ketterän julistuksen (agile-manifest), jossa määriteltiin ketterän kehityksen neljä arvoa ja 12 periaatetta. Julistuksessa kevyet menetelmät saivat nimen ketterät menetelmät (Agile Manifesto, 2001).

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 ym., 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 myös 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 vaatimusten, 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 vaikuttavina 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 & Cockburn, 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 & Cockburn, 2001).

Ketteriä menetelmiä on useita, esimerkiksi Scrum (Schwaber & Sutherland, 2011), Extreme programming (XP) (Beck, 1999) ja Kanban (Anderson, 2010). 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 & Nonaka, 1986). Scrum-menetelmä syntyi varsinaisesti vuonna 1994, kun Jeff Sutherland ryhtyi soveltamaan Takeuchin ja Nonakan ajatuksia ohjelmiston kehitykseen (Schwaber & Beedle, 2002, 11). Myöhemmin Ken Schwaber liittyi mukaan kehittämään menetelmää (Schwaber & Beedle, 2002, 11).

Vlaanderenin, Jansenin, Brinkkemperin ja Jaspersin (2011, 59) 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. 2011, 59.)

Schwaber ja Sutherland (2011), jotka ylläpitävät Scrumin virallista sääntökirjaa *the Scrum Guidea*, kutsuvat Scrumia ohjelmistonkehitystä tukevaksi viitekehykseksi. Kehys koostuu rooleista, tapahtumista, tuloksista ja säännöistä. Heidän mukaansa viitekehys tarjoaa puitteet, joiden sisällä ohjelmistoa voidaan kehittää erilaisia tekniikoita ja menetelmiä hyväksikäyttäen. Edellisestä Vlaanderenin ym. (2011) kuvauksesta poiketen, he eivät pidä Scrumia ohjelmistokehitysprosessina tai tekniikkana. (Schwaber & Sutherland, 2011.)

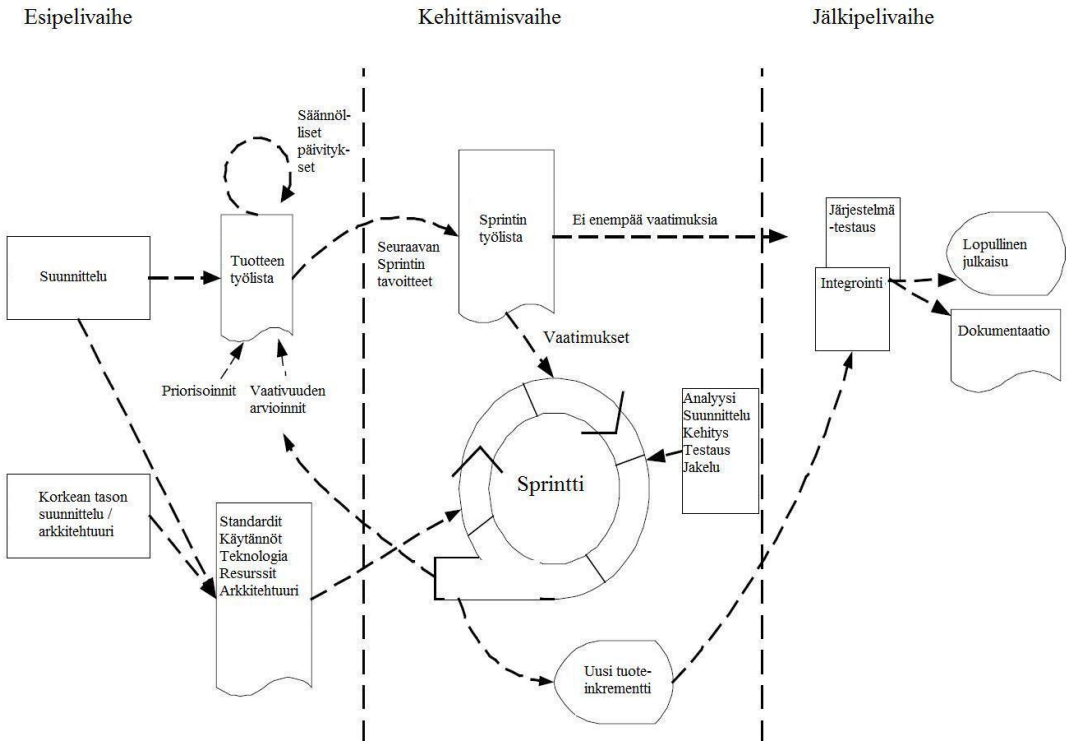
Seuraavaksi esitellään Scrumia siten, kun se on esitetty eräissä esityksissä (Schwaber, 1995; Abrahamsson, Salo, Ronkainen & Warsta, 2002) prosessina, sillä prosessikuvaus havainnollistaa Scrumin rakennetta. Tämän jälkeen esitellään Scrumin rooleja, tapahtumia ja tuotoksia. Scrumin sääntöjen kuvaaminen sisältyy näihin alalukuihin (Schwaber & Sutherland, 2011).

2.3.1 Prosessi

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

Kehittämisvaiheessa (eli pelivaiheessa, game) 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) jä-

senet 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 (postgame) 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 Scrum-tiimi

Scrum-tiimi kostuu kehitystiimistä, Scrum-mestarista ja tuotteen omistajasta (Schwaber & Sutherland, 2011). Sprintin aikana varsinaisesta kehitystyöstä vastaa *kehitystiimi,* joka kehittää sprinttiin valituista toiminnallisuuksista valmiin ohjelmistokokonaisuuden. Tiimillä on vapaus päättää, millä tavoin tuote kehitetään. Suositeltu Scrum-tiimin koko on 5–9 henkilöä. (Schwaber & Beedle, 2002, 36.)

Kehitystiimin työtä tukee *Scrum-mestari*, joka huolehtii Scrumin periaatteiden ja arvojen toteutumisesta tiimin työssä. Hän muiden muassa tukee kehitystiimiä raivaamalla pois tiimin työtä haittaavia esteitä (Schwaber & Beedle, 2002, 31). Scrum-mestari vastaa myös kehitystoiminnan prosessista ja sen kehit-

tämisestä (Cohn, 2009, 117). Pichler (2010, 9) kiteyttää Scrum-mestarin tehtävän koskevan sitä, *miten* Scrumia käytetään oikealla tavalla.

Tuotteen omistaja (product owner) on vastuussa ohjelmistoprojektin onnistumisesta ja tuotteen työlistan ylläpitämisestä. Hänellä on lopullinen päätösvalta työlistan sisällöstä ja priorisoinnista. (Schwaber & Beedle, 2002, 34.) Pichlerin (2010, 9) mukaan tuotteen omistaja vastaa siitä *mitä* kehitetään.

2.3.3 Tapahtumia

Scrum sisältää erilaisia tapahtumia, jotka ovat sprintin suunnittelupalaveri, päiväpalaveri, sprinttikatselmus ja retrospektiivipalaveri. Kaksiosainen *sprintin suunnittelupalaveri* järjestetään jokaisen sprintin alussa. Se saa kestää enintään kahdeksan tuntia kuukauden pituisessa sprintissä. Palaverin ensimmäisessä osassa päätetään, mitä toiminnallisuuksia tulevassa sprintissä tullaan kehittämään. Kehitystiimin tehtävänä on antaa palaverissa ennuste tiimin kehitysvauhdista. Muut tahot eivät voi vauhtia määritellä (Schwaber & Sutherland, 2011, 8). Tuotteen omistaja tuo palaveriin priorisoidun tuotteen työlistan, josta Scrum-tiimi yhdessä poimii kehitettäviä toiminnallisuuksia seuraavaan sprinttiin. Vaiheen aikana sprinteille määritellään myös konkreettinen tavoite. Palaverin toisessa osassa kehitystiimi suunnittelee, miten valitut toiminnallisuudet toteutetaan. Tuotteen omistaja voi olla mukana palaverissa antamassa vaatimuksista lisätietoa. Toisen vaiheen tuloksena tulisi olla käsitys siitä, miten tiimi aikoo työskennellä tavoitteiden saavuttamiseksi.

Päiväpalaveri kestää enintään 15 minuuttia ja sen aikana jokainen kehitystiimin jäsen kertoo, mitä on saanut edellisenä päivänä aikaan, mitä aikoo tehdä seuraavaksi ja millaisia hankaluuksia hän on havainnut. Päiväpalaverin avulla tiimi pystyy paremmin järjestämään keskinäistä työskentelyään tavoitteiden saavuttamiseksi. *Sprintin katselmuksessa* Scrum-tiimi ja eri sidosryhmän edustajat katselmoivat sprintin tulosta ja keskustelevat siitä. Tarkoituksena on saada näin palautetta, jota voidaan hyödyntää ohjelmiston tulevassa kehittämisessä. Kuukauden pituisissa sprinteissä palaverin enimmäispituus on neljä tuntia. Palaverin aikana kehitystiimi esittelee valmista tuotetta ja kertoo, millaisia onnistumisia ja ongelmia sprintin aikana kohdattiin sekä miten ongelmat ratkaistiin. Tuotteen omistajan tehtävä on hyväksyä, mikä ohjelmiston osa on valmis ja mikä ei, sekä antaa katselmoinnin perusteella jokin arvio siitä, milloin ohjelma todennäköisesti on valmis. *Sprintin retrospektiivi* järjestetään katselmuksen jälkeen. Sen aikana Scrum-tiimi tarkastelee työskentelytapojaan ja etsii asioita, joita se voi tehdä paremmin tulevissa sprinteissä. Myös onnistuneet asiat nostetaan esiin. Pituudeltaan palaveri on maksimissaan kolme tuntia kuukauden pituisessa sprintissä. Palaverissa laaditaan suunnitelma havaittujen kehittämiskohteiden parantamiseksi. (Schwaber & Sutherland, 2011, 8–11.)

2.3.4 Tuotokset

Scrum sisältää erilaisia tuotoksia, jotka voidaan jaotella työlistoiksi ja tuoteversioksi (Schwaber & Sutherland, 2011) Työlistoja 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 & Cockburn, 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). On huomioitava, että Schwaber ja Sutherland (2011) eivät sisällytä julkaisun työlistaa Scrumin ydinalueeseen eikä Scrum guide ota kantaa tuotteen työlistan kohtien muotoon. *Sprintin työlista* (Sprint backlog) koostuu sprinttiin kehitettäviksi valituista toiminnallisuuksista. Se luodaan sprintin suunnittelupalaverissa Scrum-mestarin, tuotteen omistajan ja kehitystiimin yhteistyönä. 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).

Schwaberin ja Sutherlandin (2011, 13) mukaan tuoteversio (increment) koostuu kaikkien aiempien sprinttien tuotteen työlistan kohdista, jotka ovat siihen mennessä valmistuneet. Jokaisen sprintin tavoitteena on saada aikaiseksi uusi tuoteversio, joka on Scrum-tiimin yhteisesti sopimien kriteerien mukainen.

2.4 Yhteenveto

Tässä luvussa esiteltiin ketterän kehittämisen taustaa ja sen keskeisiä piirteitä. Ketterän lähestymistavan synty voidaan nähdä reaktiona aiempien kehitysmenetelmien ongelmiin mukautua nopeisiin muutoksiin. Sen arvot korostavat esimerkiksi valmiin ohjelmakoodin tärkeyttä yli kattavan dokumentaation. Lähestymistavan keskeisinä piirteinä korostuvat myös iteratiivinen ja inkrementaalinen ohjelmistokehitys, mukautuvuus ja ihmisten välisen vuorovaikutuksen tärkeys. Toiseksi luvussa esiteltiin eräs ketterän lähestymistavan menetelmä, Scrum. Scrum on ohjelmistonkehitykseen tarkoitettu viitekehys, jonka sisällä ohjelmistoja kehitetään lyhyissä iteraatioissa eli sprinteissä. Scrumin määritelmä koostuu rooleista, tapahtumista, tuloksista ja säännöistä. Rooleja ovat Scrum-tiimin työtä tukeva Scrum-mestari, tuotteen omistaja, joka vastaa siitä

mitä kehitetään ja missä järjestyksessä, ja kehitystiimi, joka vastaa itseohjautuvasti ohjelmiston kehityksestä. Tapahtumia ovat sprintin suunnittelupalaveri, päiväpalaveri, sprintin katsaus ja sprintin retrospektiivi. Palaverit tarjoavat mahdollisuuksia tehostaa ja suunnitella tulevaa työtä sekä Scrum-tiimin yhteistoimintaa. Sprintin tuloksia ovat erilaiset työjonot, joissa säilytetään kehitettäviä toiminnallisuuksia ja jokaisen sprintin päätteeksi valmistuu uusi tuoteversio.

3 TUOTEHALLINTA

Tässä luvussa esitellään ensin, mitä tarkoitetaan ohjelmistotuotteella. Tässä yhteydessä tarkastellaan myös, miten ohjelmistotuotteen ja räätälöidyn ohjelmiston kehittäminen eroavat toisistaan. Toiseksi luvussa kerrotaan, mitä tarkoitetaan tuotehallinnalla ja millainen on tuotehallintaan liittyvä tuotepäällikön rooli. Tuotehallinnan eri tehtäväalueiden kokonaiskuvan hahmottamiseksi luvun lopussa esitellään erilaisia tuotehallinnan viitekehyksiä, joista yksityiskohtaisimmin kuvataan SPMBoK-viitekehystä (ISPMA, 2012). Viitekehyksistä Bekkersin ym. (2010) esitys valitaan jatkossa käsiteltävien mallien vertailupohjaksi.

3.1 Ohjelmistotuote

Ohjelmistonkehityksen historiassa ohjelmistoja on alun alkaen kehitetty räätälöidysti (tailor-made) yksittäisten tilaajien tai omiin tarpeisiin, mutta jo melko varhain 1960-luvulla kehitettiin ensimmäiset ohjelmistotuotteet (software product) (Xu & Brinkkemper, 2007, 531). Viime vuosikymmeninä ohjelmistoyritykset ovat pyrkineet yhä enemmissä määrin kehittämään räätälöityjen tuotteiden sijaan laajalle asiakasjoukolle tarjottavia tuotteita (esim. Wohlin & Aurum, 2005, 319; Artz, van de Weerd, Brinkkemper & Fieggen, 2010, 90). Aineettomana hyödykkeenä ohjelmistoa on helppo monistaa ja myydä useille tahoille, mikä tuo yrityksille mahdollisuuden saavuttaa parempia voittoja (esim. Fricker, 2012, 55, Xu & Brinkkemper, 2007, 532–533).

Ohjelmistotuotteen käsitteellä voidaan tarkoittaa hyvin erilaisia ohjelmistoja, esimerkiksi kuluttajille tarjottavia ohjelmistopaketteja, ohjelmistoa palveluna ja yrityksille suunnattuja laajoja tietojärjestelmätuotteita. Kaikille ohjelmistotuotteille on kuitenkin yhteistä se, että niitä myydään yhden tilaajan sijasta useille asiakkaille (Xu & Brinkkemper, 2007, 534). Xun ja Brinkkemperin (2007, 534) tavoin *ohjelmistotuote* määritellään tässä työssä ”paketoiduksi ohjelmistokomponenttien muodostamaksi kokoonpanoksi tai ohjelmistoperäiseksi palve-

luksi oheismateriaaleineen, joka julkaistaan ja asetetaan myyntiin tietyille markkinoille".

Ohjelmistotuotteen ja räätälöidyn ohjelmiston kehittäminen eroavat monin tavoin toisistaan. Räätälöity ohjelmisto kehitetään ohjelmiston tilaajan määrittelemien tarpeiden mukaan, jolloin vaatimusten pääasiallinen lähde on tiedossa. Kun räätälöity ohjelmisto on kehitetty tilaajan toiveiden mukaan, ohjelmistoa voidaan pitää valmiina ja kehityksestä voidaan katsoa siirryttävän ylläpitovaiheeseen. Ohjelmistotuotteen vaatimusten määrittely eroaa räätälöidystä kehittämisestä siten, että tuotteessa vaatimusten keräämisessä joudutaan huomioimaan laajemmin markkinoiden ja useiden eri sidosryhmien tarpeet. Tuotteille on myös vaikeampi määritellä tarkkaa hetkeä, jolloin ohjelmisto on lopullisesti valmis. Markkinoilla olevia ohjelmistotuotteita on tarve kehittää jatkuvalla tahdilla, sillä käyttäjien tarpeet ovat jatkuvassa muutoksessa ja kilpailijat kehittävät rinnalla omia ratkaisuja näihin tarpeisiin (Fricker, 2012, 62). Markkinoilla kilpaileminen tuo yrityksille myös painetta julkaista tuote mahdollisimman aikaisessa vaiheessa. Ohjelmistotuotteita julkaistaankin markkinoille tavallisesti useina peräkkäisinä, ominaisuuksiltaan ja julkaisuajankohdiltaan tarkoin harkittuina, julkaisuina, jotka kukin tuovat ohjelmistoon uusia ominaisuuksia. Raja ylläpidon ja kehittämisen välillä ei tuotteen kohdalla ole välttämättä selväpiirteinen vaan näitä toimintoja voidaan tehdä rinnakkain. Xu ja Brinkkemper (2007, 536–537) korostavat myös ohjelmistoarkkitehtuurin suurempaa merkitystä ohjelmistotuotteille verrattuna räätälöityyn kehittämiseen, sillä ohjelmistotuotteen arkkitehtuurin tulee mahdollistaa ohjelmiston tehokas mukauttaminen ja toisaalta esimerkiksi ohjelmiston ydinosien uudelleenkäytettävyys tulevaisuudessa. (Xu & Brinkkemper, 2007, 536–536.)

Ohjelmistotuotteiden ja räätälöidyn kehittämisen välinen raja voi olla kuitenkin häilyvä. Ohjelmistotuotteet voidaan kehittää usein räätälöityjen ohjelmistoprojektien pohjalta ja laajojen ohjelmistotuotteiden, kuten toiminnanohjausjärjestelmien, käyttöönotto vaatii usein paljon yrityskohtaista räätälöintiä. (Xu & Brinkkemper, 2007, 537.)

3.2 Tuotehallinta yleisesti

Ohjelmistotuotehallintaa (jatkossa vain tuotehallinta) voidaan luonnehtia toimialaksi, joka yhdistää yritysten liiketoiminnan ja ohjelmistotuotannon toisiinsa (Fricker, 2012, 54). Ebert (2006, 850) on määritellyt tuotehallinnan tehtäväksi *ohjelmistotuotteiden hallinnan niiden koko elinkaaren ajan niin, että yrityksen saama liiketoimintahyöty on mahdollisimman suuri*. Tuotehallinta voidaan nähdä myös tuotepäällikön tehtäväalueena, jossa yksittäinen henkilö on vastuussa tietystä tuotteesta tai tuoteryhmästä (Ebert, 2006, 850). Tuotepäällikkö toimii yrityksessä eräänlaisena tuotteesta vastaavana "minitoimitusjohtajana", joka määrittelee tuotteiden strategian niin, että tuote palvelee parhaalla mahdollisella tavalla yrityksen strategiaa ja lukuisten eri sidosryhmien tarpeita (Fricker, 2012, 59). Tuotepäällikkö huolehtii tuotteen strategian toteutuksesta koordinoimalla eri

toimijoiden kuten kehitys-, markkinointi-, myynti- ja jakeluyksiköiden työtä (Fricker, 2012, 59). Tuotepäällikön päähuomio on näin tuotteen, ei yksittäisten projektien, menestymisen tasolla. (Ebert, 2006, 851).

Tuotehallinnan käytännöistä on nähty saavutettavan useita hyötyjä. Maglyas, Nikula ja Smolander (2012, 40) viittaavat Kittlausiin ja Cloughiin (2009) todetessaan, että ”tuotehallinnalla on positiivinen vaikutus ohjelmiston laatuun, tuottavuuteen, ennustettavuuteen sillä on kriittinen rooli liiketoimintatavoitteiden hallinnassa ja saavuttamisessa”. Maglyas ym. (2012, 40) viittaavat myös Ebertin (2006) empiiriseen tutkimukseen, joissa oli tutkittu tuotehallinnan käytäntöjen vaikutuksia 178 yrityksessä. Ebertin (2006, 860) tutkimuksessa oli havaittu, että tuotehallinnan käytännöillä tuotteen markkinoilletuontiaikaa oli saatu vähennettyä 36 prosenttia ja tuotteen laatu ja aikataulussa pysymisen oli parantunut 80 prosenttia tutkimuksen lähtötilanteeseen verrattuna.

3.3 Tuotehallinnan viitekehyksiä

Tuotehallintaa kuvaamaan on luotu erilaisia viitekehyksiä. Näitä viitekehyksiä tarkastelemalla tuotehallinnan tehtävistä voidaan muodostaa parempi kokonaiskäsitys. Fricker (2012, 63) on esitellyt tuotehallintaa käsittelevässä artikkelissaan useita yleisiä viitekehyksiä, joista on poimittu esimerkeiksi tässä tarkasteltaviksi seuraavien lähteiden viitekehykset:

- van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal ja Bijlsma (2006)
- Bekkers, van de Weerd, Spruit ja Brinkkemper (2010)
- Ebert (2006)
- Kittlaus ja Clough (2009)
- SPMBoK (ISPMA, 2012)

Mainittuja viitekehyksiä kuvataan tässä hyvin yleisellä tasolla lukuun ottamatta SPMBoK-viitekehystä (ISPMA, 2012), jota esitellään muita yksityiskohtaisemmin riittävän yleiskuvan saamiseksi tuotehallinnan tehtäväalueista.

van de Weerdin ym. (2006) *ja Bekkersin ym.* (2010) viitekehykset kuvaavat kumpikin tuotehallinnan neljää ydinaluetta, ydinalueisiin sisältyviä osa-alueita ja tuotehallinnan sidosryhmiä. Vähäniitty (2010a, 32) katsoo näiden viitekehyksien kuvaavan tuotehallinnan tuotannollisia puolia (engineering aspects). Useissa lähteissä (esim. Fricker, 2012, 63; Vähäniitty, 2010a, 39) esityksiä kuvataan rinnakkain ikään kuin saman mallin eri versiona. Viitekehykset eroavat hieman toisistaan. Bekkersin ym. esityksessä viitekehyksen ydinalueet ovat portfolionhallinta, tuotesuunnittelu, julkaisunsuunnittelu ja vaatimustenhallinta. Bekkersin ym. (2010) esityksestä voidaan käyttää myös nimeä kompetenssimalli. (van de Weerd ym., 2006; Bekkers ym. 2010.)

Ebertin (2006; 2009) *viitekehys* tarkastelee tuotehallintaa tuotteen elinkaaren näkökulmasta. Viitekehyksessä esitetyt tuotteen elinkaaren vaiheet ovat strate-

ginen suunnittelu, konseptointi, markkinoille saapuminen, kehittäminen ja evoluutio. Viitekehys sisältää yhteensä 18 tuotehallinnan prosessia ja kymmenen osaamisaluetta, jotka jaottuvat elinkaaren eri vaiheisiin. (Ebert, 2009, 18.)

Kittlausin ja Cloughin (2009) *mallissa* on horisontaalisesti kuvattu kahdeksan tuotehallinnan toiminnallista aluetta, jotka ovat markkina-analyysi, tuoteanalyysi, tuotteen strategia, tuotesuunnittelu, kehittäminen, markkinointi, myynti ja jakelu sekä palvelu ja tuki. Toiminnallisiin alueisiin liittyy 49 aktiviteettia. Nämä aktiviteetit jaottuvat mallissa vertikaalisesti joko yhtiö- tai tuotetasoilla toteutettaviksi. (Fricker, 2012, 64.)

Kansainvälinen ohjelmistotuotehallintajärjestö International Software Product Management Association (ISPMA) on kehittänyt *SPMBoK-viitekehyksen* (Software Product Management Body of Knowledge) tuotehallinnan koulutustarpeita varten. Se on laadittu Ebertin (2006), van de Weerdin ym. (2006) ja Kittlausin ja Cloughin (2009) mallien pohjalta. Viitekehys koostuu seitsemästä toiminnallisesta alueesta: strateginen johtaminen, tuotestrategia, tuotesuunnittelu, kehittäminen, markkinointi, myynti ja jakelu sekä palvelu ja tuki. Toiminnalliset alueet jakautuvat kolmeen laajempaan alueeseen sen mukaisesti, millainen vastuu tuotepäälliköllä on alueista. Alueet ovat osallistumisen alue, tuotehallinnan ydinalue ja alue, jota tuotehallinta johtaa. (Fricker, 2012, 64–65.) Taulukossa 1 seitsemää toiminnallista aluetta on esitetty omilla sarakkeillaan ja kolmea tuotehallinnan vastuualuetta omilla väreillään. Taulukon ensimmäisellä rivillä on mainittu toiminnallisten alueiden ja viimeisillä rivillä vastuualueiden nimet. Muut rivit kuvaavat toiminnallisten alueiden sisältöjä.

Tuotehallinnan osallistumisen alueeseen kuuluu yksi toiminnallinen alue, *strategisen johtaminen*. Tuotepäällikkö on mukana vaikuttamassa tämän alueen tehtäviin, mutta vastuu ja johtoasema alueen tehtävissä kuuluvat tuotepäällikkötasoa ylempänä olevalle johdolle. Aluetta voidaan pitää eräänlaisena rajapintana tuotehallinnan ja ylemmän johdon välillä. Strategisen johtamisen toiminta-alueeseen kuuluu portfoliohallinnan, innovaatiojohtamisen, resurssijohtamisen, markkina-analyysin ja tuoteanalyysin osa-alueet. Pienissä organisaatioissa tuotepäällikkö voi vastata myös tuote- ja markkina-analyysien tekemisestä. (Fricker, 2012, 65–67.)

Tuotehallinnan ydinalue koostuu tuotestrategian ja tuotesuunnittelun toiminnallisista alueista, mistä tuotepäällikkö on päävastuussa. *Tuotestrategian* tehtävät alkavat tuotteen asemoimisesta markkinoille ja tuotteen määrittelystä. Alueeseen kuuluu lisäksi tuotteen jakelumallista päättäminen, päättäminen siitä kehitetäänkö tietyt ohjelmiston osat itse vai hankitaanko ne ostettuina, tuotteen liiketoimintamallin suunnittelu ja kustannuslaskelmien tekeminen. Alueeseen sisältyy myös hinnoittelun ja ekosysteemin hallinta. Ekosysteemin hallinta liittyy muun muassa siihen, millä tavoin kehitettävä tuote halutaan asemoida osaksi muiden toimijoiden muodostamaa arvoketjua. Myös juridisten oikeuksien ja immateriaalioikeuksien hallinta kuuluu alueeseen. Tämä alue liittyy muun muassa monenlaisten sopimusten laatimiseen ja aineettomien oikeuksien puolustamiseen. Lopulta toiminnalliseen alueeseen kuuluu suorituskyvyn ja riskien hallinta, missä ohjelmistotuotteen onnistumista ja riskejä mitataan ja arvioi-

daan. Tuotehallinnan ydinalueen toinen osa on *tuotesuunnittelu*. Siihen liittyy ensinnäkin tuotteen elinkaaren hallinta ja toiseksi tuotteiden tiekartoitus, jossa ennakoidaan millaisissa etapeissa tuotetta tullaan kehittämään. Alueeseen kuuluu myös julkaisunsuunnittelu. Siinä määritellään vaatimukset, jotka toteutetaan tulevassa julkaisussa ja julkaisun aikataulu. Tuotteen vaatimusten hallinta liittyy siihen, miten tuotteen vaatimuksia hallitaan markkinalähtöisessä kehittämisessä, joissa vaatimuksia kerätään laajalta sidosryhmäjoukolta. (Fricker, 2012, 67–70.)

TAULUKKO 1 SPMBoK-viitekehys (ISPMA, 2012)

Strateginen johtaminen	**Tuotestrategia**	**Tuotesuunnittelu**	**Kehittäminen**	**Markkinointi**	**Myynti ja jakelu**	**Palvelu ja tuki**
Yrityksen strategia	Tuotteen asemointi ja määrittely	Tuotteen elinkaaren hallinta	Tuotannon hallinta	Markkinoinnin suunnittelu	Myynnin suunnittelu	Palveluiden suunnittelu ja valmistelu
Portfolionhallinta	Jakelumalli	Tiekartoitus	Projektinhallinta	Asiakkaiden analysointi	Kanavien valmistelu	Palveluiden hankinta
Innovaatiojohtaminen	Hankintapäätökset	Julkaisunsuunnittelu	Projektin vaatimustenhallinta	Mahdollisuuksien hallinta	Asiakassuhteen hallinta	Tekninen tuki
Resurssijohtaminen	Liiketoimintamalli ja kustannusarvio	Tuotteen vaatimustenhallinta	Laadunhallinta	Markkinointimixin optimointi	Operatiivinen myynti	Markkinoinnin tuki
Markkina-analyysi	Hinnoittelu			Tuotteen julkistus	Operatiivinen jakelu	Myynnin tuki
Tuoteanalyysi	Ekosysteemin hallinta			Operatiivinen markkinointi		
	Juridisten oikeuksien ja Immateriaalioikeuksien hallinta					
	Suorituskyvyn ja riskien hallinta					
Osallistuu	**Tuotehallinnan ydinalue**		**Johtaa**			

Kolmas SPMBoK-viitekehyksen alue koostuu toiminnallisista alueista, joita tuotepäällikkö *johtaa* delegoiden toimintojen vastuuta. Tämän alueen ensimmäinen

toiminnallinen alue on *kehittäminen,* johon tuotepäällikkö osallistuu muun muassa innovoimalla, viestimällä vaatimuksista ja hyväksymällä toteutuneita tuloksia. Alueeseen sisältyy tuotannonhallinta (engineering management), projektin hallinta, projektin vaatimustenhallinta ja laadunhallinta. Toinen tuotehallinnan johtama toiminnallinen alue on *markkinointi*, johon sisältyvät muun muassa markkinoinnin suunnittelun ja asiakasanalyysien tekemistä. Kolmas johtamisalue on *myynti ja jakelu*. Tämä alue sisältää myynnin suunnittelua, asiakassuhteen hallintaa ja operationaalisen jakelun. Operationaalinen jakelu liittyy ohjelmistotuotteiden toimittamiseen asiakkaille. Viimeinen tuotepäällikön johtama toiminnallinen alue on *palvelu ja tuki.* Siihen liittyy erilaisia tuki- ja palvelutoimintojen suunnittelu-, valmistelu-, hankinta- ja toteutustehtäviä. (Fricker, 2012, 71–73.)

Esitellyt viitekehyksien ja erityisesti tarkemmin kuvatun SPMBoK -viitekehyksen tarkastelulla nähdään, että tuotehallinnan kenttään kuuluu monialaisia tehtäviä. Viitekehyksistä riippuen tuotehallintaan voidaan katsoa kuuluvan 28–68 erilaista prosessia, aktiviteettia, osaamisaluetta tai kyvykkyyttä (Fricker, 2012, 63).

Edellä esitellyistä viitekehyksistä Bekkersin ym. (2010) viitekehystä tullaan käyttämään tutkimuksen muiden mallien vertailupohjana, sillä kyseinen viitekehys kuvaa tämän tutkimuksen kannalta oleellista julkaisunsuunnittelun ydinaluetta ja tähän ydinalueeseen liittyviä osa-alueita. Tasavertaisena vaihtoehtona vertailupohjaksi olisi voitu valita myös van de Weerdin ym. (2006) viitekehys Bekkersin ym. (2010) viitekehyksen sijaan, mutta vertailupohjaksi päädyttiin valitsemaan esityksistä uudempi versio. Kolmas vaihtoehto vertailupohjaksi olisi ollut SPMBoK-viitekehys (ISPMA, 2012), joka myös sisältää julkaisunsuunnittelun osa-alueen. SPMBoK-viitekehys ei kuitenkaan esitä tarkasti julkaisunsuunnitteluun liittyviä osa-alueita, ja se kuvaa myös niitä tuotehallinnan osa-alueita, jotka eivät ole tämän tutkimuksen kannalta olennaisia.

3.4 Yhteenveto

Yritykset ovat yhä laajemmin siirtymässä kehittämään yksittäisten ohjelmistojen sijaan useille asiakkaille tarjottavia ohjelmistotuotteita, joilla voidaan tarkoittaa esimerkiksi paketoitua ohjelmistokokonaisuutta tai ohjelmistoperäistä palvelua. Ohjelmistotuotteen ja räätälöidyn ohjelmiston kehitystavoissa on paljon eroavaisuuksia. Ohjelmistotuotteen kehittämisessä vaatimuksia kerätään räätälöityä ohjelmistoa laajemmalta sidosryhmien joukolta. Raja kehittämis- ja ylläpitovaiheen välillä ei ole myöskään selkeä vaan tuotetta parannetaan toistuvilla julkaisuilla. Ohjelmistoarkkitehtuurin merkitys ohjelmistotuotteelle on tärkeä, jotta ohjelmistoa pystytään muokkaamaan tehokkaasti ja uusia tuotteen versioita kehitettäessä perusarkkitehtuuria ei tarvitse kehittää kokonaan uudelleen. Ero räätälöidyn ja tuotteeksi kehitetyn ohjelmiston välillä voi olla kuitenkin usein häilyvä.

Tuotehallinta voidaan nähdä yrityksen liiketoimintaa ja ohjelmistotuotantoa yhdistäväksi tekijäksi ja tuotepäällikkö eräänlaiseksi yrityksen sisäiseksi pienoistoimitusjohtajaksi, joka määrittelee tuotteen strategian yrityksen strategiaan sopivaksi ja johtaa tuotteen strategian jalkauttamista eri toimintayksiköissä.

Luvussa tarkasteltiin myös yleisellä tasolla tuotehallinnan viitekehyksiä. Näistä SPMBoK-kehystä (ISPMA, 2012) esiteltiin muita yksityiskohtaisemmin. Se tuo esiin, miten tuotepäällikkö osallistuu yrityksen ylemmän johdon kanssa strategiseen johtamisen tehtäviin, miten tuotepäällikkö on vastuussa tuotteiden suunnittelusta ja strategian päättämisestä, ja lopuksi, miten tuotepäällikkö johtaa kehittämisen, markkinoinnin, myynnin ja palvelutoimintojen alueita. Viitekehyksistä Bekkersin ym. (2010) kompetenssimallia käytetään jatkossa tutkimuksen keskeisenä vertailupohjana, sillä kehys sisältää kuvauksen julkaisunsuunnittelun ydinalueesta aliprosesseineen.

4 KETTERÄ TUOTEHALLINTA

Tässä luvussa esitetään ensin lyhyt yleiskatsaus ketterää lähestymistapaa ja tuotehallintaa yhdessä käsitteleviin tutkimuksiin. Jatkossa tästä aihepiiristä käytetään nimitystä ketterä tuotehallinta. Luvun toisessa osassa tarkastellaan, millä tavoin tuotepäällikön ja tuotteen omistajan roolien on nähty sopivan yhteen ja millaisia ristiriitoja roolien vastuualueissa on havaittu. Kolmanneksi luvussa esitellään vertailupohjana käytettävä tuotehallinnan kompetenssimalli (Bekkers ym., 2010), tarkastellaan kahta ketterää tuotehallintaa jäsentävää mallia (Leffingwell, 2011; Vähäniitty, 2010b) ja verrataan malleja keskenään. Mallien vertailussa kiinnitetään huomiota myös siihen, miten mallit kuvaavat julkaisunsuunnittelun ja ohjelmistokehityksen välistä yhteyttä.

4.1 Ketterän tuotehallinnan tutkimus

Ketterän lähestymistavan mukaisen kehittämisen on nähty tuovan organisaatioille hyötyjä muun muassa tuottavuuden ja muuttuviin vaatimuksiin reagointinopeuden parantumisena (Kittlaus, 2012, 93). Ketterän lähestymistavan menetelmät, kuten Scrum on alun alkaen kehitetty pienten kehitysorganisaatioiden käyttöön (esim. Heikkilä, 2010, 170) eikä esimerkiksi Scrum (Schwaber ja Sutherland, 2011) ota kantaa siihen, miten kehitystoiminta tulisi organisoida useiden tiimien organisaatioissa.

Ketterän lähestymistavan ja tuotehallinnan käytäntöjen yhdistämistä on toistaiseksi tutkittu verrattain vähän (Fogelström, Gorschek, Svahnberg, Olsson., 2009, 54; Kittlaus, 2012, 95). Olemassa olevissa tutkimuksissa on muun muassa tutkittu tuotehallinnan ja ketterän lähestymistavan käytäntöjen yhteensopivuutta (Fogelström ym., 2009; Kittlaus, 2012). Lisäksi on ehdotettu tapoja ketterän lähestymistavan mukaisen kehittämisen liittämiseksi organisaatioiden pidemmän aikavälin tuotesuunnitteluun ja -hallintaan (Leffingwell, 2011; Vähäniitty, 2010b; Vähäniitty & Rautiainen, 2008; Rautiainen, Lassenius & Suhonen, 2002; ja Rautainen, Lassenius, Vähäniitty & Itkonen, 2006). On myös tutkit-

tu tuotehallinnan eri osa-alueiden kuten vaatimustenhallinnan (esim. Racheva, 2008) ja julkaisunsuunnittelun (esim. Heikkilä, Jadallah, Rautiainen & Ruhe, 2010a) toteuttamista ketterässä lähestymistavassa. Edellä mainitut tutkimukset eivät kuvaa koko ketterän tuotehallinnan tutkimuksen kenttää vaan ovat esimerkkejä tähän tutkimukseen läheisesti liittyvistä tutkimusalueista.

Eräs tämän tutkimuksen aihepiiriin läheisesti liittyvä tutkimus on Vlaanderen ym. (2011) esitys, joka kuvataan tässä lyhyesti. Esitys kuvaa menetelmän, jolla voidaan varmistaa, että vaatimusten käsittelystä ei synny pullonkaulaa tuotehallinnan ja projekti- tai tiimitason kehittämisen välille. Menetelmä perustuu ajatukseen, että tuotteen työlistalla säilytetään eri tarkkuustasolla kuvattuja vaatimuksia. Työlistan ylimpinä kohtina säilytetään tärkeimmiksi arvioituja ja tarkasti kuvattuja vaatimuksia ja listan loppupäässä yleisesti kuvattuja vaatimuksia, joita ei ole suunniteltu toteutettavaksi lähiaikoina. Menetelmässä tuotepäälliköt valitsevat erityisen *tuotehallintasprintin* alussa tuotteen työlistasta joukon yleisellä tasolla kuvattuja vaatimuksia tarkennettaviksi. Tuotehallintasprintin aikana tuotepäälliköt tarkentavat vaatimukset tarkemmalle tasolle, ja sprintin päätteeksi tarkennetut vaatimukset palautetaan takaisin tuotteen työlistalle. Menetelmä muistuttaa paljon Scrumin sprinttiä (Schwaber & Sutherland, 2011), jossa kehittäjät valitsevat jokaisen sprintin alussa tuotteen työlistasta joukon käyttäjätarinoita toteutettavaksi sprintin aikana. Tuotehallintasprintti onkin samanpituinen ja samalla tavalla peräkkäin toistuva kuin Scrumin sprintit, mutta kehitystiimin ja tuotehallinnan sprintit tapahtuvat toisiinsa nähden limittäin. Vlaandrenin ym. (2011) menetelmää voidaan ajatella hyödynnettävän tämän luvun lopussa esitettävien tuotehallintaa jäsentävien mallien mukaisessa kehittämisessä. (Vlaanderen ym., 2011.)

4.2 Tuotepäällikkö ja tuotteen omistaja

Monissa ketterän tuotehallinnan tutkimuksissa on nostettu esiin tuotepäällikön ja tuotteen omistajan roolien suhteutuminen toisiinsa ja roolien tehtäväalueissa on nähty päällekkäisyyksiä. Esimerkiksi Leffingwellin (2011, 202–207) mukaan tuotteen omistajan ja tuotepäällikön tehtäväkuvaukset voivat olla päällekkäisiä tuotteen tavoitteiden, vision määrittelyn, hinnoittelun ja sijoitetun pääoman tuoton laskemisen tehtävissä.

Eri lähteissä on annettu toisistaan poikkeavia ohjeita, miten tämä ristiriitaisuus tulisi ratkaista. Joidenkin näkemyksien mukaan ketterässä tuotehallinnassa tuotteen omistaja omaksuu tuotepäällikön tehtäviä (Pichler, 2010, 2–3) tai tuotepäällikkö alkaa toimia tuotteen omistajan roolissa (Racheva ym., 2008, 2–3). On myös nähty, että tuotepäälliköt ja tuotteen omistajat voivat toimia organisaatioissa rinnakkain (Leffingwell, 2011; Heikkilä, 2010, 174, Kittlaus, 2012, 92).

Mielipide-erot rooleista jakautuvat etenkin tilanteissa, joissa kyse on pienen, muutaman tiimin kehitysprojektin sijasta laajemmasta, useiden tiimien kokonaisuudesta. Pichler (2011) näkee, että laajoissa projekteissa tuotteen omis-

tajia voi olla useita. Tällöin tuotteen omistajat voivat muodostaa hierarkkisen ryhmän, jossa yksi jäsen on johtavassa asemassa. Johtava tuotteen omistaja vastaa tällöin tuotteen vision määrittelystä ja muiden tuotteen omistajien ohjaamisesta (Pichler, 2010, 12.)

Leffingwell ei pidä realistisena sitä, että tuotepäälliköiden on mahdollista toimia laajoissa kehitysprojekteissa yhtä aikaa kehitystiimin jäseninä ja vastata samalla tuotepäällikön moninaisista tehtävistä. Tuotepäälliköitä pitäisi olla tällöin useita, jotta jokaisen tiimin toimintaan voidaan osallistua. Tuotepäälliköillä ei myöskään ole välttämättä riittävää teknistä osaamista tai kiinnostusta keskittyä teknisempiin tuotteen omistajan tehtäviin. Toisaalta Leffingwell näkee ongelmia myös siinä, että tiimeissä toimivat tuotteen omistajat ottaisivat hoitaakseen tuotehallinnan tehtäväalueet. Päätettäväksi tulisi tällöin Leffingwellin mukaan, kuka tuotteen omistajista päättää kehitettävästä tuotteesta. Tekniset tuotteen omistajat eivät myös välttämättä ole kompetenssiltaan sopivia hoitamaan tuotehallinnan laajoja tehtäväalueita. Leffingwell ehdottaakin roolijakoa, jossa tuotepäälliköiden tehtävät ovat enemmän markkina- ja asiakassuuntautuneita ja tuotteen omistajan tehtävät ratkaisu- ja teknologiasuuntautuneita. Tuotepäälliköt vastaavat tällöin julkaisujen ja tuotteen omistajat iteraatioiden toteutuksesta. (Leffingwell, 2011, 202–207.)

Kittlaus (2012) on Leffingwellin kanssa samaa mieltä tuotepäälliköiden ja tuotteen omistajien rinnakkaisista rooleista. Hän ehdottaa, että tuotepäällikön ja tuotteen omistajan vastuualueet jaetaan organisaatioissa esimerkiksi SPMBoK-viitekehystä (ISPMA, 2012) hyväksi käyttäen. Kittlaus katsoo, että tuotepäällikön tehtävät painottuvat SPMBoK-viitekehyksessä (ISPMA, 2012) tuotehallinnan ydinalueelle ja tuotteet omistajan tehtävät kehittäminen -alueelle (Kittlaus, 2012, 92.)

Päätökset rooleista ja roolien vastuualueista riippuvat kuitenkin paljon organisaatioista ja sen koosta (esim. Pichler, 2010, 3). Yhden tiimin organisaatioissa yhden henkilön voi olla mahdollista hoitaa tehokkaasti tuotepäällikön tai tuotteen omistajan rooleja (esim. Kittlaus, 2012, 92).

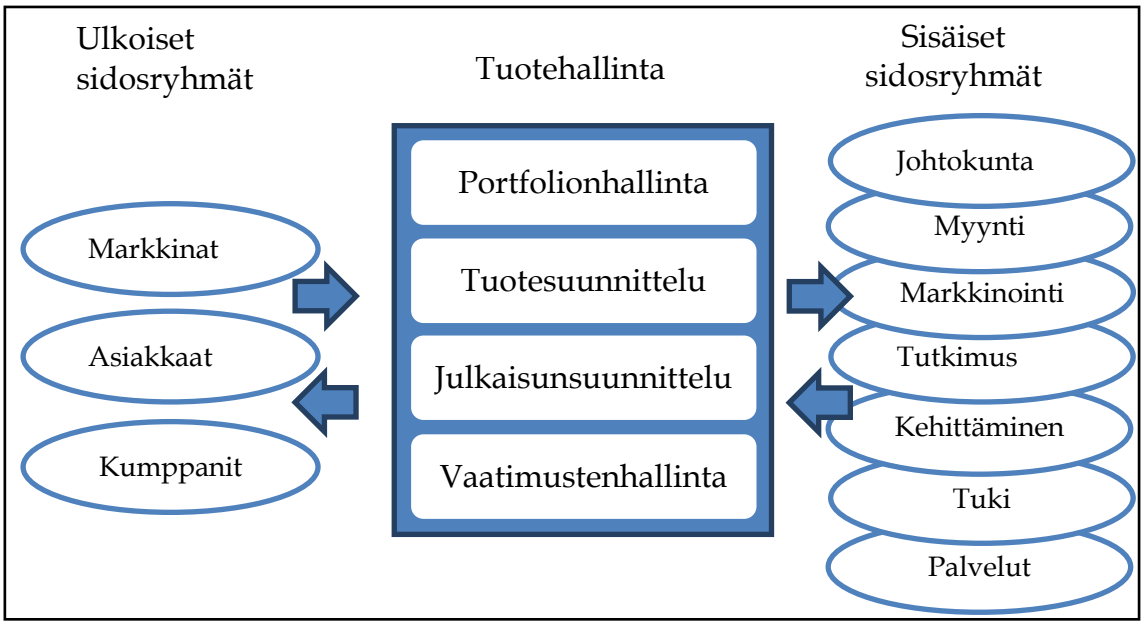
4.3 Ketterää tuotehallintaa jäsentävät mallit

Tässä alaluvussa esitetään ensin tutkimuksessa vertailupohjana käytettävä tuotehallinnan kompetenssimalli (Bekkers ym., 2010) ja toiseksi kaksi ketterää tuotehallintaa jäsentävää mallia: Big Picture-malli (Leffingwell, 2011) ja ATMAN-malli (Vähäniitty, 2010b, 124). ATMAN-mallin yhteydessä kuvataan lyhyesti myös Rautiaisen ym. (2006, 7) kontrollisyklit-mallia.

4.3.1 Tuotehallinnan kompetenssimalli

Bekkers ym. (2010) esittävät tuotehallinnan kompetenssimallin, joka kuvaa tuotehallinnan ydinalueita, sidosryhmiä, ydinalueiden sisäisiä prosesseja ja edellä

mainittujen välistä vuorovaikutusta. Mallia on luotu tiedeyhteisön ja liiketoiminnan asiantuntijoiden yhteistyönä ja se pohjautuu muun muassa kirjallisuusselvitykseen ja liiketoiminta-ammattilaisten tekemään validointiin (Bekkers ym., 2010, 2–3). Malli on esitetty myös Vlaanderenin (2010) artikkelissa. Tuotepäälliköt voivat käyttää viitekehystä vertailupohjana, jonka avulla olemassa olevia tuotehallinnan käytänteitä on mahdollista kehittää (Bekkers, 2010, 1). Tässä yhteydessä mallia ei tarkastella niin yksityiskohtaisella tavalla. Tästä syystä kuviossa 3 on esitetty yksinkertaistettu kuvaus mallista. Liitteessä 1 on alkuperäinen malli. 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 julkaisun luontiin ja jakeluun liittyviä tehtäviä. Mallin alimpana tasona on *vaatimustenhallinta*. Tämä taso sisältää tuotehallinnan jatkuvana toimintona tapahtuvan vaatimusten keräämisen, tarkentamisen ja organisoinnin. (Bekkers ym., 2010, 4–5.)

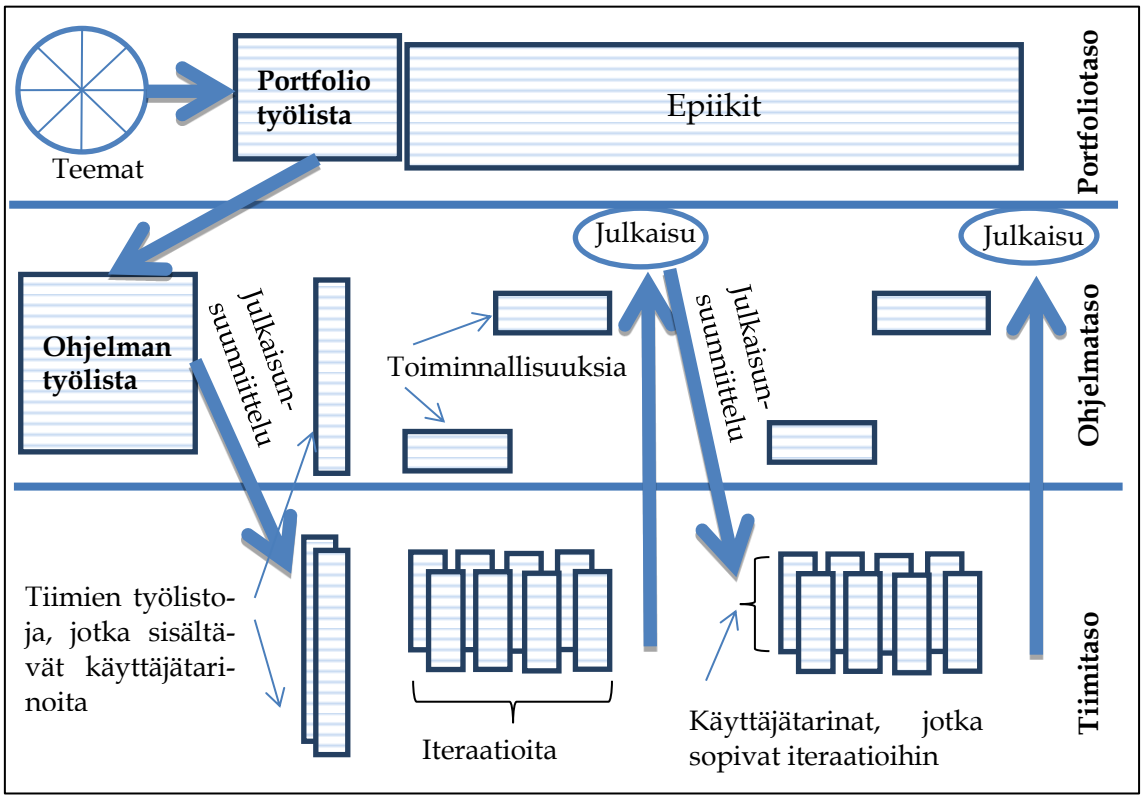


KUVIO 3 Tuotehallinnan kompetenssimalli (mukaillen Bekkers ym., 2010, 4)

Mallissa tuotehallinta on kytketty nuolin ulkoisiin sidosryhmiin, kuten markkinat, asiakkaat ja kumppanit, sekä sisäisiin sidosryhmiin kuten yrityksen johtokunta, myynti, markkinointi ja tuotekehitys & tutkimus. Kuviossa on esitetty myös kehittäminen (development), tukipalvelut (support) ja palvelut (services) (Bekkers ym., 2010). Kehittäminen vastaa projektityötä ohjelmistotuotteiden suunnittelemiseksi ja toteuttamiseksi.

4.3.2 Leffingwellin malli

Leffingwellin (2011; 2012) malli esittää miten yritykset voivat järjestää toimintaansa eri tasoilla ketterän lähestymistavan periaatteiden mukaisesti. Mallista käytetään nimitystä "Agile Enterprise Big Picture: Scaled Agile Delivery Model" eli lyhyemmin Big Picture (Leffingwell (2011, 32). Alkuperäinen malli on esitetty liitteessä 2. Sen yksinkertaistettu versio on kuviossa 4. Malli rakentuu kolmesta tasosta, jotka ovat portfolio-, ohjelma- ja tiimitaso (Leffingwell, 2011, 32). Tasoilla on esitetty erilaisia toimintoja, toimijoita, tuotoksia ja vaatimuksia sekä työlistoja. Tässä tutkimuksessa mallista keskitytään kuvaamaan kunkin tason merkitystä, tasoilla käsiteltäviä vaatimuksia ja vaatimusten muodostamaa arvoketjua. Arvoketju alkaa portfoliotasolta ja päättyy asiakkaille arvoa tuovina julkaisuina. Muita tasojen asioita kuvataan vain yleisellä tasolla (2011) eikä välttämättä tyhjentävästi.



KUVIO 4 Big Picture -malli (mukaillen Leffingwell, 2012)

Portfoliotaso on suuren, satoja tai jopa tuhansia työntekijöitä sisältävän ja useita ohjelmistotuotteita kehittävän, yrityksen käyttämä taso. Tasolla on *investointiteemoja* eli strategisia painopistealueita, joihin yritys jakaa resurssejaan. Investointiteemojen toteuttamiseksi laaditaan portfolion työlistalla säilytettäviä laaja-alaisia kehitysaloitteita eli *epiikkejä* (epics). Epiikit ovat yleisesti kuvattuja, ja niiden tarkoitus on enemmänkin antaa kuva jostakin tarpeesta kuin kuvata tarpeen ratkaisua. Tasolla on kuvattu myös erityinen portfolionhallintatoiminto,

joka vastaa muun muassa investointiteemojen määrittelystä. (Leffingwell, 2011, 83–91.)

Ohjelmatasolla useiden tiimien iteraatioissa kehittämät tuotokset yhdistyvät säännöllisessä tahdissa kehitettäviksi julkaisuiksi, jotka voidaan jakaa loppukäyttäjille. Kunkin julkaisujakson alussa pidetään julkaisunsuunnittelutapahtuma, jossa päätetään kehitettävän julkaisun sisältö ja jaetaan vaatimukset kehitettäviksi tiimeille. Tasolla vaatimukset on tarkennettu epiikeistä toiminnallisuuksiksi, joita säilytetään ohjelman työlistalla (program backlog). *Toiminnallisuudet* (features) voidaan määrittää ”järjestelmän toteuttamiksi palveluiksi jonkun sidosryhmän edustajan tarpeen tyydyttämiseksi” (Leffingwell, 2011, 75). Tasolla on esitetty myös visio ja tiekartta. *Visio* selittää kehitettävän tuotteen perimmäistä tarkoitusta. *Tiekartta* taas osoittaa, millaisia teemoja, sisältöjä ja aikatauluja seuraavan julkaisun jälkeen toteutettavissa julkaisuissa aiotaan toteuttaa. Toimijoista on esitetty tuotteenhallinta, johon liittyy tuotepäällikön rooli, ja julkaisunhallinta. Julkaisunhallinta on eräänlainen ohjausryhmä, joka valvoo ja tukee julkaisujen kehittämistä. Ohjelmatasolla on kuvattu myös laaja-alaisia järjestelmätoiminnallisuuksia kehittävät järjestelmätiimit (system team). (Leffingwell, 2011, 74–81.)

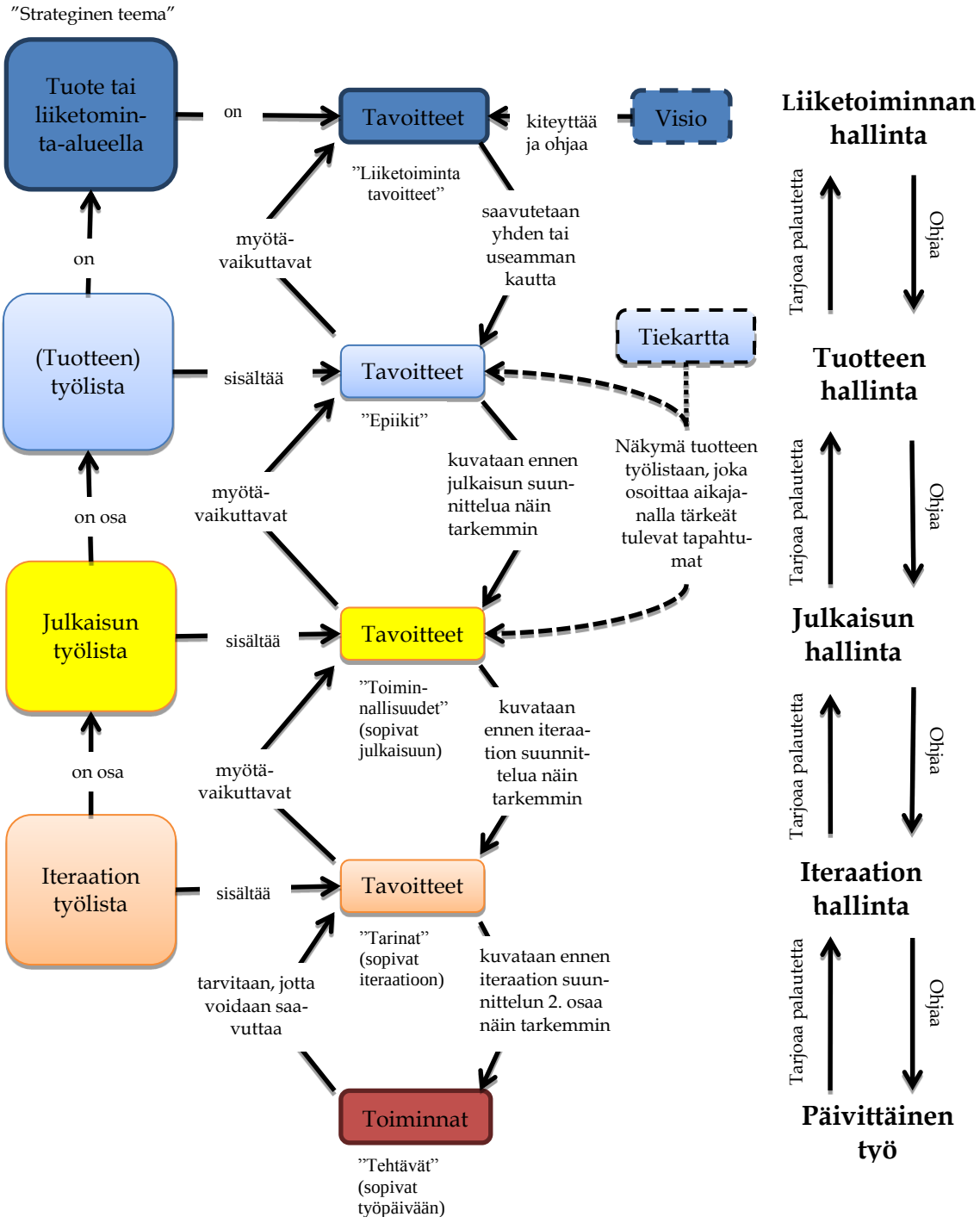
Tiimitasolla ketterät kehitystiimit tai niiden joukot kehittävät ohjelmistoja iteraatio kerrallaan. Jokaisella tiimillä on oma tiimin työlista, jossa säilytetään toiminnallisuuksista tarkennettuja käyttäjätarinoita. Käyttäjätarinat tarkennetaan tiimeissä yksityiskohtaisiksi tehtäviksi, joiden perusteella varsinainen kehitystyö toteutetaan. Tiimeihin kuuluvat tuotteen omistaja, Scrum-mestari ja kehittäjät. (Leffingwell, 2011, 47–57.)

4.3.3 ATMAN-malli

ATMAN-malli (Approach and Tool support for development portfolio MANagement) (Vähäniitty, 2010b, 124) kuvaa ohjelmistotuotteen kehitystyön jakaantumista viiteen tasoon, jotka ovat: liiketoiminnan hallinta, tuotteen hallinta, julkaisun hallinta, iteraation hallinta ja päivittäinen työ (Kuvio 5). Lisäksi malli osoittaa, millä tavoin vaatimukset on eri tasoilla määritelty ja miten ne liittyvät toisiinsa. Seuraavaksi tarkastellaan ensin ATMAN-mallin vasemmalla reunalla kuvattua vaatimushierarkiaa tasoittain Vähäniityn (2010b) mukaan. Tämän jälkeen tarkastellaan viitekehyksen oikealla puolella olevaa kontrollisyklit -mallia.

ATMAN-mallissa esitetyt vaatimukset noudattavat samoja määritelmiä kuin edellisen Leffingwellin (2012) mallin vaatimukset. *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. *Julkaisun hallinnan tasolla* tuotteen työlistan osa, *julkaisun työlista*, sisältää tulevaan julkaisuun listattuja vaatimuksia. Käyttäjien tarpeet on esitetty tällä tasolla toiminnallisuuksina. 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 toiminto toteutetaan. (Vähäniitty, 2010b, 123–125.)

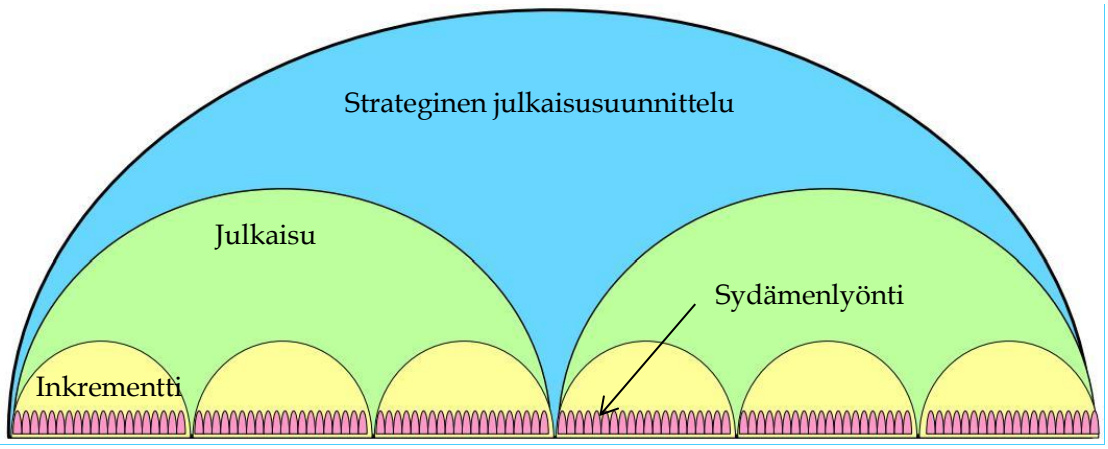


KUVIO 5 ATMAN–viitekehys (Vähäniitty, 2010b, 124)

Edellä esitetyssä kuvauksessa eri tasojen vaatimukset muodostavat hierarkian. Vähäniitty (2010b) pitää hyödyllisenä sitä, että eri abstraktiotasojen vaatimusten

liittyminen toisiinsa säilytetään, kun vaatimuksia tarkennetaan yleisemmän tason kuvauksista tarkempiin kuvauksiin. Näin voidaan esimerkiksi jäljittää, mitkä iteraation hallinnan tason käyttäjätarinat liittyvät tuotteen hallinnan tason epiikkeihin ja päinvastoin. Vähäniityn (2010b) mukaan jäljitettävyydestä voi olla hyötyä esimerkiksi tilanteessa, jossa organisaatio muuttaa strategista teemaa. Tällöin pystytään muuttamaan myös ne eri tasoilla oleva vaatimukset, jotka ovat tarkennuksia kyseisestä teemasta. Vastaavasti, kun vaatimuksia jaetaan pienemmiksi osiksi, voi syntyä tietoa, joka vaatii ylemmän tason vaatimusten muuttamista. (Vähäniitty, 2010b, 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-viitekehystä vastaa Rautiaisen ym. (2006) kontrollisykli (cycles of control) -mallia. *Kontrollisykli*mallissa on kuvattu liiketoiminnan, tuotteen ja iteraation hallinnan sekä päivittäisen työn tasot. 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 6 on esitetty kontrollisyklimalli havainnollisemmin aikajanalla. (Rautiainen ym., 2006.)



KUVIO 6 Kontrollisyklimalli aikajanalla kuvattuna (Rautiainen ym., 2006, 7)

Kuviossa 6 nähdään laajimpana kaarena strateginen suunnittelu, joka vastaa kuviossa 5 tuotetason suunnittelua. Strategisen suunnittelun ajanjaksoon sisältyy yllä esitetyssä kuviossa kaksi julkaisujaksoa, jotka kuviossa 5 vastaavat julkaisunsuunnittelun tasoa. Vastaavasti jokaisen julkaisunjakson sisällä on useita inkrementtejä (näitä voidaan kutsua mallissa myös iteraatioiksi). Jokainen iteraatio koostuu ”sydämenlyönneistä” (heartbeats), joilla tarkoitetaan yksittäisiä ohjelmistonkehityspäiviä. Jokaisen kuvion 6 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 voidaan myös päivittää korkeamman tason suunnitelmia vastaamaan nykyistä tilannetta. Kontrollipisteiden aikana organisaatio ottaa huomioon aiemmista työvaiheista saadun palautteen ja päivittää suunnitelmiaan uuden tiedon valossa. (Rautiainen ym., 2006.)

4.3.4 Mallien vertailu

Tässä alaluvussa verrataan Leffingwellin (2011) mallia ja ATMAN-mallia (Vähäniitty, 2010b) toisiinsa käyttäen vertailupohjana Bekkersin ym. (2010) kompetenssimallia. Bekkersin ym. (2010) malli soveltuu vertailupohjaksi, sillä se kuvaa tuotehallinnan keskeisiä osa-alueista mukaan lukien tämän tutkimuksen kannalta oleellisen julkaisunsuunnittelun.

Malleja verrataan ensin vuorollaan Bekkersin ym. (2010) malliin. Tämän jälkeen Leffingwellin mallia ja ATMAN-mallia verrataan toisiinsa taso kerrallaan käyttäen vertailukriteereinä tasolla vaatimusten tyyppejä ja tasojen erityispiirteitä. Yhteenveto mallien vertailusta esitetään taulukossa 2. Vertailun jälkimmäisessä osassa tarkastellaan, miten mallit kuvaavat ohjelmistokehityksen ja julkaisunsuunnittelun välisen yhteyden.

Leffingwellin malli sisältää Bekkersin ym. malliin verrattuna portfolionhallinnan tasoa vastaavan portfoliotason, julkaisunsuunnittelutasoa vastaavan ohjelmatason ja tiimitason, joka Bekkersin ym. mallissa vastaa projektitasoista kehittämistä. Leffingwellin malli ei sisällä tuotesuunnittelun tasoa, mutta sen ohjelmataso sisältää samoja piirteitä, kuten muun muassa tiekartan, joka sisältyy Bekkersin ym. tuotesuunnittelutasoon. Malli ei sisällä myöskään vaatimustenhallinnan tasoa, mutta mallin kaikilla tasoilla kuvataan erityyppisiä vaatimuksia ja vaatimusten keskinäisiä suhteita. Leffingwellin mallissa tiimitaso on kuvattu paljon yksityiskohtaisemmin kuin Bekkersin ym. mallissa.

ATMAN-malli sisältää Bekkersin ym. malliin verrattuna portfoliotasoa vastaavan liiketoiminnan hallintatason, tuotesuunnittelua vastaavan tuotteen hallinta -tason, julkaisunsuunnittelua vastaavan julkaisunhallintatason ja lopuksi iteraatio- ja päivittäisen työn tasot, jotka vastaavat Bekkersin ym. mallin kehittäminen -sidosryhmää. ATMAN-malli eroaa tasoiltaan Bekkersin ym. mallista siten, että ATMAN-malli ei sisällä vaatimustenhallintatasoa, ja sen iteraatiotaso ja päivittäisen työntasot on kuvattu yksityiskohtaisemmin kuin Bekkersin ym. mallin projektimuotoinen kehittäminen.

Leffingwellin mallissa ja ATMAN-mallissa on toisiinsa nähden paljon samankaltaisuuksia. Kummassakin mallissa vaatimustenhallinta on mukana jokaisella tasolla, sillä tasoilla kuvataan vaatimuksia ja niiden organisointia. Vaatimukset ovat malleissa täsmälleen samanlaisia, esimerkiksi käyttäjätarinoita ja toiminnallisuuksia. Malleissa on eroja tasojen määrän suhteen, mutta niissä on esitetty samanlaisia asioita, kuten tiekartoitusta ja visioita. Leffingwellin malli eroaa ATMAN-mallista merkittävästi siten, että se kuvaa rikkaammin erilaisia tuotehallinnan rooleja ja käytäntöjä. ATMAN-malli keskittyy enemmän kuvaamaan vaatimusten ja tasojen välisiä yhteyksiä riippuvuuksista.

TAULUKKO 2 Tuotehallintaa jäsentävien mallien vertailu

Bekkersin ym. malli	Leffingwellin malli	ATMAN-malli
Portfolionhallinta	**Portfoliotaso**	**Liiketoiminnan hallinta**
Vaatimukset	- Investointiteemat ja epiikit	- Strategiset teemat, liiketoimintatavoitteet ja visio
Muita piirteitä	- Portfoliotason visio ja arkkitehtuuri - Roolit: Portfoliohallinta (portfolio management)	- Ohjaa alemman tason toimintaa
Tuotesuunnittelu		**Tuotteen hallinta**
Vaatimukset		- Epiikit, tiekartta (roadmap)
Muita piirteitä		- Ohjaa alempaa tasoa ja tarjoaa palautetta ylemmälle tasolle
Julkaisunsuunnittelu	**Ohjelmataso**	**Julkaisun hallinta**
Vaatimukset	- Toiminnallisuudet ja käyttäjätarinat. Tiekartta ja visio	- Toiminnallisuudet
Muita piirteitä	- Kuvaa miten toiminnallisuudet sopivat julkaisuihin - Roolit: Järjestelmätiimit, tuote- ja julkaisunhallinta	- Kuvaa miten toiminnallisuudet sopivat julkaisuihin - Ohjaa alempaa tasoa ja tarjoaa palautetta ylemmälle tasolle
Vaatimustenhallinta	Ei ole, mutta vaatimusten organisointia on kuvattu mallin muilla tasoilla	Ei ole, mutta vaatimusten organisointia on kuvattu mallin muilla tasoilla
Kehittäminen -sidosryhmä	**Tiimitaso**	**Iteraation hallinta**
Vaatimukset	- Käyttäjätarinat ja tehtävät	- Käyttäjätarinat
	- Kuvaa miten käyttäjätarinat sopivat iteraatioihin ja tarinat toteutetaan tehtävinä - Roolit: kehitystiimit	- Kuvaa miten käyttäjätarinat sopivat iteraatioihin - Ohjaa alempaa tasoa ja tarjoaa palautetta ylemmälle tasolle
Kehittäminen-sidosryhmä		**Päivittäisen työn taso**
Vaatimukset		- Tehtävät
Muita piirteitä		- Kuvaa miten tehtävät sopivat työpäivään ja ovat tarkempia kuvauksia käyttäjätarinoista - Tarjoaa palautetta ylemmälle tasolle

Sekä Leffingwellin malli (2011) että ATMAN-malli (Vähäniitty, 2010b) osoittavat julkaisunsuunnittelun ja kehittämisen välille kaksi yhteyttä, jotka ovat eri tasojen välinen ohjaus ja vaatimusten välinen yhteys. Bekkersin ym. (2010) viitekehyksessä näitä yhteyksiä ei ole kuvattu tarkasti, vaan ohjelmistokehys kuvataan tuotehallinnan ulkopuolella olevana osana (development). Leffingwellin (2011) mallissa ohjelmatasolla tehtävä julkaisunsuunnittelu ohjaa tiimitason

kehittämistä. Mallissa ohjelmatason toiminnallisuudet tarkennetaan tiimitason käyttäjätarinoiksi, jotka puolestaan tarkentuvat toteutettaviksi tehtäviksi (Leffingwell, 2011). Myös ATMAN-mallissa (Vähäniitty, 2010a) julkaisunsuunnittelutaso ohjaa iteraatiotason toimintaa, joka taas puolestaan ohjaa päivittäistä työtä. Jokainen taso tarjoaa myös palautetta ylemmille tasoille. Näin päivittäisestä työstä kertyvä palaute vaikuttaa iteraatio- ja julkaisunsuunnittelutasoihin. Vaatimukset ovat mallissa jäljitettävissä toisiinsa.

4.4 Yhteenveto

Ketterää tuotehallintaa koskevaa kirjallisuutta on varsin vähän. Aihetta on tarkasteltu muun muassa ketterän lähestymistavan ja tuotehallinnan käytäntöjen yhteensopivuuden (esim. Fogelström ym. 2009) ja yhteen liittämisen (esim. Leffingwell, 2011) sekä tuotehallinnan eri osa-alueiden näkökulmasta. Eräänä ketterän lähestymistavan ja tuotehallinnan ristiriitakohtana on nähty päällekkäisyydet tuotepäällikön ja tuotteen omistajan rooleissa. Ratkaisuksi ongelmaan on nähty joko roolien yhdistäminen tai rinnakkain toimiminen siten, että roolien tehtäväalueita on tarkennettu. Ketterää tuotehallintaa jäsentämään on kehitetty malleja, joista kahta (Leffingwell, 2011; Vähäniitty, 2010b) esiteltiin ja vertailtiin luvussa. Vertailun perustana käytettiin Bekkersin ym. (2010) kompetenssimallia. Malleissa oli paljon samankaltaisuuksia muun muassa yhtenäisten vaatimusluokkien osalta. Merkittävimpinä eroina Leffingwellin (2011) malli kuvasi rooleja ja erilaisia käytäntöjä, kun taas Vähäniityn (2010b) malli esitti tarkemmin tasojen ja vaatimuksen välisen yhteyden. Jatkossa tässä tutkimuksessa rajaudutaan tarkastelemaan ketterän tuotehallinnan julkaisunsuunnittelua, joka sisältyy kuhunkin tuotehallinnan malliin.

5 JULKAISUNSUUNNITTELU: PROSESSEJA JA MENETELMIÄ

Tässä luvussa tarkastellaan julkaisunsuunnittelua sekä yleisesti että yksityiskohtaisemmin ketterän lähestymistavan kontekstissa. Ensiksi esitellään julkaisunsuunnittelun tehtävää ja piirteitä. Toiseksi tarkastellaan erilaisia julkaisunsuunnittelun muotoja ja selitetään miten ketterän julkaisunsuunnittelu suhteutuu näihin muotoihin. Kolmanneksi tarkastellaan julkaisunsuunnittelun haasteellisuutta. Neljänneksi esitellään julkaisunsuunnittelun tueksi kehitettyjen ehdotusten luokittelutapa ja kerrotaan, millä tavoin luvussa käsiteltävät ehdotukset on valittu. Tämän jälkeen esitellään kahdeksaa ketterään julkaisunsuunnitteluun soveltuvaa ehdotusta ja näiden ehdotusten vertailupohjana käytettävää Bekkersin ym. (2010) prosessimallia. Viimeisessä osuudessa ehdotuksia verrataan yleisten piirteiden, Bekkersin ym. mallin kattavuuden, vaatimusten valintatekijöiden ja ketterään kehittämiseen sopivuuden suhteen.

5.1 Julkaisunsuunnittelun lähtökohtia

Julkaisunsuunnittelua pidetään yhtenä ohjelmistotuotteen kehittämisen tärkeimpänä osa-alueena, sillä siinä päätetään, minkä sisältöisiä julkaisuja millekin asiakasryhmälle kehitetään ja millä aikataululla (Carlshamre, 2002, 139). Sisällön osalta päätetään, mitkä vaatimukset toteutetaan julkaisuun ja minkä laatuisina (Carlshamre, 2002, 139). Sisällön ja aikataulun lisäksi on otettava huomioon kustannukset ja työtaakka, joka kehittämisestä syntyy (Wohlin & Aurum, 2005, 247).

Saliu ja Ruhe (2005) ovat esitelleet kymmenen julkaisunsuunnittelussa huomioitavaa asiaa. Ne ovat: suunnittelun laajuus, aikahorisontti, tavoitteet, sidosryhmän edustajien osallistuminen, priorisointimekanismi, tekniset rajoitteet, resurssirajoitteet, järjestelmärajoitteet, ratkaisun luonne ja laatu, ja työkalutuki. *Suunnittelun laajuus* (scope) vaikuttaa siihen, miten monen julkaisujakson päähän suunnittelu yltää. *Aikahorisontti* (time horizon) viittaa siihen, onko jul-

kaisujakson pituus kiinteä vai joustava. Julkaisunsuunnittelun *tavoitteet* voivat liittyä useaan asiaan, kuten mahdollisimman suuren liiketoiminta-arvon tuottamiseen, tärkeyden ja riskien huomioimiseen sekä sidosryhmien tyytyväisyyteen. *Sidosryhmän edustajien osallistuminen* liittyy siihen, millä laajuudella sidosryhmän edustajan mielipiteitä on huomioitu suunnittelussa. *Priorisointimekanismi* viittaa tapaan, jolla vaatimukset järjestetään tärkeysjärjestykseen. *Tekniset rajoitteet* ovat vaatimusten toteutusjärjestykseen liittyviä rajoitteita. *Resurssirajoitteet* voivat liittyä aikataulun, budjetin, riskin ja vaativuuden asettamiin rajoitteisiin. *Järjestelmärajoitteet* liittyvät muun muassa kehitettävän järjestelmän arkkitehtuurin, koodiperustan ja virhehistorian asettamiin rajoitteisiin. *Ratkaisun luonne ja laatu* liittyvät siihen, millainen ratkaisu katsotaan suunnitteluongelmassa hyväksyttäväksi. *Työkalutuki* viittaa siihen, millaisia apuvälineitä julkaisunsuunnittelun toteuttamiseksi on olemassa. (Saliu & Ruhe, 2005, 2–3.)

Julkaisunsuunnittelun keskeisimpiä tehtäviä on vaatimusten valinta. Svahnberg ym. (2010, 241) ovat esittäneet taksonomian vaatimusten valintatekijöistä. Taksonomiaan on otettu vaikutteita muun muassa Ngo-Thelta ja Ruhelta (2007). Tekijät jakautuvat koviin rajoitteisiin (hard constraints) ja pehmeisiin tekijöihin (soft factors). Ngo-The ja Ruhe (2007, 96–97) kuvaavat omassa esityksessään *kovat rajoitteet* suunnitteluun vaikuttavina, yksiselitteisinä ja helposti ymmärrettävinä rajoitteina, kuten vaatimuksen toteuttamiseen kuluva työmäärä tunteina. *Pehmeät rajoitteet* he kuvaavat implisiittisinä tekijöinä, joita voi olla vaikea selittää yksiselitteisesti mutta jotka on tärkeä ottaa suunnittelussa huomioon, esimerkiksi riskit. Svahnbergin ym. (2010) koviin rajoitteisiin kuuluvat tekniset rajoitteet, johon kuuluvat vaatimusten riippuvuudet ja laaturajoitteet, budjetti- ja kustannusrajoitteet, resurssirajoitteet, työmäärärajoitteet ja aikarajoitteet. Pehmeisiin rajoitteisiin sisältyvät sidosryhmien vaikutus-, arvo-, riski- ja resurssinkulutustekijät. *Teknisillä rajoitteilla* tarkoitetaan vaatimuksiin itseensä liittyviä rajoitteita ja niiden toteuttamisen rajoituksia. *Vaatimusten riippuvuustekijät* liittyvät vaatimusten riippuvuuksiin, ja ne rajaavat vaatimusten toteutuksen järjestystä. *Laaturajoitteet* voivat liittyä esimerkiksi jonkin lain asettamiin rajoitteisiin. *Budjetti- ja kustannusrajoitteet* ovat rajoitteita, jotka voivat liittyä esimerkiksi vaatimusten hintaan. *Resurssirajoitteet* rajaavat kehittämisessä käytössä olevien resurssien määrää. *Työmäärärajoitteet* rajaavat kehitystyössä käytettävää työmäärän kapasiteettia. *Aikarajoitteet* liittyvät esimerkiksi siihen, että jokin vaatimus toteutetaan tiettyyn aikaan. *Sidosryhmien vaikutustekijät* ovat tekijöitä, jotka liittyvät sidosryhmien suunnittelussa mukana oloon. *Arvotekijät* ovat tekijöitä, jotka auttavat julkaisun arvon määrittelyssä tai sen maksimoimisessa. *Riskitekijät* taas auttavat julkaisun riskin määrittelyssä. *Resurssinkulutustekijät* ymmärretään (Ngo-The ja Ruhen, 2007, 97 kuvauksen perusteella) tekijöinä, joilla pyritään resurssien tasaiseen käyttöön. (Svahnberg ym., 2010, 241–244.)

Wohlinin ja Aurumin (2005) tutkivat, millä kriteereillä valitaan ensisijaisesti vaatimuksia julkaisuun. Tutkimuksessa tärkeimmiksi kriteereiksi havaittiin ei-teknisten tekijät, kuten sidosryhmien vaatimuksille asettama tärkeys, julkaisun ajankohta, se kuka vaatimuksen on antanut sekä vaatimuksen hyödyn ja kustannuksen suhde. Tutkimustuloksien perusteella näyttää tärkeältä se, että

julkaisunsuunnittelun lähestymistavoissa otetaan huomioon myös liiketoiminnan (business) ja johdon (managerial) näkökulma teknisen järjestelmänäkökulman lisäksi (Wohlin & Aurum, 2005, 254).

Julkaisunsuunnittelun yksi keskeinen osa-alue vaatimusten valinnan ohella on vaatimusten priorisointi (Berander, 2007, 5). Beranderin (2007, 44–46) mukaan vaatimusten priorisointia voidaan tehdä useasta eri näkökulmasta, esimerkiksi vaatimuksen *tärkeyden*, toteuttamatta jättämisestä syntyvän *haitan*, *kustannuksen*, kehittämiseen kuluvan *ajan* tai toteuttamiseen liittyvän *riskin* perusteella.

Liiketoiminta-arvon nopea tuottaminen on nähty yhtenä keskeisenä ketterän lähestymistavan tavoitteena (Abrahamsson, 2002, 12). Liiketoiminta-arvo on myös yksi tärkeimmistä vaatimusten priorisointiin vaikuttavista tekijöistä ketterässä kehittämisessä (Racheva ym., 2008, 1).

5.2 Julkaisunsuunnittelun muotoja

Julkaisunsuunnittelua voidaan luokitella monella tavalla. Ensiksikin se voidaan jakaa etupainotteiseen (predictive planning) ja mukautuvaan suunnitteluun (adaptive planning) (Cohn, 2006). *Etupainotteisella* suunnittelulla tarkoitetaan suunnittelutapaa, jossa jo kehitystyön alkuvaiheessa pyritään luomaan yksityiskohtainen pitkän aikavälin suunnitelma. *Mukautuvassa* julkaisunsuunnittelussa laaditaan tarkasti vain lyhyen ajan suunnitelma ja pitkän aikavälin yksityiskohdat hahmotellaan karkeammalla tarkkuustasolla. Mukautuvassa suunnittelussa suunnittelu jakautuu koko ohjelmiston elinkaaren ajalle ja suunnitelmia päivitetään säännöllisesti. Ketterässä lähestymistavassa suunnittelu on mukautuvaa. Ketterässä julkaisunsuunnittelussa vaatimukset voidaan kuvata esimerkiksi käyttäjätarinoiden tarkkuudella ja vasta iteraatiotason suunnittelussa vaatimukset tarkennetaan yksityiskohtaisemmiksi tehtäviksi (Cohn, 2006).

Julkaisunsuunnittelua voidaan luokitella myös formaalisuusasteen mukaan järjestelmälliseen (systematic planning) ja puhtaasti arviointiin perustuvaan (judgement based) suunnitteluun (Benestad & Hannay, 2011, 766). Näiden äärilaitojen välissä on hybridi-suunnittelu, jossa yhdistyy järjestelmällisen ja arviointiin perustuvan suunnittelun piirteitä (Svahnberg ym., 2010, 238). *Järjestelmällisessä suunnittelussa* julkaisusuunnitelman laatimiseen tarvittavat tekijät ja käytetyt analyysitavat on eksplisiittisesti määritelty (Benestad & Hannay, 2011, 766). Järjestelmällisen suunnittelun esimerkkejä ovat erityiset julkaisunsuunnittelumallit (Svahnberg ym., 2010). Tyypillisesti julkaisunsuunnittelumalleissa suunnittelun ongelma on mallinnettu kapsäkkiongelmaksi (knapsack problem) tai sen muunnelmaksi. (Jantunen, Lehtola, Gause, Dumdum ja Barner, 2011, 38). *Arviointiin perustuva suunnittelu* nojaa henkilöiden päätöksentekoon, ja niissä oletetaan, että julkaisusuunnittelun kysymykset pystytään ratkaisemaan projektin eri osapuolten pohdinnalla ja keskinäisillä neuvotteluilla. (Benestad & Hannay, 2011, 766). Arviointiin perustuva suunnittelua pidetään usein ketterille menetelmille ominaisena suunnittelutapana (esim. Ruhe & Saliu, 2005, 48).

Järjestelmälliselle suunnittelulle on nähty useita tarpeita, sillä pelkkään arviointiin nojaavaa suunnittelua pidetään usein riittämättömänä monimutkaisissa suunnittelutilanteissa, esimerkiksi kun täytyy huomioida usean sidosryhmän mielipide (Ruhe & Saliu, 2005). Järjestelmällisellä suunnittelulla on pyritty muun muassa parempaan päätöksenteon läpinäkyvyyteen, eri vaatimusten valintatekijöiden huomioonottamiseen (Heikkilä ym., 2010a, 1) ja kehitystyön ennustettavuuden lisäämiseen (Szoke, 2011b, 2). Järjestelmällisen suunnittelun tuloksiin on myös suhtauduttu arviointiin perustuvaa suunnittelua luottavaisemmin (Du, McElroy ja Ruhe, 2006, 439).

Ketterän lähestymistavan ja järjestelmällisen suunnittelun välillä on nähty kuitenkin useita ristiriitoja, joista Heikkilä (2010, 172) esittänyt seuraavia:

- Ketterässä kehittämisessä vaatimukset tarkennetaan yksityiskohtaisiksi tehtäviksi vasta iteraatiovaiheen suunnittelussa, kun taas järjestelmällisissä malleissa vaatimukset on kuvattava tarkasti jo julkaisusuunnittelussa.
- Ketterässä kehittämisessä vaatimusten välisiä riippuvuuksia hallitaan epäformaalisti, mutta järjestelmällisissä malleissa vaatimusten väliset riippuvuudet on määriteltävä eksplisiittisesti.
- Ketterässä kehittämisessä korostuu yksilöiden välinen yhteistyö, kun taas järjestelmällisessä mallissa ratkaisu syntyy matemaattisen mallin avulla.
- Ketterässä kehittämisessä pyritään resurssien optimaaliseen allokointiin lyhyellä aikavälillä, kun taas järjestelmällisissä malleissa pyritään luomaan suunnitelma, jossa vaatimukset on jaettu optimaalisesti koko julkaisujakson ajalle.

Näitä ristiriitoja tarkastellaan myöhemmin luvun vertailuosiossa, jossa arvioidaan eri ehdotusten sopivuutta Scrumin muotoiseen ketterään kehittämiseen.

5.3 Julkaisunsuunnittelun haasteita

Carlshamren (2002) mukaan julkaisunsuunnittelua voidaan pitää ns. ilkeänä ongelmana (wicked problem). *Ilkeät ongelmat* ovat vaikeita suunnitteluongelmia, joille ei ole olemassa optimaalisia ratkaisuja. Näiden ongelmien tarkka määrittely ei ole myöskään mahdollista eikä ongelmien ratkaisuyrityksille ole pysäytyssääntöä, joka ilmaisee milloin ongelma on parhaalla tavalla ratkaistu. (Carlshamre, 2002, 146.)

Carlshamre (2002) esittää useita esimerkkejä julkaisunsuunnittelun haasteista. Ensinnäkin julkaisusuunnitelman arvon määritys voi olla vaikeaa. Jos suunnitelmasta puuttuu joukko tärkeimmiksi arvioituja vaatimuksia, suunnitelmaa ei välttämättä koeta parhaaksi vaihtoehdoksi, vaikka se sisältäisikin kokonaisuudessaan optimaalisesti eniten liiketoiminta-arvoa. Kriteerit, joilla vaatimuksia priorisoidaan, voivat myös tarkentua suunnittelun aikana, kun vaatimuksia opitaan ymmärtämään paremmin. Päätöksenteko vaatimusten ja suunnitelmien valitsemisesta tehdään myös usein suhteellisena vertaamalla eri vaih-

toehtoja. Esimerkiksi suunnitelmien valinnassa päätöksentekoon vaikuttaa eri suunnitelmien sisältö, mutta myös se mitkä vaatimukset jäävät suunnitelmien ulkopuolelle. Monesti valmiin suunnitelman perusteella voidaan oivaltaa kokonaisvaltaisemmin asioita, joita ei ole nähty aiemmin yksittäisiä vaatimuksia tarkastellessa. Valmiista suunnitelmasta voidaan havaita esimerkiksi jokin erityinen teema. Suunnitteluprosessissa asioita voi myös olla tarve tarkastella useasta eri näkökulmasta, esimerkiksi aikajanalla tai vaatimusten riippuvuuksien tarkasteluun keskittyen, jotta suunnitelmien sopivuudesta saadaan hyvä käsitys. (Carlshamre, 2002, 146–148.)

Jantunen ym. (2011, 38–39) luokittelevat tekemänsä kirjallisuuskatsauksen perusteella julkaisunsuunnittelun haasteet yleisesti vaatimusten priorisointiin, vaatimusten välisiin riippuvuuksiin ja hyväksyttävän lopputuloksen kriteerien jatkuvaan muuttumiseen liittyviksi. *Vaatimusten priorisointi* -haasteesta Jantunen ym. (2011) kuvaavat ensin, miten prioriteettikäsite voi itsessään merkitä erilaisia asioita. Prioriteetti voidaan nähdä jonkin asian tärkeyttä mittaavana suureena tai sillä voidaan tarkoittaa siten, miten kiireellisesti vaatimus täytyy toteuttaa. Priorisointia voidaan tehdä myös useasta eri näkökulmista (Berander, 2007, 44), ja erilaiset priorisoinnin näkökulmat, kuten tärkeys, voivat olla itsessään moniulotteisia. On hyvin haasteellista päättää, mitkä ovat kulloisessakin ongelmassa oleelliset priorisoinnin näkökulmat ja miten näistä näkökulmista saadaan kerättyä oikeellista tietoa. Ongelmaa vaikeuttaa vielä usean sidosryhmän edustajan mielipiteen huomioiminen ja se, että vaatimusten prioriteetit voivat vaihdella ajallisesti. *Vaatimusten väliset riippuvuudet* -haasteesta Jantunen ym. (2011) toteavat, että vaatimusten välillä on usein monenlaisia keskinäisiä riippuvuuksia, joiden määrä lisääntyy ajan kuluessa ja esimerkiksi silloin, kun tuote esitellään uusille markkinoille. Kolmanneksi haasteeksi Jantunen ym. (2011) kiteyttävät *hyväksyttävän lopputuloksen kriteerien muutoksen.* Tällä tarkoitetaan sitä, että esimerkiksi muutokset liiketoimintaympäristössä ja teknologiassa sekä kilpailijoiden liikkeet vaikuttavat siihen, millainen suunnitelma kulloinkin nähdään hyväksyttävänä. (Jantunen ym., 2011, 38–39.)

5.4 Ketterän julkaisunsuunnittelun ehdotusten valinta

Kirjallisuudessa on esitetty lukuisia ehdotuksia julkaisunsuunnittelun tueksi. Ne eroavat toisistaan fokukseltaan ja yksityiskohtaisuustasoltaan. Esimerkiksi jotkut tarkastelevat julkaisunsuunnittelua kokonaisuudessaan, toiset vain jotain osaa siitä. Tässä alaluvussa esitetään ensin yksinkertainen luokittelu ehdotuksille ja rajaudutaan tarkastelemaan sen jälkeen vain osaa näistä.

Berander (2007) on määritellyt hierarkkisen luokituksen vaatimusten priorisointia tukeville esityksille. Luokitus muodostuu (alimmasta ylimpään) tehtävistä, tekniikoista, menetelmistä ja prosesseista. Jäsennyksessä ylemmät tasot voivat hyödyntää alemman tasojen lähestymistapoja toiminnassaan. Tehtävillä tarkoitetaan toimintoja, joita tehdään vaatimusten priorisoinnin yhteydessä. Esimerkkinä tällaisesta on vaatimusten vertaaminen toisiinsa. Tekniikalla taas

tarkoitetaan esityksiä, jotka voivat käyttää hyväkseen useampien tehtävien kautta saatuja tietoja ja suorittavat tiedoille jonkin toimenpiteen. Esimerkkinä tällaisesta on AHP (Analytic Hierarchy Process) -tekniikka (Saaty, 1980). Menetelmä on Beranderin (2007) mukaan tekniikoita monimutkaisempi (sophisticated) esitys, jossa priorisoinnissa huomioidaan useampia näkökulmia (aspects) kuin tekniikoissa. Menetelmä voi hyödyntää toiminnassaan tehtäviä ja tekniikoita. Beranderin (2007) mukaan EVOLVE-suunnittelumalli (Greer & Ruhe, 2004) on esimerkki menetelmästä. Prosesseilla tarkoitetaan askeleita priorisoinnin suorittamiseksi, ja tähän prosessiin voi sisältyä alemman tason esityksiä eli menetelmiä, tekniikoita ja tehtäviä. (Berander, 2007, 13–14.)

Tässä tutkimuksessa sovelletaan Beranderin (2007) luokittelua julkaisunsuunnittelua koskeviin ehdotuksiin seuraavin merkityksin. *Prosesseiksi* nähdään julkaisunsuunnittelun suorittamiseen tarvittavat vaiheet ja askeleet. *Menetelminä* nähdään formaalit julkaisunsuunnittelumallit (strategic release planning models), joita Svahnberg ym. (2010) ovat kuvanneet. *Tekniikoiksi* nähdään esimerkiksi julkaisunsuunnittelun suorittamisessa käytettävät tekniikat, kuten priorisointitekniikat. *Tehtävät* taas ovat askeleiden, menetelmien ja tekniikoiden aikana tehtäviä yksittäisiä toimintoja. Tämän luvun yhteydessä esitellään prosessi- ja menetelmätasoisia ketterän julkaisunsuunnittelun ehdotuksia. Tekniikkatasoisia lähestymistapoja esitellään seuraavassa -luvussa.

Julkaisunsuunnittelua tukevien ehdotusten löytämiseksi suoritettiin perusteellinen kirjallisuushaku (ks. liite 1). Haulla löydettiin yhteensä 18 ehdotusta, jotka on esitelty taulukossa 3.

TAULUKKO 3 Ketterän julkaisunsuunnittelun ehdotuksien haun tulokset

Ehdotusryhmä	Ehdotus	Viittaukset Google Scholarissa	Valitaan tarkasteluun
1	Li ym. (2006)	17	X
2	McDaid ym. (2006)	7	
	Logue ym. (2007)	5	
	Logue ja McDaid (2008a)	6	X
	Logue ja McDaid (2008b)	4	
3	Li ym. (2010)	7	X
4	Heikkilä, Jadallah, Rautiainen & Ruhe (2010a)	5	X
5	van Valkenhoef ym. (2010)	3	
	van Valkenhoef ym. (2011)	1	X
6	Szoke (2011b)	4	X
7	Heikkilä, Rautiainen ja Jansen (2010b)	3	
	Heikkilä (2010)	ei löydy	X
8	Szoke (2010)	3	
9	Nagy ym. (2010)	2	
10	Karlsson ym. (2006b)	2	
11	Fernandes ym. (2008)	1	
12	Szoke (2011a)	0	
13	Kaur (2010)	0	

Löydetyistä ehdotuksista seitsemän valittiin tarkempaan tarkasteluun käyttäen hyväksi Google Scholar -palvelun (Google, 2011) tutkimuksien viittausmääriä. Valinta tehtiin seuraavalla tavalla:

- Asetettiin ehdotukset taulukkoon.
- Merkittiin kuinka monta kertaa ehdotuksiin oli Google Scholarin mukaan viitattu.
- Ehdotukset järjestettiin viittausmäärän mukaan laskevaan järjestykseen.
- Mikäli tutkimukset esittivät samaa mallia tai olivat aiempien tutkimusten kehittyneempiä versioita, niitä käsiteltiin omana ryhmänään. Samaan ryhmään kuuluvat tutkimukset järjestettiin peräkkäin ryhmän eniten viitatun tutkimuksen jälkeen.
- Valittiin seitsemän ensimmäistä ryhmää. Näiden ryhmien uusin tai kehittynein ehdotus otettiin tarkasteluun.

Ehdotusten ryhmittelyllä haluttiin estää se, että vertailuun valikoituu samaan malliperheeseen kuuluvia ehdotuksia. Edellä mainittujen ehdotusten lisäksi tarkasteluun valittiin kaksi tunnettua, Cohnin (2006) ja Leffingwellin (2011), ehdotusta. Cohnin (2006) ehdotusta tarkastellaan omana kokonaisuutenaan ja Leffingwellin (2011) esitystä sivutaan Heikkilän (2010) esityksen yhteydessä.

Tutkimuksen kirjallisuuskatsauksessa pyrittiin hyödyntämään myös julkaisunsuunnittelun ehdotusten aiempia vertailuja. Yksi tutkimuksen kannalta keskeinen lähde on Svahnbergin ym. (2010) tutkimus, jossa vertaillaan järjestelmällisiä suunnittelumalleja. Kyseisestä tutkimuksesta löytyi ketterään kehittämiseen ehdotettu Lin ym. (2006) ehdotus, joka löydettiin myös kirjallisuuskatsauksessa. Toinen Svahnbergin ym. (2010) tutkimuksessa esiin noussut ehdotus on Loguen ja McDaidin (2008a) ketterään kehittämiseen sopivaksi ehdottama EVOLVE-malli (Greer & Ruhe, 2004). Tätä ehdotusta ei kuitenkaan poimittu tarkasteluun, sillä tutkimuksessa sivutaan Heikkilän ym. (2010a) -esityksen yhteydessä EVOLVE+ -mallia (Ngo-The & Ruhe, 2007), joka on yksi Evolve-perheen tuoreimmista ehdotuksista (Svahnberg ym., 2010, 241).

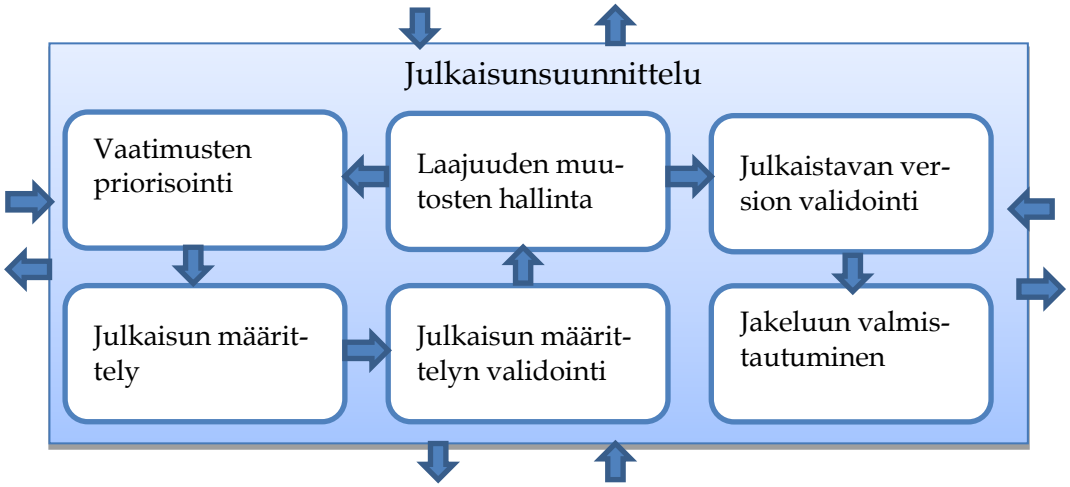
5.5 Ehdotusten kuvaukset

Tässä alaluvussa kuvataan lyhyesti edellä valitut julkaisunsuunnittelun tueksi kirjallisuudessa esitetyt ehdotukset tekijöiden mukaisessa aakkosjärjestyksessä. Ensimmäisenä kuvataan Bekkersin ym. (2010) prosessimalli, jota käytetään seuraavassa alaluvussa muiden ehdotusten vertailuun ja arviointiin.

5.5.1 Bekkersin ym. prosessimalli

Bekkersin ym. (2010) julkaisunsuunnittelun prosessimalli on osa tuotehallinnan kompetenssimallia, jota tarkasteltiin aliluvussa 4.3.1. Kuviossa 7 on esitetty,

millaisia vaiheita siihen sisältyy. Prosessi on kuvattu hyvin yleisellä tasolla. Prosessin vaiheiden kuvaukset sisältävät tiedon vaiheiden tehtävistä, mutta ei kuvausta siitä, miten vaiheiden tehtävät toteutetaan. Prosessia ei ole tarkoitettu tarkaksi ohjeeksi julkaisusuunnitelman luomiseen, vaan sitä voidaan käyttää julkaisunsuunnittelun käytäntöjen kehittämisessä (Bekkers ym., 2010, 1). Prosessi sisältää seuraavat vaiheet: vaatimusten priorisointi, julkaisun määrittely, julkaisun määrittelyn validointi, laajuuden muutosten hallinta, julkaistavan version validointi ja jakeluun valmistautuminen. Seuraavaksi kuvataan prosessin vaiheita Bekkersin ym. (2010) mukaan.



KUVIO 7 Kompetenssimallin (mukaillen Bekkers ym., 2010, 4) julkaisunsuunnitteluosa

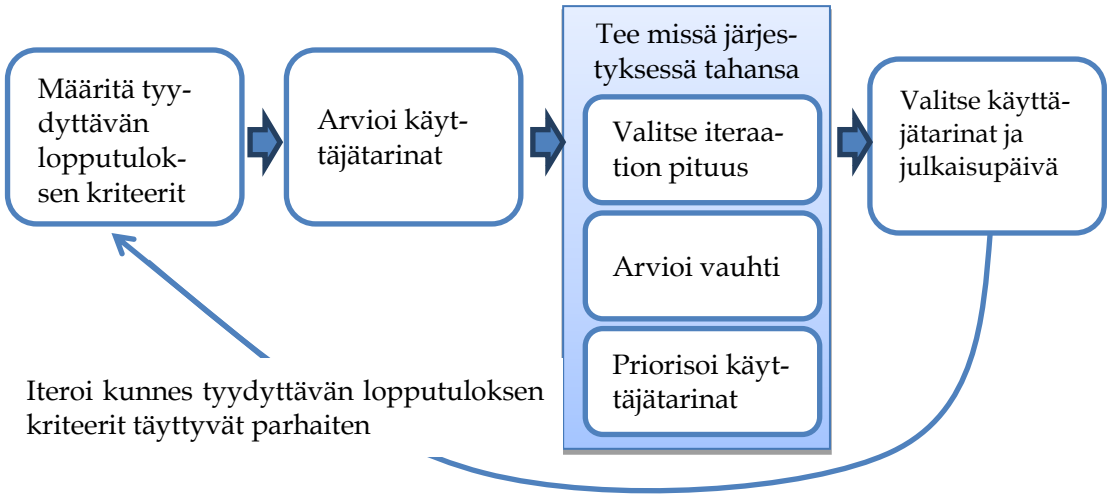
Vaatimusten priorisointi (requirements prioritization) -vaiheessa tunnistetut ja järjestetyt vaatimukset priorisoidaan. *Julkaisun määrittely* (release definition) -vaihe sisältää priorisoitujen vaatimusten valitsemisen tulevaan julkaisuun. *Julkaisun määrittelyn validointi* (release definition validation)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

5.5.2 Cohnin prosessimalli

Cohn (2006, 143) kuvaa julkaisunsuunnittelua iteratiiviseksi prosessiksi, jossa suunnitelmaa päivitetään säännöllisesti. 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 työmäärän ja kehitysvauhdin arvioinneissa käytetään yksikkönä joko tarinapisteitä tai ideaalipäiviä. *Tarinapisteet* (story points) ovat tarinoiden kokoa kuvaavia suhteellisia numeroarvoja. Jos jonkin tarinan koko arvioidaan olevan 1, tätä tarinaa kaksi kertaa vaativammaksi arvioidulle tarinalle annetaan arvo 2. Tarinapisteiden mittakaavalla ei ole merkitystä, mutta Cohn (2006, 36) suosittelee käyttämään 1–10 pisteen asteikkoa. *Ideaalipäivä* (ideal day) kuvaa, miten monta henkilötyöpäivää tarinan toteuttamiseen kuluu (Cohn, 2006, 43).

Cohnin (2006) prosessissa julkaisunsuunnittelu tapahtuu kuvion 8 mukaisissa vaiheissa. *Määritä tyydyttävän lopputuloksen kriteerit* (determine conditions of satisfaction) -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 8 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.)

Kehittäjät *arvioivat käyttäjätarinat* niiden suhteellisen koon mukaan. Arvioinnissa käytetään mittarina joko tarinapisteitä tai ideaalipäiviä. Arviointi tehdään vain niille käyttäjätarinoille, joilla on kohtuullinen todennäköisyys olla mukana tulevassa julkaisussa. Joskus voidaan arvioida myös muutamien seuraavien julkaisujen toivelistalla olevia tarinoita. (Cohn, 2006, 136.)

Valitse iteraation pituus -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, jois-

ta 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. Mitä enemmän ohjelmistonkehitykseen liittyy epävarmuutta, sitä lyhempiä iteraatioita kannattaa käyttää. (Cohn, 2006, 167.)

Priorisoi käyttäjätarinat -vaiheessa tuotteen omistaja asettaa käyttäjätarinat tärkeysjärjestykseen. Priorisointiin vaikuttavia tekijöitä ovat esimerkiksi toiminnallisuudesta saatava liiketoiminta-arvo, kehittämisen kustannus, se syntyykö toiminnallisuuden kehittämisestä merkittävää tietoa kehitystiimille, sekä mitkä riskit vähenevät, jos toiminto kehitetään. (Cohn, 2006, 80).

Arvioi vauhti -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.)

Valitse käyttäjätarinat ja julkaisupäivä -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 määrä 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.)

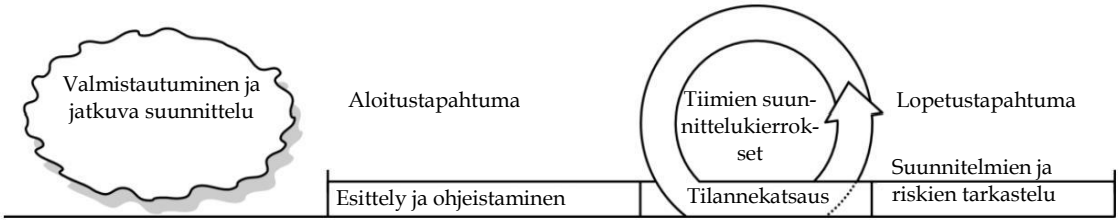
Kun käyttäjätarinat ja julkaisupäivä on valittu, tarkastellaan miten hyvin suunnitelman lopputulos täyttää suunnitelmalle asetetut kriteerit (vrt. kuvio 9). Tarvittaessa suunnitteluprosessi voidaan toistaa. Cohn (2006, 139) suosittelee myös päivittämään suunnitelmaa säännöllisesti, mieluiten joka iteraation alussa.

5.5.3 Julkaisun yhteissuunnittelu -menetelmä

Heikkilä (2010) ja Heikkilä, Rautiainen ja Jansen (2010b) ovat esittäneet julkaisun yhteissuunnittelu -menetelmän (joint release planning method), jossa usea tiimi voi suunnitella julkaisun sisältöä niin, että eri tiimien työlistoissa olevien käyttäjätarinoiden väliset yhteydet huomioidaan ja tiimien tavoitteet ovat yhteneväiset. Myös Leffingwell (2011) on esittänyt hyvin samanlaisen tapahtu-

man. Seuraavaksi tarkastellaan ensin lähestymistapaa Heikkilän (2010) mukaan ja lopuksi kuvataan lyhyesti Leffingwell (2011, 319–338) esitystä.

Heikkilän (2010) julkaisun yhteissuunnittelu -menetelmän kuvauksessa on esitetty suunnittelutapahtumaa edeltäviä, tapahtumien aikaisia ja niiden jälkeen tehtäviä aktiviteetteja. Tapahtuman rakennetta on esitetty kuviossa 9.



KUVIO 9 Julkaisun yhteissuunnittelu (Heikkilä, 2010, 175)

Suunnittelutapahtumaa edeltää *valmistautuminen ja jatkuva suunnittelu* -vaihe. Vaiheessa tehdään mm. van de Weerdin 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 niitä, joilla on realistinen mahdollisuus sopia 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 tila, jossa tiimeille, sidosryhmien edustajille, esiintymispaikoille ja tauoille on omat alueensa. Tapahtuman järjestäjäksi (facilitator) voidaan hankkia tai asettaa henkilö, jolla on kokemusta prosesseihin liittyvästä valmentamisesta, esimerkiksi kokenut Scrum-mestari. (Heikkilä, 2010, 174–178.)

Varsinainen suunnittelutapahtuma alkaa *esittely ja ohjeistaminen* -vaiheella, jossa tiimejä opastetaan tarpeen mukaan julkaisunsuunnittelun arviointi- ja suunnittelutekniikoiden käytössä. Tiimeille painotetaan, että he ovat vastuussa toiminnallisuuksien välisten riippuvuuksien selvittämisestä ja siitä, että tiimien tulisi aktiivisesti olla yhteydessä muihin tiimeihin ja sidosryhmiin avun saami-

seksi. 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 toiminnallisuuksina kuvatut vaatimukset antavat suuntaa kehitystyölle. Eri asiantuntijatahot, kuten arkkitehdit, esittävät tuotteesta oman näkökulmansa mukaisen esityksen. Lukuun ottamatta aivan ensimmäistä suunnittelutapahtumaa tapahtumissa käydään myös läpi ohjelmistoprojektin edistymisaste. (Heikkilä, 2010, 178.)

Suunnittelutapahtuman toisessa, *tiimien suunnittelukierrokset ja tilannekatsaukset,* vaiheessa tiimit jakautuvat omille aluilleen ja aloittavat oman julkaisusuunnitelmansa laatimisen. Tässä yhteydessä tiimit tarkentavat suunnitelmiensa toiminnallisuudet tarkemmiksi käyttäjätarinoiksi, ratkaisevat käyttäjätarinoiden välillä olevat riippuvuudet ja jaottelevat käyttäjätarinat tuleviin iteraatioihin. Tiimit ovat valmiita päivittämään suunnitelmiaan iteratiivisesti sitä mukaan, kun tiimien suunnitelmien käyttäjätarinoiden välillä paljastuu riippuvuuksia. Havaitut ristiriidat suunnitelmien välillä pyritään ratkaisemaan välittömästi ja tarvittaessa tuotepäälliköiden avustuksella. Tapahtuman onnistumisen kannalta on tärkeää, että tiimit saavat eri sidosryhmien edustajilta apua sidosryhmien asiantuntija-alueen ongelmien ratkaisemiseen ja että tiimit ymmärtävät keskinäisen viestinnän tärkeyden. Havaitut riskit on myös tärkeä pyrkiä ratkaisemaan mahdollisimman nopeasti. Vaiheen aikana järjestetään säännöllisesti tilannekatsauksia, joihin osallistuu jokaisesta tiimistä yksi edustaja. Katsauksissa tiimien edustajat kertovat tiiminsä työn edistymisestä ja siitä, millaisia käyttäjätarinoiden riippuvuuksia tiimi on havainnut. (Heikkilä, 2010, 178–180.)

Tapahtuman viimeisessä vaiheessa, *suunnitelmien ja riskien tarkastelussa,* esitetään jokaisen tiimin yleisellä tasolla kuvaama suunnitelma ja esiin nousseet tiimin työhön vaikuttavat riskit. Havaitut riskit käydään yksitellen lävitse ja ratkaistaan tai minimoidaan jollain tavalla (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, tuotehallinta (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.)

Leffingwellin (2011) malli muistuttaa hyvin paljon Heikkilän (2010) esitystä. Leffingwellin (2011) esityksessäkin suunnittelutapahtuma alkaa esittelyllä, jossa tilaisuus avataan ja eri sidosryhmien edustajat kuvaavat omasta näkökulmastaan tuotteen visiota ja esimerkiksi arkkitehtuuriratkaisuja. Tämän jälkeen tiimit laativat yhtenäisessä tilassa tiimikohtaiset suunnitelmat, jonka aikana tiimien edustajat kokoontuvat säännöllisesti tilannekatsaukseen jakamaan tietoa suunnittelun tilanteesta. Lopulta suunnitelmia esitellään ja käydään yhdessä

lävitse. Leffingwellin kuvauksessa tapahtuma jakautuu kahden päivälle. Ensimmäisenä päivänä tehdään luonnostasoiset ja toisena päivänä lopulliset suunnitelmat. Leffingwellin (2011) ja Heikkilän (2010) kuvaukset eroavat jonkin verran yksityiskohdiltaan. Eräänä erona Leffingwell kuvaa hieman Heikkilää yksityiskohtaisemmin riskien käsittelyä. Havaitut riskit pyritään ratkaisemaan heti havaitsemisen jälkeen, mutta jos niitä ei pystytä ratkaisemaan, ne kirjataan ylös myöhempää käsittelyä varten. Selvittämättömät riskit käydään suunniteltutapahtuman toisen päivän lopussa yksitellen läpi ja jokainen riski joko ratkaistaan, valtuutetaan jonkin tiimin valvottavaksi, hyväksytään sellaisenaan ilman toimenpiteitä tai pyritään jollain tavoin minimoimaan. Leffingwell pitää myös tärkeänä, että tiimit sitoutuvat suunnitelmiensa toteuttamiseen. Hän suosittelee, että kukin tiimi äänestää tapahtuman lopussa suunnitelmansa toteutumiskelpoisuudesta. Mikäli tiimi ei näytä luottavan suunnitelmaan, suunnittelutilaisuutta voidaan jatkaa esimerkiksi seuraavana päivänä, kunnes on laadittu suunnitelmat, joihin tiimit voivat sitoutua. (Leffingwell, 2011, 319–338.)

5.5.4 SCERP-menetelmä

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

Julkaisunsuunnitteluun osallistuu tuotepäällikkö ja kriittisten sidosryhmien edustajat. Heikkilän ym. (2010a) 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. työmäärän 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. (Heikkilä ym., 2010a, 3.)

Kun alustavat toiminnallisuudet on valittu, *toiminnallisuudet priorisoidaan* valittujen arviointikriteerien mukaisesti. Arviointikriteereinä voidaan käyttää esimerkiksi toiminnallisuudesta saatava liiketoiminta-arvoa tai sen kiireellisyyt-

tä, 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. (Heikkilä ym., 2010a, 3.)

Työmäärän arviointi -vaiheessa sidosryhmien edustajat, tyypillisesti kehittäjät, arvioivat kunkin toiminnallisuuden vaatiman työmäärän. Arvioinnissa voidaan käyttää jotain tunnettua arviointitekniikkaa, kuten suunnittelupokeria (esim. Grenning, 2002). (Heikkilä ym., 2010a, 4.)

Optimaalisten suunnitelmavaihtoehtojen laskeminen -vaiheessa tuotepäällikkö käyttää Ngo-Then ja Ruhen (2007) esittämää EVOLVE+ -optimointimenetelmää, jonka syötteenä käytetään sidosryhmän edustajien tietoja ja heille asetettuja painoarvoja, alustavien toiminnallisuuksien joukkoa, arvioituja toiminnallisuuksien työmääriä, tietoa siitä kuinka nopeasti ohjelmistoja voidaan kehittää yhden julkaisun aikana ja sidosryhmien kriteeripainotuksia. Tuloksena saadaan viisi optimaalista julkaisusuunnitelmavaihtoehtoa. (Heikkilä ym., 2010a, 4.)

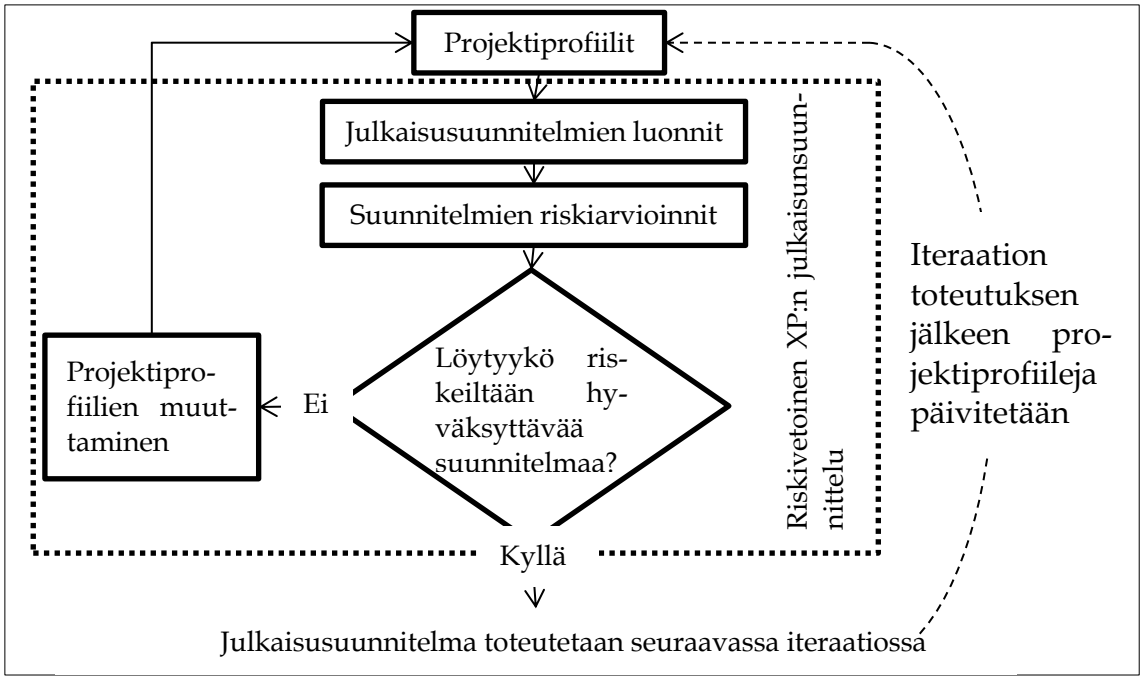
Julkaisusuunnitelmien priorisointi -vaiheessa jokaisen sidosryhmän edustajia pyydetään arvioimaan edellisessä vaiheessa tuotettua viittä suunnitelmaa ja priorisoimaan suunnitelmat samalla tavoin kuin yksittäiset toiminnallisuudet arvioitiin toiminnallisuuksien priorisointi -vaiheessa. Tuloksena saadaan sidosryhmien edustajien suositukset toteutettavasta suunnitelmasta. (Heikkilä ym., 2010a, 4.)

5.5.5 Lin, Huangin, Shun ja Lin menetelmä

Li, Huang, Shu ja Li (2006) esittävät menetelmän eXtreme programming (XP) -menetelmän yhteyteen, missä huomioidaan suunnitelmiin liittyvät riskit. Menetelmästä rajaudutaan tässä yhteydessä vain itse lähestymistavan vaiheiden kuvaamiseen (Li ym., 2006, 424–426), ei sen liittämistä XP-menetelmän mukaiseen prosessiin. Menetelmän kuvauksessa julkaisujakson pituutena esitetään yksi iteraatio, mutta menetelmää voidaan käyttää yhtä hyvin useammankin iteraation sisältävän julkaisun suunnitteluun.

Menetelmän neljä vaihetta ovat: toteutuskelpoisten julkaisusuunnitelmien luonti, suunnitelmien riskiarvioinnit, vaihe, jossa päätetään löytyykö tehdyistä suunnitelmista riskeiltään hyväksyttävää suunnitelmaa ja projektiprofiilien muuttaminen (kuvio 10). Vaiheita edeltää projektiprofiilien määrittely. Projektiprofiileilla tarkoitetaan projektin toteutuskriteerejä, jotka voivat koskea esimerkiksi järjestelmän laajuutta, hintaa, aikataulua ja tuotteen laatua. (Li ym., 2006. 424.)

Julkaisunsuunnitelmien luonti -vaiheessa tehdään haluttu määrä toteutumiskelpoisia julkaisusuunnitelmia erityistä algoritmia käyttäen. Toteutumiskelpoisuus tarkoittaa, että suunnitelmissa ovat mukana ne vaatimukset, joiden toteutus on arvioitu mahdolliseksi suunnitelmaa koskevan julkaisujakson



KUVIO 10 Riskivetoinen XP:n julkaisunsuunnittelu (Li ym., 2006, 424)

aikana. Algoritmin syötteinä huomioidaan kehittäjien määrittelemät vaatimusten koot tunteina, vaatimusten väliset riippuvuudet ja suunniteltavan kehitysjakson kehitysvauhti tunteina. Lisäksi syötteenä toimivat asiakkaiden määrittelemät vaatimusten liiketoiminta-arvot. Li ym. (2006) ehdottavat käyttämään liiketoiminta-arvon määrittelyssä erityistä AHP-tekniikkaa (Saaty, 1980), jonka avulla vaatimuksille voidaan määrittää suhteelliset liiketoiminta-arvot. (Li ym., 2006, 425.)

S*uunnitelmien riskiarvioinnissa* tunnistetaan ensin olemassa olevat riskit. Li ym. (2006) käyttävät riskien tunnistamiseen taulukossa 4 esitettyä taksonomiaa.

TAULUKKO 4 Riskien taksonomia (Li ym., 2006, 426)

Riskin tyyppi	**Riski**	**Kuvaus**
Vaatimuksiin liittyvät riskit	Epävakaa vaatimus	Epävakaan liiketoimintaympäristön vuoksi vaatimus voi muuttua
	Epämääräinen tarina	Tarina on epäselvä liiketoiminnan tavoitteiden tai järjestelmän suunnittelun näkökulmasta
Arviointiin liittyvät riskit	Koko	Tarinan koko on arvioitu väärin
	Tiimin tuottavuus	Tiimin tuottavuus on arvioitu väärin
Teknologiaan liittyvät riskit	Arkkitehtuuriristiriidat	Miten uudet tarinat liitetään nykyiseen arkkitehtuuriin?
	Vaikea implementointi	Miten tarinat implementoidaan?
Henkilöstöön liittyvät riskit	Asiakas	Asiakkaat eivät tunne toimialan liiketoimintaa riittävän hyvin

Riskien tunnistamisen jälkeen arvioidaan riskien todennäköisyyksiä ja niiden mahdollisista toteutumisista aiheutuvia tappioita asteikolla pieni, kohtalainen

ja suuri. Lin ym. (2006, 426) mukaan tappioita voidaan arvioida toteutuneen riskin vaikutuksina kehitettävän julkaisun laajuuteen, aikatauluun ja laatuun. Taulukossa 5 on esitelty, miten näiden arvioiden mukaan voidaan pohtia riskin vakavuutta (risk exposure).

TAULUKKO 5 Laadullinen menetelmä riskien vakavuuden arvioon (Li ym., 2006, 426)

Riskien vakavuus		**Todennäköisyys**		
		Pieni	**Kohtalainen**	**Suuri**
Aiheutuva tappio	**Pieni**	Vähäinen	Merkittävä	Kriittinen
	Kohtalainen	Merkittävä	Kriittinen	Ei hyväksyttävä
	Suuri	Kriittinen	Ei hyväksyttävissä	Ei hyväksyttävissä

Kun riskien vahinkoalttiudet on laskettu, jokaiselle riskille annetaan sen vakavuuden mukainen arvo. Vakavuudeltaan pieniksi arvioiduille riskeille annetaan arvo 1, merkittäville arvo 2, kriittisille arvo 3 ja ei hyväksyttäville riskeille arvo 4. Lopuksi suunnitelman sisältämien riskien arvot lasketaan yhteen, jolloin saadaan suunnitelmakohtaiset riskiarvot.

Kolmannessa vaiheessa verrataan luotujen suunnitelmien riskejä ja ratkaistaan, *löytyykö riskeiltään hyväksyttävää suunnitelmaa*. Lin ym. mukaan projektin alkuvaiheessa voidaan hyväksyä herkemmin sellaisia suunnitelmia, joissa laajuuteen liittyvät riskit ovat suuria, jotta liiketoimintatavoitteita päästämään mahdollisimman nopeasti selkeyttämään. Projektin loppuvaiheessa sen sijaan on tärkeämpi valita alhaisen riskin suunnitelmia, jotta julkaisu pystytään pitämään aikataulussaan. (Li ym., 2006, 426).

Mikäli jokin suunnitelmista hyväksytään, ohjelmistoa ryhdytään toteuttamaan suunnitelman mukaan, mutta muussa tapauksessa *projektiprofiileja muutetaan* yhteistyössä asiakkaan kanssa havaittujen riskien pienentämiseksi.

5.5.6 Lin, van den Akkerin, Brinkkemperin ja Diepenin malli

Li, van den Akker, Brinkkemper ja Diepen (2010) esittävät julkaisunsuunnittelun mallin, jossa yhdistyy vaatimusten valinta ja aikataulutus. Malli sopii erityisesti tilanteisiin, joissa samaa julkaisua kehitetään usean tiimin voimin. Li ym. näkevät useiden tiimien kehitysprojekteissa ongelmalliseksi sen, että tiimit voivat joutua vaatimusten välisten riippuvuuksien vuoksi odottamaan muiden tiimien työn valmistumista ennen kuin he pystyvät jatkamaan omaa kehittämistään. Mallin avulla voidaan luoda sisällöltään aikataulutettu julkaisusuunnitelma, jonka toteuttamisessa tiimit joutuvat odottamaan mahdollisimman vähän aikaa toistensa töiden valmistumista ja jossa julkaisun tuoma liiketoimintaarvo on samalla mahdollisimman suuri. Lin ym. esityksessä ei painoteta mallin soveltuvuutta ketterän lähestymistavan yhteyteen, mutta siinä kuvataan Scrumia yhtenä mekanismina, jolla mallissa voidaan mukautua paremmin muutoksiin ja suunnitelmassa käytettyjen lähtötietojen ali- tai yliarviointiin. (Li ym., 2010, 377–378.)

Li ym. (2010) kuvaavat kaiken kaikkiaan kolme matemaattista ILP (Integer Linear Programming) -mallia (Li ym., 2010, 378). Tässä tutkimuksessa keskitytään kuvaamaan näistä viimeistä mallia, joka yhdistää kahden muun mallin vaatimusten valinta- ja aikataulutusominaisuudet.

Lin ym. (2010) mallin syötteinä käytetään vaatimusten odotettua tuottoa (revenue) ja kustannusta (cost), tietoja vaatimusten välisistä riippuvuuksista, tiimien määrää ja tiimikohtaista kehityskapasiteettia sekä projektin pituutta. Vaatimusten tuotto arvioidaan absoluuttisesti, esimerkiksi euromääräisesti, mutta Li ym. mukaan myös suhteellisia arvolukuja voidaan käyttää. Vaatimuksiin liittyvät kehitystehtävät päätetään jo siinä vaiheessa, kun ne kerätään (Li ym., 2010, 377). Näin vaatimukset voidaan jakaa pienempiin osiin eli töihin (jobs), joita tiimit voivat itsenäisesti kehittää toisistaan riippumatta. Jokaisen työn kustannukset ja tiimien kehityskapasiteetti arvioidaan työpäivinä. Mallissa oletetaan, että vain yksi tiimi toteuttaa yhtä työtä ja tiimit suorittavat aina aloittamansa työn toteutuksen alusta loppuun (suunnittelemalla, toteuttamalla ja testaamalla sen) tekemättä välissä muihin töihin liittyviä toteutustehtäviä. Lisäksi oletetaan, että tiimit eivät voi aloittaa sellaisten vaatimusten toteuttamista, jotka riippuvat vielä toteuttamattomista vaatimuksista. Malli toimii määräaikalähtöisesti eli siinä oletetaan, että projektilla on jokin valmistumisaikatavoite. (Li ym., 2010, 377–381.)

Mallissa on vaatimusten riippuvuuksia jaoteltu kuuteen kategoriaan. *Yhdistelmäriippuvuudessa* (combination) kaksi vaatimusta on kehitettävä yhtäaikaisesti. *Seuraamusriippuvuudessa* (implication) vaatimus vaatii toimiakseen sen, että tietty vaatimus on aiemmin toteutettu. *Pois rajaavassa riippuvuudessa* (exclusion) kahdesta vaatimusta vain toinen voidaan toteuttaa eli riippuvuusparin vaatimuksen toteuttaminen rajaa pois toisen vaatimuksen toteuttamisen. *Tulo-* (revenue-based) ja *kustannusperusteiset* (cost-based) *riippuvuudet* liittyvät siihen, että jonkin toisen vaatimuksen toteuttamisella voi olla vaikutusta vaatimuksen tuottamaan arvoon tai sen kustannuksiin. *Aika riippuvuudessa* (time-related) vaatimus on toteutettava ajallisesti tietyn vaatimuksen jälkeen. Lin ym. mukaan edellisistä riippuvuuksista viisi ensin mainittua ovat oleellisia vaatimusten valinta -mallissa ja vaatimusten aikataulutuksessa huomioitavia vaatimuksia ovat seuraamukseen, kustannuksiin ja aikaan liittyvät rajoitteet. (Li ym., 2010, 378–379.)

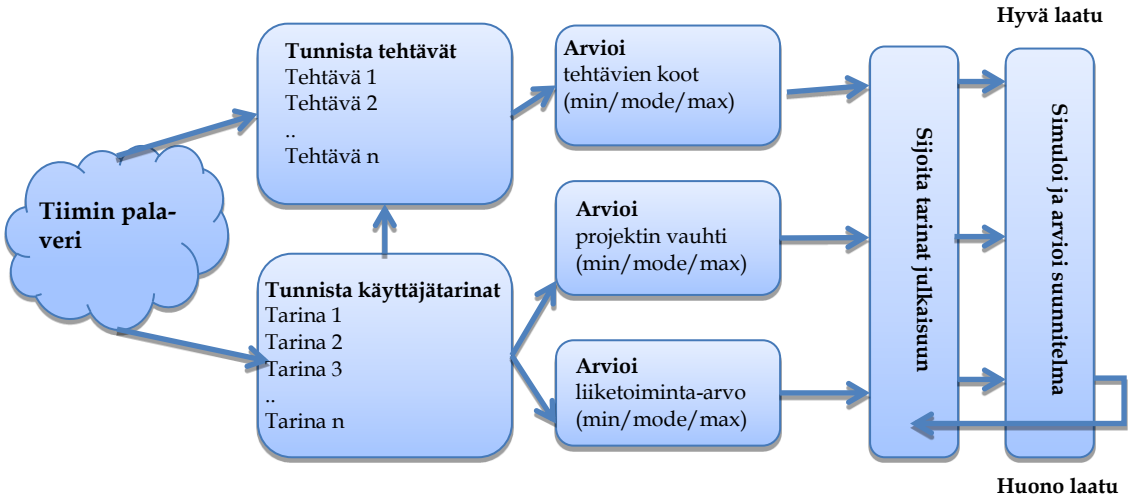
Lin ym. (2010) esittävät myös, miten heidän lähestymistavassaan suunnitelmaa voidaan päivittää, jos jotkut mallin syöttötietoina toimivat tekijät, kuten vaatimusten koko, liiketoiminta-arvo tai vaatimusten välisten riippuvuudet, tarkentuvat ja muuttuvat. Julkaisusuunnitelma voidaan laatia niin, että vaatimukset on jaettu aikataulutettuina valmiiksi iteraatioihin. Esityksen mukaan tässä suunnittelussa pyritään valmiiksi iteraatiotasolla optimaaliseen suunnitelmaan. Iteraatioiden välissä suunnitelmaa voidaan päivittää tarkentuneilla syöttötiedoilla uudelleen suunnitelman tarkentamiseksi. Li ym. pitävät lähestymistapansa ja sen oletuksien suhteen ongelmallisena sitä, että suurikokoiset vaatimukset eivät välttämättä sovi Scrumin lyhyisiin iteraatioihin. Tästä syystä he suosittelevatkin, että Scrumia käytettäessä suurikokoiset vaatimukset tar-

kennetaan riittävän yksityiskohtaiselle tasolle esimerkiksi Vlaanderenin ym. (2011) esityksen mukaisesti. (Li ym., 2010, 393–394.)

Mallin yhteydessä ei esitellä tarkasti millaisen prosessin osana se toimii. Matemaattisen mallin syötteet kuvataan ikään kuin annettuina ja niistä todetaan vain, että ennen mallin käyttöä ne on oltava määriteltynä. Rooleista mainitaan julkaisunsuunnittelussa päätöksiä tekevä tuotepäällikkö (Li ym., 2010, 392).

5.5.7 Loguen ja McDaidin menetelmä

Logue ja McDaid (2008a) esittävät julkaisunsuunnittelumenetelmän, jossa tilastollista menetelmää hyödyntäen tuotetaan joukko suunnitelmia, joihin käyttäjätarinat on sijoitettu optimaalisella tai lähes optimaalisella tavalla. Menetelmän etuna on se, että se sallii epävarmuuden käyttäjätarinoiden koon ja niiden tuottaman liiketoiminta-arvon sekä projektin vauhdin suhteen. (Logue & McDaid, 2008a, 437.) Käyttäjätarinoiden koko arvioidaan joko tarinapisteiden tai ideaalisten päivien mukaan. Loguen ja McDaidin (2008a) artikkeli ei selitä tarkemmin, soveltuuko prosessi pidemmän ajan kuin seuraavaan julkaisun suunnitteluun. Kuviossa 11 on kuvattu menetelmän vaiheet. Vaiheet on selitetty jäljempänä.



KUVIO 11 Loguen ja McDaidin (2008a, 439) menetelmä

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

Kun edellä mainitut arvioinnit on tehty, tuotteen omistaja *sijoittaa käyttäjätarinat julkaisuun*. Tämän jälkeen tehtävien aika-arvioiden perusteella suorite-

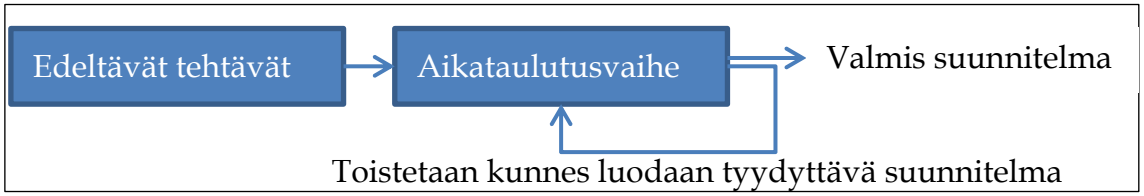
taan erityinen Monte Carlo -simulaatio, jonka tuloksena saadaan jakauma, joka osoittaa minä ajankohtana julkaisu saadaan kaikkein todennäköisimmin kehitettyä. Käytettyä menetelmää on tarkasteltu hieman tarkemmin Loguen ja McDaidin (2008a) artikkelissa, mutta menetelmän yksityiskohdat sivuutetaan tässä esityksessä. Simulointimenetelmää käytetään myös projektin työskentelyvauhdin arviointiin, jolloin saadaan laskettua todennäköinen aika julkaisun valmistumiselle kalenteripäivissä. Lopuksi simulointimenetelmää käytetään julkaisuun valittujen käyttäjätarinoiden todennäköisen liiketoiminta-arvon laskemiseen. (Logue & McDaid, 2008a, 439–441.)

Edellä esitetty menetelmä ei tähtää suoraan optimaalisen julkaisusuunnitelman tekemiseen, vaan sen avulla tuotteen omistaja voi verrata keskenään erilaisten käyttäjätarinavalintojen liiketoiminta-arvoa ja niiden todennäköistä kehitysaikaa keskenään. Näin ollen esitetty lähestymistapa voidaan nähdä julkaisunsuunnittelumenetelmän lisäksi päätöksenteon tukivälineenä.

5.5.8 Szoken malli

Szoke (2011b) esittää yhden tiimin kehitysorganisaatioille ketterään kehittämiseen soveltuvan julkaisunsuunnittelun matemaattisen optimointimallin. Mallissa yhdistyvät julkaisunsuunnittelu, jonka Szoke näkee koskevan vaatimusten julkaisuun valitsemista, sekä julkaisun aikataulutus (release scheduling), joka Szoken mukaan koskee vaatimusten sijoittamista julkaisusuunnitelman iteraatioihin (Szoke, 2011b, 8). Szoken (2011b, 37) mukaan julkaisun aikataulutusongelma on erityisen usean kapsäkin ongelman (multiple knapsack problem) laajennus, jota Szoke kutsuu ketteräksi julkaisun aikataulutusongelmaksi (ARSP, Agile Release Scheduling Problem). Mallia voidaan käyttää joko tiettyyn määräaikaan sidotun suunnitelman laatimiseen, jolloin pyritään maksimoimaan julkaisujakson aikana toteutuva liiketoiminta-arvo, tai laajuuteen sidotusti, jolloin tietty vaatimusten laajuus pyritään kehittämään mahdollisimman lyhyessä kehitysajassa. Toiminnan lopputuloksena syntyy julkaisusuunnitelma, jossa toiminnallisuudet on sijoitettu iteraatioihin. Mallissa esitetään kehittäjien ja projektipäällikön roolit (Szoke, 2011b).

Szoken (2011b, 10) mallin mukaisen toiminnan voidaan nähdä koostuvan aikataulutusta edeltävistä tehtävistä ja aikataulutusvaiheesta (kuvio 12).



KUVIO 12 Szoken (mukaillen 2011b, 10) mallin päävaiheet

Edeltävinä tehtävinä päätetään julkaisun pituus päivinä ja kehittäjien määrä sekä kunkin kehittäjän tehokkuuskerroin. Kerroin määräytyy sen mukaan, missä suhteessa työntekijä voi käyttää aikaansa projektiin (Szoke, 2011b, 9). Lisäksi

arvioidaan vaatimusten prioriteetit ja työmäärä päivinä sekä vaatimusten väliset riippuvuudet. Prioriteetit arvioidaan suhteellisesti jakamalla vaatimuksen arvo sen arvioidulla työmäärällä, jolloin vaatimukset voidaan asettaa toisiinsa nähden tärkeysjärjestykseen (Szoke, 2011b, 8). Riippuvuuksista määritellään toiminnallisuuksien väliset sidosriippuvuudet (coupling dependency), jotka on toteutettava yhtäaikaisesti, ja edeltäjyysriippuvuudet (precedence dependency), joilla tarkoitetaan sitä, että jokin vaatimus tulisi toteuttaa ennen toista vaatimusta (Szoke, 2011b, 11–12). Edeltäjyysriippuvuudella tarkoitetaan samanlaisista seuraamus-, tulo- ja kustannusperusteista riippuvuutta kuin mitä Lin ym. (2010) mallin yhteydessä esiteltiin aiemmin. Szoken (2011b, 38) mukaan menetelmä eroaa muista optimointimenetelmistä siinä, että vaatimusten riippuvuustilanteissa vaatimukset, joiden välillä on edeltäjyysriippuvuus, voidaan sijoittaa samaan iteraatioon.

Aikataulutusvaiheessa päätetään iteraatioiden pituus ja määrä sekä kehitystiimin koko. Vaiheessa määritellään myös iteraatioiden vauhti joko aiemman kehitysvauhdin mukaan tai laskemalla vauhti kehittäjien määrän ja heidän osallistumismahdollisuutensa mukaan. Kerättyjen tietojen avulla vaiheessa käytetään formaalia optimointimallia ja erityistä algoritmia aikataulutetun julkaisusuunnitelman luomiseksi. Aikataulutusvaihetta voidaan toistaa iteratiivisesti, kunnes suunnitelmaan ollaan tyytyväisiä. (Szoke, 2011b, 8–10.)

5.5.9 van Valkenhoefin ym. malli

van Valkenhoef, Tervonen, de Brock & Postmus (2011) esittävät matemaattiseen optimointiin perustuvan mallin, joka on tarkoitettu tukemaan eXtreme programming (XP) -menetelmän (Beck, 1999) yhteydessä tehtävää julkaisunsuunnittelua. Mallin kuvauksessa on esitetty XP:n rooleja ja miten nämä roolit (asiakas, kehittäjät ja jäljittäjä) osallistuvat mallin syöttötietojen tuottamiseen. Mallin avulla voidaan luoda julkaisusuunnitelma, jonka toteutus maksimoi saavutetun liiketoiminta-arvon ja jossa erilaiset kapasiteettirajoitteet, kehittämisnopeuteen liittyvä epävarmuus ja vaatimusten järjestykseen liittyvät tekijät on otettu huomioon. (van Valkenhoef ym., 2011, 1228). Mallin syötteet (ja niiden tuottamisesta vastaavat roolit) ovat (van Valkenhoef ym., 2011, 1229.)

- käyttäjätarinoiden ja teemojen liiketoiminta-arvot (asiakas)
- käyttäjätarinoiden koot (kehittäjät)
- toiveet vaatimusten toteutusjärjestyksistä (kehittäjät)
- tiedot vaatimusten teknisistä toteutusjärjestyksistä (kehittäjät)
- kehittämisvauhdin ennuste (jäljittäjä).

Malli käyttää syötteenä kahdenlaisia vaatimuksia, teemoja ja käyttäjätarinoita. *Teemalla* tarkoitetaan tässä yhteydessä vaatimusta, joka on luotu liittämällä yhteen toisiinsa läheisesti liittyviä käyttäjätarinoita. van Valkenhoef ym. neuvovat arvioimaan käyttäjätarinoiden koon ja liiketoiminta-arvon suhteessa muihin aiemmin arvioituihin ja nykyisiin jo arvioituihin käyttäjätarinoihin sekä käyt-

tämään arvioinnissa asteikkoa 1–5. Teeman liiketoiminta-arvo arvioidaan korkeammaksi kuin mikä on teeman sisältämien käyttäjätarinoiden arvojen summa, sillä teeman toteuttamisesta syntyy van Valkenhoef ym. mukaan synergiaetuja suhteessa siihen, että vain jotkin kyseiseen teemaan liittyvistä käyttäjätarinoista toteutettaisiin. van Valkenhoef ym. (2011, 1231) esittävät tutkimuksessaan kolme menetelmää teemojen arvottamiseen. Malli käyttää syöttötietoina myös kahdenlaisia tietoja vaatimusten välisistä toteutusjärjestyksistä. *Toiveet vaatimusten toteutusjärjestyksestä (preference precedence)* liittyvät siihen, että jotkin vaatimukset tuottavat arvoa vasta sen jälkeen, kun jokin aiempi vaatimus on toteutettu. Tästä syystä asiakas voi haluta määritellä toivottuja toteutusjärjestyksiä vaatimuksille. *Tekniset syyt vaatimusten toteutusjärjestykseen* liittyvät siihen, että jotkut vaatimukset on toteutettava ennen kuin tiettyjä muita vaatimuksia voidaan toteuttaa. *Kehittämisvauhdin ennusteella (velocity prediction)* viitataan erityiseen log-normal -jakaumaan, joka antaa ennusteen käyttäjätarinoiden määrästä, jotka voidaan toteuttaa julkaisujakson aikana tietyillä todennäköisyyksillä. Jakauman käytöllä pyritään huomioimaan nopeuden arviointiin liittyvä epävarmuus. van Valkenhoef ym. esittävät yhden menetelmän ja peukalosääntöjä jakauman luomiseksi. (van Valkenhoef, 2011, 1228–1231.)

Mallin tuloksena saadaan joukko käyttäjätarinoita, jotka on jaettu valmistumistodennäköisyytensä mukaan ryhmiin. Jokaiselle ryhmälle on määritelty etukäteen tietty arvo väliltä [0,1] liittyen todennäköisyyteen, jolla vaatimukset pystytään toteuttamaan tulevan julkaisujakson aikana. Esimerkiksi arvo 0.9 merkitsee, että 90 prosentin todennäköisyydellä ryhmän vaatimukset pystytään toteuttamaan seuraavaan julkaisuun. Hyvin tärkeiden vaatimusten joukolle voidaan asettaa arvo 0.9, seuraavaksi tärkeimmille vaatimuksille 0.7 ja kolmannelle luokalle 0.3. Luotu julkaisusuunnitelma toimii varsinaisen julkaisunsuunnittelutapahtuman tukena. (van Valkenhoef ym., 2011, 1229.)

5.6 Ehdotusten vertailu

Tässä alaluvussa vertaillaan edellä kuvattuja ehdotuksia (menetelmiä, malleja, prosesseja) neljällä tavalla. Ensin verrataan ehdotusten *yleisiä piirteitä.* Toiseksi verrataan ehdotusten sisältämien *aktiviteettien kattavuutta suhteessa Bekkersin ym. (2010) kompetenssimallin* julkaisunsuunnitteluosan tehtäviin. Tässä vertailussa tarkastellaan myös ehdotusten niitä *aktiviteetteja, jotka eivät sisälly Bekkersin ym.(2010) kompetenssimalliin*. Vertailujen tarkoituksena on saada tietoa, millaisia vaiheita ehdotuksiin sisältyy. Kolmanneksi vertaillaan ehdotusten sisältämiä *vaatimusten valintatekijöitä* käyttämällä pohjana Svahnbergin ym. (2010, 241–244) valintatekijöiden taksonomiaa (vrt. luku 5.1). Tämä vertailu tehdään vain niille ehdotuksille, joissa on mukana järjestelmällistä suunnittelua. Vertailun tavoitteena on antaa kuvaa formaalien tai hybridi-menetelmien piirteistä. Vertailun neljännessä osassa pohditaan, *millä tavoin ehdotukset sopivat ketterään kehittämiseen*. Pohdinnassa kiinnitetään huomiota tutkimuksessa esiinnousseisiin järjes-

telmällisen suunnittelun ja ketterän kehittämisen ristiriitakohtiin ja Scrumin kuvaukseen.

5.6.1 Yleiset piirteet

Tässä osiossa tarkastellaan ehdotusten formaaliutta, rooleja, käsiteltävien vaatimusten tarkkuustasoa ja tuloksia. Yhteenveto vertailusta on esitelty taulukossa 6.

Julkaisunsuunnittelun *formaaliusasteeltaan* Cohnin (2006) ja Heikkilän (2010) ehdotukset ovat tulkittavissa tyypiltään arviointiin perustuvaksi. Lin ym. (2010) ja Szoken (2011b) ehdotukset edustavat formaaliusasteeltaan järjestelmällistä suunnittelua. Loput ehdotukset ovat hybridi-muotoisia, sillä niissä yhdistyi sekä arviointiperustaiseen että järjestelmälliseen suunnitteluun nojaavan suunnittelun piirteitä. Hybrideistä ehdotuksista Lin ym. (2006) ja van Valkenhoefin ym. (2011) esityksien mukaan suunnitelmat tehdään optimointimallia hyväksikäyttäen, minkä jälkeen suunnitelmia arvioidaan tai käytetään varsinaisen suunnittelun tukena. Heikkilän ym. (2010a) menetelmässä suunnitelmat luodaan optimointimallilla, mutta optimointimallin käyttöä ennen ja sen jälkeen sidosryhmien edustajat osallistuvat suunnitteluprosessiin. Loguen ja McDaidin (2008a) menetelmässä suunnitelmat tehdään manuaalisesti, jonka jälkeen suunnitelmien arvioinnissa käytetään tilastollista optimointimallia. Rajanveto järjestelmällisten ja hybridien ehdotusten välille on hankalaa. Järjestelmällisiksi ehdotuksiksi katsotaan ne ehdotukset, joissa oletetaan, että valmis suunnitelma joko hyväksytään tai hylätään sellaisenaan, kun se on tuotettu jonkinlaisella optimointimallilla tai matemaattisella algoritmilla.

Ehdotuksissa yleisimpänä *roolina* esiintyvät kehittäjät, jotka ovat mukana seitsemässä ehdotuksessa. Tuotteen omistaja ja tuotepäällikkö esiintyvät kolmessa ehdotuksessa. Asiakkaan ja erilaisten sidosryhmien/asiantuntijoiden rooli mainitaan kahdessa esityksessä. On huomioitava, että sidosryhmillä voidaan tarkoittaa myös asiakkaita.

Useimmat ehdotukset käyttivät suunnitelmien luonnissa käyttäjätarina- tai toiminnallisuustasoisia *vaatimuksia*. Poikkeuksena on Lin ym. (2010) malli, jossa esitellään vain yleisesti vaatimukset ja vaatimuksista tarkennetut työt. Li ym. (2010) kuvaa töitä vaatimusten pienempinä osina, jotka tiimit voivat itsenäisesti kehittää. Toinen poikkeus Loguen ja McDaidin (2008a) menetelmä, missä käyttäjätarinat tarkennetaan suunnittelun aikana tehtäviksi.

Ehdotuksissa *tuloksina* syntyy julkaisusuunnitelmia, jotka vaihtelevat valmiusasteiltaan ja laajuudeltaan, lukumääriltään ja tarkkuudeltaan. Yhden tiimin suunnitelmat voidaan luoda Cohnin (2006) tai Szoken (2011b) ja useiden tiimien Heikkilän (2010) ja Lin ym. (2010) ehdotuksilla. Lin ym. (2010) mallissa tulos on kuvattu tarkimmalla tasolla, sillä siinä vaatimukset on aikataulutettu iteraatioittain. Samoin Loguen ja McDaidin (2008a) menetelmän tuloksena syntyy tehtävinä kuvattu suunnitelma. van Valkenhoefin ym. (2011) mallin mukaan tuloksena syntyy tärkeysryhmiin priorisoitu suunnitelmaehdotus, ja Lin ym. (2006) sekä Heikkilän ym. (2010a) tavoilla voidaan luoda useita suunnitelmia.

TAULUKKO 6 Ehdotusten yleispiirteitä

Ehdotuksen piirre	**Cohn, 2006**	**Heikkilä, 2010**	**Heikkilä ym., 2010a**	**Li ym., 2006**	**Li ym., 2010**	**Logue & McDaid, 2008a**	**Szoke, 2011b**	**van Valkenhoef ym., 2011**
Formaalius-aste	Arviointiin perustuva	Arviointiin perustuva	Hybridi	Hybridi	Järjestelmäl-linen	Hybridi	Järjestelmäl-linen	Hybridi
Roolit	Kehittäjät, tuotteen omistaja	Kaikki Scrum-roolit, tuo-tepäällikkö, asiantunti-jat, järjestäjä	Kehittäjät, tuotepääl-likkö, sidos-ryhmät	Kehittäjät, asiakas	Tuotepääl-likkö	Kehittäjät, tuotteen omistaja	Kehittäjät, projekti-päällikkö	Kehittäjät, asiakas, jäljittäjä
Vaatimuk-set	Käyttäjä-tarinat	Toiminnalli-suudet, käyttäjätari-nat	Toiminnalli-suudet	Käyttäjä-tarinat	Vaatimukset ja työt	Käyttäjäta-rinat, tehtä-vät	Toiminnalli-suudet	Teemat, käyttäjäta-rinat
Tulos	Julkaisu-suunnitelma	Usean tii-min julkai-susuunni-telmat	Usea suun-nitelma	Riskiarvioi-dut suunni-telmat	Iteraatio-suunnitel-mat	Arvioidut suunnitel-mat	Aikataulu-tettu julkai-susuunni-telma	Priorisoidut käyttäjäta-rinat

5.6.2 Vertailu Bekkersin ym. malliin

Tässä alaluvussa käytetään ehdotusten vertailuun Bekkersin ym. (2010) prosessimallia. Malli on kuvattu kohdassa 5.5.1. Ensiksi tutkitaan, mitkä Bekkersin ym. (2010) mallin aktiviteetit löytyvät ehdotuksista. Yhteenveto tuloksista on esitetty taulukossa 7. Toiseksi selvitetään, mitä Bekkersin ym. (2010) mallin ulkopuolelle jääviä aktiviteetteja ehdotuksissa on. Yhteenveto tästä on esitetty taulukossa 8. Bekkersin ym. (2010) mallissa kuvattuja julkaisunsuunnittelun vaiheita ovat:

- vaatimusten priorisointi
- julkaisun määrittely
- julkaisun määrittelyn validointi
- laajuuden muutosten hallinta
- julkaistavan version validointi
- jakeluun valmistautuminen.

Vaatimusten priorisointi on tunnistettavissa jokaisesta kahdeksasta ehdotuksesta. Cohnin (2006) prosessimallin mukaan priorisoinnissa voidaan ottaa huomioon esimerkiksi käyttäjätarinoiden liiketoimintahyöty ja kustannukset. Lin ym. (2006), Lin ym. (2010), van Valkenhoefin ym. (2011) ja Szoken (2011b) ehdotuksissa priorisointiin vaikuttavat sellaiset tekijät kuin vaatimusten riippuvuudet, arvot ja koot. Loguen ja McDaidin (2008a) menetelmän priorisointitekijänä mainitaan liiketoiminta-arvo. SCERP-menetelmässä (Heikkilä ym., 2010a) sidosryhmän edustajat osallistuvat priorisointiin arvioiden priorisointitekijöitä kuten liiketoiminta-arvoa, kiireellisyyttä ja vaatimuksen toteuttamattomuudesta syntyvän haittaa. Julkaisunyhteissuunnittelu -menetelmässä (Heikkilä, 2010) ei ole kuvattu tarkkaan millä kriteerein priorisointi tehdään, mutta kuvauksen mukaan se tehdään tuotteen omistajan ja tuotepäällikön yhteistoiminnalla sekä tiimien yhteistyössä.

Kaikki kuvatut ehdotukset sisältävät *julkaisun määrittelyn*. Osassa ehdotuksissa (Cohn, 2006; Heikkilä, 2010; Heikkilä ym., 2010a) käytetään vaatimusten esivalintaa, jolloin suunnitteluprosessiin otetaan mukaan vain ne vaatimukset, joilla on realistinen mahdollisuus olla mukana tulevissa julkaisuissa. Loguen ja McDaidin (2008a) menetelmässä tuotteen omistaja valitsee julkaisuun sijoitettavat vaatimukset. Lin ym. (2006), Lin ym. (2010, Szoken (2011b) ja van Valkenhoefin ym. (2011) ehdotuksissa vaatimusten valinta tehdään formaalien optimointimallien ja algoritmien avulla.

Julkaisun määrittelyn validoinnin suhteen on arvioitu, otatetaanko ehdotuksissa kantaa siihen, millä tavoin valmis suunnitelma hyväksytetään yrityksen sisäisillä sidosryhmillä. Cohnin (2006) sekä Loguen ja McDaid (2008a) ehdotuksissa viitataan siihen, että tuotteen omistaja päättää suunnitelman valitsemisesta. Heikkilän (2010) menetelmästä syntyy käsitys, että suunnitelma laaditaan

TAULUKKO 7 Ehdotusten kattavuuden vertailu Bekkersin ym. (2010) mallin avulla

Julkaisun-suunnittelun vaiheet	**Cohn, 2006**	**Heikkilä, 2010**	**Heikkilä ym., 2010a**	**Li ym., 2006**	**Li ym., 2010**	**Logue & McDaid, 2008a**	**Szoke, 2011b**	**van Valkenhoef ym., 2011**
Vaatimusten priorisointi	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa
Julkaisun määrittely	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa
Julkaisun määrittelyn validointi	Kattaa	Kattaa	Ei kata	Ei kata	Ei kata	Kattaa	Ei kata	Ei kata
Laajuuden muutosten-hallinta	Kattaa	Kattaa	Ei kata	Kattaa	Kattaa	Kattaa	Kattaa	Kattaa
Julkaistavan version validointi	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata
Jakeluun valmistau-tuminen	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata	Ei kata

yhteisessä julkaisunsuunnittelutapahtumassa, jossa myös yrityksen sisäisiä sidosryhmän edustajia (ml. tuotteen omistajat ja tuotepäälliköt) on mukana. Tätä käsitystä vahvistaa myös Leffingwellin kuvaus Heikkilän (2010) menetelmää vastaavasta yhteissuunnittelusta, jossa julkaisutapahtuman päätteeksi tiimit äänestävät sitoutumisestaan suunnitelmaan. Edellä mainittujen ehdotusten suhteen julkaisun määrittelyn validointi on arvioitu tästä syystä kattavaksi. Muissa ehdotuksissa tehtävään ei ole otettu selkeästi kantaa.

Eräänlainen *laajuuden muutosten hallinta* -mekanismi voidaan nähdä sisältyvän lähes jokaiseen (7/8:sta) tarkasteltuun ehdotukseen iteraatiotason kehittämisen muodossa. Ehdotuksissa oletetaan, että ohjelmistoa kehitetään iteraatioissa ja jokaisen iteraation jälkeen julkaisusuunnitelmia voidaan päivittää. Julkaisun yhteissuunnittelu -menetelmän (Heikkilä, 2010) yhteydessä tuodaan myös esiin, miten tärkeää on usean tiimin projektissa pitää tietoa esillä tiimien etenemisestä ja, että tuotepäälliköt rajaavat tarpeen mukaan kehitettävän ohjelmiston laajuutta, jos työ ei edisty alkuperäisten suunnitelmien mukaan. Vain SCERP-menetelmässä (Heikkilä ym., 2010a) ei tuoda esiin iteraatiotason kehittämisen näkökulmaa eikä siinä oteta muullakaan tavoin kantaa muutoksien hallintaan. Tämä ei kuitenkaan tarkoita, etteikö menetelmän yhteydessä ohjelmistoa voitaisi kehittää iteraatioissa. Lin ym. (2010) malli sisältää iteraatioissa tehdyn kehittämisen, mutta ei varsinaista iteraatiotason suunnittelua.

Mikään vertailussa mukana ollut ehdotus ei ota kantaa siihen, millä tavoin julkaistava *versio validoidaan* ennen sen jakelua tai *valmistellaan käyttäjille*.

Yleisimpiä ehdotusten aktiviteetteja (taulukko 8), jotka eivät sisälly Bekkersin ym. (2010) prosessimalliin, olivat vaatimusten koon ja työmäärän arvioinnit (6/8). Koon ja työmäärän arviointia käsitellään samassa aktiviteetti-

TAULUKKO 8 Aktiviteetit, jotka eivät sisälly Bekkersin ym. (2010) malliin

Ehdotukset	**Aktiviteetteja**
Cohn, 2006	Tyydyttävän lopputuloksen kriteerien ja iteraation pituuden valinnat, käyttäjätarinoiden ja tiimin vauhdin arvioinnit.
Heikkilä, 2010	Esittely ja ohjeistaminen, suunnitelmien ja riskien katselmointi.
Heikkilä ym., 2010a	Sidosryhmien valinta, vaatimusten työmäärän arviointi ja suunnitelmien priorisointi.
Li ym., 2006	Käyttäjätarinoiden koon ja liiketoiminta-arvon arvioinnit, käyttäjätarinoiden välisten riippuvuuksien määrittely, iteraatioon mahtuvan työmäärän arviointi, suunnitelmien riskiarvioinnit, päätös hyväksytäänkö suunnitelman riskit, projektiprofiilien muuttaminen.
Li ym., 2010	-
Logue & McDaid, 2008a	Käyttäjätarinoiden tunnistaminen ja jakaminen tehtäviksi. Käyttäjätarinoiden koon ja arvon arvioinnit. Tiimin vauhdin arviointi ja optimointimallin käyttö.
Szoke, 2011b	Julkaisun ja iteraation pituuden sekä iteraatioiden määrän päättämiset, vaatimusten työmäärän, riippuvuuksien ja iteraation vauhdin määrittämiset.
van Valkenhoef, 2011	Käyttäjätarinoiden ja teemojen koon ja arvon arvioinnit sekä vaatimusten riippuvuuksien määrittely. Käyttäjätarinoiden yhdistäminen teemoiksi ja tiimin kehitysnopeusennusteen teko.

luokassa, sillä näyttää, että termejä käytetään ehdotuksissa vaihtelevalla tavalla, mutta aktiviteeteilla pyritään pääasiasiassa ratkaisemaan kuinka paljon työtä jonkin vaatimuksen toteuttaminen vaatii. Muita yleisiä aktiviteetteja olivat tiimin työskentelynopeuden tai iteraation mahtuvan työmäärän arvioinnit (5/8). Viimeksi mainittuun lasketaan mukaan myös van Valkenhoefin ym. (2011) malliin sisältyvä kehitysnopeusennusteen teko. Vaatimusten arvon arviointi ja riippuvuuksien välinen määritys on mukana kolmessa ehdotuksessa. Vaatimusten arvon arviointi muistuttaa aktiviteettina vaatimusten priorisointia, sillä esimerkiksi Lin ym. (2006) arvon arvioinnin yhteydessä käytetään AHP-priorisointitekniikkaa. Ehdotuksissa arvon arvioinniksi kutsuttuja aktiviteetteja ei kuitenkaan suoraan tulkittu priorisointiaktiviteeteiksi. Aktiviteetteja, joita esiintyy kahdessa ehdotuksessa, ovat iteraation pituuden valitseminen ja tyydyttävän lopputuloksen kriteerien valinta/kriteerien uudelleen muuttaminen. Näiden lisäksi on useita yksittäisiä aktiviteetteja.

5.6.3 Vaatimusten valintatekijät

Tässä kohdassa tarkastellaan, millaisia tekijöitä ehdotuksissa on otettu huomioon vaatimuksia valittaessa julkaisuun käyttäen pohjana Svahnbergin ym. (2010, 241–244) valintatekijöiden taksonomiaa. Tarkastelun ulkopuolelle rajattiin vain arviointiin nojaavat ehdotukset (Cohn, 2006 ja Heikkilä, 2010), sillä näissä valintatekijöitä ei ole eksplisiittisesti rajattu. Tarkastelu tehtiin siten, että ensin arvioitiin, millaisia kovia rajoitteita ja millaisia pehmeitä tekijöitä ehdotuksiin sisältyy käyttäen apuna Ngo-Then ja Ruhen (2007) kuvausta kovista ja pehmeistä tekijöistä. Tekijät, jotka liittyvät suunnitelmien tavoitteisiin, nähtiin pehmeinä tekijöinä. Esimerkkinä tällaisesta on liiketoiminta-arvo suunnittelussa, jonka tavoitteena on mahdollisimman suuren liiketoiminta-arvon tuottaminen. Alla on selitetty kustakin ehdotuksesta tunnistetut vaatimusten valintatekijät. Sulkeisiin on merkitty, millä tavoin tekijä arvioitiin Svahnbergin ym. (2010) luokittelussa. Taulukossa 9 on esitetty kooste vertailun tuloksista.

Heikkilän ym. (2010a) SCERP-menetelmässä kovina rajoitteina ovat toiminnallisuuksien työmäärä ja julkaisun kehityskapasiteetti (työmäärärajoitteet). Pehmeiksi tekijöiksi nähdään sidosryhmien mielipiteen huomioiminen vaatimusten esivalinnassa, priorisoinnissa ja julkaisujen priorisoinnissa (sidosryhmien vaikutustekijät). Toisaalta menetelmässä priorisointi voidaan tehdä usean eri kriteerin perusteella, koska esimerkiksi priorisointikriteerit voivat vaihdella.

Lin ym. (2006) mallin kovia rajoitteita ovat vaatimusten väliset riippuvuudet (vaatimusten riippuvuudet), käyttäjätarinoiden koot ja iteraation vauhti (työmäärärajoitteet). Pehmeitä tekijöitä Lin ym. (2006) prosessimallissa ovat liiketoiminta-arvo (arvotekijät) ja riskianalyysi (riskitekijät).

Lin ym. (2010) mallissa kovia rajoitteita ovat riippuvuudet vaatimusten välillä (vaatimusten riippuvuudet), julkaisujakson pituus (aikarajoite) ja vaatimusten kustannukset työtunteina (resurssirajoite). Pehmeitä tekijöitä ovat vaatimusten tuotto (arvotekijät) ja tehokkaasti eri tiimeille aikataulutetut vaatimukset (resurssinkulutustekijät).

TAULUKKO 9 Vaatimusten valintatekijät Svahnbergin ym. (2010) taksonomiassa

Vaatimusten valintatekijät	**Heikkilä ym. 2010a**	**Li ym. 2006**	**Li ym. 2010**	**Logue & McDaid 2008a**	**Szoke 2011b**	**van Valkenhoef ym. 2011**
Kovat rajoitteet						
Tekniset rajoitteet						
Vaatimusten riippuvuudet		X	X		X	X
Laaturajoitteet						
Budjetti- ja hintarajoitteet						
Resurssirajoitteet			X		X	
Työmäärärajoitteet	X	X		X		X
Aikarajoitteet	*		X		X	
Pehmeät tekijät						
Sidosryhmien vaikutustekijät	X					
Arvotekijät	*	X	X	X	X	X
Riskitekijät	*	X		X		X
Resurssinkulutustekijät			X		X	

X = sisältyy ehdotukseen, * = ehdotuksen yleisyyden vuoksi voi sisältyä

Loguen ja McDaidin (2008a) menetelmän kovia rajoitteita ovat tehtävien koot ja projektin vauhti (työmäärärajoitteet). Pehmeitä tekijöitä ovat liiketoiminta-arvo (arvotekijät) ja epävarmuuden huomioiminen (riskitekijät).

Szoken (2011b) menetelmässä kovia rajoitteita ovat vaatimusten väliset riippuvuudet (vaatimusten riippuvuudet), kehittäjien määrän ja osallistumismahdollisuuden mukaan laskettu tiimin kehitysvauhti iteraatioissa (resurssirajoite) ja julkaisun pituus (aikarajoitteet). Pehmeitä tekijöitä ovat liiketoiminta-arvo (arvotekijät) ja käyttäjätarinoiden tehokas aikataulutus iteraatioihin (resurssinkulutustekijät).

van Valkenhoefin ym. (2011) mallissa koviksi rajoitteiksi voidaan tulkita vaatimusten tekniset ja toivotut toteutusjärjestykset, käyttäjätarinoiden yhdistäminen teemoiksi (vaatimusten riippuvuudet) sekä käyttäjätarinoiden koko (työmäärärajoitteet). Pehmeinä tekijöinä on huomioitu kehityksen epävarmuus, joka huomioidaan kehitysvauhdin ennusteella (riskitekijät), ja käyttäjätarinoiden liiketoiminta-arvo (arvotekijät).

Yhteenvetona voidaan todeta, että yleisimmät kovat rajoitteet ovat työmäärärajoitteet ja vaatimusten riippuvuudet, joita kumpiakin on neljässä ehdotuksessa. Pehmeistä tekijöistä arvotekijät nousevat esiin yleisimpinä tekijöinä, sillä niitä esiintyy viidessä ehdotuksessa. Kolmessa ehdotuksessa huomioidaan myös riskejä. Arvioinnissa riskiluokkaan on luokiteltu myös ne tekijät, jotka liittyvät epävarmuuden huomioiminen. Lin ym. (2010) ja Szoken (2011b) ehdotukset sisältävät ehdotuksista monipuolisimmin valintatekijöitä, sillä niissä kummassakin huomioidaan viittä eri tekijää.

Vertailussa hyödynnettyä valintatekijöiden luokittelua on käytetty aiemmin Svahnbergin ym. (2010) tutkimuksessa, jossa arvioitiin myös tässä tutki-

muksessa esiteltyä Lin ym. (2006) menetelmää. Svahnbergin ym. (2010) ja tämän tutkimuksen arviot Lin ym. (2006) menetelmästä eroavat hieman. Tässä tutkimuksessa ehdotuksen valintatekijöiksi nähdään riskitekijät ja työmäärärajoitteet toisin kuin Svahnbergin ym. (2010) vertailussa. Svahnbergin ym. (2010) arviossa Lin ym. (2006) vaatimusten valintatekijänä on budjetti- ja hintarajoite toisin kuin tässä tutkimuksessa. Näyttääkin siltä, että luokittelukriteerit on tutkimuksissa voitu nähdä eritavoin. Esimerkiksi tässä tutkimuksessa riskitekijöiksi on nähty koko julkaisusuunnitelmaa koskeva riskiarviointi, kun Svahnbergin ym. (2010) vertailussa riskitekijät on voitu mahdollisesti nähdä suppeammin vain yksittäisen vaatimuksen riskiarviointiin liittyvinä. Näiden kahden vertailun lopputulokset eivät siis välttämättä ole suoraan vertailukelpoisia.

5.6.4 Sopivuus ketterään lähestymistapaan

Tässä kohdassa arvioidaan aiemmin esiteltyjen ehdotusten sopivuutta ketterän kehittämisen yhteyteen. Arvioinnissa huomioidaan luvussa 5.2 esiin nousseita ketterän lähestymistavan ja järjestelmällisen suunnittelun ristiriitakohtia. Ketterää kehittämistä tarkastellaan Scrumin mukaisena. Tästä syystä on arvioitu, miten eri ehdotukset sopivat yhteen Scrumin roolien kanssa, ja mukaan on otettu myös kriteeri, jonka mukaan tarkempi suunnittelu tehdään iteraation aikana. Arvioinnissa kiinnitetään huomiota seuraaviin kysymyksiin:

- Onko vaatimukset kuvattu suunnittelun alussa riittävän yleisesti, esimerkiksi toiminnallisuus- tai käyttäjätarinatasoisina?
- Onko ehdotuksen tuloksena syntyvä suunnitelma yksityiskohdiltaan yleinen, esimerkiksi vähintään käyttäjätarinatasoinen?
- Tehdäänkö suunnittelua myös iteraatioissa?
- Täytyykö vaatimusten väliset riippuvuudet olla määritelty yksityiskohtaisesti suunnittelun alussa?
- Mahdollistaako julkaisunsuunnittelun päätöksenteko yksilöiden mielipiteen ja ajatusten huomioimisen eli onko ehdotus formaalisuudeltaan joko epäformaali tai hybridi?
- Ovatko ehdotusten roolit Scrumin kanssa yhteensopivia?

Arviointikriteerit on järjestetty tärkeysjärjestykseen tutkijan arvion mukaan siten, että tärkeimmät kriteerit on mainittu ensin. Tärkeimpinä arviointikriteereinä tutkija pitää suunnittelussa tarvittavien vaatimusten riittävän yleistä tasoa ja sitä, että valmis suunnitelma ei ole liian yksityiskohtainen, sillä nämä vaikuttavat siihen, pysyykö suunnittelu mukautuvana. Myös iteraatioissa tehtävä suunnittelu nähdään suunnittelun mukautuvuuden kannalta tärkeänä. Vaatimusten välisten riippuvuuksien kriteeri ymmärretään niin, että ketterässä lähestymistavassa vaatimusten riippuvuuksia selvitetään neuvottelemalla ja esimerkiksi yhteisen tilan seinätaululle hahmotellen, kuten esimerkiksi Heikkilän (2010) menetelmässä on kuvattu. Heikkilän (2010) menetelmässähän riippuvuudet määritellään usean tiimin yhteisen julkaisunsuunnittelutapahtuman

aikana, jossa tiimit samalla tarkentavat toiminnallisuudet käyttäjätarinoiksi. Suunnittelumallien formaalius kriteerinä on hieman ongelmallinen, sillä on vaikea nähdä, että formaalien mallien käyttö sulkisi täysin pois yksilöiden välisen yhteistyön hyödyntämisen suunnittelussa. Voidaan ajatella, että jotain täysin formaalia toimintatapaa käytetään suunnittelun tukivälineenä antamaan vaihtoehtoisia näkökulmia suunnittelun ratkaisuksi. Myöskään yhteensopivuus Scrumin roolien kanssa ei ole kaikkein tärkein vertailutekijä, sillä rooleja voidaan helposti mukauttaa. Arviointi on esitetty taulukossa 10.

TAULUKKO 10 Ehdotusten sopivuus ketterään kehittämiseen

Yhteensopivuus-kriteerit	**Cohn, 2006**	**Heikkilä, 2010**	**Heikkilä ym. 2010a**	**Li ym, 2006**	**Li ym. 2010**	**Logue & McDaid 2008a**	**Szoke, 2011b**	**van Valkenhoef ym. 2011**
Vaatimukset toiminnallisuus- tai käyttäjätarinatasoisia	Kyllä	Kyllä	Kyllä	Kyllä	Epäselvä	Kyllä	Kyllä	Kyllä
Suunnitelman vaatimukset kuvattu yleisemmin kuin tehtävinä	Kyllä	Kyllä	Kyllä	Kyllä	Epäselvä	Ei	Kyllä	Kyllä
Suunnittelua tehdään myös iteraatioissa	Kyllä	Kyllä	Epäselvä	Kyllä	Ei	Kyllä	Kyllä	Kyllä
Riippuvuuksia hallitaan epäformaalisti	-	Kyllä	-	Ei	Ei	-	Ei	Ei
Suunnittelu on arviointiin perustuvaa tai hybridisuunnittelua	Kyllä	Kyllä	Kyllä	Kyllä	Ei	Kyllä	Ei	Kyllä
Roolit Scrumin mukaiset	Kyllä	Kyllä	Kyllä	Ei	Ei	Kyllä	Ei	Ei

Arvioiduista ehdotuksista parhaiten arviointikriteerit täyttävät Cohnin (2006) prosessimalli ja Heikkilän (2010) menetelmä sekä Heikkilän ym. (2010a) SCERP-menetelmä. SCERP-menetelmä sopii sillä oletuksella, että siinä suunnittelua tehdään julkaisunsuunnittelu- ja iteraatiotasoilla.

Toisen ryhmän muodostavat Lin ym. (2006) ja van Valkenhoefin ym. (2011) mallit, jotka tyydyttävät osittain kriteerit. Näissä vaatimusten riippu-

vuudet tulee kuvata tarkasti suunnittelun yhteydessä eivätkä roolit ole täysin yhteensopivia Scrumin roolien kanssa. Malleja on kuitenkin mahdollista mukauttaa Scrumiin sopivaksi esimerkiksi niin, että tuotteen omistaja vastaa asiakkaan roolista ja van Valkenhoefin ym. (2011) esityksessä esitetty jäljittäjän roolin ottaa hoitaakseen jokin kehitystiimeistä. Lin ym. (2006) mallissa on myös huomioitava, että siinä julkaisu tehdään oletuksena iteraation välein eikä erillistä iteraation suunnitteluvaihetta esitetä. Mallin on kuitenkin yleisyytensä vuoksi helppo nähdä soveltuvan Scrumin mukaiseen julkaisunsuunnitteluun.

Szoken (2011b) menetelmä on sopiva vaatimusten ja tuloksen tarkkuuden suhteen, mutta sen formaalisuus ja tarve määrittää yksiselitteisesti vaatimusten väliset riippuvuudet sekä eroavaisuus Scrumin rooleista on arvioitu ketterään kehittämiseen sopivuutta haittaavina tekijöinä. Menetelmä ei välttämättä kannusta yksilöiden väliseen yhteistyöhön, mutta toisaalta ei ole estettä, etteikö sitä voitaisi käyttää yhtenä tukivälineenä. Myös Loguen ja McDaidin (2008a) menetelmä sijoitettiin tähän kolmanteen kategoriaan. Se sopii muuten vertailukriteereihin, mutta siinä suunnittelun lopputulos syntyy liian yksityiskohtaisella tasolla.

Lin ym. (2010) malli ei näytä vertailun mukaan sopivan erityisen hyvin Scrumin mukaisen ketterään kehittämiseen. Lin ym. (2010) mallin sopivuus kahden ensimmäisen kriteeriin suhteen on hieman epäselvä. Mallissa vaatimusten tarkkuutta ei kuvata kovin selvästi, ja siinä esitetään, että vaatimukset jaetaan pienempiin töihin, joiden suunnittelusta, toteuttamisesta ja testaamisesta tiimit vastaavat. Tältä osin vaatimukset kuulostavat yleisiltä, mutta ongelmalliselta tuntuu kuvaus, jonka mukaan vaatimusten keräämisen yhteydessä selvitetään myös vaatimusten kehitystehtävät, jotta työt voidaan jakaa itsenäisiin kokonaisuuksiin. Tästä syntyy kuva, että vaatimukset on määriteltävä jo etukäteen liian yksityiskohtaisella tavalla (Li ym., 2010, 377). Muilta piirteiltään malli ei täytä kriteerejä, sillä se ei esimerkiksi sisällä iteraatiossa tapahtuvaa suunnittelua.

Edellä mainitusta vertailusta keskeisenä ristiriitakohtana nousee esiin riippuvuuksien hallinta epäformaalisti –kriteeri, johon Lin ym. (2006), van Valkenhoefin ym. (2011) ja Szoken (2011b) ehdotukset on arvioitu yhteensopimattomiksi. Tämän ristiriitakohdan osalta olisi tarve pohtia vielä syvemmin, millaisia tehtäviä vaatimusten riippuvuuksien selvittämisessä on otettava huomion ja tekeekö riippuvuuksien määrittely ehdotuksista vähemmän mukautuvia.

Pohdinnan ulkopuolelle jäivät tässä arvioinnissa muut suunnitteluun vaikuttavat tekijät, kuten vaatimustenvalintatekijöiden ja julkaisunsuunnittelun liittyvien muiden aktiviteettien vaikutus Scrum-yhteensopivuuteen. Lähestymistapojen yksilöllisistä vaatimuksista on mahdollista saada tarkempi kuva niiden kuvauksista.

5.7 Yhteenveto

Julkaisunsuunnittelua pidetään yhtenä tärkeimmistä ohjelmistotuotannon osa-alueista, sillä siinä päätetään kehitettävän julkaisun sisällöstä ja aikataulusta. Suunnittelussa on huomioitava useita eri vaatimusten valintatekijöitä ja priorisointinäkökulmia. Ketterässä lähestymistavassa tärkeä priorisointitekijä on liiketoiminta-arvo. Julkaisunsuunnittelu on hyvin haastava tehtävä, johon ei olemassa optimaalisia ratkaisuja. Haastavuutta lisäävät vaatimusten priorisoinnin moniulotteisuus, niiden väliset riippuvuudet ja liiketoimintaympäristön muutoksista johtuva hyväksyttävän suunnittelun tuloksen muuttuvuus (Jantunen ym., 2011, 38–39). Ketterälle kehittämiselle ominaista on mukautuva suunnittelu ja arviointiin perustuva päätöksenteko. Ketterän lähestymistavan suunnittelun ei ole nähty sopivan hyvin suunnittelun tehostamiseksi kehitettyjen järjestelmällisten julkaisunsuunnittelumallien kanssa käytettäväksi, sillä järjestelmällinen suunnittelu johtaa herkästi etupainotteiseen suunnitteluun.

Tässä luvussa tarkasteltiin kahdeksaa ketterän julkaisunsuunnittelun yhteyteen esitettyä ehdotusta. Niitä vertailtiin yleispiirteiden, kattavuuden (vertailukohtana Bekkersin ym. (2010) malli), vaatimusten valintatekijöiden (vertailukohtana Svahnbergin ym. (2010, 251–254) luokittelu) ja Scrumin mukaiseen ketterän kehittämiseen sopivuuden suhteen. Ehdotuksista kaksi (Cohn, 2006; Heikkilä, 2010) osoittautui arviointiin ja kaksi (Li ym, 2010; Szoke, 2011b) järjestelmälliseen suunnitteluun perustuviksi loppujen kuuden ollessa näiden välimuotoja. Yleisimpiä rooleja ehdotuksissa olivat kehittäjät (7/8) ja toiseksi yleisimpiä tuotteen omistajat ja tuotepäälliköt (3/8). Ehdotusten tuloksina syntyvät suunnitelmat vaihtelivat lukumääriltään ja tarkkuudeltaan.

Bekkersin ym. (2010) malliin verrattuna kaikki ehdotukset sisälsivät vaatimusten priorisointi- ja julkaisun määrittely -aktiviteetteja sekä suurelta osin myös iteratiiviseen kehittämisen myötä laajuuden muutostenhallintaelementin (7/8). Lisäksi niissä yleisesti esiintyviä aktiviteetteja olivat vaatimusten koon ja tiimin työskentelynopeuden arvioinnit. Järjestelmällisten ja hybridi-ehdotusten yleisimpinä vaatimusten valintatekijöinä nousivat esiin vaatimusten arvotekijät (5/6), työmäärärajoitteet (4/6) ja vaatimuksen riippuvuudet (4/6) sekä riskitekijät (3/6).

Ehdotuksista Scrumin mukaiseen ketterään kehittämiseen arvioitiin sopivan parhaiten Cohnin (2006), Heikkilän (2010) ja Heikkilän ym. (2010a) ehdotukset. Myös Lin ym. (2006) ja van Valkenhoefin ym. (2011) sekä osittain myös Szoken (2011b) ja Loguen ja McDaidin (2008a) ehdotukset nähtiin tietyin varauksin soveltuvan Scrumin mukaiseen kehittämiseen. Niiden eräänä merkittävinä soveltumattomuustekijänä arvioitiin olevan vaatimusten välisten riippuvuuksien eksplisiittisen määrittelyn tarve.

6 JULKAISUNSUUNNITTELUN TEKNIIKOITA

Tässä luvussa esitellään julkaisunsuunnitteluntekniikoita. Ensin täsmennetään, mitä tarkoitetaan tekniikalla yleisesti ja julkaisunsuunnittelun yhteydessä erityisesti. Sen jälkeen annetaan esimerkkejä julkaisunsuunnittelun tekniikkatyypeistä. Tekniikoista rajaudutaan tarkastelemaan syvemmin ketterän kehittämisen yhteydessä käytettyjä vaatimusten priorisointitekniikkoja ja koon arviointitekniikkoja. Priorisointitekniikkojen osalta kerrotaan ensin lyhyesti priorisoinnin mitta-asteikoista ja sen jälkeen kuvataan neljä tekniikkaa ja verrataan niitä keskenään. Koon arvioinnille esitetään kaksi tekniikkaa.

6.1 Julkaisunsuunnittelun tekniikoista yleisesti

Tutkimuksessa käytetystä kirjallisuudesta ei löytynyt selkeää määritelmää julkaisunsuunnitteluteknikoille (release planning techniques). Tämän vuoksi käsitettä johdettiin yleisen tekniikkakäsitteen merkityksestä käsin. Leppänen (2005) määrittelee tekniikan preskriptiiviseksi malliksi, joka ohjaa toimintaa ja voi tarjota avuksi proseduurin ja suuntaviivoja" (Leppänen, 2005, 676). Tämän määritelmän mukaisesti tässä tutkimuksessa *julkaisunsuunnittelun tekniikoiden* nähdään sisältävän joukon normatiivisia sääntöjä jonkin toiminnan ohjaamiseksi julkaisunsuunnittelun kontekstissa. Tällaista toimintaa ovat esimerkiksi edellisessä luvussa julkaisunsuunnittelun ehdotuksissa yleiset aktiviteetit, kuten vaatimusten priorisointi, julkaisun määrittely, vaatimusten koon ja työmäärän arvioinnit sekä vaatimusten arvon arviointi.

Suoritetulla kirjallisuushaulla tunnistettiin vaatimusten priorisointitekniikoita (esim. Karlsson, 1996), koon arviointitekniikoita (esim. Grenning, 2002) ja esimerkiksi tiimin nopeuden arviointitekniikka. Vaatimusten arvon arvioinnista heräsi kysymys, onko arvon arvioinnille olemassa erillinen tekniikkaryhmänsä, sillä se voidaan nähdä myös priorisointina vaatimuksen arvon mukaan (kuten Lin ym. 2006 ehdotuksessa). Rachevan ym. (2008, 6) esittelemässä ketterän vaatimusten priorisoinnin käsitemallissa vaatimusten arviointi on priorisointia

edeltävä aktiviteetti, ja Rachevan ym. (2008) mukaan se on olennainen osa jokaista priorisointitekniikkaa. Esimerkiksi AHP-tekniikassa tehtävien vaatimusten keskinäisen vertailujen yhteydessä voidaan ajatella tapahtuvan vaatimusten arvon arviointia. Cohnin (2006, 102–109) esittämä tekniikka, jossa vaatimuksen tulevaa tuottoa arvioidaan absoluuttisella rahamääräisellä arvolla, voidaan kuitenkin nähdä arvon arviointitekniikaksi, koska tekniikassa arvioidaan vain yksittäisen vaatimuksen rahallista tuottoa.

Kirjallisuushaussa ei suoraan löytynyt tekniikoita, joita kutsutaan julkaisun määrittely- tai vaatimusten valinta -tekniikoiksi. Heikkilä ym. (2010a, 2) mainitsee, että Cohnin (2006) prosessimallissa ja Scrumissa yleisesti julkaisun määrittely tehdään yksinkertaisesti ns. ahneella menetelmällä (greedy algorithm), jossa vaatimukset sijoitetaan julkaisusuunnitelmaan yksitellen tärkeysjärjestyksessä, kunnes suunnitelmaan ei enää mahdu uusia vaatimuksia. Toinen esimerkki mahdollisesta julkaisun määrittelytekniikasta on Cohnin (2006, 84–86) ohje valita ensin eniten arvoa tuovat ja riskiä sisältävät, toiseksi paljon arvoa ja vähän riskiä sisältävät ja kolmanneksi vähän arvoa ja vähän riskiä sisältävät vaatimukset toteutettavaksi. Tässä ohjeessa ajatuksena on, että kehittämällä alussa riskejä sisältäviä vaatimuksia riskejä pystytään jo kehityksen alkuvaiheissa parhaiten eliminoimaan. Samalla pyritään tuottamaan mahdollisimman paljon arvoa. Ohjetta voidaan ajatella julkaisun määrittely -tekniikkana ainakin tilanteessa, jossa vaatimusten arvo ja riski on jo etukäteen arvioitu.

Cohn (2006, 177–186) on esittänyt muutamia yleisiä suuntaviivoja tiimin työskentelynopeuden arviointiin, joita voidaan ajatella työskentelynopeuden arviointitekniikoiksi, mutta tässä yhteydessä niitä ei kuvata tarkemmin.

Edellä esitellyt tekniikkatyypit perustuvat edellisessä luvussa esiteltyjen julkaisunsuunnittelun ehdotusten yleisiin aktiviteetteihin. Tyypittely ei välttämättä ole täysin kattavaa, vaan olemassa voi olla myös muita tekniikkatyyppejä, joita voidaan nimittää julkaisunsuunnittelun tekniikoiksi. Aihetta voi olla tarve pohtia vielä syvemmin erillisessä tutkimuksessa.

Seuraavaksi tarkastellaan edellä kuvatuista tekniikkatyypeistä vaatimusten priorisointiin ja vaatimusten koon arviointiin liittyviä tekniikoita.

6.2 Priorisointitekniikoita

Tässä alaluvussa esitellään ensin mitta-asteikkoja, joilla priorisoinnin lopputuloksia voidaan esittää. Toiseksi tarkastellaan esityksiä, joissa ketterään kehittämiseen sopivia priorisointitekniikoita on esitelty. Kolmanneksi kuvataan viittä priorisointitekniikkaa. Lopuksi verrataan tekniikoita niiden käyttämän mitta-asteikon, sopivuutta suurten vaatimusmäärien priorisointiin, priorisointiin osallistuvien roolien, vaikeusasteen ja ehdotetun käyttötavan näkökulmasta.

6.2.1 Priorisoinnin mitta-asteikkoja

Priorisoinnin tuloksia voidaan esittää erilaisilla mitta-asteikoilla (Berander, 2007, 47–48). Karlssonin, Höstin ja Regnellin (2006a, 327) mukaan yleisiä mitta-asteikkoja ovat köyhimmästä rikkaampaan luokittelu-, järjestys-, välimatka- ja suhdeasteikko. Asteikkojen *rikkaudella* tarkoitetaan sitä, että rikkaammilla asteikoilla on mahdollista tehdä enemmän matemaattisia toimenpiteitä kuin köyhemmillä asteikoilla (Kärkkäinen & Högmander, 2008, 19) ja rikkaammat asteikot ovat köyhempiä asteikkoja tarkempia. Välimatka-asteikolla ei Karlssonin ym. (2006a, 327) mukaan ole selvää soveltamiskohdetta vaatimusmäärittelyssä, joten sitä ei esitellä tarkemmin. *Luokittelu- eli nominaaliasteikolla* (nominal scale) kohteet luokitellaan niiden laadullisten erojen mukaan. Asteikon luokkia ei voida asettaa keskinäiseen tärkeysjärjestykseen eikä niiden tärkeyseroja voida arvioida. Esimerkki luokka-asteikon käytöstä on vaatimusten jaottelu sen mukaan minkä järjestelmän osaan ne liittyvät. *Järjestysasteikossa* (ordinal scale) kohteet ovat tärkeysjärjestyksessä, mutta asteikossa ei voida päätellä, kuinka paljon tärkeämpi jokin asteikon kohde on toiseen asteikon kohteeseen verrattuna. Brodien ja Woodmanin (2011) mukaan järjestysasteikollisia lopputuloksia antavat tekniikat voidaan jakaa ryhmittely- (grouping) ja järjestystekniikoihin (ranking). *Ryhmittely*tekniikoissa vaatimukset jaetaan niiden tärkeyden mukaan prioriteettiryhmiin ja prioriteettiryhmät ovat toisiinsa nähden tärkeysjärjestyksessä (Brodie & Woodman, 2011, 81). *Järjestys*tekniikoissa vaatimukset asetetaan keskinäiseen tärkeysjärjestykseen niin, että n määrällä vaatimuksia tärkein vaatimus on ensimmäisellä sijalla ja vähiten tärkeä n:llä sijalla (Brodie ja Woodman, 2011, 81). *Suhdeasteikossa* nähdään kohteiden välinen ero ja kohteet voidaan myös asettaa järjestykseen. Asteikossa on mukana nollapiste ja siinä voidaan verrata, kuinka paljon tärkeämpi jokin asteikon kohde on toiseen kohteeseen nähden. (Kärkkäinen & Högmander, 19–20.)

6.2.2 Ketterien priorisointitekniikkojen tutkimus

Priorisointitekniikoita on tutkittu kirjallisuudessa laajasti (esim. Brodie & Woodman, 2011), mutta ei kuitenkaan kovin kattavasti ketterän kehittämiseen liittyen. Berander (2007, 48–52) on esittänyt viisi yleistä priorisointitekniikkaa, jonka lisäksi hän kuvaa XP-menetelmästä lähtöisin olevaa suunnittelupeliä (Beck, 1999). Szoke (2011b) kuvaa yleisiä tekniikoita ja mainitsee yhtenä tekniikkana ketterän kehittämisen kirjallisuudesta peräisin olevat MoSCoW-säännöt (Craddock, Richards, Tudor, Roberts, & Godwin, 2012, 20). Heikkilä ym. (2010a) kuvaa tekniikoita sidosryhmien osallistumisen näkökulmasta ja viittaa muun muassa suunnittelupeliin ja Cohnin (2006) ketterä suunnittelua käsittelevän kirjan neljään priorisointitekniikkaan. Lin ym. (2006) XP-menetelmän yhteydessä käytettäväksi tarkoitetun julkaisunsuunnittelumallin yhteydessä käytetään AHP-priorisointitekniikkaa (Saaty, 1980). Löydetyistä lähteistä kattavimmin ketterään kehittämiseen sopivia priorisointitekniikoita käsittelee Racheva ym. (2008, 4–5), sillä heidän tutkimuksessaan esitellään 15

ketterän kehittämisen yhteydessä käytettyä tekniikka. Tässä yhteydessä näitä tekniikoita ei esitellä yksityiskohtaisesti, mutta tekniikoiden nimet lähdetietoineen on esitelty liitteessä 4. Esittelyssä ovat mukana edellä mainitut suunnittelupeli- ja MoSCoW-tekniikat. Rachevan ym. (2008, 5–6) mukaan heidän tutkimuksessaan esitellyissä tekniikoissa priorisointi on epämuodollista ja perustuu subjektiiviseen arviointiin. Rachevan ym. (2008, 5) tekniikat voidaan jakaa niihin, joissa vaatimuksia vertaillaan pareittain, ja niihin, joissa vaatimukset ryhmitellään niiden tärkeyden mukaan.

Edellä esitellyistä tutkimuksista on valittu tarkempaan tarkasteluun hyvin yksinkertaisia tekniikoita kuten MoSCoW- (Craddock ym., 2012, 20) ja suunnittelupokeritekniikka (Beck, 1999) sekä tunnettuja tekniikoita kuten AHP:hen perustuva Karlssonin (1996) parivertailutekniikka, 100 dollarin tekniikka (Leffingwell, 2011, 236) ja Kano-malli (Cohn, 2006, 113; Leffingwell, 2011, 269).

6.2.3 MoSCoW -tekniikka

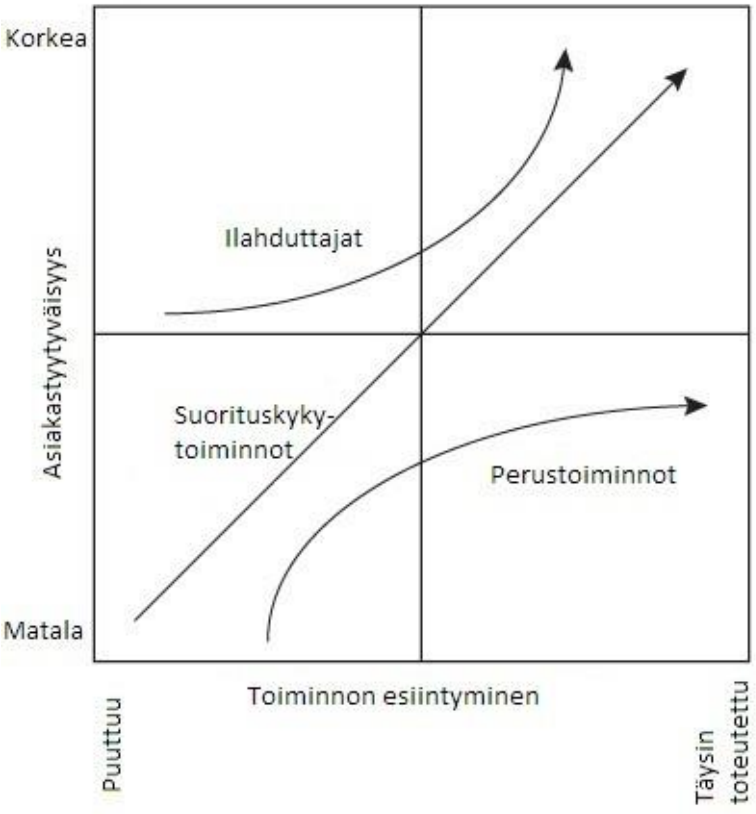
MoSCoW on esimerkki ryhmittelytekniikasta (Brodie & Woodman, 2010, 81). Siinä vaatimukset jaotellaan tärkeyden mukaan neljään ryhmään, jotka ovat englanninkielisiltä nimiltään Must have, Should have, Could have ja Won't have. *Must have* -ryhmän vaatimukset ovat välttämättömiä tuotteen toiminnan kannalta ja ilman niitä tuote on käyttökelvoton. *Should have* -vaatimukset ovat tärkeitä vaatimuksia, joita voidaan kiireettömässä tilanteessa pitää pakollisina. Ne eivät ole kuitenkaan elintärkeitä tuotteen menestymisen kannalta. *Could have* -vaatimukset ovat hyödyllisiä, mutta ne voidaan jättää muita vaatimuksia helpommin pois seuraavasta julkaisusta. *Won't have* -vaatimukset jätetään varmuudella seuraavan julkaisun ulkopuolelle, mutta ne voidaan toteuttaa myöhemmin. Won't have vaatimuksia voidaan ajatella myös toivelistana ohjelmiston tulevista ominaisuuksista (Hatton, 2008, 518). (Craddock ym., 2012, 20.)

6.2.4 Kano-malli

Kano-mallin avulla vaatimuksia voidaan luokitella niiden tuottaman asiakastyytyväisyyden mukaan. Mallia tarkastellaan seuraavaksi Cohnin (2006) mukaan käyttäen Pichlerin (2010) esityksen mukaisia kategorianimiä: perustoiminnot (basic functions), suorituskykytoiminnot (performance functions) ja ilahduttajat (delighters). Toimintojen vaikusta asiakastyytyväisyyteen on havainnollistettu kuviossa 13.

Perustoimintojen on oltava mukana tuotteessa, jotta tuote voi menestyä. Näiden toiminnallisuuksien lisääminen tuo vain vähän asiakastyytyväisyyttä sen jälkeen, kun toiminnot ovat täyttäneet tietyn perustarpeen. Cohnin (2006) antamassa esimerkissä hotellihuoneen vuode vastaa perustoimintoa, jota ilman hotellihuone olisi puutteellinen. *Suorituskykytoimintojen* määrän tai laajuuden lisääntyessä asiakastyytyväisyys kasvaa samassa suhteessa. Mitä paremmin nämä toiminnallisuudet toimivat tai mitä enemmän niitä on, sitä tyytyväisempiä asiakkaat ovat. Cohn (2006) vertaa tätä luokkaa hotellihuoneen kokoon.

Ilahduttavat toiminnallisuudet lisäävät paljon asiakastyytyväisyyttä, mutta niiden puuttuminen ei kuitenkaan laske asiakastyytyväisyyttä neutraalia tasoa alemmas. Cohnin (2006) mukaan ilahduttavia toiminnallisuuksia kutsutaan usein tiedostamattomiksi tarpeiksi. Cohn antaa esimerkkinä näistä toiminnoista hotellin kuntosalin kuntopyörään integroidun television, joka tuo iloa, mutta jota ei välttämättä odoteta. (Cohn, 2006, 112–114.)



KUVIO 13 Kano-malli (mukaillen Cohn, 2006, 113)

Cohnin (2006) mukaan vaatimusten valinnassa painopiste tulee olla perustoiminnoissa. Perustoimintojen on oltava mukana tuotteessa, kun tuote julkaistaan markkinoille. Näiden toimintojen osittainenkin toteuttaminen voi kuitenkin täyttää asiakkaiden tarpeet, sillä toiminnallisuuksien määrän tai laajuuden lisääminen lisää vain vähän asiakastyytyväisyyttä. Cohnin (2006) mukaan suorituskykytoiminnallisuuksia tulisi pyrkiä sisällyttämään suunnitelmaan mahdollisimman paljon. Lisäksi julkaisusuunnitelmaan tulisi sisällyttää muutamia ilahduttavia toiminnallisuuksia. Cohn toteaa, että ajan kuluessa toiminnallisuudet muuttuvat, jolloin ilahduttavat toiminnallisuuksista voi tulla suorituskykytoimintoja ja suorituskykytoiminnoista perustoimintoja. (Cohn, 2006, 113–114.)

Vaatimusten ryhmittely Kano-mallin luokkiin on mahdollista tehdä esimerkiksi suoraan vaatimuksia luokittelemalla tai järjestämällä kysely asiakkaille tai sidosryhmien edustajille. Kyselyssä kustakin arvioitavasta toiminnallisuudesta esitetään kaksi kysymystä, joista ensimmäinen liittyy vastaajan suh-

tautumiseen toiminnallisuuden sisältymiseen ja toinen toiminnallisuuden puuttumiseen. Kullakin kysymykselle on viisi vastausvaihtoehtoa: 1. Pidän asiasta, 2. Odotan asian olevan niin, 3. Neutraali suhtautuminen asiaan, 4. Siedän asian, 5. En pidä asiasta. Vastauksia voidaan tulkita taulukon 11 avulla. (Cohn, 2006, 114–116.)

TAULUKKO 11 Kano-mallin kyselyn tuloksien tulkintataulu (Cohn, 2006, 116)

		Toiminnon puuttuminen -kysymys				
		Pidän	**Odotan**	**Neutraali**	**Siedän**	**En pidä**
Toiminnon sisältyminen -kysymys	**Pidän**	K	I	I	I	S
	Odotan	Pä	E	E	E	P
	Neutraali	Pä	E	E	E	P
	Siedän	Pä	E	E	E	P
	En pidä	Pä	Pä	Pä	Pä	K
P = perustoiminto, S = suoritustoiminto, K = kyseenalainen, I = ilahduttava, E = ei väliä, Pä = päinvastainen						

Tulkintataulun riveillä on esitetty, miten kyselyn tuloksia voidaan tulkita edellä esiteltyjen perustoimintojen, suoritustoimintojen ja ilahduttavien toimintojen lisäksi ei väliä (indifferent), kyseenalainen (questionable) ja päinvastainen (reverse) -tuloksilla. Cohn (2006) ei selitä, mitä muilla taulukon tiedoilla tarkoitetaan, mutta Lacey (2012) kuvaa artikkelissaan niitä tarkemmin. Laceyn (2012) mukaan *ei väliä* -tulos tarkoittaa, että vaatimus on vähiten tärkeä ja tuottaa todennäköisesti vain vähän tai ei lainkaan liiketoiminta-arvoa. Esimerkki mahdollisuus muuttaa ohjelmiston käyttöliittymän väriä voi jollekin henkilölle olla ei väliä -toiminnallisuus, johon vastaaja voi suhtautua itsestään selvänä (odotettavana) ominaisuutena, mutta, jonka puuttumisenkin hän toisaalta sietää. Toisin sanoen ominaisuus ei vaikuta suuresti tyytyväisyyteen tai tyytymättömyyteen. *Kyseenalainen* ja *päinvastainen* -tuloksissa vaatimukseen liittyvien kysymyksien vastaukset ovat ristiriidassa keskenään ja Laceyn (2012) mukaan on syytä tarkistaa, että kyselyn kysymykset on muotoiltu ymmärrettävästi. Kyseenalainen tai päinvastainen tulokset voivat johtua esimerkiksi siitä, että kysymysparit on muotoiltu virheellisesti eivätkä ne eivät kunnolla vastaa toisiaan tai siitä, että vastaaja ei ole ymmärtänyt kysymyksiä.

Kun kysely on tehty, vastaukset kootaan yhteen ja lasketaan jakauma sille, mihin ryhmiin käyttäjät arvioivat kutakin vaatimusta. Jakaumasta voidaan esimerkiksi nähdä, että suurin osa vastaajista pitää tiettyä vaatimusta ilahduttavana toimintona. (Cohn, 2006, 116.)

6.2.5 Suunnittelupeli

Suunnittelupeli on XP -menetelmän (Beck, 1999) julkaisunsuunnittelun käytäntö, jota on sovellettu myös priorisointitekniikkana (esim. Racheva ym., 2008; Brodie & Woodman, 2010). Seuraavaksi suunnittelupeliä esitetään Karlssonin, Thelinin, Regnellin, Beranderin ja Wohlinin (2007) mukaan.

Vaatimukset priorisoidaan suunnittelupelissä sekä arvon että toteuttamisen riskin perusteella. Arvioitavat vaatimukset voidaan kirjata esimerkiksi kahden korttipinon lapuille, joista toisia käytetään arvon ja toisia riskin mukaan priorisointiin. Asiakkaat ryhmittelevät vaatimukset kolmeen ryhmään, jotka ovat toiminnan kannalta: (a) välttämättömät, (b) ei välttämättömät, mutta liiketoiminta-arvoa merkittävästi tuovat, ja (c) vaatimukset, jotka olisi mukava sisällyttää tuotteeseen. Samaan aikaan kehittäjät jakavat vaatimukset riskin mukaan ryhmiin, jotka ovat: (a) pystytään arvioimaan tarkasti, (b) pystytään arvioimaan kohtuullisen hyvin, ja (c) ei pystytä arvioimaan lainkaan. Samaan aikaan kehittäjät voivat myös arvioida vaatimusten kehittämisen vaadittavaa työmäärää. Ryhmiin jaon jälkeen, kunkin ryhmän vaatimukset järjestetään tärkeysjärjestykseen. Tämän jälkeen saman näkökulman mukaan priorisoidut ryhmät yhdistetään yhdeksi pinoksi, jolloin vaatimukset ovat kummankin pinoissa näkökulmien mukaan tärkeysjärjestyksessä. Tekniikan lopputuloksena syntyy näin sekä arvon että riskin mukaan järjestetyt vaatimukset. (Karlsson ym., 2007, 6.)

Järjestetyistä vaatimuksista voidaan valita ensin toteutettavaksi eniten arvoa ja paljon teknistä riskiä sisältävät vaatimukset. Ajatuksena on, että riskejä pystytään minimoimaan, kun tartutaan ensimmäisenä vaatimuksiin joiden toteutuksen työmäärästä ei vielä kunnolla käsitystä (Racheva, Daneva, Herrmann, 2010, 288–289).

6.2.6 100 dollarin tekniikka

100 dollarin tekniikka (100 dollard test) on tunnettu tekniikka, jonka suorittamiseen voi osallistua yhtä aikaa usea arvioija. Priorisointi aloitetaan niin, että osallistujat olettavat käytössään olevan 100 (tai esimerkiksi 1000) yksikköä mielikuvitusvaluuttaa. Seuraavaksi jokainen osallistuja jakaa valuuttansa vapaasti vaatimuksille sen mukaisesti miten arvokkaiksi ne näkevät. Kun kaikki valuutat on sijoitettu, kunkin vaatimuksen saama valuuttamäärä lasketaan yhteen. Jos rahat ovat jakautuneet hyvin tasaisesti, voidaan arvioijille halutessa antaa uudet 100 valuuttayksikköä jaettavaksi, mutta tällä kertaa valuutta on jaettava yhtenä 50 yksikön ja kahtena 25 yksikön osuutena (Hatton, 2008, 517). Eri arvioijien arvioinnille voidaan halutessa antaa erilaisia painoarvoja (Berander, 2007, 50). Priorisoinnin lopputuloksena syntyy suhdeasteikollinen vaatimusten jaottelu. (Leffingwell, 2011, 236.)

Tekniikka ei välttämättä sovi käytettäväksi useaan kertaan peräkkäin samalla ryhmällä ja samoilla vaatimuksilla tehtäväksi, sillä arvioijilla on tieto siitä, miten muut tulevat todennäköisesti äänestämään. Jotkin arvioijat voivat tällöin taktikoida ja sijoittaa kaiken valuuttansa jollekin erityisesti pitämälleen vaatimukselle, joka ei välttämättä tulisi muuten arvioitua korkealla prioriteetilla.

Tekniikka ei välttämättä sovellu hyvin suurten vaatimusmäärien priorisointiin. Regnell, Höst, Natt och Dag, Beremark ja Hjelm (2001) tutkivat hajautettua priorisointia, jossa arvioijat jakoivat vaatimuksille 100000 mielikuvitusyksikköä ja vaatimuksia oli 58 kappaletta. Tutkimuksessa havaittiin, että priorisoijat sijoittivat valuuttaa vain rajatulle joukolle (Beranderin (2007, 135)

mukaan vain keskimäärin 34 prosentille) vaatimuksia, jolloin erityisesti vähäisillä yksiköillä arvioitujen vaatimusten välisen tärkeyden erottelukyky jäi alhaiseksi. Beranderin (2007, 135) mukaan suuria vaatimusmääriä arvioitaessa arvioijien voi olla vaikea ylläpitää kokonaiskäsitystä priorisoitavista kohteista. (Berander, 2007, 50–51.)

6.2.7 Parivertailutekniikka

Karlssonin (1996) *parivertailutekniikka* (pair-wise comparison technique) perustuu Saatyn (1980) analyyttiseen hierarkiaprosessiin (AHP). Tekniikassa jokaista mahdollista vaatimusparia verrataan vuorollaan toisiinsa ja jokaisessa vertailussa arvioidaan vaatimusparien tärkeyden välistä eroa. Tärkeämmäksi arvioitu vaatimus arvioidaan tärkeyden voimakkuuden mukaan arvoilla 3, 5, 7 tai 9, missä 3 merkitsee vähäistä ja 9 äärimmäistä tärkeyseroa. Vertailuparissa vähemmän tärkeäksi arvioidulle vaatimukselle annetaan tärkeämmäksi arvioidun vaatimuksen käänteisluku. Mikäli vaatimukset arvioidaan yhtä tärkeiksi, kummallekin vaatimukselle annetaan arvo 1. Esimerkiksi jos A-vaatimus arvioidaan B-vaatimusta kohtalaisesti tärkeämmäksi, A-vaatimukselle asetetaan arvo 5 ja B:lle A:n arvon käänteisluku 1/5. Vertailun tulokset sijoitetaan vertailumatriisiin, josta on annettu esimerkki taulukossa 12.

TAULUKKO 12 Esimerkki vertailumatriisista (Karlsson, 1996, 112).

	Vaatimus A	**Vaatimus B**	**Vaatimus C**
Vaatimus A	1	5	1/3
Vaatimus B	1/5	1	1/7
Vaatimus C	3	7	1

Tekniikassa tarvittavien vertailujen määrä voidaan laskea kaavalla n * (n - 1) / 2, jossa n on vaatimusten lukumäärä. Esimerkiksi kymmenellä vaatimuksella parivertailu tehdään 45 kertaa ja 20 vaatimukselle 95 kertaa. (Karlsson, 1996, 111.)

Kun kaikki vertailut on tehty, lasketaan kunkin vaatimuksen ominaisarvo, jonka jälkeen vaatimuksille pystytään määrittelemään suhdeasteikolla tärkeys esimerkiksi prosenttilukuina ilmaistuna. Tekniikan lopputuloksena syntyy suhteellisella asteikolla arvioidut vaatimusten prioriteetit. (Karlsson, 1996, 112.)

Tekniikassa on mahdollista tarkistaa, onko vertailussa ollut epäjohdonmukaisuuksia. Vertailun epäjohdonmukaisuudesta esimerkkinä on tilanne, jossa A, B, C vaatimuksia vertailtaessa A on arvioitu B:tä, B C:tä ja C A:ta tärkeämmäksi. Tässä yhteydessä epäjohdonmukaisuuden tarkistustapaa ei esitellä tarkemmin. (Karlsson, 1996, 112)

Parivertailutekniikasta on kehitetty työkalulla käytettävä versio, jota kutsutaan *työkalutuetuksi parivertailutekniikaksi* (Tool Supported Pair-Wice Comparison, TSPWC). TSPWC-tekniikassa tarvittavien vaatimusparien vertailumäärä voidaan pienentää jopa 75 prosenttia pienemmäksi kuin parivertailutekniikas-

sa, mutta samalla tekniikan epäjohdonmukaisuusarvion laatu heikkenee. (Karlsson ym., 2007, 7.)

Karlsson ym. (2007) ovat vertailleet keskenään parivertailutekniikkaa, suunnittelupeliä ja toiseksi suunnittelupeliä ja TSPWC-tekniikkaa. Vertailussa keskityttiin tutkimaan eroja tekniikoiden käytön helppoudessa, priorisoinnin nopeudessa ja päätöksenteon tarkkuudessa. Kokeissa priorisoitiin enintään 16 vaatimusta. Suunnittelupeli havaittiin tutkimuksessa parivertailutekniikkaa helpommaksi ja nopeammaksi käyttää. TSPWC-tekniikka oli yhtä helppokäyttöinen kuin suunnittelupeli. Se oli verrattavista tekniikoista nopeakäyttöisin. Tekniikoiden päätöksenteon tarkkuudessa ei havaittu merkittäviä eroavaisuuksia. (Karlsson ym., 2007, 3.)

6.2.8 Priorisointitekniikan valinta

Berander (2007) mukaan priorisointitekniikan valinnassa keskeistä on päättää, mikä on mahdollisimman nopea ja kustannustehokas tekniikka haluttuun käyttötarkoitukseen, mutta joka silti tuottaa riittävän laadukkaan lopputuloksen. Kriittisimmissä päätöksissä hän kehottaa käyttämään kehittyneempiä priorisointitekniikkoja. (Berander, 2007, 53.)

Rachevan ym. (2008) mukaan priorisointitekniikan valintaan vaikuttaa projektin tausta, kokemukset aiempien tekniikoiden käytöstä ja myös projektipäällikön kokeneisuus. Tekniikan valinnassa tulee ottaa huomioon ainakin priorisoitavien vaatimusten ja priorisointiin osallistuvien tahojen määrä, vaatimusten muutosalttius (volatility) ja se, millaisia tiedonsaantitapoja on käytettävissä. (Racheva ym., 2008, 8.)

Seuraavaksi verrataan tekniikoita niiden tuloksena syntyvän mitta-asteikon tarkkuuden, sopivuuden keskisuurten vaatimusmäärien eli noin yli 20 vaatimuksen priorisointiin, priorisointitekniikan käyttäjien, tekniikan vaikeuden ja käyttötapaehdotuksen mukaan. Yhteenveto vertailusta on esitetty taulukossa 13.

Mitta-asteikkoiltaan tekniikat ovat suhde-, järjestys- tai luokitteluasteikollisia. MoSCoW-tekniikasta syntyy prioriteettiryhmiin sijoitetut vaatimukset ja suunnittelupelissä taas järjestykseen asetetut vaatimukset. Kano-mallin tulos tulkitaan luokitteluasteikolliseksi, sillä mallin vaatimuskategoriat ovat laadullisesti toisistaan poikkeavia. Tekniikan lopputuloksena ei näytä syntyvän selkeitä prioriteettiryhmiä, sillä jokaisen luokittelun vaatimukset tuottavat eri tavalla asiakastyytyväisyyttä.

Bebenseen, van de Weerdin ja Brinkkemperin (2010, 68) mukaan MoSCoW -tekniikka ja suunnittelupeli sopivat hyvin myös keskisuurten tai suurien vaatimusmäärien priorisointiin. Näillä kokoluokilla he tarkoittavat muutamaa tusinaa eli karkeasti arvioiden noin 20–40 vaatimusta. Kano-mallin käyttäjäkyselyssä käyttäjien tulisi vastata jokaista vaatimusta kohden kahteen kysymykseen. Esimerkiksi 30 vaatimuksen arvioimiseksi vastaajien tulisi vastata 60 kysymykseen. Tällaiset kysymysmäärät eivät tunnu aivan kohtuuttomilta vastaajien näkökulmasta, sillä yksittäisiin kysymyksiin vastaaminen käy

TAULUKKO 13 Priorisointitekniikoiden vertailu

Vertailu-kriteeri	**MoSCoW (Craddock ym., 2012)**	**Kano-malli (Cohn, 2006)**	**Suunnittelu-peli (Beck, 1999)**	**100 dollarin tekniikka (Leffing-well, 2011)**	**Parivertailu (Karlsson, 1996)**
Priorisoinnin tuloksen mitta-asteikko	Järjestys-asteikko (ryhmittely)	Luokittelu-asteikko	Järjestys-asteikko (järjestäminen)	Suhdeas-teikko	Suhdeas-teikko
Sopivuus yli 20 vaati-muksen priorisointiin	Sopii	Sopii	Sopii	Sopii kohta-laisesti (enin-tään n. 20–40 vaatimuksel-le)	Ei sovi hyvin
Tekniikan käyttäjät	Ei määritelty	Useat sidos-ryhmät	Asiakkaat ja kehittäjät	Yksi tai use-ampi henkilö	Ei määritelty
Vaikeusaste	Helppo	Keskinker-tainen	Helppo	Helppo	Vaikea
Ehdotus käyttötavas-ta	Vaatimusten esivalinnan yhteydessä tehtävä prio-risointi	Tuotteen ominaisuuk-sien suunnit-telu	Yleinen prio-risointitek-niikka kette-rään kehit-tämiseen	Yleinen prio-risointitek-niikka kette-rään kehit-tämiseen	Tarkkuutta vaativa prio-risointi

todennäköisesti nopeasti. Eniten kuluu todennäköisesti aikaa kysymyksien laatimiseen. Tekniikka soveltunee suurten vaatimusmäärien arviointiin ainakin vastaajien näkökulmasta. Lehtolan ja Kauppisen (2004, 166) tutkimuksen mukaan parivertailutekniikalla yli 20 vaatimuksen priorisoiminen käy aikaa vieväksi tarvittavien vertailujen suuren määrän takia. Tästä syystä tässä arvioidaan, että tekniikka voi sopia parhaiten alle 20 vaatimuksen priorisoimiseen. 100 dollarin tekniikka ei sovi välttämättä hyvin yli 50 vaatimuksen priorisointiin, kuten tekniikan kuvauksessa aiemmin todettiin. Tämän perusteella arvioidaan, että tekniikkaa voidaan käyttää maksimissaan keskisuurten eli 20–40 vaatimusten priorisointiin.

MoSCoW- ja parivertailutekniikoiden kuvauksissa ei määritellä mitään tiettyä käyttäjäryhmää. Käyttäjä voinee olla taho, joka priorisointia normaalisti suorittaa, esimerkiksi tuotteen omistaja tai tuotepäällikkö. Suunnittelupelin kuvauksessa priorisointipäätöksen tekee asiakas ja kehittäjät osallistuvat prosessiin arvioimalla vaatimusten riskejä. Scrumin yhteydessä tuotteen omistaja voi tehdä asiakkaan sijasta priorisointia. 100 dollarin tekniikassa priorisointiin voi helposti osallistua useita eri henkilöitä, kuten myös Kano-mallissa, jossa voidaan tehdä kysely vaatimusten kiinnostavuudesta useille eri henkilöille.

Hatton (2008, 524) pitää MoSCoW- ja 100 dollarin tekniikoita helppokäyttöisinä ja nopeina. Kano-mallin helppokäyttöisyydestä ei löydetty kirjallisuudesta suoraa arvioita, mutta tekniikka arvioidaan helppokäyttöisyydeltään keskinkertaiseksi. Tekniikan käyttö voi olla aikaa vievää etenkin kyselyn laatimisessa ja suorittamisessa, mutta muulla tavoin tekniikka ei ole käytöltään kovin monimutkainen. Suunnittelupeli vaikuttaa helpolta tekniikalta omaksua ja

käyttää, sillä se koostuu periaatteessa vaatimusten ryhmittelystä ja järjestykseen asettamisesta. Tekniikka arvioidaan tästä syystä helpoksi. Parivertailutekniikka arvioidaan hankalaksi käyttää, sillä esimerkiksi Hatton (2008, 524) luokittelee AHP-tekniikan, johon parivertailutekniikka perustuu, vaikeakäyttöiseksi.

Käyttötarkoituksiltaan MoSCoW-tekniikka voi karkeatasoisuutensa vuoksi soveltua hyvin vaatimusten esivalintaan. Esimerkiksi Hattonin (2008, 523) mukaan MoSCoW-tekniikka sopii hyvin projektin alkuvaiheessa tehtävään priorisointiin, kun vaatimukset on kuvattu yleisesti. Voidaan ajatella, että esivalinnassa Must have -vaatimukset voidaan ottaa suoraan mukaan ja Won't have -vaatimukset rajataan esivalinnan ulkopuolelle. Should have ja Could have -vaatimusten mukaan ottamista voidaan pohtia tarkemmin tai näitä luokkia voidaan mahdollisesti priorisoida jollain tarkemmalla priorisointitekniikalla vaatimusten valitsemiseksi. Tekniikkaa voidaan ajatella käytettävän myös yleisenä helppona priorisointitekniikkana, kun vaatimuksia valitaan itse julkaisusuunnitelmaan. Kano-malli soveltuu hyvin ohjelmistotuotteen kehittämiseen, sillä sen avulla voidaan saada tietoa asiakkaiden mieltymyksistä ja tarpeista. Sen avulla voidaan varmistaa, että kehitettävästä tuotteesta muodostuu tasapainoinen kokonaisuus, jossa on varmasti mukana riittävät pakolliset ominaisuudet, mutta myös käyttäjien tyytyväisyyttä nostavia ilahduttavia ominaisuuksia. Koska käyttäjäkyselyn tekeminen on todennäköisesti työlästä, tekniikkaa voi olla tehokasta käyttää vain aika-ajoin, mutta ei välttämättä joka julkaisunsuunnittelutapahtuman yhteydessä. Kano-malli voikin soveltua parhaiten käytettäväksi hieman julkaisunsuunnittelua yleisemmällä suunnittelun tasolla. Esimerkiksi Pichler (2010, 39) esittelee Kano-mallia vision määrityksen yhteydessä. Suunnittelupeli soveltuu kuvauksensa mukaan hyvin ketterän lähestymistavan kontekstiin, sillä se nojautuu yksilöiden väliseen vuorovaikutukseen. Se on myös helppokäyttöinen ja sen lopputuloksena syntyy järjestetty vaatimusten lista, joka voi riittää hyvin esimerkiksi tuotteen työlistan järjestämiseen. 100 dollarin tekniikka sopii keveytensä vuoksi hyvin ketterän julkaisunsuunnittelussa tehtävään priorisointiin. Siinä priorisointiin voi osallistua yhtäaikaisesti useita henkilöitä ja koska tekniikan lopputuloksena syntyy suhdeasteikollinen vaatimusten jaottelu, sen avulla saadaan halutessa yksityiskohtaista tietoa vaatimusten välisten tärkeyserojen suuruudesta. Parivertailutekniikka voi sopia pienten vaatimusryhmien tarkkaan priorisoimiseen. Sen yhtenä etuna on, että priorisoinnin tulokset voidaan tarkistaa epäjohdonmukaisuuksien varalta. Se ei vaikuta kuitenkaan parhaalta priorisointitekniikalta ketterään kehittämiseen vaikeakäyttöisyytensä vuoksi. Se on myös aikaa vievä tekniikka, jos priorisoitavia vaatimuksia on paljon, ainakin ilman työkalutukea.

6.3 Koon arviointitekniikat

Tässä alaluvussa kerrotaan ensin koon ja työmäärän arvioinnin eroista. Tämän jälkeen kerrotaan yleisesti koon arvioinnista, jonka jälkeen tarkastellaan suunnittelupokeritekniikkaa ja lyhyemmin oikotiearviointitekniikkaa.

6.3.1 Koon ja työmäärän arvioinnin erot

Tämän tutkimuksen julkaisunsuunnitteluehdotuksissa vaatimusten koon (size) ja työmäärän (effort) arviointitermejä käytetään hieman vaihtelevalla tavalla. Koon arviointia tehdään McDaidin ja Loguen (2008a), van Valkenhoefin ym. (2011) ja Lin ym. (2006) ehdotuksissa. Koko ilmaistaan niissä ideaalipäivinä tai tarinapisteinä (Logue & McDaid, 2008a), tarinapisteinä (van Valkenhoef ym., 2011) tai työtunteina (Li, 2006). Vaatimusten työmääriä arvioidaan Heikkilän ym. (2010a) ja Szoken (2011b) ehdotuksissa. Näistä ensimmäisessä arvioinnin yksikköä ei ole määritelty, mutta jälkimmäisessä arviointi tehdään henkilöpäivinä. Cohnin (2006) esityksessä käytetään vaatimusten koon arviointi -termiä, mutta myös työmäärä ja kehittämisen kustannustermejä käytetään. Arvioinnin yksikkönä Cohn (2006) käyttää tarinapisteitä tai ideaalipäiviä.

Cohnin (2006, 36) mukaan tarinapisteiden koon määrittelyyn ei ole tarkkaa kaavaa, vaan koon arviossa voidaan huomioida muun muassa kehittämiseen tarvittavaa työmäärää, kehittämisen monimutkaisuutta ja kehittämiseen sisältyvää riskiä. Vaikuttaakin siltä, että Cohn näkee tarinapisteiden koon arvioinnin työmäärän arviointia laajempana kokonaisuutena. Eräässä blogikirjoituksessaan Cohn (2010) näyttää kuitenkin painottavan, että työmäärän arviointi on tarinoiden suhteellisessa koon arvioinnissa tärkeintä.

van der Straaten (2012) esityksessä koon arviointi määritellään vaatimuksen arvioinniksi suhteellisena suureena joko tarinapisteinä tai ideaalipäivinä. Vaatimusten työmäärän arvioinnilla tarkoitetaan esityksessä eksaktia, esimerkiksi todellisina päivinä (actual days) tai tunteina (actual hours) ilmaistua, aikaarvioita. Tämä termien erottelu vaikuttaa selväpiirteiseltä ja se sopii yhteen kaikkien ehdotuksien, paitsi Lin ym. (2006) esityksen, arviointitermien käytön kanssa. Tässä tutkimuksessa koon ja työmäärän arvioinnin määrittely nähdään kuvatun jaottelun mukaisena. Täten Cohnin (2006) esityksessä tehtävä arviointi nähdään koon arviointina. Koon ja työmäärän arviointiin käytettävistä tekniikoista käytetään jatkossa yksinkertaisesti koon arviointitekniikat -nimitystä.

6.3.2 Yleisesti koon arvioinnista

Cohnin (2006, 49–51) mukaan ketterässä kehittämisessä on tärkeämpää pyrkiä vaatimusten koon arvioinnissa nopeuteen ja riittävän hyviin tuloksiin kuin äärimmäisen tarkkoihin arvioihin. Cohnin (2006) mukaan arviointien hiomisesta huolimatta niihin jää aina epävarmuutta ja hiomisesta saatava hyöty saattaa jäädä pieneksi käytettyyn työmäärään nähden. Pichlerin (2010, 67) mukaan vaatimusten arviointiin osallistavat vain kehittäjät, jotka osallistuvat julkaisun kehittämiseen. Vaatimusten työmäärää arvioidaan yleensä kehittäjien yhteistyönä ryhmässä (Cohn, 2006, 51).

Kuten Cohnin (2006) julkaisunsuunnittelun ehdotuksessa todettiin, vaatimusten koon arviointi tehdään tavallisesti joko suhteellisina käyttäjätarinoina tai ideaalipäivinä. Iteraatiotason suunnittelussa tehtävien koot arvioidaan tunteina (Cohn, 2006, 148). Käyttäjätarinoiden suhteellisessa arvioimisessa on tär-

keää huomata, että arviot ovat aina tiimikohtaisia ja tiimeille kehittyy omat arviointiskaalansa (Pichler, 2010, 65).

Cohnin (2006) mukaan vaatimusten koon ja tiimin nopeuden arviointi on ketterässä julkaisunsuunnittelussa erotettu toisistaan. Ensin arvioidaan vaatimusten koko, tämän jälkeen johdetaan vaatimusten kokojen perusteella aikaarvio siitä, kuinka kauan vaatimusten kehittäminen kestää, ja lopulta laaditaan aikataulu vaatimusten toteuttamiseksi (Cohn, 2006, 33–35). Cohnin (2006) mukaan sillä, että tiimin kehitysvauhti päätellään aiemman työnopeuden mukaan, on automaattisesti arviointivirheitä tasoittava vaikutus. Mikäli tiimi on arvioinut vaatimusten koot alakanttiin ja ei ole pystynytkään toteuttamaan iteraation aikana vaatimuksia täydessä laajuudessa, tiimi voi automaattisesti korjata aikatauluarvioita oikeaan suuntaan laskemalla aikataulun viimeisen tiimin nopeuden mukaan. (Cohn, 2006, 39–40.)

Vaatimusten koon arvioinnissa tavallisesti käytetyt tekniikat ovat Cohnin (2006) mukaan analoginen arviointi, asiantuntija-arviot ja vaatimusten osiin jakaminen. Analogisessa arvioinnissa arvioitavaa kohdetta arvioidaan suhteessa aiemmin arvioituihin kohteisiin. Asiantuntija-arviossa asiantunteva ammattilainen arvioi tietojensa ja kokemuksensa mukaan, kuinka kauan tietyn tehtävän suorittamisen kestää. Cohnin mukaan asiantuntija-arvio ei yksinään sovellu hyvin käyttäjätarinoiden arviointiin, sillä käyttäjätarinoiden toteuttaminen vaatii usean osaajan työpanosta ja yksittäisen asiantuntijan voi olla vaikea arvioida tarkasti työpanoksien määrää. Yksilöllisillä asiantuntija-arvioinneilla voi olla myös taipumusta ylioptimistisuuteen varsinkin silloin, kun arvioidaan omia työtehtäviä (Jørgensen, 2004). Osiin jakamisessa arvioitava kohde jaetaan pienempiin osiin, jonka jälkeen osien koot arvioidaan. Lopuksi osien kokoarviot lasketaan yhteen, jolloin saadaan arvio vaatimuksen koosta. (Cohn, 2006, 54–56.)

6.3.3 Suunnittelupokeri

Yksi tunnetuimmista ketterän julkaisunsuunnittelun vaatimusten koon arviointiin käytettävistä tekniikoista on Grenningin (2002) esittelemä suunnittelupokeri (planning poker). Se voidaan luokitella ryhmäarviointitekniikaksi (group estimation technique), jossa arvioon päädytään yksilöiden yhteistyöllä (Mahnič & Hovelja, 2012, 2087). Tekniikkaa esitellään seuraavaksi Cohnin (2006, 56) mukaan.

Suunnittelupokeriin osallistuu koko kehitystiimi. Tuotteen omistaja osallistuu myös tapahtumaan, mutta hän ei suorita tarinoiden arviointia. Suunnittelupokeri alkaa siten, että yksi osallistujista lukee yhden vaatimuksen, joka voi olla kuvattuna esimerkiksi käyttäjätarinana. Tämän jälkeen kukin tiimin jäsen arvioi itsenäisesti arvioitavan vaatimuksen suuruuden pitäen arvionsa aluksi muilta tiimin jäseniltä piilossa. Arvioinnissa voidaan käyttää esimerkiksi kullekin tiimin jäsenelle jaettua korttisarjaa, joka sisältää kortteja, jotka on numeroitu 1, 2, 3, 5, 8, 13, 20, 40 ja 100 luvuin. Kun jokainen tiimin jäsen on päättänyt arvionsa, arviot paljastetaan muulle tiimille. Mikäli jokainen tiimin jäsen päätyy

täsmälleen samaan arvioon, tarinan kooksi tulee tämä arvio. Mikäli arviot ovat eriäviä, pienimmän ja suurimman arvon esittäneet perustelevat arvionsa. Tämän jälkeen ryhmä voi keskustalla arvioista muutaman minuutin ajan, jonka jälkeen ryhmän jäsenet arvioivat tarinaa uudelleen kuten aiemmin. Prosessia jatketaan niin kauan, että jokainen ryhmän jäsen päätyy samaan kokoarvioon. Kaikki käyttäjätarinat arvioidaan samalla tavalla. Cohnin (2006) mukaan suunnittelupokeria voidaan pelata aivan projektin tai ensimmäisen iteraation alussa ja myös säännöllisesti iteraatioiden kehittämisen aikana. Cohn (2006) ehdottaa, että uusia vaatimuksia voidaan arvioida aina iteraatioiden päätteeksi. Uudet käyttäjätarinat voidaan esimerkiksi asettaa tietylle sovitulle paikalle näkyville ja tiimien jäsenet voivat aina sopivan tilaisuuden tulleen arvioida pienissä ryhmissä uusien vaatimusten työmäärät. (Cohn, 2006, 56–58.)

Suunnittelupokerissa tiimin jäsenet pääsevät tasapuolisesti ilmaisemaan mielipiteensä ilman, että vain äänekkäimpien arvioijien mielipiteet nousevat esiin. Koska arviot ilmaistaan tekniikassa samanaikaisesti, sen avulla voidaan vähentää niin sanottua ankkurointi-ilmiötä (anchoring), jossa ensimmäisen arvioijan antama, mahdollisesti epärealistinenkin, arvio vaikuttaa ohjaavasti seuraavaksi arvion antavien henkilöiden mielipiteeseen. (Haugen, 2006, 2.)

Suunnittelupokerin arvioiden tarkkuudesta ja optimistisuusasteesta on hieman ristiriitaisia tutkimuksia. Useissa tutkimuksissa on todettu, että ryhmäpäätöksenteossa voi muodostua optimistisempia päätöksiä verrattuna yksilöllisesti tehtyihin päätöksiin (Mahnič & Hovelja, 2012, 2088). Toisaalta Moløkken-Østvoldin ja Haugenin (2007) tutkimuksessa suunnittelupokeri oli havaittu tarkemmaksi ja vähemmän ylioptimisia tuloksia tuottavaksi kuin tekniikka, jossa arvio lasketaan yksilöittäin tehtyjen arvioiden keskiarvon mukaan. Kyseisen tutkimuksen kokeessa arvioitavat vaatimukset olivat kuitenkin olleet tehtävätasoisia ja olivat kehittäjille tuttuja, sillä koe tehtiin projektin ollessa jo käynnissä. Mahnič ja Hovelja (2012) tekivät Moløkken-Østvoldin ja Haugenin (2007) tutkimusta vastaavat tutkimuksen, jossa arvioinnit tehtiin julkaisujakson alussa ja vaatimukset oli kuvattu käyttäjätarinoina. Koe suoritettiin kokemattomille viimeisen vuoden tietojenkäsittelytieteen opiskelijoille ja kokeneille ohjelmistoalan ammattilaisille. Kokemattomilla arvioijilla sekä suunnittelupokerin että ja yksilöarvioiden keskiarvoon perustuvan tekniikan arviot olivat ylioptimistia ja optimistisuus oli suurempaa suunnittelupokerin tuloksissa. Kokemattomien arvioijien suunnittelupokerilla tehdyt arviot olivat myös keskiarvotekniikkaa epätarkempia. Sen sijaan kokeneiden ryhmän suunnittelupokerilla tehdyt arviot olivat keskiarvoarvoon perustuvia arvioita vähemmän optimistisia. Tekniikoiden tarkkuudessa ei ollut merkittävää eroa. Kokeneiden ryhmän kummankin tekniikan arviot olivat opiskelijoiden arvioita tarkempia. Tutkimuksessa ei voitu täysin selvästi osoittaa, asiantuntijoiden suunnittelupokerin arviot olisivat olleet keskiarvon perusteella tehtyjä arvioita tarkempia.

Haugenin (2006) aiemmassa tutkimuksessa suunnittelupokeria verrattiin yksinkertaiseen tekniikkaan, jossa kokoarviot sovittiin epämuodollisesti keskustelun jälkeen. Tämän tutkimuksen mukaan suunnittelupokerin tulokset olivat tarkempia toiseen tekniikkaan verrattuna, kun arvioitavat vaatimukset oli-

vat tuttuja kehittäjille, mutta epätarkempia, kun kehittäjillä ei ollut aiempaa tuntemusta arvioitavista vaatimuksista.

Edellä mainittujen tutkimuksien perusteella näyttääkin tärkeältä, että tarkkuuden lisäämiseksi ja optimistisuuden vähentämiseksi, suunnittelupokeriin osallistuu kokeneita henkilöitä tai henkiöitä, jolla on aiempaa kokemusta arvioitavista vaatimuksista.

6.3.4 Oikotiearviointi

Pichler (2010, 67–68) kuvaa *oikotiearviointi* (fast-track estimation) -tekniikkaa, jota voidaan käyttää vaatimusten koon nopeaan arviointiin suunnittelupokerin sijaan. Toiminta aloitetaan jakamalla esimerkiksi tyhjä seinä osiin niin, että jokainen osa vastaa tiettyä vaatimuskokoluokkaa. Tuotteen työlistalla olevat vaatimukset tulostetaan lapuille ja laput asetetaan pöydälle. Seuraavaksi vaatimukset arvioidaan niin, että tiimin jäsenet ottavat yksitellen lappuja pöydältä, arvioivat itsenäisesti vaatimuksen koon ja sijoittavat ne seinälle arvon mukaiseen osioon. Tiimin jäsenet voivat vapaasti siirtää seinällä olevia lappuja, jos he huomaavat, että jokin seinällä olevista lapuista ei ole sopivassa kokoluokassa. Pichlerin (2010, 68) mukaan tekniikka on nopea käyttää, mutta sen tulokset ovat suunnittelupokeria epätarkemmat.

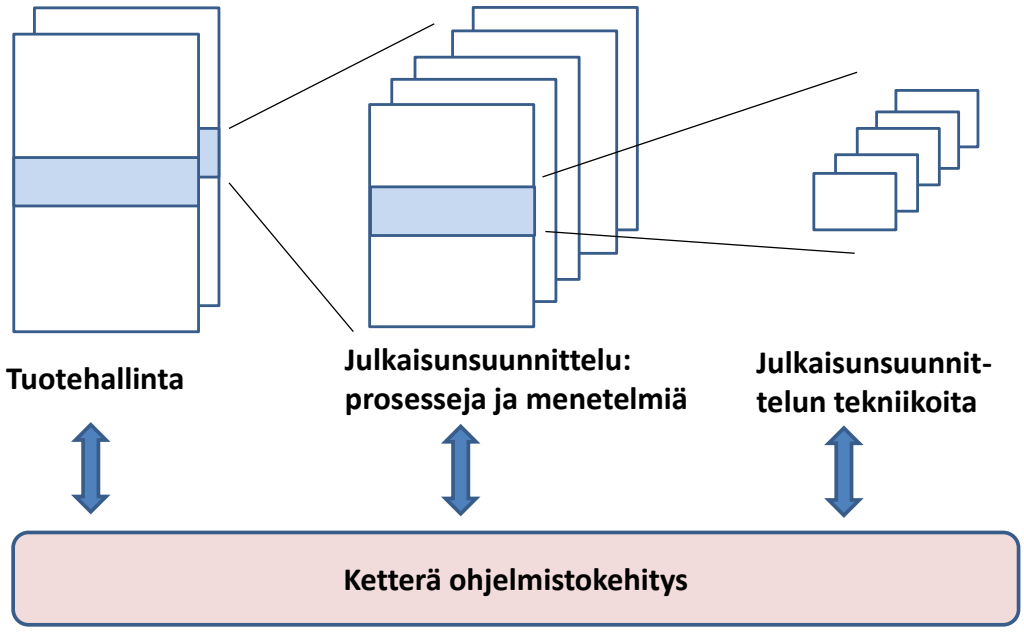
6.4 Yhteenveto

Tämän luvun alussa julkaisunsuunnittelun tekniikka määriteltiin normatiiviseksi ohjeeksi, joka ohjaa toimintaa julkaisunsuunnittelun kontekstissa. Esimerkkejä näistä tekniikkatyypeistä ovat vaatimusten priorisointi-, koon arviointi-, arvon arviointi- ja julkaisun määrittelytekniikat. Priorisointi- ja koon arviointitekniikoita tarkasteltiin luvussa lähemmin. Priorisointitekniikoiden yhteydessä todettiin käyttävän yleisesti luokittelu-, järjestys- ja suhdeasteikkoja. Ketterään kehittämisen priorisointitekniikoita käsitelleistä tutkimuksista todettiin, että Racheva ym. (2008) on käsitellyt tekniikoita löydetyistä lähteistä laajimmin. Priorisointitekniikoista keskityttiin tarkastelemaan MoSCoW- (Craddock ym., 2012, 20), Kano-malli- (Cohn, 2006, 112–116), suunnittelupeli- (Beck, 1999), parivertailu- (Karlsson, 1996) ja 100 dollarin (Leffingwell, 2011, 236) tekniikkoja. Priorisoinnin tulokset olivat tekniikoissa luokka-, järjestys- tai suhdeasteikollisia. Tekniikat olivat helppokäyttöisiä ja sopivat noin 20–40 vaatimuksen vaatimusmäärien priorisointiin parivertailutekniikkaa lukuun ottamatta. Suunnittelupelin käyttäjiksi mainitaan asiakkaat ja kehittäjät. 100 dollarin tekniikassa sekä Kano-mallissa priorisointi voidaan suorittaa useille käyttäjille. Muilta osin tekniikoiden käyttäjärooleja ei ole tekniikoissa määritelty. MoSCoW-tekniikan arvioitiin sopivan hyvin vaatimusten esivalintaan, Kano-mallin tuotesuunnitteluun, suunnittelupeli ja 100 dollarin tekniikat peruspriorisointitekniikoiksi ketterään kehittämiseen ja parivertailutekniikka suuren tarkkuustason vaativaan

priorisointiin. Priorisointitekniikoiden jälkeen tarkasteltiin vaatimusten koon arviointitekniikoista suunnittelupokeria (Grenning, 2002) ja oikotiearviointia (Pichler, 2010, 67–68). Suunnittelupokeri mahdollistaa muun muassa tiimin jäsenten tasapuolisen osallistumisen, mutta tekniikan tarkkuudesta ja optimistisuuteen ohjaamisesta on hieman ristiriitaisia tuloksia. Mahničin ja Hoveljan (2012) ja Haugenin (2006) tutkimuksien perusteella näyttää tärkeältä, että suunnittelupokeriin osallistuu myös kokeneita asiantuntijoita tai henkilöitä, joilla on tuntemusta arvioitavista vaatimuksista.

7 KOKONAISKUVA KETTERÄSTÄ JULKAISUNSUUNNITTELUSTA

Tässä luvussa esitetään kokonaiskuva ketterästä julkaisunsuunnittelusta kokoamalla yhteen edellisissä luvuissa esiteltyjä tuloksia. Tarkastelussa keskitytään erityisesti ketterän ohjelmistokehityksen yhteensopivuuden näkökulmaan. Samalla tarkastellaan myös miten tutkimuksen eri osa-alueet ja niiden sisältämät tuotehallintamallit, prosessit ja menetelmät sekä tekniikat liittyvät toisiinsa. Kuvio 14 hahmottaa tätä kokonaiskuvaa.



KUVIO 14 Ketterän julkaisunsuunnittelun kokonaiskuva

Tuotehallinta voidaan nähdä yrityksen liiketoimintaa ja ohjelmistotuotantoa yhdistävänä tehtäväalueena, joka hallinnoi tuotteita niiden koko elinkaaren ajan (esim. Fricker, 2012; Ebert, 2006). Sen vastuualueisiin kuuluu esimerkiksi tuotesuunnittelua, mutta myös yrityksen strategian johtamiseen osallistumista ja yri-

tyksen muiden toimijoiden ohjaamista tuotekehityksessä (ISPMA, 2012). Organisaatioiden tuotehallinnan ja ketterän kehittämisen yhdistämiseksi on luotu erilaisia ketterän tuotehallinnan viitekehyksiä, joista kahta, Big Picture -mallia (Leffingwell, 2011) ja ATMAN-mallia (Vähäniitty, 2010b) tarkasteltiin tässä tutkimuksessa. Kummankin mallin osana on sekä iteraatioissa tehtävä ohjelmistokehitys että julkaisunsuunnittelu.

Ketterän tuotehallinnan viitekehykset näyttävät sopivan hyvin ketterään ohjelmistokehitykseen, sillä Scrumin sprinteissä kehittämisen voi nähdä sisältyvän tuotehallintamallien iteraatiotasoihin. Scrumin tuotteen ja sprintin työlistat vastaavat myös Leffingwellin mallin ohjelman ja tiimien työlistoja ja ATMAN-mallin kanssa työlistat ovat samannimisiä. Leffingwellin mallin portfoliontyölista ja ATMAN-mallin julkaisuntyölistat näyttävät laajentavan tai tarkentavan Scrumin työlistoja. Leffingwellin mallissa on samoja rooleja kuin Scrumissa, mutta siinä esitetään Scrumiin verrattuna uusia vastuualueita ja rooleja, kuten tuotepäällikkö. Koska Leffingwellin mallissa sekä tuotepäälliköt että tuotteen omistajat osallistuvat tuotteenhallinnan tehtäviin, tuotteen omistajan vastuualue voi olla Scrumin kuvaukseen verrattuna rajoittuneempi.

Julkaisunsuunnittelu nähdään tässä tutkimuksessa prosessina, jonka tuloksena syntyy yleisen tason suunnitelma, joka ohjaa iteraatiotason yksityisyyskohtaisempaa ja lyhyemmällä aikajänteellä tehtävää suunnittelua. Ketterää julkaisunsuunnittelua jäsentämään ja tukemaan on luotu erilaisia prosesseja ja menetelmiä, joista tutkimuksessa tarkasteltiin kahdeksaa. Niitä arvioitiin niiden yleisten piirteiden, niiden sisältämien aktiviteettien, vaatimusten valinta tekijöiden ja ketterään kehittämiseen sopivuuden suhteen.

Ketterään kehittämiseen sopivuuden arvioinnissa prosessit ja menetelmät jaettiin neljään ryhmään. Parhaiten ketterään ohjelmistokehitykseen arvioitiin sopivan Cohnin (2006), Heikkilän (2010) ja Heikkilän ym., (2010a) prosessit ja menetelmät. Niissä vaatimuksia kuvataan yleisesti, suunnittelu perustuu arviointiin ja roolit ovat Scrumin mukaisia. Toisen ryhmän Lin ym. (2006) ja van Valkenhoefin ym. (2011) menetelmät eroavat ensimmäisestä ryhmästä yhteensopivuusongelmana nähdyn vaatimusten välisten riippuvuuksien tarkan määrittelyn vuoksi. Kolmanneksi ketterään kehittämiseen sopivat varauksin Szoken (2010b) sekä Loguen ja McDaidin (2008a) menetelmät, joissa ensimmäisessä suunnittelu kuvataan järjestelmällisenä ja toisen lopputuloksena syntyy liian tarkka suunnitelma. Lin ym. (2010) menetelmä ei sovi hyvin ketterään kehittämiseen sen formaalisuuden ja iteraatioissa tehtävän suunnittelun puuttumisen vuoksi.

Julkaisunsuunnittelun prosesseista ja menetelmistä Szoken (2011b), Heikkilän ym., (2010a) ja Heikkilän (2010) prosessit sopivat yhteen ATMAN-mallin (Vähäniitty, 2010b) kanssa, sillä mallissa ja prosesseissa vaatimuksia käsitellään toiminnallisuuksina. Rikkaasti kuvattuun Leffingwellin (2011) tuotehallintamalliin sopii vain Heikkilän (2010) prosessi, sillä Leffingwellin mallissa julkaisunsuunnittelua tehdään usean tiimin yhteistapahtumana. Kun julkaisunsuunnittelun ehdotuksia sovelletaan tuotehallintamallien kanssa, on hyvä huomioida, että esityksissä tuotepäälliköiden ja tuotteen omistajien rooleissa saattaa olla

päällekkäisyyksiä. Tuotepäällikön ja tuotteen omistajan tehtäväalueiden yhdistämisessä voidaan noudattaa esimerkiksi Leffingwellin (2011) mallin antamaa tapaa tai vaihtoehtoisesti voidaan käyttää vastuualueiden jakamiseen Kittlausin (2012, 92) ajatusta tehtävien jaosta jonkin tuotehallinnan viitekehyksen (esim. SPMBoK, ISPMA, 2012) avulla. Yleisenä ohjeena tuotehallinnan mallien ja julkaisunsuunnittelun prosessien ja menetelmien yhdistämisessä voidaan ajatella myös sitä, että yksityiskohdiltaan rikkaampaa Leffingwellin mallia käytetään esimerkkimallina, jota voidaan käyttää tavoitteena (vrt. Leffingwell, 2011, 31) tai josta voidaan poimia käytäntöjä omaan toimintaan. Löyhempää ATMAN-mallia voidaan ajatella käytettävän runkona tuotehallinnan käytäntöjen ja julkaisunsuunnittelun ehdotusten yhdistämisessä.

Ketterän julkaisunsuunnittelun yleisiä aktiviteetteja ovat vaatimusten priorisointi, julkaisun määrittely, laajuuden muutostenhallinta, koon ja työmäärän arviointi sekä tiimin työvauhdin arviointi. Aktiviteettien tueksi on olemassa erilaisia tekniikoita, joista tutkimuksessa kuvattiin vaatimusten priorisointi- ja koon arviointitekniikoita. Esitellyt priorisointitekniikat ovat Kano-malli (Cohn, 2006), MoSCoW- (Craddock ym., 2012), suunnittelupeli- (Beck, 1999), (Leffingwell, 2011), parivertailu- (Karlsson, 1996) ja 100 dollarin (Leffingwell, 2011) tekniikka. Koon arviointitekniikoista tarkasteltiin suunnittelupokeria (Cohn, 2006) ja oikotiearviointia (Pichler, 2010). Priorisointitekniikoista parhaiten ketterään kehittämiseen sopivat MoSCoW- suunnittelupeli- ja 100 dollarin tekniikat, sillä ne olivat kaikki kevyitä käyttää. Koon arviointitekniikat sopivat myös kuvauksensa mukaan hyvin ketterään kehittämiseen. Suunnittelupokeri mahdollistaa koko tiimin osallistumisen arviointiin ja oikotiearviointi on hyvin nopea ja kevyt käyttää.

Tekniikoista Kano-mallia voidaan käyttää muiden priorisointitekniikoiden tukena tärkeiden vaatimusten tunnistamisessa, sekä myös vaatimusten esivalinnassa siten, että tekniikalla tunnistetut ”ei väliä” -ryhmän vaatimukset harkitaan jättämään suunnittelun ulkopuolelle. Muita priorisointitekniikoita voidaan käyttää kaikkien prosessien ja menetelmien yhteydessä, paitsi niissä menetelmissä, joissa priorisointi tapahtuu formaalilla tavalla (Li ym., 2006; Li ym., 2010; Szoke, 2011b; Heikkilä ym., 2010a; van Valkenhoef ym., 2011). Priorisointitekniikoita, joiden tuloksena syntyvät suhdeasteikolliset tulokset (parivertailu- ja 100 dollarin tekniikka), voidaan käyttää myös Lin ym. (2006) ja Lin ym. (2010) menetelmissä arvon arviointiin. Koon arviointitekniikat sopivat jokaisen tarkastellun ehdotuksen kanssa käytettäväksi koon ja työmäärän arviointiin, lukuun ottamatta Loguen & McDaidin (2008a) menetelmää, jossa vaatimuksille annetaan pessimistinen, todennäköinen ja positiivinen arvo. Saattaa olla liian raskasta käyttää tekniikoita näiden kolmen arvon arviointiin. Toisaalta suunnittelupokeria voidaan soveltaa pelaamalla sitä vain yksi kierros ja sijoittamalla pienin kierroksen arvio pessimistiseksi, keskiarvo-arvio todennäköiseksi ja suurin arvio positiiviseksi arvoksi. Koon arviointitekniikoita voidaan ajatella käytettävän myös Heikkilän ym. (2010a) ja van Valkenhoefin ym. (2011) ehdotuksissa arvon arviointiin, sillä arvot arvioidaan niissä suhteellisesti tietylle arvoalueelle.

8 YHTEENVETO

Tämän tutkimuksen tarkoituksena oli luoda kirjallisuuteen perustuen kokonaiskuva ketterän lähestymistavan mukaisesta julkaisunsuunnittelusta osana tuotehallintaa. Tutkimusongelma oli: *millaista tukea löytyy kirjallisuudesta ketterään julkaisunsuunnitteluun.* Tämän pohjalta laaditut tutkimuskysymykset olivat:

- Mitä tarkoitetaan tuotehallinnalla ja millainen on ketterä tuotehallinta?
- Mitä tarkoitetaan julkaisunsuunnittelulla, millaisia julkaisunsuunnittelun prosesseja, menetelmiä ja tekniikoita on olemassa ja mitkä näistä soveltuvat ketterän ohjelmistokehityksen yhteyteen?
- Miten julkaisunsuunnittelu on yhteydessä ketterään tiimi- tai projektimuotoiseen ohjelmistonkehitykseen?

Seuraavassa kuvataan tutkielman keskeisiä tuloksia. Tutkimuksessa esiteltiin ensin ketterän lähestymistavan taustaa, pääpiirteitä ja Scrum-menetelmää. Lähestymistavan keskeisinä piirteinä korostuvat iteratiivinen ja inkrementaalinen ohjelmistokehitys, mukautuvuus ja ihmisten välisen vuorovaikutuksen tärkeys. Scrumista esiteltiin sen taustaa, prosessia, rooleja, tapahtumia ja tuotoksia. Scrum on ohjelmistokehitykseen tarkoitettu viitekehys, jonka mukaan ohjelmistoja kehitetään lyhyissä iteraatioissa.

Toiseksi tutkimuksessa luotiin yleiskuva tuotehallinnasta. Tuotehallintaa voidaan hahmottaa liiketoimintaa ja ohjelmistotuotantoa yhdistäväksi toimialaksi ja vastuualueeksi, joka hallinnoi ohjelmistotuotteita. Monet viitekehykset kuvaavat tuotehallinnan ydin- ja vastuualueita. Yksi tuotehallinnan ydinalueita esittävä viitekehys (Bekkers ym., 2010) valittiin tutkimuksen keskeiseksi vertailupohjaksi.

Tuotehallintaa ketterässä lähestymistavassa tarkasteltiin aiheen tutkimuksen sekä tuotepäällikön ja tuotteen omistajan roolien yhteensovittamisen osalta. Ketterää tuotehallintaa on yleisesti tutkittu verrattain vähän ja roolien sovittamiseen kirjallisuudessa on esitetty toisistaan poikkeavia näkemyksiä. Lisäksi kuvattiin ja verrattiin kahta (Leffingwellin, 2011; Vähäniitty, 2010b) ketterää

tuotehallintaa jäsentävää mallia niiden piirteiden selvittämiseksi. Malleissa ohjelmistokehitys on kiinteänä osana iteraatiotasojen kautta ja vaatimusten hallintaa tehdään mallien jokaisella tasolla. Mallit kuvaavat myös, miten julkaisunsuunnittelu on yhteydessä tiimi- ja projektimuotoiseen kehittämiseen tasojen välisen ohjaus- ja palautesuhteen sekä vaatimusten yhteyden kautta.

Julkaisunsuunnittelusta kuvattiin ensin sen lähtökohtia, muotoa ja haasteita. Julkaisunsuunnittelu todettiin ohjelmistokehityksen tärkeäksi, mutta haastavaksi, tehtäväksi. Ketterä julkaisunsuunnittelu on mukautuvaa eli suunnittelu tapahtuu siinä joustavasti sekä julkaisu- että iteraatiotasoilla. Toiseksi kuvattiin kahdeksaa ketterän julkaisunsuunnittelun tueksi ehdotettua prosessia ja menetelmää ja vertailtiin niitä useiden kriteerien mukaan. Suunnittelu on kahdessa prosessissa ja menetelmässä arviointiin perustuvaa, kahdessa järjestelmällistä ja lopuissa hybridi-suunnittelua. Rooleista yleisimmät ovat kehittäjät ja vaatimuksista käyttäjätarinat ja toiminnallisuudet. Aktiviteeteista yleisimmät ovat vaatimusten priorisointi, julkaisun määrittely, laajuuden muutostenhallinta sekä koon ja työmäärän arviointi. Järjestelmällistä suunnittelua sisältävien menetelmien yleisimmät vaatimusten valintatekijät liittyvät arvoon, työmäärään ja vaatimusten riippuvuuksiin. Prosesseista ja menetelmistä kolme sopi hyvin ketterään ohjelmistokehitykseen, neljä vaihtelevin rajoittein ja yksi heikosti.

Tutkimuksen julkaisunsuunnittelun tekniikat voidaan luokitella vaatimusten priorisointi-, koon arviointi-, arvon arviointi-, julkaisun määrittely- ja iteraation vauhdin arviointitekniikoihin. Näistä tarkasteltiin priorisointi- ja vaatimusten koon arviointitekniikoita. Tarkastellut viisi priorisointitekniikkaa ovat pääosin helppokäyttöisiä ja sopivat keskisuurten vaatimusmäärien priorisointiin. Tekniikoista kolme sopii hyvin ketterään kehittämiseen. Koon arviointitekniikoista tarkasteltiin kahta tekniikkaa, joista tunnetumman (Grenning, 2002) osalta tutustuttiin sitä arvioiviin tutkimuksiin.

Lopuksi tutkielmassa muodostettiin kokonaiskuva tutkimuksen eri osa-alueista. Kokonaiskuvassa tarkasteltiin erityisesti osa-alueiden liittymistä ketterään ohjelmistokehitykseen ja toisiinsa. Tuotehallinnan mallit sopivat hyvin ketterän kehittämisen yhteyteen. Vain osa prosesseista ja menetelmistä sopi suoraan tuotehallintamalleihin. Koon arviointitekniikoita voitiin käyttää lähes kaikissa prosesseissa ja menetelmissä sekä priorisointitekniikoita osassa niitä.

Tutkimusta ja sen tuloksia voidaan hyödyntää monella tavalla. Työssään julkaisunsuunnittelun tehtävissä toimiva voi saada yleiskuvan julkaisunsuunnittelun prosesseista, menetelmistä ja teknikoista sekä arvioita ja vertailutietoja, joiden perusteella hän voi tehdä valintoja käsillä oleviin tilanteisiin. Tutkimus tarjoaa myös kattavan viitevalikoiman alan tutkimuksiin.

Tutkimukseen liittyy joitakin rajoitteita. Kokonaiskuvan luonnissa päähuomio oli julkaisunsuunnittelujen prosesseissa ja menetelmissä. Tuotehallinnasta ja ketterää tuotehallintaa käsittelevistä malleista luotiin laaja yleiskuva, mutta niihin ei keskitytty aivan yhtä tarkasti kuin julkaisunsuunnittelun prosesseihin ja menetelmiin. Julkaisunsuunnittelun tekniikoita käsiteltiin muita alueita suppeammin. Tarkemmin eri julkaisunsuunnittelutekniikoista tarkasteltiin vain priorisointi- ja koon arviointitekniikoita. Tarkastelussa ei pyritty sa-

manlaiseen kattavuuteen kuin julkaisunsuunnittelun prosesseissa ja menetelmissä. Päähuomion pitäminen julkaisunsuunnistelun ehdotuksissa oli perusteltua, sillä kaikkia osa-alueita ei tutkimuksen luonteen ja aiheiden laajuuden vuoksi voitu käsitellä yhtä tarkasti.

Tutkimuksessa erilaiset vertailut olivat keskeisessä asemassa ehdotusten piirteiden tarkastelussa. Vertailupohjista yksi tärkeimmistä oli Bekkersin ym. (2010) kompetenssimalli. Sen avulla pystyttiin vertaamaan sekä tuotehallinnan malleja että julkaisunsuunnittelun ehdotuksia. Bekkersin ym. (2010) prosessimallin vertailukriteerejä olisi voitu hieman tarkentaa, sillä joitain vertailukriteerejä kuten esimerkiksi julkaisun määrittelyn validointi -kriteeriä voitiin tulkita monesta eri näkökulmasta. Tutkimuksessa pyrittiin tuomaan esille käytetyt näkökulmat. Erilaisten näkökulmien huomioonottamiseen liittyy myös julkaisunsuunnittelun prosessien ja menetelmien vaatimustenvalintatekijöiden vertailun hieman eroavat tulokset verrattuna aiempaan Svahnbergin ym. (2010) vertailuun. Samanlaisista vertailukriteereistä huolimatta tulokset eivät ole välttämättä näkökulmaerojen vuoksi suoraan verrannolliset.

Tutkimuksen tiedonhakutapa perustui pääasiassa tieteellisistä tietokannoista tehtyihin hakuihin. Tietokantahaussa löytyi yhteensä 18 julkaisunsuunnittelun prosessia ja menetelmää, jotka ryhmiteltiin samankaltaisuuden mukaan 13 ryhmään. Ryhmistä seitsemästä valittiin edustajat tarkasteluun. Ottaen huomioon, että tarkasteluun valittiin ryhmiä, joihin oli viitattu eniten, kirjallisuuskatsauksesta tutkimukseen mukaan poimitut ehdotukset edustavat melko hyvin tunnetuimpia ehdotuksia. Tiedonhaun ulkopuolella ovat ehdotukset, joiden kuvauksessa ei ole mainittu sopimista ketterään julkaisunsuunnitteluun. Koska tiedonhaku keskittyi tieteellisiin tietokantoihin, tarkastelun ulkopuolella ovat myös ketterää lähestymistapaa käsittelevät kirjat lukuun ottamatta tutkimuksessa mukana olleita esityksiä. Tiedonhaku pyrittiin tekemään mahdollisimman huolellisesti ja jo löydettyjen ehdotusten lähteitä hyödyntäen, mutta on mahdollista, että jokin tutkimus on jäänyt tiedonhaussa huomaamatta.

Tutkimus luo pohjaa monenlaiselle jatkotutkimukselle. Tutkimuksessa tarkasteltuja osa-alueita voidaan tarkastella syvällisemmin, tutkimuksessa käytettyjen vertailun kriteerejä voidaan entisestään tarkentaa ja tarkastelua voidaan laajentaa. Ensimmäisenä jatkotutkimusaiheena voisi olla julkaisunsuunnittelun tekniikoiden tarkastelu tätä tutkimusta syvemmin ja kattavammin. Tutkimuksessa esiteltyjä tekniikoita hahmoteltiin yleisesti ja kahteen tekniikkatyyppiin keskittyen. Toisena jatkotutkimusaiheena olisi mielenkiintoista selvittää tarkemmin, millaisia ristiriitoja ketterän lähestymistavan ja perinteisten julkaisunsuunnittelun käytäntöjen välillä on. Ristiriitakohtien vertailussa tässä tutkimuksessa jäi vielä avoimeksi se, missä määrin vaatimusten riippuvuustekijöiden määrittely vaikuttaa ketterään julkaisunsuunnitteluun. Julkaisunsuunnittelun prosesseissa ja menetelmissä on nähtävissä toisiinsa nähden hieman raskaampia ja kevyempiä riippuvuuden määrittelykeinoja. Kysymys siitä, mitkä näistä keinosta ovat ketterän kehittämisen yhteydessä riittävän hyviä ja sopivia vaatimusten riippuvuuksien kartoittamiseen, kaipaa vielä lisäselvennystä.

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

Anderson, D. (2010). Kanban: successful evolutionary change for your technology business. Sequim: Blue Hole Press.

Artz, P., van de Weerd, I., Brinkkemper, S. & Fieggen, J. (2010). Productization: transforming from developing customer-specific software to product software. Teoksessa P. Tyrväinen, S. Jansen & M. Cusumano (toim.), *Proceedings of the 1st International Conference on Software Business (ICSOB 2010)* (s. 90-102). Berlin: Springer-Verlag.

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

Bebensee, T., van de Weerd, I. & Brinkkemper, S. (2010). Binary priority list for prioritizing software requirements. Teoksessa R. Wieringa & A. Persson (toim.), *Requirements Engineering: Foundation for Software Quality*. (s. 67–78) Berlin: Springer-Verlag.

Beck, K. (1999). *Extreme programming explained: embrace change*. Reading: Addison-Wesley.

Bekkers, W., van de 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.

Benestad, H. & Hannay, J. (2011). A comparison of model-based and judgement-based release planning in incremental software projects. Teoksessa *Proceedings of the 33rd International Conference on Software Engineering.* (s. 766–775). New York, NY: ACM.

Berander, P. (2007). Evolving prioritization for software product management. *Blenkinge Institute of Technology Doctoral Dissertation Series 7*. Väitöskirja. Ronneby: Blekinge Institute of Technology.

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

Brodie L. & Woodman, M. (2011). Prioritization of stakeholder value using metrics. Teoksessa L.A. Maciaszek & P. Loucopoulos (toim.), *Evaluation of Novel Approaches to Software Engineeting* (s. 74–88). Berlin: Springer-Verlag.

Carlshamre, P. (2002). Release planning in market-driven software product development: provoking an understanding. *Requirements Engineering, 7*(3), 139–151.

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

Cohn, M. (2009). *Succeeding with Agile – software development using scrum*. Upper Saddle River, NJ: Addison-Wesley.

Cohn, M. (2010). It's effort, not compelixity. Haettu 26.4.2013 osoitteesta http://www.mountaingoatsoftware.com/blog/its-effort-not-complexity

Craddock, A., Richards, K., Tudor, D., Roberts, B. & Godwin, J. (2012). The DSDM agile project framework for scrum. Haettu 28.3.2013 osoiteesta http://www.dsdm.org/wp-content/uploads/2012/05/The-DSDM-Agile-Project-Framework-v1-1.pdf

Du, G., McElroy, J. & Ruhe, G. (2006). Ad hoc versus systematic planning of software releases – a three-staged experiment. Teoksessa J. Münch & M. Vierimaa (toim.), *Product-Focused Software Process Improvement* (s. 435–440). Berlin: Springer-Verlag.

Ebert, C. (2006). The impact of software product management. *Journal of Systems and Software, 80,* 850–861.

Ebert, C. (2009). Software product management. *CrossTalk, The Journal of Defence Software Development*, *22*(1). 15–19. Haettu 2.10.2012 osoitteesta http://www.crosstalkonline.org/storage/issue-archives/2009/200901/200901-0-Issue.pdf

Fernandes, M.C., Alencar, A.J., Schmitz, E.A., Alves, C.H. & Ferreira, A.L. (2008). A multicriteria approach to the xp release plan that maximizes business performance in uncertain environments. Teoksessa G. Louis & K. Crowther (toim.), *Proceedings of the 2008 Systems and Information Engineering Design Symposium* (s. 184–189*).* Charlottesville, VA: University of Virginia.

Fogelström, N.D., Gorschek, T., Svahnberg, M. & Olsson, P. (2009). The impact of agile principles on market-driven software product development. *Journal of Software Maintenance and Evolution: Research and Practice, 22*(1), 53–80.

Fricker, S. (2012). Software product management. Teoksessa A. Maedche, A. Botzenhardt & L. Neer (toim.), *Software for people* (s. 53–81). Berlin: Springer-Verlag.

Google (2011). Google Scholar. Haettu 3.12.2012 osoitteesta http://scholar.google.fi/

Greer, D. & Ruhe, G. (2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology, 46*(4). 243–253.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. Haettu 1.11.2012 osoitteesta

http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

Hatton, S. (2008). Choosing the right priotization method. Teoksessa F. K. Hussain & E. Chang (toim.), *Proceedings of the 19th Australian Software Engineering Conference (ASWEC)*. (s. 517–526). Los Alamitos, CA: IEEE Computer Society.

Haugen, N. (2006) An empirical study of using planning poker for user story estimation. Teoksessa J. Chao, M. Cohn, F. Maurer, H. Sharp & J. Shore (toim.), *Proceedings of the AGILE 2006 Conference*. (s. 23–34). Los Alamitos, CA: IEEE Computer Society.

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. (2010a). 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-43), [CD-ROM],* (s. 1–10). IEEE Computer Society.

Heikkilä, V., Rautiainen, K. & Jansen, S. (2010b). A revelatory case study on scaling agile release planning. Teoksessa M. Chaudron (toim.), *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010)* (s. 289–296). Los Alamitos, CA: IEEE Computer Society.

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

ISPMA, International Software Product Management Association (2012). Software product management body of knowledge (SPMBoK). Haettu 19.11.2012 osoitteesta http://ispma.org/spmbok/

Jantunen, S., Lehtola, L., Gause, D.C., Dumdum, U.R. & Barnes, R.J. (2011). The challenge of release planning - visible but not seen? Teoksessa *Proceedings of the 5th International Workshop on Software Product Management (IWSPM 2011)* (s. 36–45).

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software 70.* 37–60.

Karlsson, J. (1996). Software requirements prioritizing. Teoksessa *Proceedings of the 2nd International Conference on Requirements Engineering* (s. 110–116). Washington, DC: IEEE Computer Society.

Karlsson, L., Höst, M. & Regnell, B. (2006a). Evaluating the practical use of different measurement scales in requirements prioritisation. Teoksessa *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering* (s. 326–335). New York, NY: ACM.

Karlsson, L., Regnell, B. & Thelin, T. (2006b). A case study in retrospective analysis of release planning in agile project. Teoksessa *Proceedings of the 1st Workshop on the Interplay of Requirements Engineering and Project Management in Software Projects* (s. 1–9).

Karlsson, L., Thelin, T., Regnell, B., Berander, P. & Wohlin, C. (2007). Pair-wise comparisons versus planning game partitioning – experiments on requirements prioritisation techniques. *Empirical Software Engineering 12*(1), 3–33.

Kaur, P. (2010). Reinforcement learning based approach for adaptive release planning in an agile environment. Teoksessa *Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE)*(s. 1–4).

Kittlaus, H. & Clough, P (2009). Software product management and pricing. New York, NY: Springer.

Kittlaus, H. (2012). Software product management and agile software development: conflict and solutions. Teoksessa A. Maedche, A. Botzenhardt & L. Neer (toim.), *Software for people* (s. 83–96). Berlin: Springer-Verlag.

Kärkkäinen, S. & Högmander, H. (2008). *Tilastomenetelmien peruskurssi (5. painos).* Jyväskylän yliopisto: Matematiikan ja tilastotieteen laitos.

Lacey, M. (2012). Prioritization. Haettu 5.4.2012 osoitteesta http://msdn.microsoft.com/en-us/library/hh765981(v=vs.110).aspx

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*. Upper Saddle River, NJ: Addisson-Wesley.

Leffingwell, D. (2012). Scaled agile framework. Haettu 8.5.2013 osoitteesta http://scaledagileframework.com

Lehtola, L. & Kauppinen, M. (2004). Empirical evaluation of two requirements prioritization methods in product development projects. Teoksessa T. Dingsøyr (toim.), *Proceedings of the 11th European Software Process Improvement Conference (EuroSPI 2004)*. (s. 161–170). Berlin: Springer-Verlag.

Leppänen, M. (2005). An ontological framework and a methodical skeleton for method engineering – a contextual approach. Väitöskirja. *Studies in Computing 52.* Jyväskylä: University of Jyväskylä.

Li, C., van der Akker, M., Brinkkemper, S., Diepen, G. (2010) An integrated approach for requirement selection and scheduling in software release planning. *Requirements Engineering, 15*(4), 375–396.

Li, M., Huang M., Shu F. & Li J. (2006). A risk-driven method for extreme programming release planning. Teoksessa L.J. Osterweil, H.D. Rombach & M.L. Soffa (toim.), *Proceedings of 28th International Conference on Software Engineering (ICSE 2006)* (s. 423–430). New York, NY: ACM.

Logue, K., McDaid, K. & Greer, D. (2007). Allowing for task uncertainties and dependencies in agile release planning. Teoksessa *Proceedings of the 4th Software Measurement European Forum* (s. 275–284).

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

Logue, K. & McDaid, K. (2008b). Handling uncertainty in agile requirement prioritization and scheduling using statistical simulation. Teoksessa G. Melnik, P. Kruchten & M. Poppendieck (toim.), *Proceedings of the Agile 2008 Conference* (s. 73–82). Los Alamitos: CA: IEEE Computer Society.

Maglyas, A., Nikula, U. & Smolander, K. (2012). Lean solutions to software product management problems. *IEEE Software 29*(5), 40–46.

Mahnič, V. & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software, 85*, 2085–2095.

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

McDaid, K., Greer, D., Keenan, P., Prior, P. & Coleman, G. (2006). Managing uncertainly in agile release planning. Teoksessa K. Zhang, G. Spanoudakis & G. Visaggio (toim.), *Proceedings of 18th International Conference on Software Engineering and Knowledge Engineering (SEKE 2006)* (s. 138–143). San Francisco, CA: SEKE.

Moløkken-Østvold, K. & Haugen, N. (2007). Combining estimates with planning poker – an empirical study. Teoksessa J. Grundy & J. Han (toim.), *Proceedings of the 18th Australian Engineering Conference* (s. 349–358). Los Alamitos, CA: IEEE Computer Society.

Nagy, A., Njima, M. & Mkrtchyan, L.(2010). A bayesian based method for agile software development release planning and project health monitoring. Teoksessa F. Xhafa, S. Demetriadis, S. Caballe & A. Abraham (toim.), *Proceedings of the 2nd International Conference on Intelligent Networking and Collaborative Systems (INCoS 2010)* (192–199). Los Alamitos, CA: IEEE Computer Society.

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

Pichler, R. (2010). *Agile product management with Scrum: creating products that customer love*, Upper Saddle River, NJ: Addison-Wesley.

Racheva, Z., Daneva, M. & Buglione, L. (2008). Supporting the dynamic reprioritization of requirements in agile development of software products. Teoksessa *Proceedings of the 2nd International Workshop on Software Product Management (ISWPM 2008)* (s. 49–58). Washington, DC: IEEE Computer Society.

Racheva Z., Daneva, M. & Herrmann, A. (2010). A conceptual model of client-driven agile requirements prioritization. Teoksessa P. Loucopoulos & J.L. Cavarero (toim.), *Proceedings of the 4th International Conference on Research*

Challenges in Information Science (RCIS 2010) (s. 287–298). IEEE Computer Society.

Rautiainen, K., Lassenius, C. & Sulonen, R. (2002). 4CC: a framework for managing software product development. *Engineering Management Journal, 14*(2), 27–32.

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/pacingsoftwareproduct development.pdf

Regnell, B., Höst, M., Nat och Dag, J., Beremark, P. & Hjelm, T. (2001). An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering*, *6*(1), 51–62.

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

Ruhe, G. & Saliu, M. (2005). The art and science of software release planning. *IEEE Software, 22*(6). 47–53.

Saaty, T. (1980). *The Analytic Hierarchy Process*. New York, NY: McGraw-Hill.

Saliu, O. & Ruhe, G. (2005). Supporting software release planning decisions for evolving systems. Teoksessa *Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop (NASE SEW-29)* (s. 14–26). Los Alamitos, CA: IEEE Computer Society.

Schwaber, K. (1995). SCRUM development process. Teoksessa J. Sutherland, D. Patel, C. Casanave, G. Hollowell & J. Miller (toim.), *Proceedings of OOPSLA'95 Workshop on Business Object Design and Implementation* (s. 170–175). New York, NY: ACM.

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

Schwaber, K. & Sutherland, J. (2011). The scrum guide – scrumin määritelmä ja pelisäännöt. Haettu 6.11.2012 osoitteesta http://www.scrum.org/ Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20FI.pdf

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

van der Straaten, O. (2012). Guide line: agile estimation. Haettu 29.4.2013 osoitteesta http://epf.eclipse.org/wikis/openup/core.mgmt.common. extend_supp/guidances/guidelines/agile_estimation_A4EF42B3.html

Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Saleem, S. B. & Shafique, M. U. (2010). A systematic review on strategic release planning models. *Information and Software Technology, 52*(3), 237–248.

Szoke, A. (2010). Bin-packing-based planning of agile releases. Teoksessa L. Maciazek, C. González-Pérez & S. Jablonski, *Evaluation of Novel Approaches to Software Engineering* (s. 133–146). Berlin: Springer-Verlag.

Szoke, A. (2011a). A feature partitioning method for distributed agile release planning. Teoksessa A. Sillitti, O. Hazzan, E. Bache & X. Albaladejo (toim.), Agile Processes in Software Engineering and Extreme Programming (s. 27–42). Berlin: Springer-Verlag.

Szoke, A. (2011b). Conceptual scheduling model and optimized release scheduling for agile environments. *Information on software technology, 53*(6), 574–591.

Takeuchi, H. & Nonaka, I. (1986). The new product development game. *Harvard Business Review, 64*(1), 137–146.

van Valkenhoef, G., Tervonen, T., de Brock, B. & Postmus, D. (2010). Product and release planning practices for extreme programming. Teoksessa A. Sillitti, A. Martin, X. Wang & E. Whitworth (toim.), *Agile Processes in Software Engineering and Extreme Programming* (s. 283–243). Berlin: Springer-Verlag.

van Valkenhoef, G., Tervonen, T., de Brock, B. & Postmus, D. (2011). Quantitative release planning in extreme programming. *Information and Software Technology, 53*(11), 1227–1235.

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., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: applying SCRUM principles to software product management. *Information and Software Technology, 53*(1), 58–70.

Vähäniitty, J. & Rautiainen, K. (2008). Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development. Teoksessa *Proceedings of the 1st International Workshop on Software Development Governance* (s. 25–28). New York, NY: ACM.

Vähäniitty, J. (2010a). Agile product and portfolio management – crucial for competitiveness. Teoksessa V. Heikkilä, K. Rautiainen & J. Vähäniitty (toim.), *Towards Agile Product and Portfolio Management* (s. 31–37). Espoo: Aalto University.

Vähäniitty, J. (2010b). Agile product management. Teoksessa V. Heikkilä, K. Rautiainen & J. Vähäniitty (toim.), *Towards Agile Product and Portfolio Management* (s. 115–125). 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 M. Glinz & R. Lutz, *Proceedings of the 14th IEEE International Conference on the Requirements Engineering* (s. 319–322). Washington, DC: IEEE Computer Society.

Wohlin, C. & Aurum, A. (2005). What is important when deciding to include a software requirement in a project or release? Teoksessa J. Vemer & G.H. Guilherme (toim.), *Proceedings of the International Symposium on Empirical Software Engineering (ISESE 2005)* (s. 246–255). Los Alamitos, CA: IEEE Computer Society.

Xu, L. & Brinkkemper, S. (2007). Concept of product software. *European Journal of Information Systems, 16*, 531–541.

LIITE 1 TUTKIMUKSEN TIEDONHANKINTATAPA

Tutkimuksessa käytettiin keskeisinä lähteinä Cohnin (2006) ja Leffingwell (2011) teoksia. Tämän lisäksi marraskuussa 2012 haettiin tietoa usealla hakulauseella tieteellisiä artikkeleja sisältävistä tietokannoista, jotka olivat ACM, IEEE eXplore, SpringerLink ja Science Direct. Hakujen päätarkoituksena oli löytää julkaisunsuunnittelun ehdotuksia ja -tekniikoita, joiden on esitetty soveltuvan ketterään kehittämiseen. Sopiviksi ehdotuksiksi ja tekniikoiksi katsottiin myös ne esitykset, joita oli jonkin toisen tutkimuksen yhteydessä ehdotettu käytettäväksi ketterän kehittämisen yhteydessä. Toisena tavoitteena oli kerätä yleistä taustamateriaalia aihepiiristä, mm. tuotehallinnasta.

1. haku, joka kohdistettiin tutkimuksien koko teksteihin:

("release planning" OR "release plan" OR "planning release" OR "agile planning" OR "product planning")
AND
(agile OR xp OR "extreme programming" OR xp OR scrum OR kanban OR lean)

Edellisellä hakulauseella etsittiin IEEE eXplore –tietokannan kohdalla vain tutkimusten metatiedoista (otsikko, abstrakti ja avainsanat) hakutulosten määrän rajoittamiseksi. IEEE eXplore -tietokannan haku suoritettiin kokotekstihakuna niin, että hakulausetta supistettiin seuraavanlaiseksi:

("release planning" OR "release plan" OR "planning release" OR "agile planning")
AND
(agile OR xp OR "extreme programming" OR scrum)

Ensimmäisen haun tuloksista luettiin vain otsikko, abstrakti ja johdanto. Talteen kerättiin julkaisunsuunnitteluun tai tekniikoihin liittyviä tutkimuksia. Löydetyt tutkimukset jaoteltiin karkeasti yleisiin kategorioihin esimerkiksi ”julkaisunsuunnittelun ehdotukset”, ”julkaisunsuunnittelun tekniikat”, ja ”vaatimusten priorisointia koskevat” ja ”julkaisunsuunnittelun ehdotuksia sivuavat” -ryhmiin. Haun tarkoituksena oli löytää ketterään julkaisunsuunnittelun ehdotuksia.

2. haku, jossa haettiin tietoja tutkimuksien abstrakteista:

"release planning" OR "release plan" OR "planning release" OR "agile planning" OR "requirements prioritisation" OR "requirements prioritization" OR "requirements selection"

Toisen haun tuloksista luettiin otsikko ja abstrakti. Talteen otettiin kaikki tutkimukset, jotka sopivat ensimmäisen haussa käytettyihin yleisiin kategorioihin erottelematta liittyivätkö tutkimukset ketterään kehittämiseen. Haun tarkoituksena oli kerätä yleistä tutkimuksen aihepiiriin liittyvää taustamateriaalia. Aja-

tuksena oli, että suuresta tutkimus joukosta voidaan tunnistaa varmemmin aihepiirin kannalta keskeisiä lähteitä. Samalla pyrittiin löytämään uusia ketterän julkaisunsuunnittelun ehdotuksia.

3. hakulause, joka kohdistettiin tutkimuksien koko teksteihin:

"*software product management*"

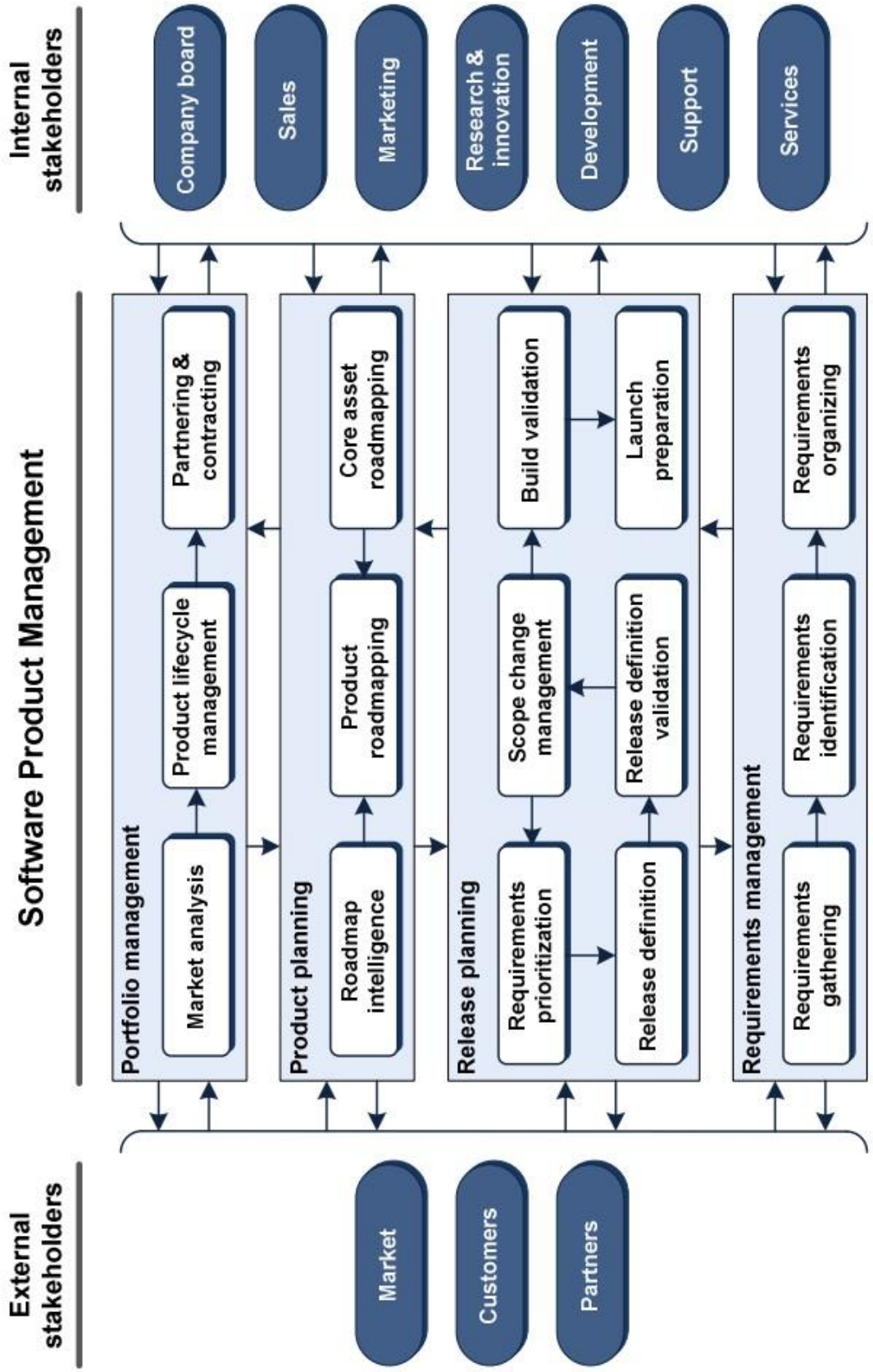
Kolmannen haun tuloksista luettiin otsikko ja abstrakti. Tutkimus otettiin talteen, jos se näytti käsittelevän ohjelmistotuotehallintaa tai ensimmäisen haun yhteydessä käytettyjä kategorioita (esimerkiksi ”julkaisunsuunnittelun ehdotukset”). Haun tarkoituksena oli löytää taustatietoa tuotehallinnan kuvaamiseen.

Hauissa tutkimukset sivuutettiin seuraavissa tapauksissa:

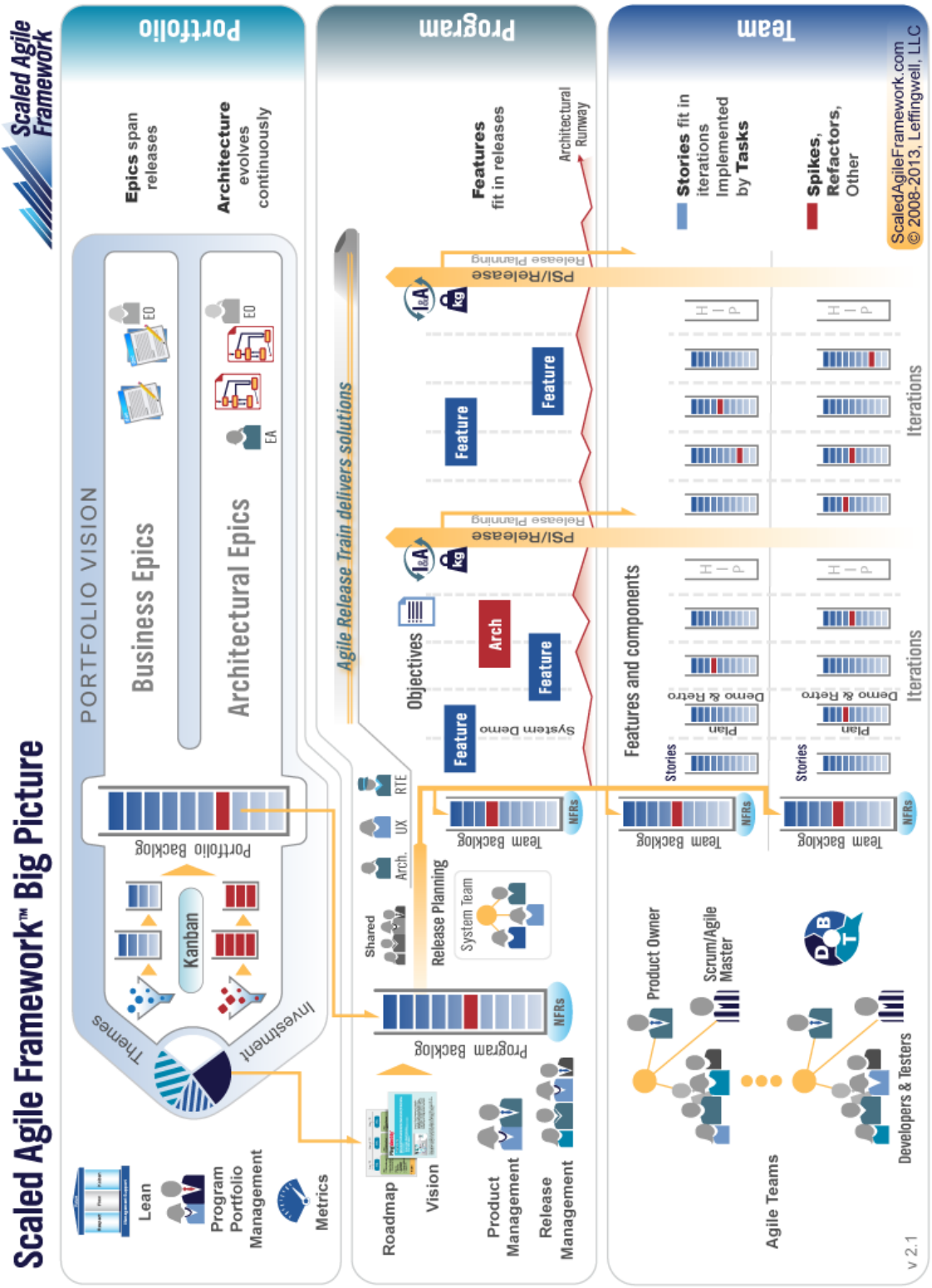
- Vain abstrakti oli luettavissa.
- Tutkimus oli toisen tutkimuksen arvostelu.
- Tutkimus ei ollut englanninkielinen.
- Tutkimus käsitteli vain työkaluja.

Related work –osuuksiin tutustuminen:
Löydettyjen ketterien julkaisunsuunnittelun ehdotusten ”related work” tai vastaavat osuudet luettiin tarkasti. Tarkoituksena oli näin etsiä ehdotuksia joita kirjallisuuskatsauksessa ei ollut löydetty.

LIITE 2 BEKKERSIN YM. (2010) KOMPETENSSIMALLI



LIITE 3 LEFFINGWELLIN (2012) BIG PICTURE -MALLI



LIITE 4 RACHEVAN YM. (2008) ESITTÄMÄT PRIORISOINTITEKNIIKAT

Lähteiden esitystyyliä on muokattu vastaamaan paremmin tutkimuksessa käytettyä tyyliä. Joidenkin lähteiden www-osoitteita on korjattu osoittamaan suoraan oletettuun lähdeartikkeliin.

Tekniikan nimi	**Lähde**
Round-the-group prioritization	Berteig, M. (2006). Methods of Prioritization. Saatavilla osoitteesta http://www.agileadvice.com/2006/03/20/agilemanagement/methods-of-prioritization/
Ping Pong Balls	Schwaber, K. & Beedle M. (2004). *Agile software development with Scrum*, Microsoft Press.
$100 allocation (cumulative voting)	Leffingwell, D. & Widrig, D. (2003). Managing software requirements: A use case approach (2nd edition). Boston, MA: Addison-Wesley
Multi-voting system	Tabaka, J. (2006). *Collaboration explained: facilitation skills for software project leaders.* Addison-Wesley.
MoSCoW	ORACLE (2007). Getting starter with use case modeling. An Oracle White Paper. Saatavilla osoitteesta http://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf
Pair-wise analysis	Gottesdiener, E. (2007). At a glance: other prioritization methods. Saatavilla osoitteesta http://www.ebgconsulting.com/Pubs/Articles/At%20a%20Glance-Other%20Prioritization%20Methods-supplement-EBG%20Consulting.pdf
Weighted criteria analysis	
Dot voting	
Analytic Hierarchy Process (AHP)	Saaty, T. (1980). *The Analytic Hierarchy Process*, New York, NY: McGraw-Hill.
Binary Search Tree	Ahl, V. (2005). An Experimental Comparison of Five Prioritization Methods. Master's Thesis. Ronneby: School of Engineering, Blekinge Institute of Technology.
Ranking based on product definition	Fraser, J. (2002). Setting Priorities. Saatavilla osoitteesta http://www.adaptivepath.com/ideas/essays/archives/000018.php
Planning Game	Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley.
Quality functional deployment (QFD)	Crow, K. (2002). Customer-focused development with QFD. Saatavilla osoitteesta http://www.npd-solutions.com/qfd.html Gottesdiener, E. (2007). At a glance: other prioritization methods. Saatavilla osoitteesta http://www.ebgconsulting.com/Pubs/Articles/At%20a%20Glance-Other%20Prioritization%20Methods-supplement-EBG%20Consulting.pdf
Wieger's matrix approach	Wiegers, K. (1999). First things first: prioritizing requirements. *Software Development 7*(9). Saatavilla osoitteesta http://www.processimpact.com/articles/prioritizing.html
Mathematical programming techniques for release planning.	Li, C., van den Akker, J., Brinkkemper, S & Diepen, G. (2007). Integrated requirement selection and scheduling for the release planning of software product. Teoksessa *Requirements Engineering: Foundations of Software Quality* (s. 93-108). Berlin: Springer-Verlag.