Helena Maukonen

KETTERÄN OHJELMISTOKEHITYKSEN KYPSYYSMALLIEN VERTAILU

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2015

TIIVISTELMÄ

Maukonen, Helena
Ketterän ohjelmistokehityksen kypsyysmallien vertailu
Jyväskylä: Jyväskylän yliopisto, 2015, 83 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja: Leppänen, Mauri

Teknologian nopea kehittyminen ja liiketoimintaympäristön muutokset vaativat ohjelmistokehitykseltä nopeaa reagointikykyä ja lyhyttä vasteaikaa haluttujen ohjelmistotuotteiden ja palvelujen tuotannossa. Ratkaisuksi on usein nähty siirtyminen ketterien menetelmien käyttöön. Ketterien menetelmien käyttö pidemmällä aikavälillä on kuitenkin tuonut tarpeen arvioida organisaation, projektin ja tiimin ketterän kehittämisen tilaa ja suunnitella tapoja parantaa sitä. Organisaation tai sen osan tilaa tai kehitysvaihetta on totuttu kuvaamaan ja arvioimaan kypsyysmallien avulla. Koska perinteiset kypsyysmallit sopivat huonosti ketterän ohjelmistokehityksen arviointiin, on sille alettu kehittää omia kypsyysmalleja.

Tämän tutkielman tarkoituksena on kuvata ja verrata ketterään ohjelmistokehitykseen esitettyjä kypsyysmalleja. Kypsyysmalleja on etsitty käyttämällä tutkimustietokantoja ja Google-hakuja. Mukaan otettiin yksitoista kypsyysmallia. Kustakin mallista kerrotaan, mitä tarkoitusta varten malli on kehitetty, minkälaisista tasoista se koostuu sekä onko mallia käytetty ja/tai validoitu. Työssä määritellään myös seitsemän kriteeriä, joita käyttäen malleja vertaillaan monipuolisesti. Mallien vertailukriteereinä käytetään menetelmä-sidonnaisuutta, kohdealuetta, käyttötarkoitusta, rakennetta, esitystä, käyttöä sekä testausta ja validointia.

Tutkimuksen tuloksia voidaan hyödyntää myös käytännön työssä, sillä tutkimus antaa yleiskuvan mallitarjonnasta ja auttaa valitsemaan sopivan mallin oman organisaation käyttöön.

Asiasanat: ketterä ohjelmistokehitys, ketterät menetelmät, XP, Scrum, kypsyysmalli, ketterän ohjelmistokehityksen kypsyysmalli

ABSTRACT

Maukonen, Helena
A Comparison of Maturity Models for Agile Software Development
Jyväskylä: University of Jyväskylä, 2015, 83 p.
Information Systems, Master's Thesis
Supervisor: Leppänen, Mauri

The rapid development of technology and changes in the business environment require quick reactions and short response time in the production of desired software products and services. A solution is often seen to be in the transition to agile methods. However, the use of agile methods in the longer term has raised a need to assess an organization, a project, and a team in terms of agility, and to plan ways to improve it. It is common to use maturity models to describe and assess the state of an organization or part of it. Since conventional maturity models, such as CMM and CMMI, poorly suit to agile software development evaluation, new maturity models specific to the agile approach have been developed.

The purpose of this study is to describe and compare maturity models presented for agile software development. Maturity models for the review have been sought through research databases and Google. Eleven maturity models were chosen. For each model, we describe the purpose for which the model has been developed, what kind of levels it consists of, as well as whether the model has been used and / or validated. The thesis also defines criteria by which models are compared to each other in a versatile manner. The following criteria are used: method specificity, target domain, purpose, structure, presentation, use, testing and validation.

The results of the study can be utilized in practical work as the study provides an overview of the models available and it will help to choose a suitable model for the organization.

Keywords: agile software development, agile method, XP, Scrum, maturity model, agile software development maturity model

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Viime aikoina yhä useampi yritys on muuttanut ainakin osan ohjelmistokehityksestään ketterän lähestymistavan mukaiseksi (VersionOne, 2014; Rodriquez, Markkula, Oivo & Turula, 2012). Myös monissa suurissa yrityksissä on siirrytty käyttämään ketteriä toimintatapoja (Laanti, 2013). Ketterän kehittämisen avulla voidaan pyrkiä muun muassa parantamaan tuottavuutta, nopeuttamaan prosesseja, vähentämää kuluja, reagoimaan nopeammin muutoksiin ja näin vastaamaan paremmin asiakkaiden tarpeisiin (Dybå & Dingsøyr, 2008; Abrahamsson, Salo & Ronkainen, 2002; Pikkarainen, Haikara, Salo, Abrahamsson & Still, 2008). Ketterät menetelmät ottavat käyttäjät paremmin huomioon, ja näin heidän sitoutumisensa paranee. Yritysjohdon kannalta tärkeimmät edut ovat tuottavuuden lisääntyminen, prosessin läpinäkyvyys ja nopea reagointikyky (Dybå & Dingsøyr, 2008).

Yritysten käytettyä ketteriä toimintatapoja jonkin aikaa, herää mielenkiinto selvittää, miten niiden toiminta sijoittuu ketterän lähestymistavan arvojen ja periaatteiden suhteen ja missä niillä olisi vielä kehitettävää (Highsmith, 2006). Yritykset kaipaavat jonkinlaista mittavälinettä, jonka avulla he voisivat arvioida omaa ketterää ohjelmistokehitystään ja tehdä suunnitelmia prosessinsa parantamiseksi. Yrityksissä on totuttu käyttämään standardeja lähtökohtana toiminnan laadukkaalle kehittämiselle. Sertifioinnin tai arvioinnin tulokset välittävät asiakkaalle laatukuvan toimittajasta.

Organisaation tai sen osan tilaa tai kehitysvaihetta on totuttu kuvaamaan ja arvioimaan kypsyysmallien avulla. *Kypsyysmallien* avulla voidaan analyyttisesti arvioida ja mitata organisaation ja prosessien kypsyyttä (maturity) tai/ja kyvykkyyttä (capability). Mallit ohjaavat organisaatioita toimimaan järjestelmällisesti ja parantamaan toimintatapaansa arvioinnin tuloksena (Mettler & Rohner, 2009). Kypsyysmallit koostuvat tyypillisesti viidestä tai kuudesta tasosta. Kypsyysmalleja on kehitetty erilaisiin tarkoituksiin kuten esimerkiksi ohjelmistokehitykseen, testaukseen ja tietoturvallisuuteen (De Bruin, Freeze, Kulkarni & Rosemann, 2005). Portaittaisessa (staged) mallissa kypsyystaso esitetään prosessialueiden kypsyyden perusteella ja jatkuvassa (continuous) mallissa tarkastellaan prosessialueiden kyvykkyys- tai kypsyystasoja. Perinteisen tie-

tojärjestelmäkehityksen prosessien arviointiin on kehitetty lukuisia arviointimalleja. Viime vuosina on laajalti käytetty CMM- (Capability Maturity Model) (Paulk, Curtis, Chrissis & Weber, 2003) ja CMMI- (Capability Maturity Model Integration) malleja (SEI, 2010), kun on arvioitu organisatorista kypsyyttä ja prosessien kyvykkyyttä.

Ketterän kehittämisen arviointiin ja etenkin kokemattomille tiimeille perinteiset kypsyysmallit, kuten CMM, ovat kuitenkin liian laajoja ja raskaita. Tällaisten mallien suoraviivainen käyttöönotto on epäkäytännöllistä ja aikaa vievää, niin kokemattoman tiimin koulutuksen, kuin mallin toteuttamisenkin suhteen. Nämä mallit ovat myös sopimattomia sen takia, että ne korostavat prosessin kyvykkyyttä ja kypsyyttä, mutta eivät pysty tukemaan kokemattomia tiimejä tuottamaan parempia ohjelmistoja eivätkä tuomaan liiketoiminta-arvoa. (Lui & Chan, 2005.).

Tarve saada arviointitukea ja linjausta prosessin parantamiseksi on johtanut erityisesti ketterää kehittämistä varten tarkoitettujen kypsyysmallien esittämiseen (Proulx, 2010). Kirjallisuudessa on esitelty useita erilaisia ketterän ohjelmistokehityksen kypsyysmalleja (Schweigert, Vohwinkel, Korsaa, Nevalainen & Biro, 2013a). Osa niistä on tarkoitettu käytettäväksi tietyn ketterän menetelmän yhteydessä, kuten Scrumin (Schwaber, 2004) tai XP:n (Beck, 1999) yhteydessä, osa puolestaan on menetelmäriippumattomia ja niitä voidaan soveltaa ketterään kehittämiseen yleisesti. Osaa menetelmistä voidaan käyttää sekä projekteille että koko organisaatiolle, osa on tarkoitettu vain projekteille tai tiimeille. Vielä ei ole kuitenkaan löydetty yhteisesti hyväksyttyä, ketterille menetelmille tarkoitettua kypsyysmallia (Schweigert, Nevalainen, Vohwinkel, Korsaa & Biro, 2012; Schweigert, Vohwinkel, Korsaa, Nevalainen & Biro, 2013b).

Tämän tutkielman tavoitteena on kuvata kirjallisuudessa esitettyjä ketterän ohjelmistokehityksen kypsyysmalleja ja verrata niitä toisiinsa. Työn tutkimusongelma voidaan muotoilla seuraavasti: *Millaisia yhtäläisiä ja erilaisia piirteitä ketterään ohjelmistokehitykseen tarkoitetuilla kypsyysmalleilla on*? Tämä voidaan jakaa seuraaviin tutkimuskysymyksiin: *Mitä tarkoitetaan ketterällä ohjelmistokehityksellä ja millaisia ketteriä menetelmiä on olemassa? Mitä tarkoitetaan kypsyysmallilla ja millaisia kypsyysmalleja on olemassa? Millaisia kypsyysmalleja ketterään ohjelmistokehitykseen on esitetty ja miten ne vertautuvat toisiinsa?*

Kypsyysmalleja etsitään tekemällä hakuja tutkimus-tietokantoihin (esimerkiksi IEEExplore, ACM Digital Library), tutustumalla alan konferenssien (esimerkiksi XP- ja AGILE-konferenssit) kokoomateoksiin ja ohjelmistokehitystä koskeviin lehtiin sekä suorittamalla Google-hakuja. Mallien vertailua varten määritellään joukko vertailukriteerejä. Tutkimus on luonteeltaan käsitteellis-teoreettinen.

Tarkasteltaviksi valittiin ensisijassa malleja, jotka jo nimeltään ovat kypsyysmalleja (esimerkiksi The Agile Maturity Model, (Ambler, 2010)). Mukaan otettiin myös esityksiä, joita ei kutsuta kypsyysmalleiksi, mutta jotka sisältävät tasoja, joita voidaan käyttää ketteryyden arvioimiseen. Tarkasteltaviksi valittiin seuraavat mallit: The Agile Maturity Model (Ambler, 2010), Seven Dimensions of Agile Maturity in the Global Enterprise (Benefield, 2010), The Agile Maturity

Model (Gujral & Jayaraj, 2008), The Agile Maturity Model Applied to Building and Releasing Software (Humble & Russell, 2009), A Road Map for Implementing eXtreme Programming (Lui & Chan, 2005), Simple Lifecycle Agility Maturity Model (Malik, 2007), Maturity Model for eXtreme Programming (Nawrocki, Walter & Wjochiechowski, 2001), The Agile Maturity Map (Packlick, 2007), Agile Maturity Model (AMM) (Patel & Ramachandran, 2009), An "Agile Maturity Model?" (Pettit, 2006), Agile Maturity Model (AMM) (Proulx, 2010), Agile Adoption and Improvement Model (Qumer & Henderson-Sellers, 2008), A Disciplined Approach to Adopting Agile Practices: the Agile Adoption Framework (Sidky, Arthur & Bohner, 2007), Scrum Maturity Model (Yin, Figueiredo & da Silva, 2011).

Tutkielma on jaettu kuuteen lukuun. Luvussa 2 kuvataan lyhyesti ketterien menetelmien taustaa, niiden yleisiä periaatteita, mahdollisia hyötyjä sekä haasteita. Tämän jälkeen kahta suosituinta ketterää menetelmää esitellään hieman tarkemmin. Luvussa 3 kuvataan kypsyysmalleja yleisesti sekä esitellään hieman tarkemmin kypsyysmallit CMM ja CMMI. Luvussa 4 kutakin valittua ketterän ohjelmistokehityksen kypsyysmallia kuvataan tarkemmin. Kuvauksista käyvät ilmi, mitä tarkoitusta varten malli on kehitetty, minkä rakenteinen se on, millä periaatteella se on rakennettu sekä onko mallia käytetty ja validoitu. Luvussa 5 vertaillaan esiteltyjä ketterään ohjelmistokehitykseen tarkoitettuja kypsyysmalleja määriteltyjen kriteerien mukaisesti. Tutkielma päättyy yhteenvetoon.

2 KETTERÄ OHJELMISTOKEHITYS

Tässä luvussa tarkastellaan ketterää ohjelmistokehitystä. Aluksi kerrotaan ketterästä lähestymistavasta, sen arvoista ja periaatteista. Tässä yhteydessä mainitaan myös ketteryyttä koskevista erilaisista käsityksistä sekä ketterästä ohjelmistokehityksestä koetuista hyödyistä ja ongelmista. Tämän jälkeen kuvataan kahta yleisintä ketterää kehittämismenetelmää, XP:tä (eXtreme Programming) ja Scrumia.

2.1 Ketterä lähestymistapa

Ketterän ohjelmistokehityksen yhteiset perusarvot ja periaatteet on määritelty Agile-manifestissa (Beck ym., 2001). Tämä ketterän ohjelmistokehityksen julistus syntyi vuonna 2001, kun ryhmä ohjelmistoalan ammattilaisia kokoontui yhteen ja sopi perusarvot ja periaatteet ketterille menetelmille. Tällöin otettiin myös ensi kertaa termi ”ketterä” (agile) käyttöön, kuvaamaan näitä uusia ohjelmistokehityksen menetelmiä (Abrahamsson ym., 2002). Ennen manifestin esittämistä oli ohjelmistokehityksessä ollut jo kevyitä kehittämismenetelmiä ja tekniikoita (Larman & Basili, 2003; Abbas, Gravell & Wills, 2008; Schwaber, 1995; Schwaber, 2000; Beck, 1999; Stapleton, 1997, Highsmith, 2000), mutta nyt nekin liitettiin yhteisen nimen alle.

Agile-manifesti (Beck ym., 2001) määrittelee neljä arvoa seuraavasti:

> Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:
>
> - Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
> - Toimivaa sovellusta enemmän kuin kattavaa dokumentaatiota
> - Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja

- Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän. (Beck ym., 2001.).

Fowlerin ja Highsmithin (2001) mukaan ilmaisujen molemmat osat ovat tärkeitä, mutta kyse on siitä, kumpi on vielä tärkeämpää. Tärkeysjärjestys perustuu siihen, kumpi arvoista edustaa paremmin ketteriä arvoja. Ensimmäinen arvo korostaa taitavien yksilöiden ja heidän välisensä vuorovaikutuksen tärkeyttä. Käytännössä tämä arvo voidaan nähdä esimerkiksi tiiviinä työympäristönä, joka kannustaa keskusteluihin ja luovuuteen. Mutta myös huippuyksilöt ja –tiimit tarvitsevat kunnolliset työkalut ja menetelmät pystyäkseen toimimaan täysipainoisesti. Toisen arvon tarkoituksena on helpottaa kehittäjien dokumentointityötä, koska ketterässä ohjelmistokehityksessä tyypillisesti julkaistaan ohjelmistosta uusia versioita tiheään tahtiin. Versioiden tarkka dokumentointi vaatisi kohtuuttoman määrän työtä, joten työryhmille annetaan päätösvalta riittävän dokumentoinnin tuottamiseksi. Asiakkaan kannalta on kuitenkin tärkeämpää saada toimiva sovellus kuin kattava dokumentaatio. Kolmas arvo perustuu siihen, että asiakkaan vaatimukset ymmärretään paremmin, jos asiakas tekee yhteistyötä ohjelmistokehittäjien kanssa. Tiivis yhteistyö auttaa molempien osapuolien edustajia ymmärtämään paremmin valmistettavaa ohjelmistoa, jolloin on todennäköisempää, että toimitettu ohjelmisto vastaa asiakkaan odotuksia. Neljännen arvon avulla korostetaan joustavuutta ja nopeaa reagointia muuttuneisiin vaatimuksiin ja olosuhteisiin. Usein nopea reagointikyky on edellytyksenä projektin onnistumiselle. Tämä edellyttää sitä, että kehittäjillä on valta tehdä muutosten aiheuttamat korjaukset ja että kehittäjät ovat myös valmiita tekemään tarvittavat korjaustoimenpiteet. (Fowler & Highsmith, 2001; Abrahamsson, Salo, Ronkainen & Warsta, 2002)

Agile-manifesti (Beck ym., 2001) esittää myös kaksitoista ketterän ohjelmistokehityksen periaatetta. Näiden periaatteiden mukaan tärkein tavoite on pitää huolta asiakastyytyväisyydestä muun muassa toimittamalla asiakkaan tarpeita täyttäviä versioita ohjelmistosta tarpeeksi aikaisessa vaiheessa sekä säännöllisesti. Myös päivittäinen yhteistoiminta ja kasvokkain kommunikointi asiakkaan ja ohjelmistokehittäjien välillä sekä kehitystiimin kesken on ehdottoman tärkeää, jotta muuttuviin vaatimuksiin pystytään vastaamaan nopeasti.

Agile-manifesti ei määrittele tarkasti, millainen ketterä menetelmä on. Manifestia kohtaan onkin esitetty kritiikkiä, jonka mukaan se on liian epämääräinen käytettäväksi tieteellisen työn pohjana (Laanti, Similä & Abrahamsson, 2013). Conboy ja Fitzgerald (2004) väittävät, ettei se sisällä riittävää pohjaa johtamisteorioista ja filosofiasta. Abrahamssonin ym. (2002) mukaan ketterä menetelmä opastaa toimimaan lisäävästi (incremental), korostaa yhteistyötä ja asiakkasta, on yksinkertainen ja helposti opittava sekä muutoksiin sopeutuva. Qumerin ja Henderson–Sellersin (2006a) mukaan joustavuus, nopeus, keveys, oppiminen ja reagointikyky ovat ketteryyden ominaisuuksia. Highsmithin ja Cockburnin (2001) mukaan ketteryydessä on kyse muutoksen luomisesta ja muutokseen vastaamisesta. Heidän mukaansa ketterissä menetelmissä ei ole

uutta niissä esiintyvät käytänteet, vaan ihmisten tunnistaminen menestymisen avaintekijöiksi yhdessä tehokkuuteen keskittymisen ja ohjattavuuden kanssa. Nämä yhdessä muodostavat yhdistelmän arvoja ja periaatteita, jotka määrittelevät ketterän kehittämisen (Higshmith & Cockburn, 2001). Conboyn (2009) mukaan menetelmän tulee, ollakseen ketterä, vaikuttaa yhdellä tai useammalla seuraavista tavoista: aikaansaada muutosta, ennakoida muutosta, reagoida muutokseen tai oppia muutoksesta. Lisäksi ketterän menetelmän pitää edistää taloudellisuutta, laatua sekä yksinkertaisuutta ja olla jatkuvasti valmiina valmistamaan ohjelman osa (komponentti) käyttöä varten.

Boehmin (2002) mukaan ketterillä menetelmillä tarkoitetaan keveitä, joustavia ja nopeasti muutoksiin reagoivia ohjelmistokehitysmenetelmiä. Ketterissä menetelmissä kehitysprosessi toteutetaan lyhyinä iteratiivisina ja inkrementaalisina sykleinä, jolla ohjelmistoa kasvatetaan vähitellen. Prosessissa vaaditaan, että asiakas on aktiivisesti mukana määrittelemässä, priorisoimassa ja verifioimassa vaatimuksia. Olennainen osa kehitysprosessia on itseohjautuvien (self-organizing) tiimien käyttäminen, joissa niiden annetaan itse päättää työn organisoinnista ja tavasta. Kehitysprosessi ei ole sidottu tiettyyn kaavaan, vaan sen pitäisi antaa muodostua ja täydentyä projektin edetessä. Kehitysprosessille on kuitenkin tyypillistä, että toiminnalliset vaatimukset priorisoidaan ja tärkeimmät niistä toteutetaan ensimmäisinä. (Boehm, 2002.)

Ketteriä menetelmiä on tutkittu jonkin verran myös empiirisesti. Useissa tutkimuksissa (esim. Bustard, Coleraine, Wilkie & Greer, 2013; Rodriguez, Markkula, Oivo & Turula, 2012) todetaan, että ketterät menetelmät ovat helposti omaksuttavissa, ja että ne myös toimivat hyvin. Tutkimuksissa korostetaan lisääntynyttä asiakastyytyväisyyttä, prosessin kehittymistä sekä lopputuotteen laadun paranemista (Boehm & Turner, 2003; Highsmith, 2004; Anderson, 2005). Dybån ja Dingsøyrin (2008) tekemässä systemaattisessa kirjallisuuskatsauksessa todetaan kommunikaation sekä palautteenannon lisääntyneen ketteriä menetelmiä käyttämällä. Myös säännölliset tapaamiset tiimin kesken lisäsivät yhteistyötä ja antoivat kaikille paremman kuvan työn etenemisestä. Lisäksi asiakkaat nähtiin arvokkaina resursseina ja vastavuoroisesti asiakkaat kokivat pääsevänsä aktiivisesti osallistumaan ja vaikuttamaan prosessiin. Prosessin kontrolli, läpinäkyvyys ja laatu kasvoivat jatkuvan integroinnin ja hallittavan kokoisten tehtävien ansiosta.

Ketteriä menetelmiä käytettäessä, muutostarpeiden nopean huomioimisen ansiosta, kehitetty tuote vastaa paremmin asiakkaan sekä teknologia- ja liiketoimintaympäristön tarpeita ja vaatimuksia (Cao & Ramesh, 2008; Figueiredo, 2009). Kun kehitystiimi tekee säännöllistä yhteistyötä asiakkaan kanssa ja saa jatkuvasti palautetta tekemästään työstä, kehitettävä tuote vastaa paremmin asiakkaan vaatimuksia ja on käyttökelpoinen. Sekä kehitystiimin että asiakkaan tyytyväisyys kehitettävää tuotetta kohtaan kasvaa myös (Mann & Maurer, 2005). Ongelmien raportointi ja avun pyytäminen heti ongelmien ilmaannuttua, vähentää huomattavasti tuotteeseen liittyviä virheitä, helpottaa ja nopeuttaa virheiden korjaamista ja auttaa kehitystiimiä pysymään aikataulussa (Coram & Bohner, 2005; Schatz & Abdelshafi, 2005).

Dybån ja Dingsøyrin (2008) tutkimuksessa tunnistettiin myös joitakin ketteriin menetelmiin liittyviä ongelmia. Ketterässä kehittämisessä tapahtuvan jatkuvan testauksen on todettu vaativan paljon resursseja ja myös integroidun testausympäristön luominen on ollut haastavaa. Jos ohjelmiston arkkitehtuurin suunnitteluun ei ole iteratiivisessa suunnittelussa kiinnitetty tarpeeksi huomiota, on tämä saattanut johtaa huonoihin suunnitteluratkaisuihin. Ketterien menetelmien skaalautuvuudessa on myös todettu olevan ongelmia. Tiimien välinen kommunikaatio saattoi olla heikkoa, vaikka tiimin sisäinen kommunikaatio toimikin. Johtajat eivät aina ymmärtäneet rooliaan tiimin työskentelyn mahdollistajana. Suurempi rooli kehitystyössä saattoi olla asiakkaalle stressaavaa.

Ketterien menetelmien on väitetty sopivan vain tietynlaisille projekteille (esimerkiksi pienet tiimit ja sovellukset) (Boehm & Turner, 2003; McMahon, 2005). Ketterien menetelmien käyttöönotto edellyttää usein suuria muutoksia. Käyttöönotto voikin olla työlästä, aikaa vievää ja haastavaa, mikä puolestaan voi aiheuttaa muutosvastarintaa kehitystiimin keskuudessa. (Schatz & Abdelshafi, 2005.). Koska kehitystiimit ovat pieniä, kehittäjiltä vaaditaan usein laajaa, esimerkiksi tuotteen suunnitteluun, toteutukseen ja testaamiseen liittyvää, osaamista (Pikkarainen & Wang, 2011). Projekteihin voi kuitenkin olla vaikea löytää kehittäjiä, joilla on tarpeeksi laaja osaaminen ja kokeneemmat kehittäjät voivat joutua neuvomaan kokemattomampia kehittäjiä, mikä vie aikaa varsinaiselta työnteolta (Conboy ym., 2011). Jos yrityksissä puolestaan arvostetaan kehittäjien erityisosaamista tai kehittäjät haluavat säilyttää oman erityisosaamisensa, kehittäjien välinen tiedon jakaminen voi olla vähäistä (Moe ym., 2010).

Ketteriä menetelmiä on julkaistu parin vuosikymmenen aikana lukuisia. Näistä tunnetuimpia ovat XP (eXtreme Programming) (Beck, 1999; Beck & Anders, 2004), Scrum (Schwaber, 2000; Schwaber & Sutherland, 2013), Kanban (Anderson, 2010), FDD (Feature Driven Development) (Palmer & Felsing, 2002) ja Adaptive Software Development (Highsmith, 2000). Käytetyimpiä ovat Scrum, XP ja niiden erilaiset variantit (VersionOne, 2014; Rodriquez ym., 2012). Ne ovat myös hyvin dokumentoituja menetelmiä (Salo ja Abrahamsson, 2008). Seuraavaksi kuvataan hieman lähemmin kahta menetelmää, XP:tä ja Scrumia.

2.2 XP

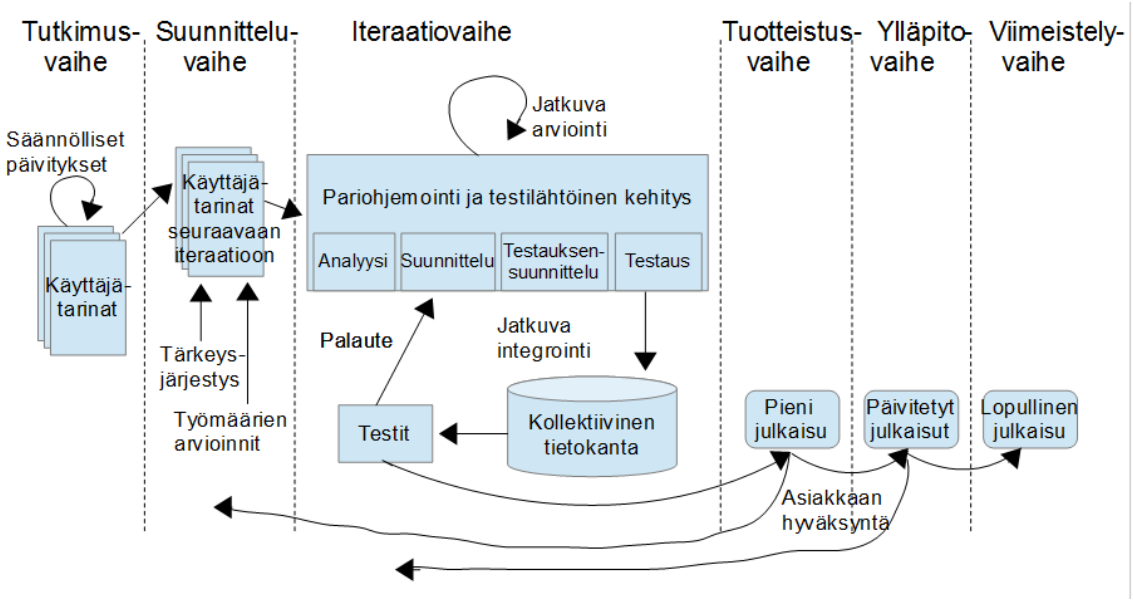
XP (eXtreme Programming) on kehitetty alun alkaen pieniä ja keskisuuria työryhmiä varten, jotka kehittävät ohjelmistoja jatkuvasti muuttuvien vaatimusten keskellä (Beck, 1999). Ennen menetelmän julkaisua monet XP-menetelmän käytännöt olivat jo olemassa, Beck vain esitteli nämä yhdessä paketissa. XP muodostaa aikaisempia käytäntöjä yhdistelemällä uuden tavan kehittää ohjelmistoja (Abrahamsson ym., 2002). Menetelmässä hyväksi havaitut käytännöt ja periaatteet viedään äärimmäisyyksiin (to the extreme). XP-menetelmä sisälsi alun perin 13 käytännettä, joiden yhteiskäytöllä pyrittiin saavuttamaan ketterämpi kehitys. Myöhemmin Beck ja Anders (2004) julkaisivat uuden version menetelmästä, joka sisältää useampia käytänteitä (kolmetoista pääkäytännettä ja yksi-

toista muuta käytännettä). Seuraavassa kerrotaan lyhyesti XP:n käytänteistä alkuperäisen version (Beck, 1999) mukaisesti.

Pariohjelmointi, tarkoittaa ohjelmointitapaa, jossa kaksi henkilöä työskentelee yhdessä samalla tietokoneella. *Koodin yhteisomistajuus* puolestaan tarkoittaa sitä, että tiimin jäsenet yhdessä omistavat koodin ja ovat vastuussa siitä. Kuka tahansa tiimistä voi koska tahansa parantaa mitä tahansa osaa ohjelmistosta. *Testivetoinen kehittäminen* tarkoittaa testien kirjoittamista ennen varsinaisen ohjelmakoodin tuottamista. Testit eivät kuitenkaan mene läpi ennen kuin ohjelmakoodi on toteutettu. *Jatkuvassa integroinnissa* uusi ohjelmakoodi integroidaan usein koodikantaan ja sille ajetaan aina automaattiset testit. Tällä pyritään minimoimaan ongelmat, jotka seuraavat mittavien muutosten integroinneista. Jos integrointi epäonnistuu, koodi tulee korjata välittömästi. *Refaktorointi* tarkoittaa ohjelmakoodin parantamista uudelleen kirjoittamalla, muuttamatta varsinasta toiminnallisuutta. *Suunnittelupeli* tarkoittaa tekniikkaa, jolla asiakkaat priorisoivat käyttäjätarinat ja päättävät julkaisujen sisällön ja aikataulun kehittäjien työmääräarvioihin perustuen. *Pienet julkaisut* tarkoittavat sitä, että ohjelmiston ensimmäinen versio julkaistaan hyvin pian ja uusia julkaisuja tehdään sen jälkeen useasti, jopa päivittäin. *Metaforat* tarkoittavat kielikuvia, joilla helpotetaan kehittäjien ja asiakkaan välistä kommunikaatiota. *Yksinkertainen suunnittelu* tarkoittaa sitä, että pyritään mahdollisimman yksinkertaiseen, halutut tarpeet täyttävään ohjelmiston rakenteeseen. Ylimääräinen tai päällekkäinen osa toteutuksesta poistetaan välittömästi turhana. *Läsnäoleva asiakas* (on-site customer) tarkoittaa, että asiakkaan tulee olla paikalla ja kehitystiimin käytettävissä kokoaikaisesti. *Koodaussäännöt* -käytänne tarkoittaa, että jokaisen tiimin jäsenen tulee noudattaa yhtenäistä ohjelmointitapaa, jolloin henkilökohtaisista eroista johtuvat poikkeamat tuotetussa lähdekoodissa vähenevät ja jopa katoavat. Yhteisomistajuuden ja pariohjelmoinnin soveltaminen edellyttävät standardoituja työskentelymenetelmiä ohjelmoinnissa. *40-tunnin työviikko* tarkoittaa työajan tarkkailua ja rajoittamista siten, ettei ylitöitä tehdä kahtena peräkkäisenä viikkona. Tällä pyritään takaamaan tiimin jäsenten jaksaminen ja ylläpitämään luovuutta. *Avoin työtila* tarkoittaa, että tiimi on työtä varten järjestäytyneenä samaan tilaan, jossa on tarpeelliset välineet kehittämiselle. Joissain lähteissä tätä käytännettä ei ole laskettu käytänteisiin mukaan.

XP:n ohjelmistoprosessi jakaantuu kuuteen vaiheeseen (Beck, 1999), jotka ovat tutkimus, suunnittelu, iteraatio, tuotteistus, ylläpito ja viimeistely (kuvio 1).

Tutkimusvaiheessa asiakkaat kuvaavat tulevan ohjelmiston vaatimuksia käyttäjätarinoiksi, jotka kirjoitetaan erilliseille korteille (story cards). Samaan aikaan ohjelmoijat tutkivat ja kehittävät arkkitehtuuri- ja teknologiaratkaisuja. Projektin työntekijät puolestaan tutustuvat käytettävään teknologiaan, työkaluihin sekä uusiin käytäntöihin. Kehitettävästä järjestelmästä tehdään myös mahdollisesti prototyyppi käytettävän teknologian ja arkkitehtuurivaihtoehtojen tutkimiseen. Tutkimusvaiheen kesto vaihtelee, projektista riippuen, muutamasta viikosta muutamaan kuukauteen. (Beck, 1999b; Abrahamsson ym., 2002.)



KUVIO 1 XP-menetelmän prosessi (Abrahamsson ym., 2002, 19).

Suunnitteluvaiheessa käyttäjätarinat priorisoidaan ja niiden vaatima työmäärä arvioidaan. Suunnitteluvaihe kestää vain muutaman päivän ja sen tuloksena saadaan ensimmäisen julkaisun sisältö sekä aikataulu. Ensimmäisen julkaisun tuottamiseen kuluva aika on normaalisti alle kaksi kuukautta. (Beck, 1999b; Abrahamsson ym., 2002.)

Iteraatiovaiheessa suunnitteluvaiheessa valittu sisältö ja aikataulu hajotetaan useisiin, yhdestä neljään viikkoa kestäviin iteraatioihin. Ensimmäiseen iteraation valitaan tarinat, jotka muodostavat järjestelmän arkkitehtuurin. Myöhemmissä iteraatioissa toiminnallisuus on tärkeämmässä roolissa. Iteraatioiden lopussa tehdään toiminnalliset testit ja järjestelmä on toimintavalmis. Asiakas valitsee mitkä tarinat kussakin iteraatiossa toteutetaan. (Beck, 1999b; Abrahamsson ym., 2002.)

Tuotteistusvaihe on julkaisun jälkeistä palautteen antamista, suorituskyvyn testausta ja laadun varmistamista. Tuotteistamisen aikana otetaan myös jatkokehittämisideoita talteen. Jos tässä vaiheessa tulee uusia vaatimuksia tai muutoksia, niiden osalta päätetään, lisätäänkö ne nykyiseen julkaisuun vai siirretäänkö tulevaan. (Beck, 1999b; Abrahamsson ym., 2002.)

Ylläpitovaiheen alkaessa järjestelmä on tuotteistettu. Tuotteistettua järjestelmää ylläpidetään ja siihen luodaan mahdollisesti myös uutta toiminnallisuutta. Ylläpitoon kuuluu mm. asiakastuki, tekninen tuki ja virheiden korjaaminen. Tässä vaiheessa kehittämisen tahti usein hidastuu ja työstä tulee rutiininomaisempaa. Ylläpitovaihe on usein XP:n prosessin yleisin vaihe. (Beck, 1999b; Abrahamsson ym., 2002.)

Viimeistelyvaihe alkaa, kun järjestelmä täyttää sille asetetut vaatimukset, eikä asiakkaalla ole enää uusia tarinoita toteutettavaksi. Viimeistelyvaiheeseen voidaan päätyä myös silloin, jos järjestelmä ei täytä sille asetettuja vaatimuksia tai sitä ei syystä tai toisesta kannata enää kehittää. Viimeistelyvaiheessa kirjoitetaan tarvittava dokumentaatio. (Beck, 1999b; Abrahamsson ym., 2002.)

XP:ssä on joitakin ominaispiirteitä, joita ei kaikissa ketterissä menetelmissä käytetä. XP hyödyntää pariohjelmoinnin tuomia etuja. Pariohjelmoinnissa virheiden minimoimiseksi kaikki kirjoitettu koodi on syntynyt kahden ohjelmoijan yhteistyönä (Paulk, 2001). Näin millään kirjoitetulla koodin osalla ei ole yksittäistä omistajaa, vaan kaikki saavat parannella ja muokata kirjoitettua koodia (Paulk, 2001). Projekteissa käytettävien tiimien koko on määritelty maksimissaan 10 ihmisen kokoiseksi. Rajoituksen tarkoituksena on parantaa tiimin sisäistä kommunikaatiota (Beck, 1999). XP kiinnittää huomiota myös työntekijöiden jaksamisen rajoittamalla työviikon 40 tuntiin (Vanderburg, 2005).

2.3 Scrum

Scrum-menetelmä julkaistiin ensimmäisen kerran vuonna 1995 OOPSLA-konferenssin (Conference of Object-Oriented Programming, Systems, Languages and Applications) kokoomateoksessa (Schwaber, 1995). Sen jälkeen siitä on julkaistu useampia kirjoja (Schwaber & Beedle, 2004; Schwaber, 2004). Ajantasaisin kuvaus Scrum-menetelmästä on kuvattu Scrum Guide -nimisessä esityksessä (Schwaber & Sutherland, 2013). Seuraavaksi selitetään Scrum–menetelmää pääosin Scrum Guiden mukaisesti.

”Scrum on viitekehys, jossa ihmiset voivat ratkaista monimutkaisia ongelmia kehittäessään tuotteita tuottavasti ja luovasti mahdollisimman korkealla lisäarvolla.” (Schwaber & Sutherland, 2013,3). Scrum on menetelmänä kevyt ja helppotajuinen, mutta sitä on vaikea hallita hyvin. Scrum-menetelmää on käytetty tuotekehityksessä 1990-luvun alusta lähtien. Schwaberin ja Sutherlandin (2013) mukaan Scrumia voi kuvata parhaiten viitekehyksenä, jonka sisällä voi käyttää useita erilaisia prosesseja ja tekniikoita. Scrum itsessään ei ole tuotekehitysprosessi tai –tekniikka. Scrumin avulla voidaan tuotehallinnon ja -kehityksen menetelmien vaikutukset tehdä näkyviksi ja sitä kautta mahdollistetaan menetelmien parantaminen. (Schwaber & Sutherland, 2013.)

Scrumin keskeiset elementit ovat Scrum-tiimit rooleineen, tapahtumat, tuotokset ja säännöt. Jokaisella elementillä on oma tarkoituksensa ja jokainen elementti on tärkeä osa Scrumin onnistumista. Scrumin sääntöjen tehtävänä on sitoa yhteen roolit, tapahtumat ja tuotokset sekä ohjata niiden välistä vuorovaikutusta. (Schwaber & Sutherland, 2013.)

Ajattelutapa Scrumin taustalla on empirismi eli empiirinen prosessinhallintateoria. Sen mukaan tieto perustuu havaintoihin, kokemukseen ja päätösten tekemiseen tunnettujen tosiasioiden pohjalta. Scrumin lähestymistapa on iteratiivis-inkrementaalinen (toistava ja lisäävä). Lähestymistavan avulla Scrum pyrkii optimoimaan ennustettavuutta ja kontrolloimaan riskejä. Empiirisellä prosessinhallinnalla on kolme peruspilaria: *läpinäkyvyys* (prosessin merkittävät tekijät määritellään ja sovitaan yhdessä, jotta tarkastelijoilla on yhteinen näkemys siitä, mitä tarkastellaan), *tarkastelu* (Scrumin käyttäjien tulee säännöllisesti tarkastella Scrumin tuotoksia ja työn edistymistä) ja *sopeuttaminen* (jos huomataan, että syntyvää tuotetta olisi mahdoton hyväksyä, tulee prosessia tai käytet-

täviä materiaaleja säätää). Scrumissa on neljä muodollista tapahtumaa tarkasteluun ja sopeuttamiseen: sprintin suunnittelu, päivittäinen palaveri, sprintin katselmointi ja sprintin jälkitarkastelu. (Schwaber & Sutherland, 2013.)

Scrum-tiimi koostuu tuoteomistajasta, kehitystiimistä ja Scrum-mestarista. Scrum-tiimit ovat itseohjautuvia ja monitaitoisia. Itseohjautuvat tiimit päättävät itse, kuinka parhaiten tekevät työnsä ulkoisen ohjauksen sijaan. Monitaitoisilla tiimeillä on lisäksi kaikki työn tekemiseen vaadittava osaaminen ilman riippuvuuksia tiimin ulkopuolisiin henkilöihin. Scrumin tiimimalli on suunniteltu joustavuuden, luovuuden ja tuottavuuden optimoimiseksi. (Schwaber & Sutherland, 2013.)

Tuoteomistaja on vastuussa tuotteen arvon ja kehitystiimin työn arvon maksimoimisesta. Tuoteomistaja on vastuussa myös tuotteen kehitysjonon hallinnasta. Kehitystiimi koostuu ammattilaisista, jotka muuttavat tuotteen kehitysjonon sisällön potentiaalisesti julkaisukelpoiseksi "valmiiksi" tuoteversioksi jokaisessa sprintissä. Ainoastaan kehitystiimin jäsenet osallistuvat tuoteversion kehitykseen. Scrum-mestari vastaa siitä, että kaikki ymmärtävät ja käyttävät Scrumia. Scrum-mestarit tekevät tämän varmistamalla, että Scrum-tiimit pitäytyvät Scrumin teoriassa, käytännöissä ja säännöissä. Scrum-mestari on Scrum-tiimin palveleva johtaja. (Schwaber & Sutherland, 2013.)

Scrumin ennalta sovitut tapahtumat luovat prosessiin säännöllisyyttä ja ne vähentävät muiden kuin Scrum-palavereiden tarvetta. Jokaisella Scrumin tapahtumilla on maksimipituus. Tapahtumat voidaan päättää ennen niiden maksimipituuden täyttymistä, kunhan aikaa on käytetty riittävästi, eikä prosessissa pääse syntymään hukkaa. Ainoa poikkeus tästä on itse sprintti, joka sisältää muut tapahtumat. (Schwaber & Sutherland, 2013.)

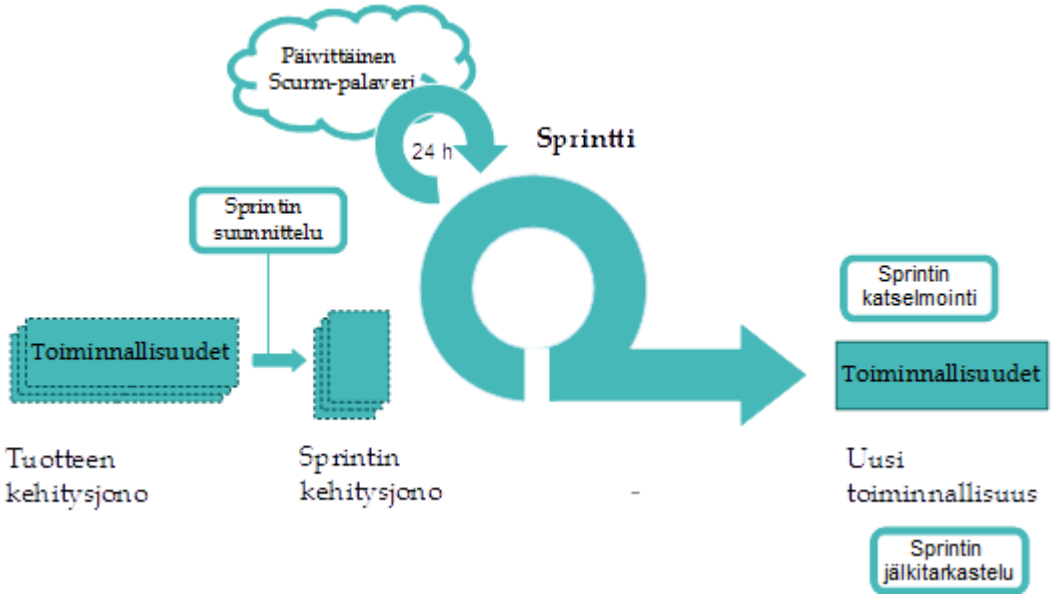
Scrumin ydin on *sprintti.* Sprintti on maksimissaan kuukauden pituinen projekti, jonka aikana tuotetaan käyttökelpoinen ja mahdollisesti julkaisukelpoinen tuoteversio. Sprintin pituus pysyy samana koko kehityksen ajan. Sprintin päätyttyä aloitetaan välittömästi uusi sprintti. Sprintit koostuvat sprintin suunnittelupalaverista, päiväpalaverista, kehitystyöstä, sprintin katselmoinnista ja sprintin jälkitarkastelusta, retrospektiivistä. (Schwaber & Sutherland, 2013.)

Sprintin aikana tehtävä työ suunnitellaan *sprintin suunnittelupalaverissa*. Tämä suunnitelma luodaan yhteistyössä koko Scrum-tiimin kesken. Suunnittelupalaverissa päätetään, mitkä tuotteen kehitysjonosta (product backlog) siirretään sprintin kehitysjonoon (sprint backlog). Sprintin suunnittelu rajataan enintään kahdeksaan tuntiin kuukauden mittaiselle sprintille. Lyhemmille sprinteille varataan yleensä vähemmän aikaa. Scrum-mestari varmistaa, että sprintti suunnitellaan ja että osallistujat ymmärtävät tapahtuman tarkoituksen. Scrum-mestari opastaa Scrum-tiimiä pitämään tapahtuman sen aikarajan sisällä. (Schwaber & Sutherland, 2013.)

Päivittäisen palaverin kesto on määrätty viideksitoista minuutiksi riippumatta tiimin jäsenten määrästä. Tässä palaverissa jokainen tiimin jäsen vastaa kolmeen kysymykseen: mitä on tehnyt viimeisen palaverin jälkeen, mitä aikoo tehdä seuraavaan palaveriin mennessä ja onko ilmennyt ongelmia. (Schwaber 2004.)

Sprintin jälkeen järjestetään *sprintin katselmointi*. Tässä kokouksessa tiimi esittelee projektin sidosryhmille ne uudet toiminnallisuudet, jotka ovat valmistuneet sprintin aikana. Puolivalmiita tai keskeneräisiä tuotoksia kokouksessa ei esitellä. Sidosryhmät voivat tässä kokouksessa vapaasti esittää kysymyksiä ja kommentteja tiimille. Sprintin katselmointi kestää enintään 4 tuntia. (Schwaber 2004.)

Sprintin lopuksi järjestetään *sprintin jälkitarkastelu*, retrospektiivi, johon on varattu aikaa 3 tuntia. Tarkasteluun osallistuu tiimi, Scrum-mestari sekä mahdollisesti tuoteomistaja. Tarkastelussa jokainen tiimin jäsen vastaa kysymyksiin mikä meni hyvin sprintin aikana ja mitä voidaan tehdä paremmin seuraavassa sprintissä (Schwaber 2004). Scrum-menetelmän prosessi on kuvattu kuviossa 2.



KUVIO 2 Scrum-menetelmän prosessi (Schwaber 2004).

2.4 Yhteenveto

Tässä luvussa muodostettiin yleiskuva ketterästä ohjelmistokehityksestä. Aluksi kerrottiin yleisesti ketterästä lähestymistavasta, sen arvoista ja periaatteista. Tässä yhteydessä kerrottiin myös ketteryyttä koskevista erilaisista käsityksistä sekä ketterästä ohjelmistokehityksestä koetuista hyödyistä ja ongelmista. Lopuksi esiteltiin tarkemmin kahta ketterää menetelmää, XP:tä ja Scrumia. XP-menetelmästä esiteltiin lyhyesti sen käytänteet sekä kuvattiin prosessi yleisellä tasolla. Scrum-menetelmästä käytiin läpi sen keskeiset periaatteet ja termit, kuten menetelmään kuuluvat roolit ja tapahtumat. Myös Scrum-menetelmän prosessi esiteltiin.

3 KYPSYYSMALLIT

Tässä luvussa kerrotaan kypsyysmalleista. Aluksi kerrotaan yleisesti kypsyysmalleista, niiden taustasta, rakenteesta ja periaatteista. Sen jälkeen kuvataan kaksi kypsyysmallia (CMM, CMMI) hieman tarkemmin. Lopuksi kerrotaan, miten perinteiset kypsyysmallit nähdään ketterän kehittämisen näkökulmasta.

3.1 Kypsyysmalleista yleisesti

Kypsyysajattelu on lähtöisin 1970-luvulta, jolloin Richard Nolan esitti Stages-of-Growth-mallin. Nolanin (1973) mallissa oli ensin neljä kypsyysvaihetta (vaiheet 1-4), mutta myöhemmin siihen lisättiin vielä kaksi vaihetta (vaiheet 5-6) (King & Kraemer, 1984). Malli kuvaa informaatioteknologian käyttöönottoa yrityksissä (Mettler 2010). Nolanin malli laukaisi tutkimusbuumin alalla, ja vaikka mallia kritisoitiinkin, se on laajasti käytetty ja vaikuttanut useiden kypsyysmallien kehittämiseen (Pöppelbuß, Niehaves, Simons & Becker, 2011). Yksi näistä oli Quality Management Maturity Grid (OMMG) (Crosby, 1979). Jokelan, Siposen, Hirasawan ja Earthyn (2006) mukaan kypsyysmallit ovat saaneet alkunsa juuri laadunhallinnasta (Quality Management). Heidän mukaansa Crosbyn QMMG-malli on toiminut nykyisten kypsyysmallien aloittajana (Jokela ym. 2006). QMMG-mallissa on määritelty ruudukkoon organisaation tyypillinen käyttäytyminen viidellä eri kypsyystasolla, laatujohtamisen jokaiselle kuudelle eri vaiheelle (aloitus, laajentamien, ohjaaminen, integrointi, tiedon hallinnointi ja kypsyys) (Crosby, 1979; Crosby, 1986). Mallin mukaan yritykset kehittyvät viiden kypsyystason kautta – epävarmuus, herääminen, valaistuminen, viisaus ja varmuus - matkallaan kohti erinomaista laadun hallintaa (Crosby, 1979; Crosby, 1986). QMMG on yksi ensimmäisistä malleista, jonka tasot ovat portaittaisia. Humphrey ja Radice tiimeineen kehittivät IBM:llä QMMG-mallia edelleen. Vuonna 1986 Humphrey toi tehdyt konseptit SEI:lle (Software Engineering Institute), jossa niitä käytettiin CMM-mallin (Capability Maturity Model) ensim-

mäisen version pohjana (Paulk, 2009). Kypsyysmallit nousivat suureen suosioon, kun ensimmäinen CMM ilmestyi (Pöppelbuß ym., 2011).

Kypsyysmalleilla pyritään arvioimaan jonkin kohteen kypsyyttä (maturity) tai/ja kyvykkyyttä (capability) tietyn kriteeristön pohjalta. Tyypillisesti mallit koostuvat kypsyystasoista ja näkökulmista, joiden kautta kohdetta arvioidaan. (Jokela ym., 2006) Kypsyysmalli tarjoaa määrämuotoisen ja käytännössä koetellun viitekehityksen kohteen arviointiin. Malleilla on useita esitystapoja niiden rakenteesta ja laajuudesta riippuen. *Portaittaisessa* (staged) mallissa kypsyystaso esitetään prosessialueiden kypsyyden perusteella ja *jatkuvassa* (continuous) mallissa tarkastellaan prosessialueiden kyvykkyys- tai kypsyystasoja. Taulukon muodossa esitetty kypsyysmalli edustaa yksinkertaisempaa muotoa (Mettler 2010).

Kypsyysmalleilla on kolme käyttötarkoitusta, kuvaileva, ohjaileva ja vertaileva (Becker, Knackstedt & Pöppelbuß, 2009; De Bruin ym. 2005). Ensinnäkin kypsyysmallia voidaan käyttää *kuvailevaan* tapaan määrittämällä organisaation nykytila ennalta määritellyn kriteeristön pohjalta (Becker ym. 2009). Kypsyysmallia voidaan myös käyttää *ohjailevasti*, eli toiminnan kehittämiseen. Tämä edellyttää kuitenkin sen, että kypsyysmalli sisältää kypsyystasojen lisäksi selkeät toimintaohjeet parannusten tekemiseksi (Becker ym. 2009). Mallit ohjaavat organisaatioita toimimaan järjestelmällisesti ja parantamaan toimintatapaansa arvioinnin tuloksena. Kolmanneksi, kypsyysmalli voi toimia *vertailu*työkaluna (De Bruin ym. 2005).

3.2 CMM

Yksi tunnetuimmista kypsyysmalleista on Software Engineering Instituten (SEI) julkaisema CMM (Capability Maturity Model) –malli (Humphrey, 1989). Se on monen myöhemmin julkaistun kypsyysmallin perusta. CMM poikkeaa Crosbyn (Crosby, 1979; Crosby, 1986) laaturuudukosta siten, että se pitää sisällään sarjan kumulatiivisia avainprosessialueita (Key Process Areas, KPA), joista pitää suoriutua sitä mukaan kun kypsyystaso nousee. Jokainen KPA on jaettu viiteen luokkaan. Nämä luokat sisältävät käytännöt, jotka toteuttavat tärkeimpien prosessialueiden päämäärät (Fraser, Moultrie & Gregory, 2002).

CMM-mallista on muodostunut useita johdannaisia. Komi-Sirviön (2004) mukaan tunnetuin näistä on SW-CMM (Software Capability Maturity Model). SW-CMM on portaittainen malli, joka keskittyy organisaation kyvykkyyden rakentamiseen. Se yksilöi muutaman kohteen, joihin keskittyä, ja kuvaa reitin prosessien parantamiseksi (Paulk, 2001). SW-CMM:n lisäksi on kehitetty malleja erilaisiin tarpeisiin ohjelmistotuotantoon ja muille alueille. Paulk (2001) mainitsee muiden muassa seuraavat: SE-CMM (Systems Engineering Capability Maturity Model), SA-CMM (Software Acquisition Capability Maturity Model), EMM (Engineering Maturity Model). Renken (2004) on lisäksi tutkinut mallia PM-CMM (People Capability Maturity Model).

SEI kehitti vuonna 1996 CMM-mallin rinnalle IDEAL–mallin (SEI, 2006). Komi-Sirviön (2004) mukaan IDEAL on kehitetty tukemaan SW-CMM -pohjaista ohjelmistokehityksen prosessien parantamista. IDEAL ei ole kypsyystasomalli, vaan se tarjoaa etenemissuosituksen toimintatavaksi prosessien parantamiseen ja sitä voidaan käyttää erilaisten kypsyysmallien rinnalla. CMM-malli ja sen SPI (Software Process Improvement) -menetelmä IDEAL (SEI 2006) olivat maailman käytetyimpiä ja tunnetuimpia malleja organisaatioissa, joiden toimiala on ohjelmistotuotanto (Ngwenyama & Nielsen 2003). Renken (2004) mukaan CMM-mallin voima perustuu mitattaviin määriteltyihin prosesseihin.

Seuraavassa esitetään SW-CMM version 1.1 viisiportaisen kypsyystasomallin tasot ja niiden prosessialueet (Paulk, 2001):

1 *Alkutilanne* (Initial). Alkutilanteessa organisaation menestyminen on täysin riippuvainen siellä työskentelevistä ihmisistä ja heidän tietotaidostaan. Prosessi ei ole ennustettavissa, ja se on heikosti valvottu. Prosessialueet: ei varsinaisia prosessialueita.
2 *Toistettava* (Repeatable). Organisaatiossa on yhteiset pelisäännöt, joiden noudattamista vaaditaan ja valvotaan. Prosessit voidaan toistaa. Prosessialueet: vaatimustenhallinta, ohjelmistoprojektin suunnittelu, projektin seuranta ja ohjaus, alihankinnan hallinta, laadunvarmistus ja konfiguraation hallinta
3 *Määritelty* (Defined). Prosessi on määritelty, sitä noudatetaan ja sitä pystytään kehittämään. Prosessialueet: organisaation prosesseihin keskittyminen, prosessien määrittely, koulutuksen suunnittelu, integroitu ohjelmistokehityksen hallinta, ohjelmistotuotteiden kehitys, ryhmien välinen koordinointi ja vertaiskatselmointi.
4 *Johdettu* (Managed). Prosessia mitataan ja mittaustuloksia käytetään prosessin kehittämiseen. Prosessialueet: mitattava prosessienhallinta ja ohjelmistotuotannon laadunhallinta.
5 *Optimoiva* (Optimizing). Tietoa kerätään automaattisesti ja sitä käytetään prosessin optimoimiseksi. Prosessialueet: virheiden ehkäiseminen, teknologian muutostenhallinta ja prosessien muutostenhallinta.

CMM-mallin ongelmina voidaan pitää sitä, ettei se sovellu sellaisenaan kaikille organisaatioille. Malli ei määrittele sitä, minkälaisille organisaatioille se soveltuu (Kollanus, 2003). Lisäksi esimerkiksi riskien hallinnan puutetta on arvosteltu (Kollanus, 2003). SEI ei ole enää kehittänyt CMM-mallia, koska vuonna 2000 julkaistiin CMMI (Capability Maturity Model Integration) (SEI, 2006).

3.3 CMMI

CMM:n seuraaja ja korvaaja CMMI (CMM Integration) julkaistiin 2000, ja sen uusin versio 1.3 vuonna 2010 (SEI, 2006; SEI, 2010). CMMI-malliin yhdistettiin

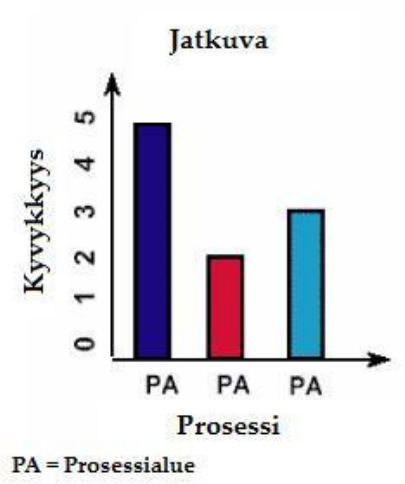
kolme CMM-mallia: SW-CMM, SE-CMM ja IPD-CMM (Kollanus, 2007). CMMI kattaa viisi kypsyystasoa ja 22 prosessialuetta, jotka liittyvät tavalla tai toisella ohjelmistotuotteiden kehitykseen. CMMI perustuu ajatukseen siitä, että loistavatkaan yksilöt tai tuotteet eivät yksin riitä, kun kokonaisuus on riittävän laaja ja monimutkainen. Taustalla toimiva prosessi voi noudattaa määrätietoisesti kehitettyä toimintamallia tai olla huonosti määritelty ja spontaanisti kehittynyt. Lähtökohtana on käytössä olevien osaprosessien kypsyysasteen tunnistaminen viisiportaisella asteikolla. Tämän jälkeen prosessia voidaan ryhtyä korjaamaan ja parantamaan porras kerrallaan (SEI, 2010).

CMMI kattaa ohjelmistotuotannon lisäksi tuotekehityksen. CMMI-mallissa on kaksi esitystapaa, portaittainen ja jatkuva. Portaittainen esitystapa keskittyy organisaation prosesseihin kokonaisuudessaan ja kartoittaa prosessien kehittämistä ennalta määrättyjen prosessialueiden ryhmittelyiden perusteella (viisi kypsyystasoa). Jatkuva esitys keskittyy yksittäisten prosessialueiden kehittämiseen (kyvykkyys-tasot 0-5) (Marçal ym., 2008; Pikkarainen & Mäntyniemi, 2006). CMMI mallin viisi kypsyystasoa ja 22 prosessialuetta ovat seuraavat (Phillips, 2003):

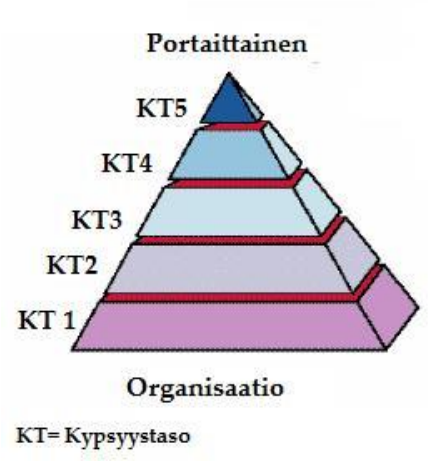
1 *Suoritettu* (Performed). Prosessi on ennustamaton, heikosti valvottu ja reaktiivinen. Prosessialueet: ei nimettyjä prosessialueita, koska CMMI:n mukaan jokainen organisaatio on vähintään tasolla 1.
2 *Hallittu* (Managed). Pääpaino on projektinhallinnassa. Prosessialueet: organisaation tulee täyttää seitsemän prosessialueen vaatimukset - vaatimustenhallinta, projektin suunnittelu, projektin seuranta ja ohjaus, toimittajasopimusten hallinta, mittaaminen ja analysointi, prosessien ja tuotteiden laadunvarmistus, konfiguraation hallinta.
3 *Määritetty* (Defined). Organisaatiolle on määritelty yhteiset käytänteet ja sen toiminta on yksityiskohtaisesti mietitty ja määritelty. Prosesseja voidaan räätälöidä. Prosessialueet: organisaation tulee täyttää tason 2 prosessivaatimukset ja lisäksi tasolle 3 määriteltyjen prosessien vaatimukset (11 prosessialuetta) - vaatimusten kehittäminen, tekniset ratkaisut, tuoteintegraatio, varmistus, todennus, organisatorinen prosessikeskeisyys, organisatorinen prosessien määrittely, organisatorinen koulutus, integroitu projektinhallinta, riskienhallinta, päätöksenteon analyysi ja määrätietoisuus.
4 *Tilastollisesti hallittu* (Quantitatively Managed). Prosessia mitataan ja hallitaan. Toimintaa kyetään ennakoimaan. Prosessialueet: tasolla 4 tulee täyttää edellisten tasojen ja tason 4 prosessien vaatimukset – organisatorinen prosessien esittäminen ja mitattu projektinhallinta.
5 *Optimoiva* (Optimizing). Tavoitteena prosessin jatkuva parantaminen. Kehittämiselle asetetaan mitattavat tavoitteet. Prosessialueet: tasolla 5 tulee täyttää edellisten tasojen vaatimukset ja tason 5 prosessien vaatimukset – innovatiivinen kehittäminen sekä syy-, seuraussuhteet ja ehkäisevät toimenpiteet.

CMMI prosessialueille asetetaan tavoitteita (goals), jotka pitävät sisällään käytäntöjä (practices). Tavoitteita asetetaan sekä prosessialuekohtaisesti (specific goals) että kaikille yhteisesti (generic goals). Käytännöt jaetaan prosessikohtaisiin (specific practice) ja kaikille yhteisiin (generic practice) käytäntöihin. (Phillips 2003.).

CMMI-mallin kahdelle esitystavalle (portaittainen ja jatkuva) on esitetty myös erilaisia kuvaustapoja. Kuvio 3 esittää pylväillä, miten kyvykäs organisaatio on kolmen prosessialueen osalta (jatkuva malli). Kuviossa 4 havainnollistetaan organisaation kypsyystasoja pyramidimallin avulla (portaittainen malli).



KUVIO 3 CMMI:n pylväsmäinen kyvykkyysprofiili (Phillips, 2003)



KUVIO 4 CMMI:n pyramidimainen kypsyystasomalli (Phillips, 2003)

CMMI-mallin apuna voidaan käyttää GQM (Goal-Question-Metric) –menetelmää (Basili, 1992). Menetelmässä määritetään ensin organisaation tavoitteet (goals), joista johdetaan sitten kysymykset (questions), joihin pyritään saamaan vastaukset sopivilla mittareilla (metrics). CMMI-mallia käyttävät organisaatiot voivat suorittaa arvioinnin joko itse tai käyttää SEI:n (Software Engineering Institute) sertifioimaa arvioijaa. CMMI ei pyri sertifioimaan prosessien kypsyyttä

vaan se on ainoastaan niiden kehittämisen apuväline. Arvioinnit luokitellaan A-, B- tai C-arvioinneiksi, ja SCAMPI (Standard CMMI Appraisal Method for Process Improvement) A on laajin arviointi, jonka johtaa päteväksi todettu pääarvioija (Kähkönen & Abrahamsson, 2004). Tällaisesta arvioinnista voidaan antaa julkinen tulos kypsyys- tai kyvykkyystasoina (SEI, 2006).

Myös CMMI-mallissa on joitakin heikkouksia. Patel ja Ramachandran (2009) arvostelevat CMMI-mallia sopimattomuudesta pieniin yrityksiin ja ketterään kehittämiseen. Hämäläisen (2007) mukaan CMMI-malli ei sovellu aivan pienille organisaatioille.

3.4 CMM ja CMMI ketterän kehittämisen näkökulmasta

CMM- ja CMMI-kypsyysmalleja kehitettiin aikana, jolloin ohjelmistokehitystä tehtiin pelkästään perinteisillä menetelmillä. Tällaista ohjelmistokehitystä on totuttu kutsumaan suunnitelmavetoiseksi (plan-driven) kehittämiseksi, koska sille on tyypillistä laaja etukäteissuunnittelu ja runsas dokumentointi. Ketterää ohjelmistokehitystä on pidetty suunnitteluvetoisena (Turner & Jain, 2002). Näiden kahden lähestymistavan yhdistämisen on sanottu olevan yhtä perustavanlaatuinen haaste kuin veden ja öljyn yhdistäminen (Turner & Jain 2002).

Tutkijat ovat kuitenkin eri mieltä siitä, kuinka yhteensopivia ketterät menetelmät ja kurinalaiset (rigorous) menetelmät ovat (Heeager, 2013). Vaikka ketterät menetelmät näyttävät olevan konfliktissa kurinalaisten menetelmien kanssa, useat tutkijat ovat sitä mieltä, että on mahdollista käyttää (joitakin) ketteriä käytänteitä ja periaatteita ja samaan aikaan täyttää laatuvaatimukset (Heeager, 2013). On arveltu, että on mahdollista saavuttaa CMMI tasojen 2 ja 3 prosessialueet käyttäen Scrumia ja XP–käytänteitä (Cohen, Lindvall & Costa, 2004; Paulk, 2002). Lisäksi muutamat ovat sitä mieltä, että useimmat XP projektit, jotka todella noudattavat XP-käytänteitä, voisivat saavuttaa CMMI tason 2 (Glazer, 2001; Kähkönen & Abrahamsson 2004; Paulk, 2001). Muutamissa tutkimuksissa organisaatioissa käytetyt ketterät menetelmät täyttivät CMMI:n tasojen 2 ja 3 tavoitteet (Anderson, 2005; Baker, 2005; Bos & Vriens, 2005; Vriens, 2003; Kähkönen & Abrahamsson, 2004). Boehm ja Turner (2003) puolestaan toteavat, että tason 5 konsepti jatkuvasta kehittymisestä suorituskyvyn parantamiseksi on linjassa ketterien menetelmien kanssa, kuitenkin huomaten sen, että useimmat ketterät menetelmät eivät tue kaikkia alempien tasojen elementtejä (Pikkarainen & Mäntyniemi, 2006). Suurin osa Heeagerin (2013) käsittelemistä tutkimuksista oli sitä mieltä, että yrityksen, joka käyttää laajennettua ketterää menetelmää, on mahdollista mukautua kypsyysmalleihin kuten CMMI tasot 2 ja 3. Anderson (2005) jopa esittää, että olisi mahdollista saavuttaa CMMI-taso 5 käyttämällä ketteriä menetelmiä. On myös ehdotettu, että ketterät menetelmät olisivat tavallaan vertikaalinen siivu CMMI tasoista 2-5 (Cohen, Lindval & Costa, 2004).

Usein on oletettu, että CMMI:n kanssa yhteensopivien prosessien täytyy olla raskaita, byrokraattisia ja hidas-liikkeisiä (Anderson, 2005). Ketterien mene-

telmien, kuten XP ja Scrum, on sanottu tarjoavan vähemmän byrokraattisen tavan laadukkaaseen ihmiskeskeiseen ohjelmistokehittämiseen (Bos & Vriens, 2004). Yleinen uskomus on kuitenkin ollut, että CMMI:tä seuratakseen tiimin täytyy dokumentoida vaatimukset, palaverit, suunnitelmat, riskit ja kehittämisen saavutukset, voidakseen kehittää korkealaatuisia ohjelmistoja. Toisaalta ketterissä menetelmissä tiimit voivat tuottaa korkealaatuisia ohjelmistoja käyttämällä epämuodollista ja kevyttä dokumentointia (Boehm & Turner, 2003).

Beck ja Boehm (2003) ovat sitä mieltä, että ketteryys ja kurinalaisuus eivät ole vastakohtia. Boehm (2002) myös esittää, että vaikka molempien suuntien edustajat pitäisivät niitä vastakohtina, niiden osien yhdistäminen projekteissa voi olla hyödyllistä. Glass (2001) huomauttaa, että ”yksi-koko-sopii-kaikille” lähestymistapa ei toimi. Mahnic ja Zabkar (2007, 2008) toteavat, että on mahdollista luoda ohjelmistoprosessi, joka yhdistää ketteryyden ja kurinalaisuuden. Glazer (2010) toteaa, että ketterät menetelmät ja CMMI täydentävät toisiaan ja voivat tuoda nopeita, edullisia, näkyviä ja pitkäaikaisia etuja.

Jotkut ovat esittäneet, että CMMI-kypsyysmallia ja ketteriä menetelmiä voisi käyttää yhdessä niin, että molemmista otettaisiin parhaat ominaisuudet (Boehm & Turner 2003; Paulk, 2001; Kähkönen & Abrahamsson, 2004). Asiasta on tehty kuitenkin vain muutamia tutkimuksia (Pikkarainen, 2008). Huo, Verner, Zhu ja Babar (2004) huomasivat, että ketterät menetelmät sisältävät laadunvarmistus-käytäntöjä, ja niitä on jopa enemmän kuin vesiputousmallissa.

Yksi tapa käsitellä CMMI:n ja ketterien menetelmien yhteensovittamisongelmaa, on vertailla CMMI:n ja ketterän kehittämisen periaatteita (Pikkarainen, 2008). Marcal ym. (2008) ovat tehneet selvityksen siitä, miltä osin Scrum-menetelmä vastaa CMMI:n määrityksiä. Heidän mukaansa noin kolmannes CMMI:n projektinhallinnan prosessialueiden käytännöistä tulee täytettyä Scrum-projektinhallintamallissa (Marcal ym., 2008). 16,4 % käytännöistä täyttyy osittain ja noin puolet eivät täyty ollenkaan. CMMI:n ja ketterän projektinhallinnan välillä on eroja erityisesti riskienhallinnassa, organisaationlaajuisten prosessien hallinnassa sekä systemaattisessa historiatietojen käytössä. Osa näistä eroista liittyy dokumentaation puutteeseen, mikä puolestaan johtuu Agile-manifestin (Beck ym., 2001) arvosta “Arvostamme toimivaa sovellusta enemmän kuin kattavaa dokumentaatiota”. Työn ja kustannusten arviointiin käytetään Scrumissa tuotteen kehitysjonoa sekä sprintin kehitysjonoa. Näiden arviota ei ole kuitenkaan johdettu työn koosta tai kompleksisuudesta, mitä CMMI vaatii, eikä kustannusten arvioinnista mainita Scrumin yhteydessä mitään. Budjetti ja aikataulu johdetaan Scrumissa suoraan tuotteen kehitysjonosta määritellyistä työmääräarvioista, mutta tarkemmat ohjeet niiden laatimiseksi puuttuvat. Scrumissa riskejä ei tunnisteta CMMI:n vaatimalla systemaattisella ja parametrisoidulla tavalla. Ainoa projektin suunnittelun toiminto, jota Scrum ei toteuta millään tavalla, on työtulosten tai tehtävien ominaisuuksien, kuten koon tai kompleksisuuden, määrittäminen. (Marcal ym., 2008.).

Beckin ja Boehmin (2003) mukaan XP on kurinalainen menetelmä, sillä se tarjoaa selkeän mallin siitä, mitä käytänteitä tulisi käyttää. DeMarco ja Boehm

(2002) tukevat tätä toteamusta lisäten, että XP-menetelmä sisältää enemmän suunnittelua kuin mitä CMM-malli edellyttää.

Yhteenvetona voidaan todeta, että vaikka ketterän ohjelmistokehityksen ja perinteisten kypsyysmallien periaatteissa on lähtökohtaisia eroja, on ketteriä menetelmiä soveltamalla mahdollista saavuttaa alimpia CMMI-mallin tasoja. Toisaalta CMMI-mallin soveltamista pidetään sen verran raskaana ja paljon resursseja vaativana, ettei sitä pidetä kovin soveliaana ketterän ohjelmistokehityksen yhteydessä käytettäväksi.

3.5 Yhteenveto

Tässä luvussa kerrottiin kypsyysmalleista. Aluksi kerrottiin yleisesti kypsyysmalleista, niiden taustasta, rakenteesta ja periaatteista. Sen jälkeen kuvattiin kahta kypsyysmallia (CMM, CMMI) hieman tarkemmin. Lopuksi kerrottiin, miten perinteiset kypsyysmallit nähdään ketterän kehittämisen näkökulmasta. Vaikka CMMI ja ketterät menetelmät on perinteisesti nähty lähes vastakkaisina menetelminä, ovat monet tutkijat nykyään sitä mieltä, että niitä voisi käyttää yhtä aikaa. Ongelmana on kuitenkin se, että CMM:n ja CMMI:n tapaisten mallien käyttö on raskasta, joten on toivottu vaihtoehtoista tapaa arvioida ketterän ohjelmistokehityksen kypsyyttä.

4 KETTERÄN OHJELMISTOKEHITYKSEN KYPSYYSMALLEJA

Tässä luvussa esitellään kirjallisuudesta löytyneitä ketterän ohjelmistokehityksen kypsyysmalleja. Aluksi kerrotaan, miten malleja on etsitty ja valittu tähän esitykseen. Sen jälkeen kuvataan kutakin mallia erikseen tekijöiden mukaisessa aakkosjärjestyksessä. Tavoitteena on kuvata malleja siten, että niitä voidaan arvioida ja verrata kuvausten perusteella. Kustakin mallista esitetään, mitä tarkoitusta varten malli on kehitetty, minkälaisista tasoista se koostuu sekä onko mallia käytetty ja/tai validoitu.

4.1 Mallien etsintä ja valinta

Kuten edellisestä luvusta kävi ilmi, on ohjelmistokehitykseen kehitetty kypsyysmalleja jo varsin varhaisessa vaiheessa. Nämä mallit perustuvat oletukseen siitä, että prosessit määritellään tarkasti ja niitä noudatetaan tunnollisesti joka tilanteessa. Nämä oletukset ovat varsin kaukana ketterän ohjelmistokehityksen arvioista ja periaatteista. Tästä syystä ei olekaan ihme, että ketterää ohjelmistokehitystä varten on pyritty kehittämään kypsyysmalleja, jotka sopivat paremmin ympäristöön, jossa edellytetään nopeaa reagointikykyä vaatimusten ja ympäristön muutoksiin.

Ketterän ohjelmistokehityksen kypsyysmalleja koskevia tutkimuksia on etsitty tutkimuskannoista (esim. IEEExplore, ACM Digital Library) ja käyttämällä Google Scholaria. Joitakin malleja on julkaistu vain netissä kaupallisten toimijoiden ja konsulttien toimesta. Löydetyt tutkimukset on esitetty taulukossa 1.

Taulukon neljäntoista mallin tarkastelu tässä työssä olisi ollut liian työlästä. Tästä syystä mallijoukkoa jouduttiin rajaamaan. Gujralin ja Rayarajin (2008) malli rajoittuu käsittelemään vain sovelluskehitystiimejä. Humble ja Russel (2009) puolestaan keskittyivät mallissaan ohjelmistojen kasaamiseen ja julkaisemiseen. Nämä mallit eivät kuvaa tarpeeksi laajasti ja kattavasti ketterää ohjel-

TAULUKKO 1 Ketterän ohjelmistokehityksen kypsyysmalleja

Lähde	Mallin nimi	Käyttötarkoitus
Ambler (2010)	The Agile Maturity Model (AMM)	Ketterän kehittämisen kypsyysmalli ohjelmistokehityksen tehokkuuden parantamiseen
Benefield (2010)	Seven Dimensions of Agile Maturity	Ketterän kehittämisen käyttöönoton nopeuttaminen
Gujral & Rayaraj (2008)	The Agile Maturity Model (AMM)	Sovelluskehitystiimien reagointikyvyn mittaaminen liiketoiminnan muutosten näkökulmasta
Humble & Russell (2009)	The Agile Maturity Model (AMM)	Ketterän kehittämisen kypsyysmalli ohjelmistojen kasaamiseen ja julkaisemiseen
Lui & Chan (2005)	A Road Map for Implementing eXtreme Programming	Vaiheistus XP-menetelmän käyttöönottoon
Malik (2007)	Simple Lifecycle Agility Maturity Model (SLAMM)	Ohjelmistokehitysprosessin ketteryyden mittaaminen
Nawrocki ym. (2001)	eXtreme Programming Maturity Model (XPMM)	Kypsyysmalli XP:n käyttöönotolle
Packlick (2007)	The Agile Maturity Map (AMM)	Ketterien menetelmien käytön kehittäminen organisaatiossa
Patel & Ramachandran (2009)	Agile Maturity Model (AMM)	Ketterän ohjelmistokehittämisen parantaminen ja tehostaminen
Pettit (2006)	”Agile Maturity Model”	Ketterän kehittämisen arviointi ja kehittäminen organisaatiossa
Proulx (2010)	Agile Maturity Model (AMM)	Scrumia käyttävän organisaation kypsyyden arviointi
Qumer & Henderson-Sellers (2008)	The Agile Adoption and Improvement Model (AIIM)	Organisaation ketteryyden tason ja ketterien menetelmien noudattamisen arviointi
Sidky ym. (2007)	The Agile Adoption Framework (AAF)	Ketterien käytäntöjen käyttöönoton systematisoiminen
Yin ym. (2011)	Scrum Maturity Model	Ohjelmistokehitysorganisaation auttaminen ja ohjaaminen Scrum-menetelmän käytössä

mistokehitystä, kun ajatellaan koko organisaatiota ja sen toimintaa. Luin ja Chanin (2005) malli on tarkoitettu kehittymättömille tiimeille, ja vaikka malli ei täytä yllämainittua laajuuden ja kattavuuden vaatimusta, se valittiin kuitenkin, koska se on tarkoitettu käytettäväksi XP-menetelmän yhteydessä. Näin vertailuun saatiin kaksi mallia (Lui & Chan, 2005; Nawrocki ym., 2001), jotka on tarkoitettu käytettäväksi XP-menetelmän yhteyteen. Malik (2007) on kehittänyt Excel-pohjaisen työkalun tuotekehitysprosessin kypsyyden arviointiin. Hän ei kuitenkaan selitä työkalun pohjalla olevaa mallia, joten sitä voidaan pitää

enemmän työkaluna kuin kypsyysmallina, ja näin ollen se ei tullut valituksi. Muut taulukossa mainitut mallit valittiin kuvattaviksi ja vertailtaviksi

4.2 Amblerin malli

Ambler (2010) on esittänyt viisitasoisen ketterän kehittämisen kypsyysmallin, jolla pyritään parantamaan ketterän ohjelmistokehityksen tehokkuutta. Kypsyysmallin tasot ovat retorinen (rhetorical), sertifioitu (certified), uskottava (plausible), kunnioitettava (respectable) ja mitattu (measured).

Ensimmäisen tason (retorinen) ketterien menetelmien käyttäjät uskovat, että ketterä kehittäminen on tehokkaampaa kuin perinteinen kehittäminen. He uskovat vakaasti itseensä ja asiaansa, eivätkä kunnioita johtoa, tuoteomistajia tai edes ketterää yhteisöä. Retorisella tasolla päätökset tehdään itseohjautuvissa tiimeissä. Usein on kyse pilottiprojektimaisesta ketterän menetelmän kokeilusta. Koska tällaisiin projekteihin valitaan usein taitavia osallistujia ja niillä on riittävästi johdon tukea, projektit myös usein onnistuvat. Tämä lasketaan usein pelkästään ketterien menetelmien ansioksi. On kuitenkin vaarana, että tiimi ajautuu ns. ”Scrum-but” tilanteeseen, jolloin Scrum-käytänteistä käytetään vain osaa, eikä kokonaisuuden tuomia hyötyjä saavutetakaan. Yleensä ottaen ketterien menetelmien käytössä ollaan hyvällä alulla, mutta pitkä matka on vielä edessä.

Toisella tasolla (sertifioitu) suurin osa tiimin jäsenistä on suorittanut jonkin sertifioidun ketterän kehittämisen kurssin (esimerkiksi sertifioitu Scrum Master). Tällä tasolla sertifiointia on tärkeä tuoda esille myös ulospäin, niin omassa organisaatiossa kuin asiakkaiden suuntaankin. Tasolla pysyäkseen täytyy sertifiointeja pitää voimassa. Yhä useampien suoritettua sertifiointikursseja, ketterien menetelmien käyttö organisaatiossa laajenee. Sertifioinnista huolimatta ketteristä menetelmistä ja niiden soveltamisesta ei ole vielä kovin syvällistä tietoa.

Kolmannella tasolla (uskottava) painopistettä aletaan siirtää ketterän kehittämisen strategioihin. Onnistuakseen tämä vaatii, että organisaatiossa on jo tarpeeksi laajalti ketterää tietämystä. Pienet ketterät tiimit toimivat jo hyvin, ja niissä aletaan huomata, mikä todella toimii kyseisessä organisaatiossa. Ongelmia voivat aiheuttaa suuret tai hajautetut tiimit, joita ei vielä osata johtaa ja hallita ketterästi. Sertifiointi ei ole enää niin tärkeää, vaan nyt keskitytään taitojen kehittämiseen ja niiden strategioiden ymmärtämiseen, joita vaaditaan laadukkaiden lopputuotteiden toimittamiseen.

Neljännellä tasolla (kunnioitettava) ketterien menetelmien käyttö laajenee kattamaan koko toimitusketjun, aikaisemman pelkän ohjelmistokehityksen elinkaaren sijaan. Organisaatiossa siirrytään tuottamaan laadukkaita palveluita/ratkaisuja sovellusten sijasta. Siellä ymmärretään paremmin kokonaisuutta, jossa toimitaan, ja liiketoimintaprosessien ja organisaatio-rakenteiden kehittäminen hyödyttää kaikkia. Organisaatiossa arvostetaan ja käytetään useita ketteriä menetelmiä tilanteen mukaan. Myös perinteisiä menetelmiä osataan taas

arvostaa ja ymmärretään, että niillä on omat etunsa. Ketterät tiimit ovat tällä tasolla kurinalaisia ja itseohjautuvia ja niillä on riittävä tuki- ja ohjausverkosto. Tiimit ymmärtävät myös organisaationsa asettamat rajoitukset omaan toimintaansa.

Viidennellä tasolla (mitattu) prosessista kerätään tietoja erilaisilla mittavälineillä ja integroiduilla työkaluilla, joiden avulla saadaan tarkkaa reaaliaikaista tietoa. Mitatun tiedon perusteella prosessia voidaan ohjata ja parantaa. Tässä vaiheessa ymmärretään, että organisaation johto yrittää tehdä parhaita mahdollisia päätöksiä käytettävissä olevan, usein puutteellisen, tiedon perusteella. Kun prosessista on käytettävissä luotettavampaa tietoa, näihin tietoihin perustuvat päätökset ovat luotettavammalla pohjalla. Organisaatiossa käytetään useita erilaisia menetelmiä, niin perinteisiä kuin ketteriäkin, aina sen mukaan, mikä kulloisessakin tilanteessa on tehokkainta. Myös lähestymistapaa muokataan ja skaalataan tarpeiden mukaan.

Mallista ei ole käytännön kokemuksia eikä validointitietoja.

4.3 Benefieldin malli

Benefield (2010) on tehnyt tapaustutkimuksen British Telecomin (BT) IT-osaston kehittämästä mallista. IT-ala ja varsinkin televiestintä on viime vuosina ollut suurten muutosten kourissa. Alalle tulvii uusia teknologioita ja sovelluksia on kehitettävä entistä nopeammin ja asiakkaita enemmän huomioiden. Näihin haasteisiin vastatakseen myös BT siirtyi käyttämään ketteriä menetelmiä. Menetelmät osin toimivat ja hyötyjäkin saavutettiin, mutta seurauksena oli usein ongelmia jollain toisella alueella, jolloin hyödyt hävisivät tai pienenivät. Esimerkiksi kehittämisen elinkaaren onnistunut lyhentäminen aiheutti ongelmia integroinnissa, joka taas puolestaan hidasti julkaisutahtia. Syiden selvittämisen yhteydessä alettiin kehittää mallia, joka nopeuttaisi ketterien menetelmien käyttöönottoa, ja samalla paljastaisi suurimmat riskialueet, jotta apu ja tuki voitaisiin kohdistaa nopeasti näille alueille.

Malliin valittiin seitsemän ulottuvuutta, joilla kypsyyttä arvioidaan, sekä niille viisi kypsyystasoa. Nämä seitsemän ulottuvuutta tai käytäntöä ovat automatisoitu regressiotestaus, koodin laatumittarit, automatisoitu kehittäminen, automatisoidut koonnit ja versionhallinta, keskitetty toimitus ja integrointitestaus, testilähtöinen kehittäminen sekä suorituskyky- ja skaalautuvuustestaus.

Kypsyystasot on nimetty seuraavasti:

1. Uudet parhaat käytännöt (Emergent engineering best practices),
2. Jatkuvat käytänteet komponenttitasolla (Continuous practices at component level),
3. Komponenttien välinen jatkuva integraatio (Cross component continuous integration),

4. Projektitasoinen jatkuva integraatio (Cross journey continuous integration),
5. Kysyntään juuri ajallaan vastaavat toimitukset (On demand just in time releases).

Tavoitteena *ensimmäisellä kypsyystasolla* on saavuttaa tiimien kesken yhtenäiset parhaat käytännöt ja luoda selkeä pohja, josta mallia lähdetään rakentamaan eteenpäin. Tasolle kuuluvia käytänteitä ovat yksikkötestit, koodikatselmoinnit, toistettavat koonnit, versionhallinta, koodin laatu, sekä testilähtöinen kehittäminen. Tason saavuttamiseksi tiimien tulee ymmärtää näiden käytänteiden merkitys sekä käyttää/toteuttaa niitä vähintään alustavalla tasolla.

Toisen tason tavoitteena on luoda toistettavat käytännöt komponenttitiimeille. Uudelleenkäytettävät automatisoidut koonnit, jakelut ja testitapaukset kuuluvat tämän tason saavuttamiseen.

Kolmannella tasolla näkökulma laajenee tiimeistä komponenttien väliseen integraatioon ja yhteistoimintaan. Tällä tasolla koontien ja regressiotestauksen tulee olla jo vahvasti automatisoitu. Tiimien tulee synkronisoida työnsä muiden tiimien kanssa ja suorittaa säännölliset koonti-/testikierrokset yli komponenttirajojen.

Neljännellä tasolla näkökulma laajenee edelleen ja koskee nyt koko tuotantoketjua. Useiden tuotteiden ja palveluiden tuottajien pitää pystyä toimimaan yhdessä ja tuottamaan asiakkaalle hänen haluamansa tuote. Tällä tasolla automatisointi on viety huippuunsa ja refaktoroinnista tulee rutiinia. Tuottavuus, suorituskyky sekä skaalautuvuus laajenee komponenttitasolta tuote- ja tuotantoketjutasolle.

Viidennellä tasolla tiimit ovat erittäin tuottavia ja pystyvät nopeasti ymmärtämään ja kommunikoimaan eri komponenttien ja tuotteiden väliset riippuvuudet ja sidonnaisuudet. Koodin refaktorointi on rutiinia ja se myös näkyy koodin laadussa. Uudet tuotteet pystytään nopeasti kokoamaan ja kokeilemaan. Parhaimmillaan tällä tasolla lopputuotteen testauksen, kehittämisen, operatiivisen toiminnan ja liiketoiminnan rajat hämärtyvät ja ideat voivat kehittyä vapaasti halki yrityksen ja niitä voidaan nopeasti kokeilla käytännössä.

Organisaation arviointia varten on käytettävissä taulukko, jonka riveinä ovat ulottuvuudet ja sarakkeina tasot (kuvio 5). Jokaisen ulottuvuuden taso arvioidaan omalla rivillään. Taulukkoon voidaan merkitä myös tavoitetaso. Taulukosta nähdään nopeasti vahvuudet ja heikoimmat alueet, joille seuraavaksi tulisi kohdistaa kehittämistoimia.

Benefield (2010) luonnehtii mallin käytön tuloksia organisaatiossa erittäin positiiviseksi. Mallin avulla on alustavien tulosten mukaan esimerkiksi virheiden löytyminen vasta järjestelmätestauksen yhteydessä vähentynyt yli 60 % ja näistä virheistä tuskin yksikään on ollut erittäin vakava. Myös kun koonti-jakelu-/testaussyklit saatiin paremmin automatisoitua ja toistettavimmaksi, säästyi tuhansia henkilötunteja ja samalla paljastui uusia ongelma-alueita. Näiden uusien ongelma-alueiden löytäminen aikaisessa vaiheessa mahdollistaa myös niiden ratkaisemisen aikaisemmin. Mallin käytön on sanottu laajenevan organisaatiossa edelleen.

	Ulottuvuus	Nykyinen taso	1	2	3	4	5	Tavoite-taso
1	Automatisoitu regressiotestaus							
2	Koodin laatumittarit							
3	Automatisoitu kehittäminen							
4	Automatisoidut koonnit ja versionhallinta							
5	Keskitetty toimitus ja integrointitestaus							
6	Testilähtöinen kehittäminen							
7	Suorituskyky ja skaalautuvuus testaus							

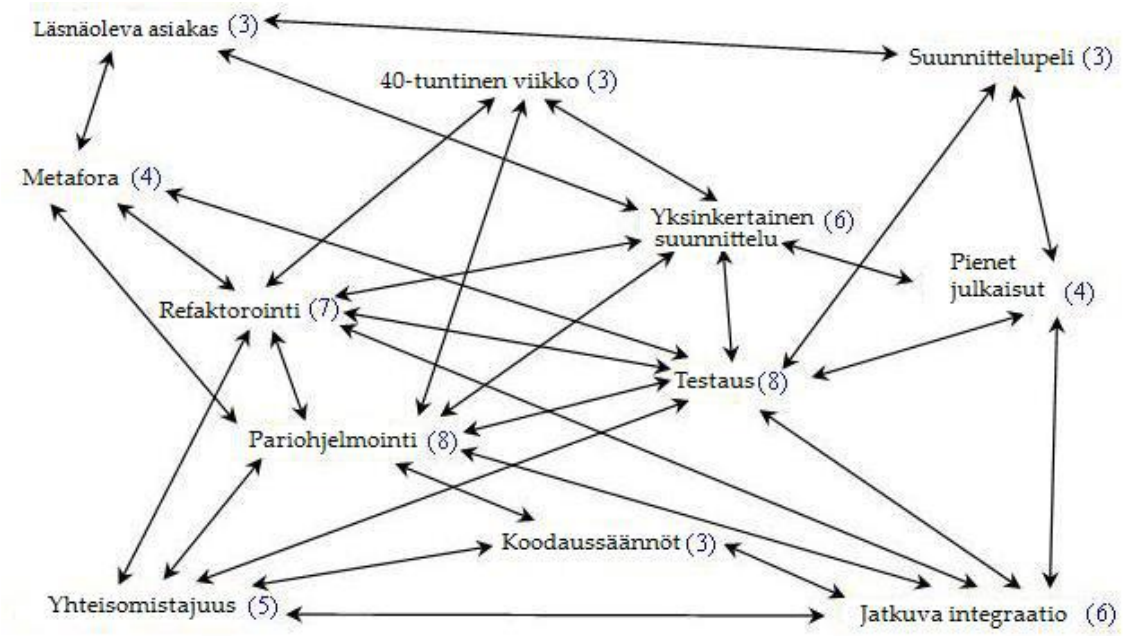
KUVIO 5 Benefieldin (2010, 5) malli

4.4 Luin ja Chanin malli

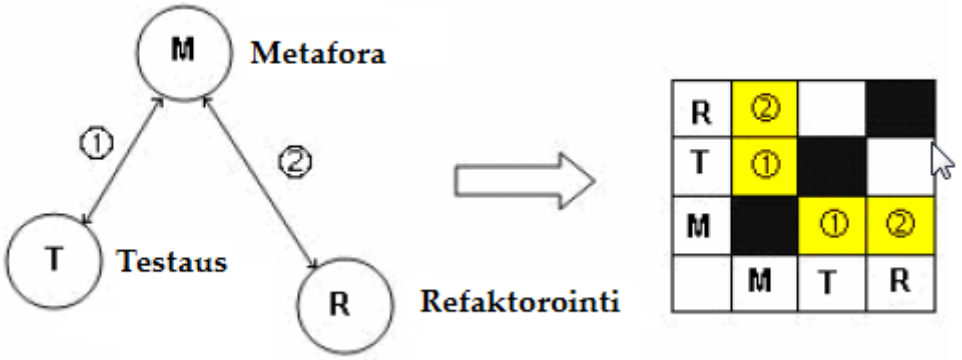
Luin ja Chanin (2005) malli on tarkoitettu tukemaan XP-menetelmän käyttöönottoa sellaisissa tiimeissä, jotka eivät vielä ole niin kehittyneitä, että voisivat käyttää muita ketterän kehittämisen kypsyysmalleja. Luin ja Chanin (2005) mukaan kehittymättömät tiimit tarvitsevat yksinkertaiset ja vaiheittaiset ohjeet. Beck (1999) havaitsi, että XP-käytänteiden välillä on vaikutussuhteita. Jotkut käytänteet vaikuttavat toisiinsa vahvistavasti ja osa XP-käytänteistä on keskeisempiä kuin toiset. Näitä XP-käytänteiden suhteita havainnollistetaan kuviossa 6. Nuoli kahden käytänteen välillä tarkoittaa, että nämä käytänteet vahvistavat toisiaan. Suluissa käytänteen perässä on esitetty kunkin käytänteen suhteiden määrä.

XP:n vahvuus tulee juuri käytänteiden yhteisvaikutuksesta. Lui ja Chan (2005) halusivat löytää uuden esitystavan sille, kuinka nämä 12 käytännettä ovat vuorovaikutuksessa keskenään. Heidän mielestään olisi ollut väärin priorisoida käytänteitä ainoastaan suhteiden määrän perusteella, sillä kaikki käytänteet ovat tärkeitä ja kuuluvat XP:n lähestymistapaan. Luin ja Chanin (2005) mielestä on kuitenkin selvää, että testaus, jolla on kahdeksan suhdetta, on hyvä lähtökohta, koska se tarjoaa suurimman määrän vaihtoehtoja seuraavaan askeleeseen.

Lui ja Chan (2005) muokkasivat XP-käytänteiden suhdegraafia visuaalisen tiedon louhinnan (visual data mining, VDM) avulla toisenlaiseen muotoon (kuvio 7) ja loivat sen perusteella oman mallinsa. Seuraavaksi kuvataan hieman heidän käyttämäänsä tekniikkaa.



KUVIO 6 XP käytänteiden suhteet (Beck, 1999)



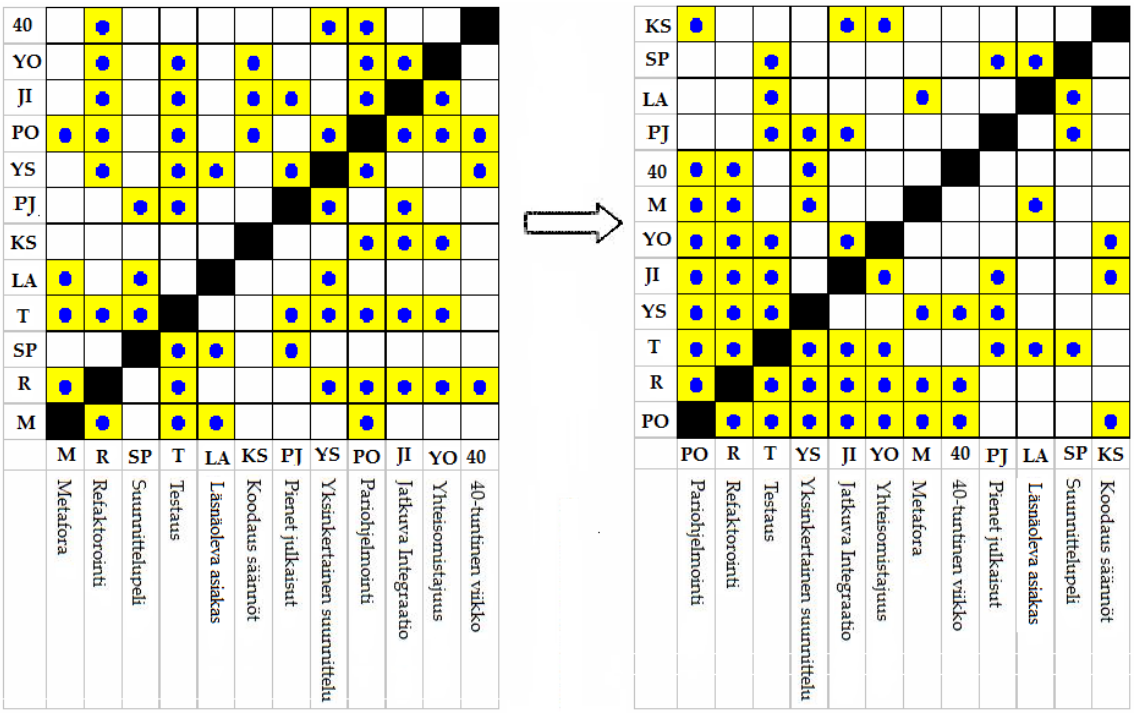
KUVIO 7 Graafinen esitys vs. Matriisi (Lui & Chan, 2005, 478)

Kuvio jossa on n solmua, voidaan esittää nxn–kokoisena matriisina, missä arvo (i, j) on 1 jos solmun i ja solmun j välillä on suhde, muuten arvo on 0. Esimerkiksi kuviossa 7 Testauksen ja Metaforan välillä on suhde. Matriisissa Testauksen ja Metaforan risteyskohdissa niiden suhde on merkitty keltaiseksi.

Näin suhdekuvio voidaan esittää uudelleen matriisimuodossa. Matriisin ei kuitenkaan muodostu selkeää tai merkityksellistä kuviota (kuvio 8, vasemmanpuoleinen kuvio). Analyysin seuraavassa askeleessa vaihdetaan sarakkeiden ja rivien järjestystä matriisissa. Lui ja Chan (2005) määrittelivät funktion $f(i,j)=(i+1,j)+(i-1,j)+(i,j+1)+(i,j-1)$, jossa $(i,j)=1$ jos matriisin ruutu rivillä i ja sarakkeessa j on merkitty, muuten $(i,j)=0$. $f(i,j)$:n kokonaisarvo matriisissa on

$$\sum_{j=1}^{12}\sum_{i=1}^{12} f(i,j)$$

Vaihtamalla kaksi riviä keskenään sekä niiden vastaavat sarakkeet, jotkut pisteet siirtyvät kauemmas toisistaan, mikä vähentää $f(i,j)$:n kokonaisarvoa. Samalla tavoin pisteet voivat siirtyä lähemmäs toisiaan, mikä puolestaan lisää kokonaisarvoa. Tämän periaatteen mukaan matriisi voidaan järjestää uudelleen niin, että kokonaisarvo muodostuu mahdollisimman suureksi. Kuvion 8 oikeanpuoleinen matriisi näyttää tuloksen järjestelyn jälkeen ja paljastaa kuviokeskittymän.



KUVIO 8 VDM:n avulla löydetty kuviokeskittymä (Lui ym., 2005, 478)

Kuviosta päätellen ja klusterianalyysia hyväksikäyttäen Lui ja Chan (2005) ovat päätyneet neljään XP-käytänteiden käyttöönottovaiheeseen. Vaiheet ovat seuraavat:

1. testaus, yksinkertainen suunnittelu, refaktorointi, koodaussäännöt
2. jatkuva integraatio
3. pariohjelmointi, yhteisomistajuus
4. metafora, 40-tuntinen viikko, pienet julkaisut, läsnäoleva asiakas, suunnittelupeli

Lui ja Chan (2005) sanovat saaneensa paljon positiivista palautetta ohjelmointitiimeiltä, jotka ovat kokeilleet mallia pyrkiessään ottamaan käyttöön XP:tä. Lui ja Chan (2005) pitävät menetelmän etuna sitä, että se opettaa samalla tiimejä

ymmärtämään, mitä tiimin pitäisi omaksua ja miksi. Mallin rajoituksena voidaan pitää sitä, että se ei aikatauluta vaiheita mitenkään. Mallista ei ole validointitietoja saatavilla.

4.5 Nawrockin, Walterin ja Wjochiechowskin malli

Nawrocki, Walter ja Wjochiechowski (2001) esittävät kypsyysmallin XP:n käyttöönotolle. Mallin nimi on eXtreme Programming Maturity Model eli XPMM. Mallista haluttiin tehdä mahdollisimman kevyt ja yksinkertainen, koska myös XP on ketteränä menetelmänä kevyt. Nawrocki ym. (2001) käyttävät osin CMM(I):tä (SEI, 2010) ja PSP:tä (Personal Software Process) (Humphrey, 2002) oman mallinsa pohjana. CMM(I):n he valitsivat sen laajan käytön ja tunnettuuden vuoksi.

CMM(I):ssä on viisi tasoa, kun taas XPMM:ssä on neljä tasoa. Molemmissa tasoille on määritelty käytänteitä, joita täytyy noudattaa tasolle pääsemiseksi. Sekä CMMI:n että XPMM:n ensimmäisellä tasolla ovat organisaatiot/projektit, jotka eivät toteuta mitään mallin määriteltyjä käytänteitä. CMMI:n ja XPMM:n tasoilla 2 yhdistävänä tekijä on suuntautuminen projektitiimeihin.

Nawrocki ym. (2001) jakavat XP-käytänteet viiteen osa-alueeseen, yleissuunnitteluun (planning, P), yksityiskohtaisempaan suunnitteluun (designing, D), koodaukseen (coding, C), testaukseen (testing, T) sekä ympäristöön (facilities, F). Jokainen näistä osa-alueista sisältää useita siihen liittyviä XP-käytänteitä, paitsi ympäristö, jossa on vain yksi XP-käytänne. Osa-alueeseen ”testaus” kuuluvat esimerkiksi seuraavat käytänteet:

T0. Kaikella koodilla täytyy olla yksikkötestit.
T1. Kaiken koodin täytyy läpäistä yksikkötestit ennen julkaisemista.
T2. Jos löydetään vika, siitä täytyy luoda testitapaus.
T3. Hyväksymistestejä ajetaan usein ja niiden tulokset julkaistaan.
T4. Automatisoituja testejä käytetään tukemaan säännöllisiä integraatiotestejä.

Nawrockin ym. (2001) mallissa on neljä kypsyystasoa: ei ollenkaan yhteensopiva (Not compliant at all), alustava (Initial), kehittynyt (Advanced) ja kypsä (Mature) (kuvio 9). Edetäkseen mallissa tasoja ylöspäin projektitiimin täytyy noudattaa kaikkia sille tasolle ja sitä alemmille tasoille asetettuja käytänteitä.

Ensimmäisellä tasolla (Ei ollenkaan yhteensopiva) käytetään hyvin vähän tai ei vielä ollenkaan XP käytänteitä. *Toisen tason* (Alustava) avainprosessialueita ovat asiakkuuksien hallinta ja tuotteen laadunvalvonta, joihin liittyy 18 XP-käytännettä. *Kolmannella tasolla* (Kehittynyt) keskitytään pariohjelmointiin, ja siihen liittyy 11 XP-käytännettä. *Neljännellä tasolla* (Kypsä) painopistealueena on projektin suorituskyky, johon liittyy kolme XP-käytännettä. Taulukossa 2 on esitetty yhteenveto tasojen 2-4 avainprosessialueista sekä niihin liittyvistä XP-käytänteistä. Taulukossa käytetään seuraavia lyhenteitä P (planning) yleis-

suunnittelu, D (designing) yksityiskohtaisempi suunnittelu, C (coding) koodaus, T (testing) testaus sekä F (facilities) ympäristö.

KUVIO 9 Nawrockin ym. (2001, 236) malli

Mallia on testattu Poznan Teknillisessä yliopistossa. Viisi projektitiimiä, joissa kussakin oli kuusi opiskelijajäsentä (kaksi päällikköä ja neljä suunnittelijaa), toimivat projekteissa XP:n mukaisesti. Heitä pyydettiin organisoimaan työnsä XPMM:n mukaan ja soveltamaan niin montaa XP käytännettä kuin mahdollista ja saavuttamaan mahdollisimman korkea taso mallissa. Testituloksia tai validointitietoja ei ole saatavissa.

4.6 Packlickin malli

Packlick (2007) esittelee tapaustutkimuksen, joka on toteutettu Sabre Airline Solutions -nimisessä yrityksessä. Yrityksessä on jo kauan käytetty ketteriä menetelmiä, kuten XP:tä, osia Scrumista sekä Lean-periaatteita ja havaittu näiden menetelmien tehokkuus sekä tuottavuuden että laadun suhteen.

Scrum-menetelmässä sprintin lopussa toteutettavat jälkitarkastelupalaverit, retrospektiivit (retrospective), ovat yksi tapa edistää ketterien menetelmien kehittämistä ja uusien kehitystarpeiden havaitsemista. Sabre Airline Solutions -yrityksessä havaittiin kuitenkin, että retrospektiivit eivät riitä kehittämään ketteriä menetelmiä riittävästi. Aivoriihien tuloksena lähtökohdaksi otettiin tavoiteperusteinen näkökulma.

TAULUKKO 2 Tasojen 2-4 avainprosessialueet ja XP-käytänteet XPMM-mallissa (Nawrocki ym., 2001)

Taso 2		**Taso 3**	**Taso 4**
Asiakkuuksien hallinta	**Laadunvalvonta**	**Pariohjelmointi**	**Projektin suorituskyky**
P0. Suunnittelupelin käyttö projektien suunnitteluun	C2. Ensin koodataan yksikkötestit	P8. Parien/tehtävien kierrätys	C0. Asiakas kehitystiimin tiloissa
P2. Käyttäjätarinoiden kirjoittaminen	C5a. Integrointi usein	C0b. Asiakas käy säännöllisesti kehitystiimin luona	C8. Ei ylitöitä
P3. Suunnitelma luo aikataulun	C7. Optimoinnin jättäminen viimeiseksi	C1. Koodi kirjoitetaan sovittujen sääntöjen mukaan	T1. Kaiken koodin täytyy läpäistä yksikkötestit ennen julkaisemista
P4. Säännölliset, pienet julkaisut	T0. Kaikella koodilla on yksikkötestit	C3. Kaikki tuotantokoodi on pariohjelmoitua	
P5. Projektin käytettyä aikaa (velocity) mitataan	T1a. Kaiken koodin täytyy läpäistä yksikkötestit ennen julkaisemista	C4. Vain yksi pari integroi kerrallaan	
P6. Projekti on jaettu iteraatiohin	T2. Löytyneestä viasta täytyy luoda testitapaus	C5b. Integrointi usein	
P7. Jokainen iteraatio alkaa suunnittelulla	T3. Hyväksyntätestejä ajetaan usein ja niiden tulokset julkaistaan	C6. Koodin yhteisomistajuus	
D1. Järjestelmän metaforan valinta		C8b. Ylityötiedot kerätään ja julkaistaan	
D3. Julkaisu aikaisin riskien minimoimiseksi		C9. Versionhallinta on käytössä	
D4. Toimintoja ei saa lisätä ilman vaatimuksia		T4. Automatisoituja testejä käytetään tukemaan säännöllisiä integraatiotestejä	
C0a. Tehokas yhteistyö asiakkaan kanssa		F0. Tiimillä on avoin työtila	

Yrityksen parhaimmat ketterän kehittämisen asiantuntijat määrittelivät työlistalle yli kolmekymmentä käyttäjätarinaa niistä ketteristä käytännöistä, joita haluttiin parantaa. Nämä käyttäjätarinat ryhmiteltiin edelleen viideksi korkean tason ketterän kehittämisen tavoitteeksi (Paclick, 2007). Alla nämä tavoitteet on esitetty järjestyksessä, jossa englanninkielisten termien alkukirjaimista syntyy sana AGILE (Paclick, 2007):

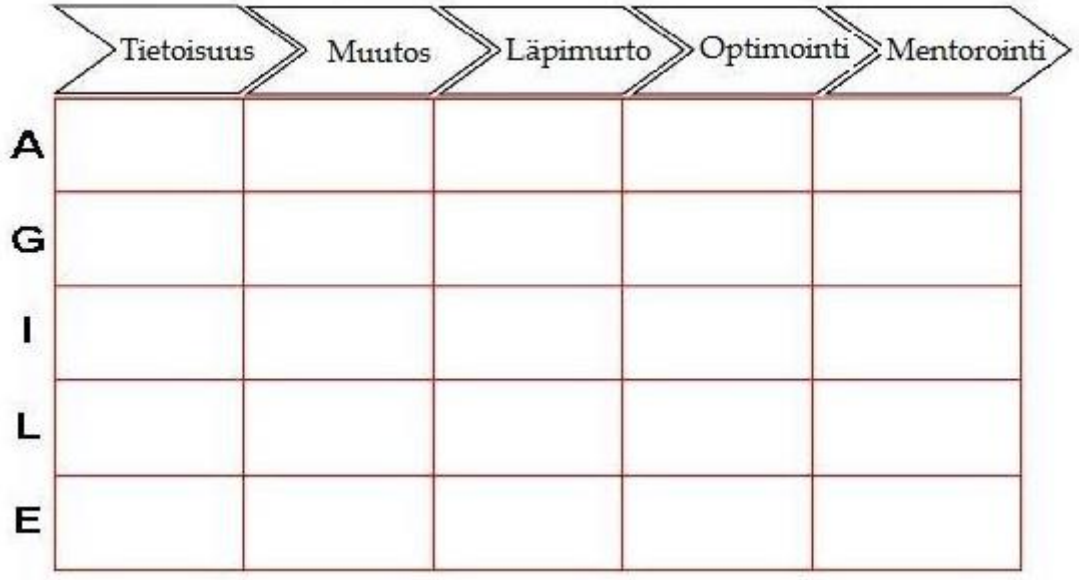
- *Hyväksymiskriteerit* **(A,** Acceptance Criteria). Toiminnallisen oikeellisuuden validointi on kriittistä.
- *Automatisoidut testit ja koonnit* **(G,** Green-Bar Tests and Builds). Koodi validoi koodia, koonnit ovat automatisoituja. Epäonnistuneet koonnit ja testit korjataan heti.
- *Iteratiivinen suunnittelu* **(I,** Iterative Planning). Suunnittele jatkuvasti, pidä prosessi läpinäkyvänä, sitouta tiimi, käytä todellisuuteen perustuvaa palautetta päätöksenteon apuna.
- *Oppiminen ja sopeuttaminen* **(L,** Learning and Adapting). Keskity myös taitojen parantamiseen ja oppimiseen ei ainoastaan tekemiseen.
- *Osaaminen* **(E,** Engineering Excellence). Noudata käytäntöjä, jotka on luotu varmistamaan ohjelmistotuotannon laatua, kuten esimerkiksi koodauskäytännöt, refaktorointi ja pariohjelmointi.

Jotta voitaisiin seurata ja arvioida missä määrin kehitystiimi on kunkin tavoitteen osalta edistynyt ketterien tavoitteiden saavuttamisessa, määriteltiin viisi kypsyystasoa:

1. *Tietoisuus* (Awareness). Tiimi ymmärtää tavoitteet ja ymmärtää tavoitteiden saavuttamisen merkityksen. Myös ymmärrys ”paremmista” käytännöistä on olemassa. Tiimi voi käyttää joitakin ketteriä perusominaisuuksia tavoitteiden saavuttamiseen.
2. *Muutos* (Transformation). Tieto siirretään aktiivisesti käytännön toimintaan. Ketteriä kehittämiskäytänteitä käytetään säännöllisesti. Sekä tiiminjäsenet että vetäjät ovat sitoutuneita tavoitteiden saavuttamiseen.
3. *Läpimurto* (Breakthrough). Tiimi käyttää jatkuvasti ketteriä periaatteita saavuttaakseen tavoitteensa, jopa stressitilanteissa.
4. *Optimointi* (Optimizing). Parannuksia tehdään jatkuvasti. Luovan innovaation henki parannuksissa näkyy.
5. *Mentorointi* (Mentoring). Tiimin suorituskyky on korkea ja se pystyy mentoroimaan muita tiimejä. Tämä edistää oppimista organisaatiotasolla.

Ketteryyden AGILE-tavoitteista ja kypsyystasoista muodostuu yhdessä ketterä kypsyysmalli (AMM, Agile Maturity Model) (kuvio 10). Käyttäjätarinat sijoitetaan mallin soluihin. Esimerkiksi käyttäjätarina ”Kehityssuunnitelma tarkistetaan joka iteraatiossa” voidaan sijoittaa riville I (iteratiivinen suunnittelu) ja sarakkeeseen ”Tietoisuus”. Käyttäjätarina ”Kaikki tiimin jäsenet pystyvät nopeas-

ti ja näkyvästi arvioimaan edistymistä suhteessa suunniteltuun lopputulokseen" sijoittuisi myös riville I (iteratiivinen suunnittelu), sarakkeeseen "Läpimurto". Kun kaikki käyttäjätarinat on sijoitettu malliin, saadaan sen avulla helposti yleiskuva tilanteesta ja voidaan asettaa tavoitteita sekä seurata edistymistä.



KUVIO 10 Packlickin (2007, 269) kypsyysmalli

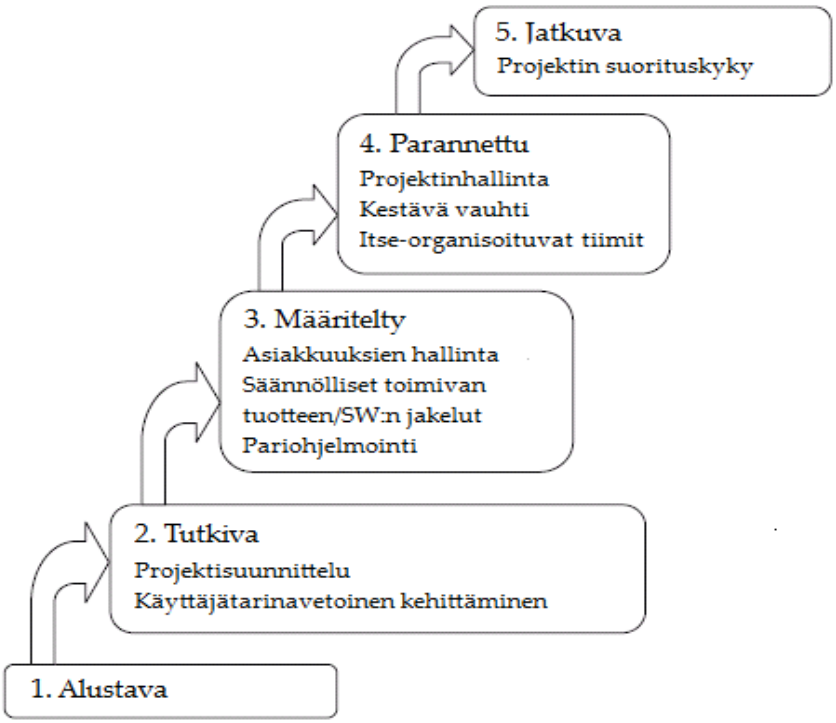
Kaksikymmentä tiimiä on käyttänyt tätä mallia Sabre Airline Solutions -yrityksessä yli kuuden kuukauden ajan. Packlickin (2007) mukaan useimmissa tiimeissä on tapahtunut huomattavia parannuksia. Seuraavassa muutamia esimerkkejä kehittymisestä: tehokkuus on parantunut, tiimit pitävät arviointipalavereita säännöllisemmin, aikaisemmin esiintyneitä ongelmia on saatu ratkaistua, tottumattomammat tiimit hakevat aktiivisemmin apua edistyneemmiltä tiimeiltä, koodin analysointityökaluja ja metriikoita käytetään aktiivisemmin.

Packlickin (2007) mukaan mallia kannustetaan käyttämään ja kehittämään edelleen organisaatiossa. Mallin tavoite-suuntautuneen lähestymistavan ansiosta parannukset on todettu saatavan tehtyä nopeammin ja ne ovat kestävämpiä. Tiimin jäsenet arvostavat ja kunnioittavat prosessia joka toimii. Arvostus ja kunnioitus lisääntyvät vielä enemmän silloin, kun tiimin jäsenet ovat itse päässeet kehittämään prosessia.

4.7 Patelin ja Ramachandranin malli

Patelin ja Ramachandranin (2009) kypsyysmalli on nimeltään Agile Maturity Model (AMM). Mallin tavoitteena on parantaa ja lisätä ketterien menetelmien käyttöä ja vahvistaa niiden avulla saavutettavia hyötyjä, joita ovat muiden muassa pienemmät tuotantokustannukset, korkea tuottavuus, korkea laatu ja asiakastyytyväisyys. Mallissa on viisi kypsyystasoa (kuvio 11): alustava (initial),

tutkiva (explored), määritelty (defined), parannettu (improved) ja jatkuva (sustained).



KUVIO 11 Patelin ja Ramachandranin (2009, 6) malli

Ensimmäisellä tasolla (alustava) ei ole määritelty ollenkaan ketterän prosessin kehittämistavoitteita. Ohjelmistokehityksen käytännöt ja prosessit ovat hyvin vähäisiä, eivätkä ne ole välttämättä toistettavissa. Suurimmat ongelmat tällä tasolla liittyvät ylitöihin, aikataulujen venymiseen, kommunikaatioon, tuotteiden laatuun sekä tuotekehityskustannuksiin.

Toisella tasolla (tutkiva) edellä mainitut ongelmat vähenevät, mutta kommunikaatio sekä koodaus- ja integrointikäytänteet pysyvät edelleen ongelma-alueina. Toisen tason organisaatio on keskittynyt projektisuunnitteluun. Suunnittelupeli ja asiakkaan määrittelemät käyttäjätarinat ovat käytössä. Asiakkaan edustaja on samassa tilassa kehitystiimin kanssa. Prosessia arvioidaan ja verrataan aikaisempiin projekteihin.

Kolmannella tasolla (määritelty) keskitytään asiakassuhteiden hoitamiseen sekä ohjelmistojen laatuun. Myös koodaussäännöt ja -käytännöt parantuvat. Testilähtöisen suunnittelun taitojen kasvaessa koodin laatu ja testaus-käytännöt parantuvat. Ajankäyttö on edelleen ongelmana. Tällä tasolla pystytään ratkaisemaan suurin osa teknisistä ongelmista, mutta organisatoriset ongelmat, jotka liittyvät tiimeihin, jäävät ratkaisematta.

Neljännellä tasolla (parannettu) organisaatiossa pystytään jo keräämään mitattua tietoa prosessista ja käytänteistä sekä tuotteen laadusta. Tällä tasolla keskitytään projektin hallintaan, ajankäyttöön, itseohjautuviin tiimeihin sekä riskienhallintaan eli enemmän tiimiin vaikuttaviin asioihin kuin itse tuotteeseen.

Viidennellä tasolla (jatkuva) oleva organisaatio kehittää jatkuvasti prosessejaan kerätyn mittaustiedon perusteella. Sekä asiakkaat että kehittäjät ovat tyytyväisiä. Tasolla keskitytään epävarmuuden hallintaan, vikojen ennalta ehkäisyyn sekä suorituskyvyn ja sisällön parantamiseen.

Patel ja Ramachandran (2009) ovat määritelleet jokaisen tason tavoitteille, myös avainprosessialueet (Key Process Areas). Jokaiselle avainprosessialueelle on lisäksi määritelty useita arviointikysymyksiä, joiden avulla voidaan arvioida, onko kyseinen taso saavutettu kulloisenkin kysymyksen osalta. Taulukossa 3 on esitetty yhteenveto tasojen tavoitteista ja avainprosessialueista.

TAULUKKO 3 Patelin ja Ramachandranin (2009) mallin tavoitteet ja avainprosessialueet.

	Tavoitteet	**Avainprosessialueet**
1 Alustava (Initial)	Prosessinkehitystavoitteita ei ole määritelty	Ei ole.
2 Tutkiva (Explored)	• Projektisuunnittelu • Vaatimustenhallinta • Sidosryhmien perehdyttämien • Arvon, yhteistyön ja suunnittelukäytänteiden parantaminen	• Projektisuunnittelu • Käyttäjätarinalähtöinen suunnittelu • Asiakkaan saatavuus paikan päälle • TDD:n käyttöönotto
3 Määritelty (Defined)	• Asiakastyytyväisyys • Kommunikaation parantaminen • Ohjelmistojen laatu • Koodauskäytäntöjen ja -standardien parantaminen	• Asiakkuuksien hallinta • Säännölliset toimitukset • Pariohjelmointi • Keskinäinen vuorovaikutus • TDD, testilähtöinen kehittäminen • Täytäntöönpano ja vuorovaikutus • Koodausstandardit
4 Parannettu (Improved)	• Tiimin voimauttaminen ja palkitseminen • Projektin hallinta • Riskienhallinta • Ei ylitöitä • Yksinkertaisuus	• Projektinhallinta • Kestävä vauhti • Itseohjautuva tiimi • Riskienhallinta • Koodioptimoinnin suunnittelu
5 Jatkuva (Sustained)	• Sisällön parantaminen • Epävarmuuden hallinta • Projektin suorituskyky • Vikojen ehkäiseminen	• Projektisuunnittelu, koontisuunnittelu • Käyttäjätarinavetoinen kehittäminen

Patelin ja Ramachandranin (2009) mallin lähestymistavasta on keskusteltu kolmen organisaation kanssa. Yhdessä niistä mallia on otettu käyttöön, ja tulokset ovat olleet positiivisia. Kahdessa muussakin organisaatiossa suhtautuminen malliin on ollut positiivista. Validointitutkimus on artikkelin kirjoittamisen aikaan meneillään, mutta siitä ei ole vielä julkaistua tietoa.

4.8 Pettitin malli

Pettit (2006) halusi luoda yksinkertaisen ja joustavan mallin ketterän kehittämisen arviointiin. Mallin avulla voidaan nopeasti arvioida nykyistä prosessia ja asettaa tavoitteita sen parantamiseksi. Mallin avulla organisaatio voi myös tunnistaa omia vahvuuksiaan ja heikkouksiaan ketterän kehittämisen alueella sekä havaita, mitkä toiminnot estävät ja toisaalta mahdollistavat IT-liiketoiminnan yhdenmukaistamista. Pettitin (2006) mukaan mallia voidaan käyttää sekä projekti- että organisaatiotasoisesti.

Pettitin (2006) mukaan ketteryyttä voidaan arvioida kaikkien järjestelmäkehityksen toimintojen osalta, vaatimusmäärittelystä testaamiseen. Jokainen näistä joko estää tai vahvistaa IT-toiminnon reagointikykyä liiketoimintaympäristön muutoksiin. Pettit (2006) esittää, miten mallia voitaisiin käyttää vaatimusten keräämisen (requirements collection) ketteryyden tarkasteluun. Eteneminen kohti ketteryyttä voi tapahtua seuraavin askelin (stages):

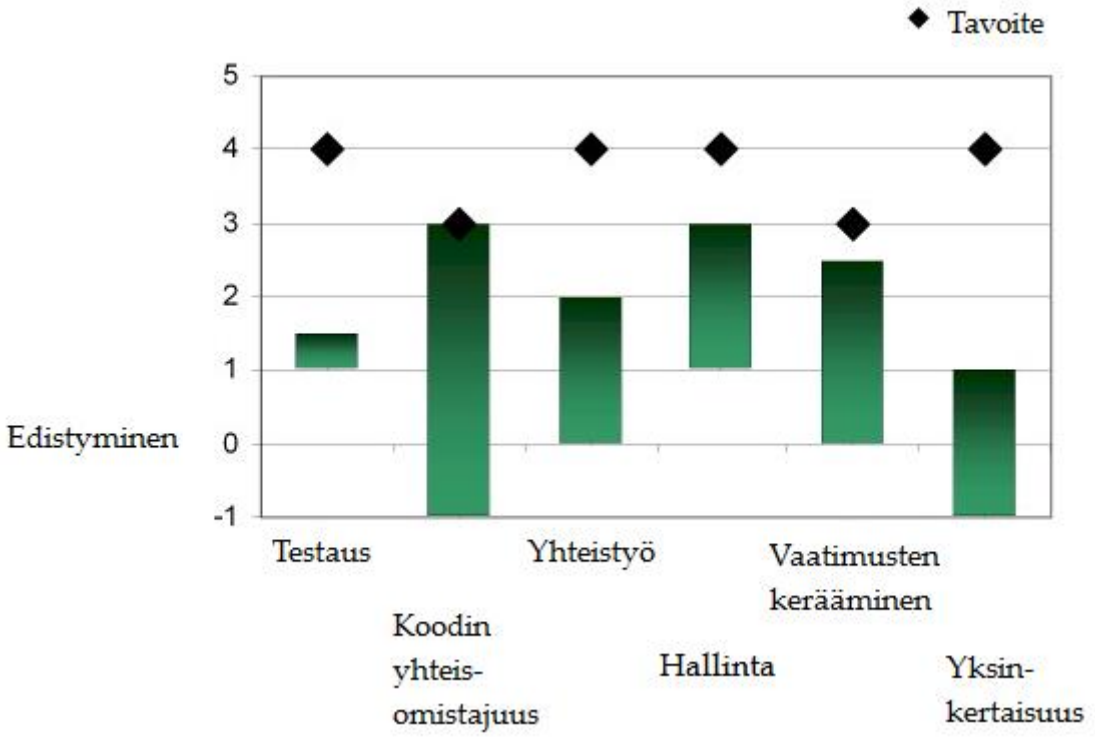
- *Vähiten ketterä*: kehittämistehtävät valitaan suuresta määrästä jäädytettyjä vaatimuksia
- *Tuotetaan kevyitä hahmotelmia*, joista kehitetään korkeamman tason vaatimuksia.
- *Hajanaisista vaatimuksista* kootaan korkean tason käyttäjä-vaatimuksia.
- *Käyttäjätapaukset lajitellaan* tiimeille sopiviksi toiminnallisuuksiksi, jotka voidaan toteuttaa aikarajoitetussa (time-boxed) iteratiivisessa kehittämisympäristössä.
- *Suunnitellaan toiminnallisia kokonaisuuksia*, jotka sisältävät testattavissa olevat hyväksymiskriteerit sekä arvion toiminnallisuuden arvosta.
- *Käyttäjätarinoita kehitetään spontaanisti*. Käyttäjätarinat eivät ole peräisin olemassa olevista lähteistä, mutta ne ovat kuitenkin ilmauksia asiakkaiden tarpeista ja kysynnästä.
- *Eniten ketterä:* Globaali arkisto toiminnallisista vaatimuksista aina liiketoiminnan kehittämiin käyttäjätarinoihin. Käyttäjätarinoissa on mukana myös arvio vaatimuksen/käyttäjätarinan tuomasta hyödystä.

Vaatimusten keräämisen lisäksi Pettit (2006) käsittelee seuraavia ohjelmistokehityksen osa-alueita:

- testaus – missä määrin testaus on integroitu päivittäiseen kehittämiseen?
- koodin yhteisomistajuus – kuinka paljon pariohjelmointia toteutetaan ja kuinka hyvin eri ihmiset pystyvät muokkaamaan koodia eri järjestelmissä?
- yhteistyö – kuinka paljon ja kuinka usein on tapahtumia, joissa kehittäjät, johto ja asiakkaat ovat yhteydessä toisiinsa ja kommunikoivat?
- hallinta – miten hyvin johdon aktiviteetit integroituvat suoraan kehitystehtäviin?

- yksinkertaisuus – missä määrin suunnittelijat ovat sidottu teknisiin suunnitelmiin vai pystyvätkö he muokkaamaan uusia vaatimuksia?

Valitut osa-alueet sijoitetaan kaavioon (kuvio 12), jossa voidaan arvioida osa-alueen alkutila, nykyinen tila sekä tavoitetila. Tilat voidaan numeroida vaihteluvälillä -1 (estää ketteryyttä) – 5 (paras ketteryyden mahdollistaja). Jokaisen osa-alueen tavoitteen saavuttaminen voidaan organisaatiokohtaisesti vaiheistaa vähiten ketterästä eniten ketterään. Kun vaiheistukset on kerran määritelty, niitä voidaan arvioida säännöllisesti ja esittää kaaviona. Kaavion avulla on helppo nähdä nykytilanne, edistyminen sekä kehitystarpeet.



KUVIO 12 Pettitin (2006) malli

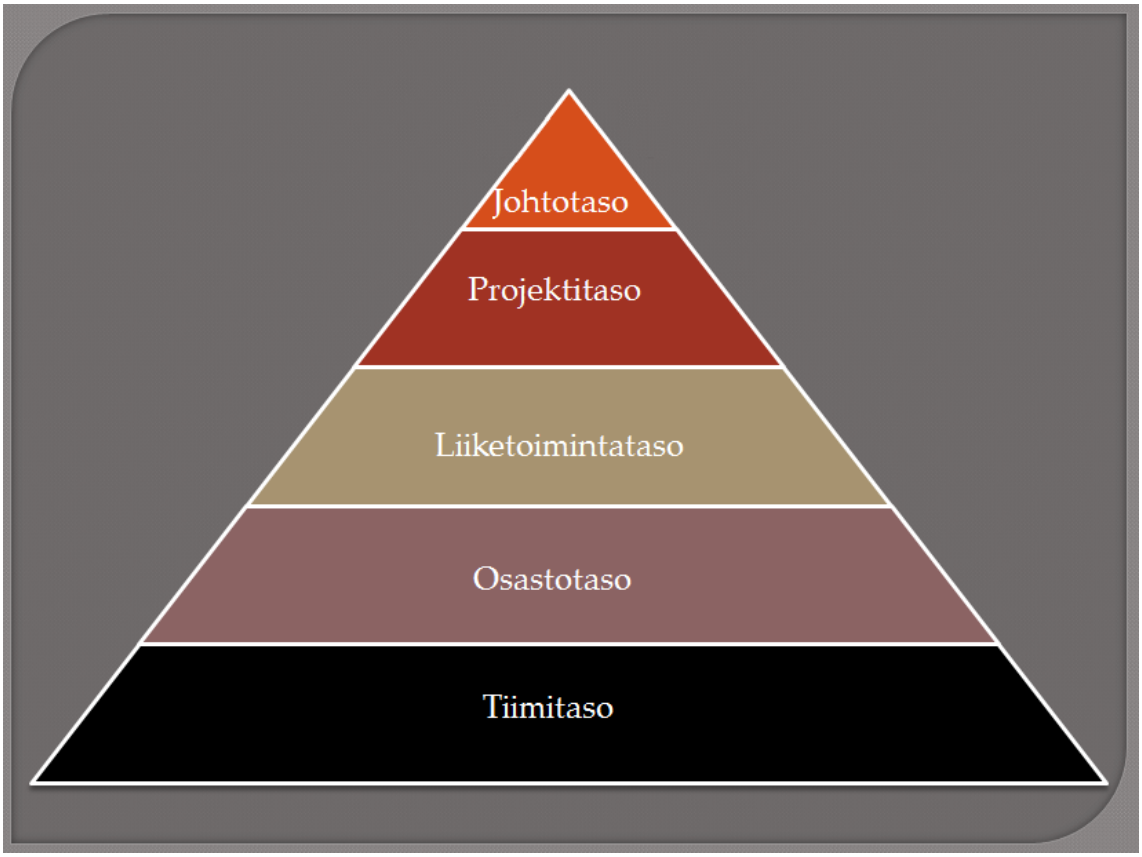
Kuvion 12 esimerkkitapauksessa ”Yhteistyö” on alussa ollut tasolla 0, arviointihetkellä ollaan edistytty tasolle 2 ja tavoitteena on päästä tasolle 4. ”Koodin yhteisomistajuus” puolestaan on lähtötasolla ollut -1 ja arviointihetkellä on edistytty tasolle 3, mikä oli alun perin asetettu tavoite-tasoksi.

Mallista ei ole käytännön kokemuksia eikä validointitietoja saatavilla

4.9 Prouxin malli

Proulxin (2010) Agile Maturity Model –malli (AMM) on tarkoitettu käytettäväksi Scrum-menetelmän yhteydessä. Hänen mielestään Scrum-menetelmään keskittyminen oli perusteltua siksi, että Scrum on suosituin ketterä menetelmä

(Forrester Research Inc., 2010). AMM-malli on kaksiulotteinen. Sen tasot on määritelty organisaatiotasojen mukaisesti seuraavasti: tiimitaso, osastotaso, liiketoimintataso, projektitaso ja johdon taso (kuvio 13).



KUVIO 13 Proulxin (2010) malli

Proulx (2010) on määritellyt jokaiselle tasolle kypsyyden. Sen lisäksi hän on esittänyt, miten kyseisen tason kypsyys näkyy muille tasoille ja mitkä ovat tulokset tason saavuttamista (taulukko 4).

Ensimmäisellä tasolla (tiimitaso) tiimissä on Scrum-mestari ja tiimi käyttää joitakin Scrumin käytänteitä, mutta ei välttämättä johdonmukaisesti. Prosessia ei ole dokumentoitu, ja se vaihtelee projektien välillä. Ketterät käytänteet on opittu itse. Tiimin ulkopuolella tuskin kukaan on kuullut ketteristä menetelmistä tai jos onkin, he eivät ole asiasta kiinnostuneita. Tästä seuraa kommunikaatiovaikeuksia. Tiimi on hieman tuottavampi ja työmoraali on hieman parantunut, mutta tiimin ja muun organisaation välillä on paljon kitkaa, koska tiimin jäsenet yrittävät opettaa muille, mitä ketteryys on ja miten sitä voisi käyttää.

Toisella tasolla (osastotaso) Scrum-menetelmää aletaan käyttää tiimeissä johdonmukaisesti. Johdonmukaisuus tiimien välillä on tosin epätasaista. Myös dokumentaation taso vaihtelee tiimien välillä. Osastotasolla ketteriä käytänteitä aletaan käyttää lisääntyvissä määrin. Ylemmillä tasoilla ketteryys ei vielä paljon näy. Projektipäälliköt tiedostavat uudet käytänteet, mutta heillä on muutosvas-

tarintaa uutta menetelmää kohtaan, koska heillä ei ole riittävästi tietoa. Tuloksena tiimit, jotka käyttävät ketteriä menetelmiä, ovat hieman tuottavampia, mutta tiimien välillä on eroja. Joidenkin tiimien tuottavuus on voinut jopa laskea alkuvaikeuksien takia ja osa tiimeistä on voinut palata käyttämään vanhoja käytänteitä.

TAULUKKO 4 Proulxin (2010) mallin tasot.

	Tiimi	Osasto	Liiketoiminta	Projekti	Johto	Tulokset
Taso 1 Tiimi	Scrum-mestari Itse opitut Scrum käytänteet osittain käytössä Prosessi ei ole dokumentoitu ja vaihtelee eri projektien välillä	Ei ole tietoinen ketteristä menetelmistä tai ei ole kiinnostunut niistä Perinteiset menetelmät käytössä	Ei ole tietoinen ketteristä menetelmistä tai ei ole kiinnostunut niistä Perinteiset menetelmät käytössä	Ei ole tietoinen ketteristä menetelmistä tai ei ole kiinnostunut niistä Perinteiset menetelmät käytössä	Ei ole tietoinen ketteristä menetelmistä tai ei ole kiinnostunut niistä Perinteiset menetelmät käytössä	Tiimi on hiukan tuottavampi ja työmoraali hiukan parempi Paljon kitkaa muiden ryhmien kanssa
Taso 2 Osasto	Scrum-mestari Scrum käytänteitä aletaan käyttää johdonmukaisesti Jonkin verran dokumentointia, mutta vaihtelee tiimien välillä Mahdollisesti käytetty jo ulkopuolisia asiantuntijoita	Pääasiassa vielä perinteiset menetelmät käytössä Vaikka yhä useammat tiimit ottavat ketteriä menetelmiä käyttöön	Ei ole tietoinen tai ei ole kiinnostunut tiimin lähestymistavasta Perinteiset menetelmät käytössä Mahdollisesti hiukan enemmän yhteistoimintaa kehitystiimin ja liiketoiminnan välillä	Tietoisuus uusista menetelmistä Muutosvastarintaa Perinteiset menetelmät käytössä Ketterien menetelmien ei nähdä sopivan suurin projeteihin	Ei ole tietoinen ketteristä menetelmistä tai ei ole kiinnostunut niistä Perinteiset menetelmät käytössä	Tiimien tuottavuus on epätasaista, osalla parantunut, osalla huonontunut, osa luopunut käytöstä Työmoraali paranee
Taso 3 Liiketoiminta	Scrum-mestari ja muut Scrumin roolit käytössä, mahdollisesti myös Scrumien Scrum Ulkopuolisia asiantuntijoita käytetty Kongresseihin osallistuminen	Rooleissa on hämmennystä Prosessi dokumentoitu ja yhtenäinen eri projektien välillä Ulkopuolisia asiantuntijoita on käytetty	Tuoteomistajan rooli selkeästi määritelty Prosessi leviää hitaasti liiketoimintapuolella	Tietoisuus lisääntyy Muutosvastarintaa Perinteiset menetelmät käytössä Ketterien menetelmien ei nähdä sopivan suurin projekteihin	Tietoisuus lisääntyy, kuitenkin vielä paljon oletuksia ja väärinymmärryksiä Yksittäinen "evankelista" tuo uutta näkemystä esille	Tiimit ovat tuottavampia Työmoraali on huomattavasti korkeampi Henkilöstöhallinnon ja projektihallinnon välillä kitkaa
Taso 4 Projekti	Scrum-mestarin ja projektipäällikön roolit ovat selkeät Mahdollisesti Scrumien Scrum Tiimit ovat itsenäisiä ja käytänteitä seurataan tarkasti.	Scrum käytänteitä käytetään johdonmukaisesti Roolit alkavat selkeytyä Prosessi dokumentoitu ja yhtenäinen eri projektien välillä Ulkopuolisia asiantuntijoita on käytetty	Roolit määritelty ja hyväksytty Inkrementaalisuus ja iteratiivisuus ovat täysin hyväksyttyjä Prosessi leviää edelleen liiketoimintapuolella	Projektipäälliköt ovat täysin tietoisia uusista käytänteistä Perinteisiin menetelmiin lisätään enemmän ketteriä lähestymistapoja Ketterygs on hyväksytty myös suurin projekteihin	Tietoisuus lisääntyy, vielä jonkin verran oletuksia ja väärinymmärryksiä Johtajien kouluttaminen alkaa ja on suosittua Yksittäinen "evankelista" promotoi näkemystä	Tiimit ovat tuottavampia Työmoraali on korkealla kaikkialla Ongelmat perinteisten roolien suhteen ovat hallinnassa
Taso 5 Johto	Scrum-mestari ja muut Scrumin roolit käytössä, mahdollisesti myös Scrumien Scrum Ulkopuolisia asiantuntijoita käytetty Kongresseihin osallistuminen	Scrum käytänteitä käytetään johdonmukaisesti Roolit ovat selkeät Prosessi dokumentoitu ja yhtenäinen eri projektien välillä	Roolit määritelty ja hyväksytty Inkrementaalisuus ja iteratiivisuus ovat täysin hyväksyttyjä Prosessi leviää johtotasolle	Projektipäälliköt ovat täysin tietoisia uusista käytänteistä Perinteisiin menetelmiin lisätään enemmän ketteriä lähestymistapoja Ketterygs on hyväksytty myös suurin projekteihin Parhaiden käytänteiden arviointi	Tiimeillä on täydet valtuudet sekä vastuu Yhteistyön ja tiimityön edistäminen Jatkuvan oppimisen tukeminen Johtamistyyli vaihtelee tiimien mukaan	Tiimit ovat tuottavampia Työmoraali on korkealla kaikkialla Kitka uutta lähestymistapaa kohtaan on hävinnyt

Kolmannella tasolla (liiketoimintataso) myös liiketoimintaihmiset ovat mukana ketterässä kehittämisessä. Yhteistyön määrä lisääntyy ja samalla myös luottamus liiketoimintaihmisten ja tiimi-osastoihmisten välillä lisääntyy. Tiimitasolla Scrum-mestarin lisäksi muutkin Scrumin rooleista ovat käytössä. Jos Scrum-tiimejä on useita, myös Scrumien Scrum (Scrum of Scrums) on käytössä. Tälle tasolle pääsemisen edellytyksenä on, että myös ulkopuolista apua on käytetty. Osastotasolla rooleissa on hämmennystä. Prosessi on dokumentoitu ja yhtenäinen eri projektien välillä. Liiketoimintatasolla tuoteomistajan (Product owner) rooli on selkeästi määritelty. Projektin-hallintatasolla on vielä muutosvastarintaa ja vaikka ketterät käytänteet ovat jo tiedossa, projekteja johdetaan edelleen vanhojen käytänteiden mukaisesti. Projektinhallintatasolla ei uskota, että ketterät menetelmät sopivat suuriin projekteihin. Johtotasolla on jo jotakin tietoa, mutta silti esiintyy vielä väärinymmärryksiä. Tuloksena tiimit, jotka käyttävät ketteriä menetelmiä, ovat tuottavampia, myös työmoraali on parempi. Henkilöstöhallinnon ja projektihallinnon välillä on kitkaa.

Neljännellä tasolla (projektitaso) projektinhallinnassakin käytetään jo osin ketteriä menetelmiä, vaikka henkilöstöhallinto vielä osin käyttää perinteisiä menetelmiä. Tiimit ovat itsenäisiä ja Scrumin käytänteitä seurataan tarkasti. Osastotasolla ketterät periaatteet ovat käytössä ja niitä käytetään johdonmukaisesti. Liiketoimintatasolla prosessi laajenee. Projektipäälliköiden keskuudessa ketterät menetelmät on täysin hyväksytty. Johdontasolla tietoisuus lisääntyy, vaikka jonkin verran on vielä oletuksia ja väärinymmärryksiä. Tuloksena tiimit ovat tuottavampia ja työmoraali on huomattavasti parempi.

Viidennellä tasolla (johtotaso) myös johtajat ovat omaksuneet ketterät menetelmät. Tiimitasolla Scrumin roolit ovat täysin käytössä. Tiimin jäsenet osallistuvat Agile-konferensseihin. Osastotasolla suurin osa Scrum-käytänteistä on omaksuttu ja niitä käytetään säännöllisesti. Prosessi on dokumentoitu ja johdonmukainen eri projekteissa. Liiketoimintatasolla tuoteomistajaroolit ovat selkeät ja osa tiimiä. Projektinhallintatasolla myös projektinjohtamiseen on otettu ketteriä käytänteitä käyttöön. Ketteryys on hyväksytty myös suuriin projekteihin. Johtotasolla tuetaan täysin ketteriä menetelmiä. Tuloksena ketterät projektit ovat tuottavampia kuin perinteiset ja työmoraali on korkealla.

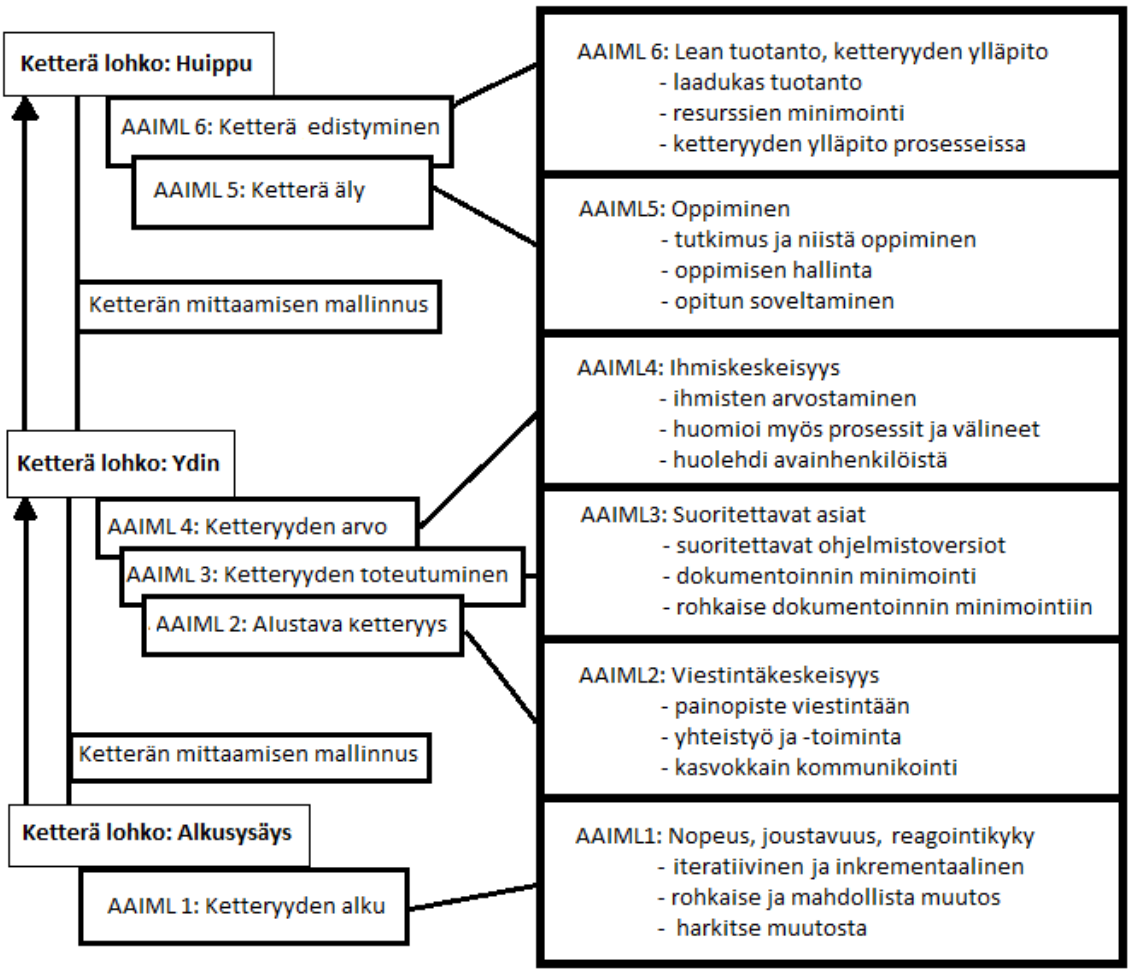
Proulx (2010) mainitsee myös kuudennen tason eli yritystason kypsyyden. Tätä tasoa hän ei kuitenkaan vielä pysty määrittelemään muuten kuin, että tällä tasolla koko organisaatio, niin ihmiset kuin prosessit ja välineet ovat ketterien periaatteiden ja arvojen mukaisia.

Mallista ei ole käytännön kokemuksia eikä validointitietoja

4.10 Qumerin ja Henderson-Sellersin malli

Qumer ja Henderson-Sellers (2008) ovat kehittäneet Agile Adoption and Improvement Model –mallin (AAIM), jonka tarkoituksena on auttaa arvioimaan miten ja kuinka hyvin ketteriä menetelmiä noudatetaan organisaatiossa. Se auttaa myös arvioimaan organisaation tämänhetkisen ketteryyden tasoa. Malli tar-

joaa myös suuntaviivat systemaattisen ketterän kehittämisympäristön luomiseen sekä ketterien käytänteiden käyttöön. Malli koostuu kolmesta lohkosta; alkusysäys (prompt), ydin (crux) ja huippu (apex). Lohkot sisältävät yhteensä kuusi tasoa (kuvio 14).



KUVIO 14 Qumerin ja Henderson-Sellersin (2008, 1908) malli

Ensimmäinen lohko, *alkusysäys*, on alkupiste ketterän kehittämisen aloittamiselle. Lohkossa on vain yksi taso, *ketteryyden alku* (Agile Infancy) (AAIML1). Tällä tasolla ei organisaatiossa vielä käytetä ketteriä menetelmiä, vaan luodaan mahdollisuudet ketterän kehittämisen aloittamiseen. Tämä tapahtuu keskittymällä kolmeen ketterän kehittämisen perusasiaan, nopeuteen, joustavuuteen ja reagointikykyyn.

Toinen lohko, *ydin*, keskittyy ketterien käytäntöjen ja prosessien luomiseen. Se koostuu kolmesta tasosta: alustava ketteryys (Agile Initial) (AAIML2), ketteryyden toteutuminen (Agile Realization) (AAIML3) ja ketteryyden arvo (Agile Value) (AAIML4). Tasolla *alustava ketteryys* keskitytään kommunikoinnin ja yhteistyön mahdollistamiseen. Tasolla *ketteryyden toteutuminen* keskitytään suoritettavien ohjelmistoversioiden tuottamiseen sekä dokumentoinnin minimoimiseen. Kolmen alimman tason saavuttaminen luo perustan neljännen ta-

son saavuttamiseen. Neljännellä tasolla, *ketteryyden arvo*, keskitytään ihmisiin, prosesseja ja työkaluja kuitenkaan unohtamatta. Ihmisiin keskittyminen koskee sekä kyseistä organisaatiota että sen asiakkaita.

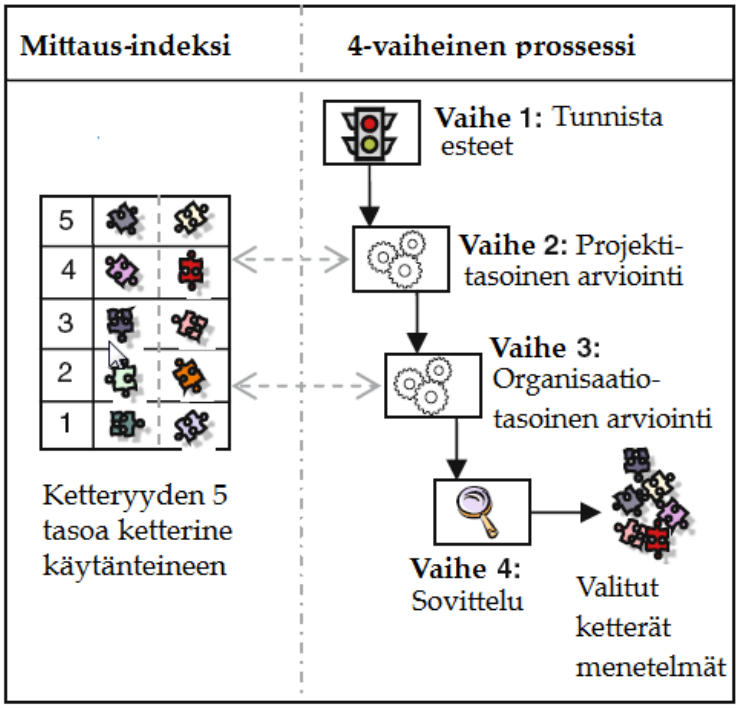
Kolmas lohko, *huippu*, keskittyy oppimiseen, laadunvarmistukseen sekä ketteryyden ylläpitoon. Lohko koostuu kahdesta tasosta, jotka ovat ketterä äly (Agile Smart) (AAIML5) ja ketterä edistyminen (Agile Progress) (AAIML6). Tasolla *ketterä äly* keskitytään oppimiselle edullisen ympäristön luomiseen. Tämä ympäristö koskee niin ihmisiä, prosesseja, tuotteita kuin työkalujakin. Taso luo perustan organisaation oppimiselle ja kehittämiselle. Korkeimmalla tasolla, *ketterä edistyminen*, keskitytään luomaan tuotantoympäristö joka on lean (laadukas tuotanto minimoiduin resurssein mahdollisimman nopeasti) sekä ylläpitämään ketterän kehittämisen käytäntöjä.

Kussakin lohkossa ohjelmistoprosessin ketteryyttä mitataan kvantitatiivisesti käyttämällä ”ketterän mittaamisen mallinnus” -lähestymistapaa (4-Dimensional Analytical Tool, 4-DAT työkalu) (Qumer & Henderson-Sellers, 2006b).

Mallia on testattu kahdella tapaustutkimuksella, kahdessa eri organisaatiossa (Qumer & Henderson-Sellers, 2008). Molemmissa organisaatioissa projektin alussa prosessi otettiin käyttöön käyttämällä AAIML-mallia. Koska työ oli molemmissa organisaatioissa vasta aluillaan, ne olivat vasta AAIM-tasolla 1. Mallia oli kuitenkin tarkoitus seurata edelleen ja molemmilla organisaatioilla oli tavoitteena saavuttaa tasot 2 ja 3. Qumerin ja Henderson-Sellersin (2008) mukaan organisaatiot käyttivät menestyksekkäästi ketteriä käytäntöjä saavuttaakseen halutut liiketoimintatavoitteet. Tapaustutkimukset toivat esille, että on järkevämpää siirtyä ketterien menetelmien käyttöön vaiheittain ja vähitellen, sen sijaan että tehtäisiin siirtyminen nopeasti yhdellä kertaa. Tapaustutkimusten perusteella saatujen kokemusten johdosta oli käynnistetty kolmas tapaustutkimus, mutta siitä ei ole vielä julkaistuja tuloksia.

4.11 Sidkyn, Arthurin ja Bohnerin malli

Sidky, Arthur ja Bohner (2007) ovat esittäneet mallin nimeltään Agile Adoption Framework (AAF) ketterien käytäntöjen käyttöönoton systematisoimiseksi. Malli koostuu kahdesta osasta, mittausindeksistä (SAMI, the Sidky Agile Measurement Index), jolla mitataan organisaation ja projektin kyvykkyyttä ottaa käyttöön ketteriä käytäntöjä, sekä nelivaiheisesta prosessista (kuvio 15). Prosessi auttaa määrittelemään, onko organisaatio valmis ottamaan ketterät menetelmät käyttöön sekä ohjaamaan sen ketterää potentiaalia oikeaan suuntaan. Prosessin avulla selvitetään, mitä ketteriä käytänteitä organisaatiossa pystytään käyttämään ja mitä niistä tulisi ottaa käyttöön.

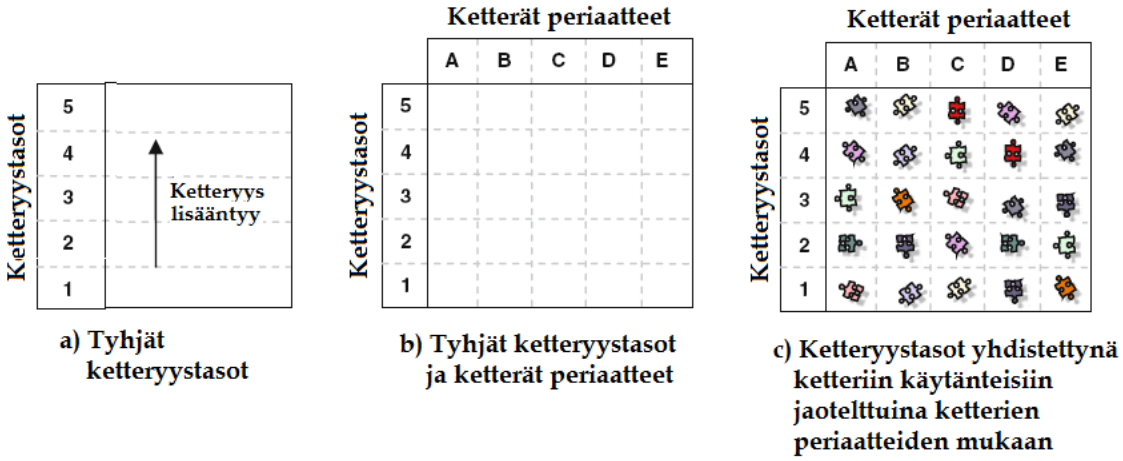


KUVIO 15 Sidkyn ym. (2007, 204) malli

Malli ei ole menetelmäriippuvainen, vaan sitä voidaan käyttää kaikkien ketterien menetelmien yhteydessä. Osien käyttö erikseen ei riitä, vaan tarvitaan molemmat osat. Käyttäminen vaatii, että organisaatiossa on joku asiantuntija, joka tuntee ketterät menetelmät hyvin. Myös ulkopuolista asiantuntijaa voidaan käyttää. Mallia käytetään sekä projekti että organisaatiotasolla (Sidky ym., 2007).

SAMI koostuu neljästä komponentista (kuvio 16) (Sidky ym., 2007):

- *Ketteryystasot*, joille ketterät käytänteet on sijoitettu. Ajatuksena on, että tietyt käytänteet tuovat parannuksia ohjelmistojen kehitysprosessiin (kuvio 16a).
- *Ketterät periaatteet*, eli periaatteet joita täytyy noudattaa, jotta kehittämisprosessi olisi ketterä (kuvio 16b).
- *Ketterät käytänteet ja käsitteet*, eli ne konkreettiset toimet ja käytännön tekniikat, joita käytetään johdonmukaisesti ohjelmistoprojektien kehittämiseen ja hallintaan ketterien periaatteiden mukaisesti (kuvio 16c).
- *Indikaattorit* eli kysymykset, joita arvioija käyttää organisaation tai projektin valmiuksien arviointiin ketterien käytänteiden omaksumiseen. Tällaiset indikaattorit liittyvät ihmisiin, kulttuuriin ja ympäristöön.



KUVIO 16 SAMI:n komponentit (ei indikaattoreita) (Sidky ym., 2007, 205)

SAMI:n viisi ketteryystasoa pohjautuvat Agile manifestin (Beck ym., 2001) perusperiaatteisiin. Jokainen taso koostuu kokoelmasta ketteriä käytänteitä, jotka luovat ja ylläpitävät tasolle kuuluvia ketterän laadun tekijöitä. SAMI:n tasot ovat:

- Taso 1: *Yhteistyö* (Collaborative). Tämä taso tarkoittaa viestinnän ja yhteistyön edistämistä kaikkien sidosryhmien välillä. Yhteistyö on ketterän ohjelmistokehityksen perusta
- Taso 2: *Evolutionäärisyys* (Evolutionary). Evolutiivinen kehitys tarkoittaa varhaista ja jatkuvaa ohjelmistojen toimitusta. Tämä on olennaista, koska kaikissa ketterissä menetelmissä varhainen ja jatkuva toimitus on oletuksena.
- Taso 3: *Tehokkuus* (Effective). Tämän tason tavoite on lisätä kehitysprosessin tehokkuutta hyväksymällä käytäntöjä, jotka johtavat korkealaatuiseen ja toimivaan ohjelmistotuotteeseen. Tämä on tarpeen valmistauduttaessa kehitysprosessin seuraavalle tasolle, jossa voidaan vastata jatkuvaan muutokseen vaarantamatta kehitettävää ohjelmistoa.
- Taso 4: *Mukautuvuus* (Adaptive). Tällä tasolla luodaan ketterän laadun perusteet varmistamalla prosessin kyky reagoida muutoksiin. Tällä tasolla on tärkeää eritasoisten palautteiden määrittely ja niihin vastaaminen.
- Taso 5: *Kattavuus* (Encompassing). Ketteryydestä on tullut olennainen osa organisaation kulttuuria ja on tärkeää, että ympäristö kuvastaa ja tukee ohjelmistokehitysprosessiin ketterää luonnetta. Tämä taso keskittyy luomaan kaiken kattavan ympäristön ylläpitämään ja edistämään ketteryyttä koko organisaatiossa.

Agile-manifestissa (Beck ym., 2001) on 12 periaatetta, jotka kuvaavat ketterän kehittämisen luonnetta. Näiden 12 periaatteen huolellisen ryhmittelyn ja yhdistämisen tuloksena on SAMI:a varten määritelty viisi ketterää periaatetta:

1. *Muutoksen omaksuminen tuottaa asiakkaalle lisäarvoa*. Siksi asenne, jonka mukaan muutokset ovat tervetulleita ja haluttuja, pitäisi säilyttää läpi koko ohjelmistojen kehitystyön.
2. *Ohjelmistojen toimittaminen säännöllisesti*. Aikainen ja säännöllinen toimivan ohjelmiston toimitus on ratkaisevaa, koska se tarjoaa asiakkaalle konkreettisen kohteen arvioitavaksi. Tämä palaute on välttämätöntä tulevien iteraatioiden suunnittelulle.
3. *Ihmiskeskeisyys*. Luottamus ihmisiin ja heidän väliseen vuorovaikutukseen on ketterän ohjelmistokehittämisen kulmakivi.
4. *Tekninen osaaminen*. Ketterät kehittäjät ovat sitoutuneet tuottamaan mahdollisimman korkealaatuista koodia, koska korkealaatuinen koodi on välttämätöntä nopeatempoisissa kehitysympäristöissä, kuten ketterissä.
5. *Yhteistyö asiakkaan kanssa*. On välttämätöntä että asiakkaan, kehittäjän ja muiden asianomistajien välillä on runsasta ja säännöllistä vuorovaikutusta. Tämä varmistaa sitä, että kehitettävä tuote täyttää asiakkaan tarpeet.

Kuvion 16c tavoin Sidky ym. (2007) ovat sijoittaneet ketteriä käytäntöjä matriisiin sen mukaan, mihin ketterän kehittämisen periaatteeseen ne pääasiassa liittyvät ja miten kypsää käsitystä ketteryydestä ne edustavat.

Sidkyn ym. (2007) nelivaiheisen prosessin tavoitteena on tehdä jatkamis/lopettamis -päätös (1. Esteiden tunnistaminen) ja tunnistaa käyttöönotettavat ketterät käytänteet (2. Projekti-tasoinen arviointi, 3. Organisaation valmiuden arviointi, 4. Sovittelu).

Prosessin ensimmäisen vaiheen, *esteiden tunnistaminen*, tarkoitus on tarjota arviointiprosessi, jonka avulla voidaan tunnistaa ne avaintekijät, jotka voisivat estää onnistuneen ketterien menetelmien käyttöönoton. Sidky ym. (2007) esittävät kolme tällaista syytä: väärät syyt ketterien menetelmien käyttöönottoon (ketteryys ei tuo lisäarvoa), puutteellinen pääoma (ketterien menetelmien käyttöönottoon ei ole tarpeeksi varoja tai tukea) ja puuttuva johdon tuki (johdon tulee olla sitoutunutta ketteryyteen). Syyt voivat vaihdella organisaatioittain.

Toisen vaiheen, *projekti-tasoinen arviointi*, avulla tunnistetaan korkein mahdollinen ketteryyden taso, jonka projekti voi saavuttaa, eli tavoitetaso. Tavoitetason tunnistaminen toteutetaan käyttäen SAMI:a.

Kolmannessa vaiheessa, *organisaation valmiuden arviointi*, määritellään missä määrin organisaatio on valmis sopeutumaan projektin tavoitetasoon. Myös tämän vaiheen arviointi toteutetaan SAMI:n avulla.

Neljännessä vaiheessa, *sovittelu*, sovitetaan projektin tavoitetaso organisaation valmiustasoon ja määritellään käyttöönotettavat ketterät käytänteet. Jos organisaation valmius on suurempi tai samalla tasolla kuin projektien tavoitetaso, ei sovittelua tarvita. Sovittelu tarvitaan ainoastaan, jos projektien tavoitetaso on korkeammalla kuin organisaation valmiustaso.

Mallia on validoitu esittämällä se 28 ketterän yhteisön jäsenelle. Validointi toteutettiin 90 minuutin mittaisilla istunnoilla, joissa malli ensin esiteltiin, siitä keskusteltiin ja sen jälkeen osallistujat täyttivät kyselylomakkeen. SAMI ja pro-

sessi arvioitiin erikseen. SAMI:n osalta kyselylomakkeessa olivat alueet kattavuudelle, käytännöllisyydelle, tarpeellisuudelle sekä merkityksellisyydelle. Yhteenvetona SAMI:n osalta todettiin, että yli 75 % vastaajista piti SAMI:a kattavana, hyödyllisenä ja tarpeellisena.

Prosessille oli alueet ymmärrettävyys, käytännöllisyys, tarpeellisuus, valmiusaste sekä tehokkuus. Myös prosessin osalta palaute oli positiivista. Suurin osa vastaajista (noin 80 %) oli samaa mieltä (joko täysin tai lähes) kaikkien viiden alueen osalta, eikä yksikään vastaaja ollut täysin eri mieltä minkään alueen osalta.

4.12 Yinin, Figueiredon ja da Silvan malli

Yin, Figueiredo ja da Silva (2011) ovat esittäneet mallin Scrum-menetelmän mukaisen ohjelmistokehityksen kypsyyden arviointiin. Mallin nimenä on Scrum Maturity Model. Sen tarkoituksena on auttaa ja ohjata ohjelmistokehitysorganisaatioita Scrum-menetelmän käytössä ja käyttöönotossa sekä rohkaista niitä kehittämään itse itseään. Erityistä huomiota kiinnitetään asiakkaan rooliin. Malli sopii myös organisaatioille, joille Scrum menetelmänä ei ole vielä tuttu, mutta jotka haluavat ottaa sen vaiheittain käyttöön. Yin ym. (2011) ovat ottaneet vaikutteita malliinsa mm. CMMI:stä (SEI, 2010) sekä AMM:stä (Agile Maturity Model) (Patel & Ramachandran, 2009).

Mallissa on viisi tasoa. Tasot ovat seuraavat:

1. Alustava (initial)
2. Hallittu (managed)
3. Määritelty (defined)
4. Mitattavasti hallittu (quantitatively managed)
5. Optimoiva (optimizing)

Ensimmäisellä tasolla (alustava) organisaatiossa käytetään Scrumia, mutta prosessin kehityksellä ei ole tavoitteita. Organisaatiossa ei ole myöskään määritelty, miten Scrumia tulisi käyttää. Ongelmana ensimmäisellä tasolla on, että projektit eivät pysy aikataulussa eivätkä budjetin rajoissa. Kommunikaatio eri sidosryhmien välillä ei toimi, ja lopputuotteen laatu ei ole sitä mitä haluttiin. Projektit toimivat kukin omalla tavallaan ja mahdollinen onnistuminen on taitavien yksilöiden ansiota.

Toisella tasolla (hallittu) organisaatiolla on jo selkeitä päämääriä ja tavoitteita Scrumin käytön kehittämiselle. Organisaatiossa on yhteiset käytänteet ja ehdotuksia niiden mittaamiseen. Päästäkseen tälle tasolle organisaation täytyy täyttää kaksi päämäärää (Scrumin perushallinta ja vaatimustenhallinta), ja niihin kuuluvat tavoitteet. Taso voidaan saavuttaa vaikka joitakin tavoitteita ei vielä osattaisikaan toteuttaa täysin oikein tai niissä olisi vielä parannettavaa. Yin ym. (2011) kuvaavat tasovaatimuksia lomakkeiden avulla. Lomakkeisiin merkitään päämäärä, tavoitteet, sekä siihen kuuluvat käytänteet ja ehdotetut

mittarit. Kuviossa 17 on esitetty Scrum kypsyysmallin toisen tason päämäärät ja tavoitteet. Organisaatiot, jotka ovat toisella tasolla, kohtaavat vähemmän ongelmia kuin ensimmäisellä tasolla olevat organisaatiot. Toisella tasolla on kuitenkin edelleen kommunikaatio-ongelmia asiakkaiden kanssa, sekä myös ongelmia liittyen aikataulun ja budjetin ylitykseen.

Päämäärä	Tavoite	Käytänteet	Ehdotettu mittari
Scrumin perushallinta	Scrumin roolit ovat käytössä	(…)	(…)
	Scrumin välineet ovat käytössä	(…)	(…)
	Scrumin palaverit ovat käytössä ja niihin osallistutaan	(…)	(…)
	Scrumin prosessin kulkua kunnioitetaan	(…)	(…)
Ohjelmisto-vaatimusten-hallinta	Tuoteomistaja on selkeästi määritelty	(…)	(…)
	Tuotteen kehitysjonoa hallitaan	(…)	(…)
	Sprintin suunnittelupalaverit onnistuvat	(…)	(…)

KUVIO 17 Scrum kypsyysmallin tason 2 päämäärät ja tavoitteet (Yin ym., 2011, 23)

Kolmannella tasolla (määritelty) keskitytään asiakassuhteeseen ja aikataulussa pysymiseen. Kolmannen tason kaksi päämäärää ovat asiakassuhteen hallinta ja iteraatioiden hallinta. Kuviossa 18 esitetään tason 3 päämäärät ja tavoitteet. Tason 3 organisaatioissa jo useat projektit onnistuvat, pysyvät aikataulussa ja budjetissa sekä toimittavat halutun tuotteen, mutta johtaminen ei ole vielä samanlaista kaikissa projekteissa.

Päämäärä	Tavoite	Käytänteet	Ehdotettu mittari
Asiakas-suhteen hallinta	”Valmiin” määritelmä on olemassa	(…)	(…)
	Tuoteomistaja on saatavilla	(…)	(…)
	Sprintin katselmointipalaverit	(…)	(…)
Iteraatioiden hallinta	Sprintin kehitysjonoa hallitaan	(…)	(…)
	Iteraatiot ovat suunniteltuja	(…)	(…)
	Vauhtia (velocity) mitataan	(…)	(…)

KUVIO 18 Scrum kypsyysmallin tason 3 päämäärät ja tavoitteet (Yin ym., 2011, 24)

Neljännen tason (mitattavasti hallittu) organisaatiolla on standardoitu ja säännöllinen kehitysprosessi, jota johto tukee ja jonka suorituskykyä mitataan, analysoidaan ja kehitetään. Suurin osa projekteista onnistuu, mutta prosessi ei ole vielä täysin optimoitu. Tason 4 päämäärät ja tavoitteet on esitetty kuviossa 19.

Päämäärä	Tavoite	Käytänteet	Ehdotettu mittari
Standardoitu projektin-hallinta	Kvantitatiivinen projektinhallinta	(…)	(…)
Prosessin suorittukyvyn hallinta	Mittaaminen ja analysointi	(…)	(…)

KUVIO 19 Scrum kypsyysmallin tason 4 päämäärät ja tavoitteet (Yin ym., 2011, 24)

Viidennen tason (optimoiva) organisaatiot ovat huippuja Scrum menetelmän käytössä. Ne keskittyvät jatkuvaan itsensä kehittämiseen, saavuttaakseen yhä paremman asiakastyytyväisyyden ja paremman tyytyväisyyden jokaisen sidosryhmän kannalta. Tasolla viisi on vain yksi päämäärä, johon liittyy neljä tavoitetta (kuvio 20).

Päämäärä	Tavoite	Käytänteet	Ehdotettu mittari
Suoritus-	Onnistunut päivittäinen Scrum-palaveri	(…)	(…)
kyvyn	Onnistunut sprintin jälkitarkastelu	(…)	(…)
hallinta	Syiden analysointi ja johtopäätökset	(…)	(…)
	Positiiviset indikaattorit	(…)	(…)

KUVIO 20 Scrum kypsyysmallin tason 5 päämäärät ja tavoitteet (Yin ym., 2011, 25)

Mallin käytettävyyttä, tehokkuutta ja sen tuomaa arvoa on testattu ja validoitu usealla tavalla (Yin ym., 2011). Ensin on haastateltu kahta Scrumin, ketteryyden ja CMMI:n asiantuntijaa ja sen jälkeen on tehty kolmessa yrityksessä (myöhemmin yritykset X, Y ja Z) yhteensä kuusi arviointia. Asiantuntija 1 piti mallin kolmea ensimmäistä tasoa onnistuneina, mutta vaikka tasojen 4 ja 5 päämäärät ja tavoitteet olivat hyviä, tasot tarvitsisivat yksityiskohtien tarkennusta ja käytäntöjä tasojen toteuttamiseen. Asiantuntija 2 oli malliin tyytyväinen ja antoi arvokasta palautetta mallin jatkokehittämiseen.

Arvioinnit yrityksissä tehtiin kolmessa osassa. Ensin malli esiteltiin lyhyesti ja sen jälkeen osallistujat täyttivät esiarviointi-lomakkeen. Seuraavaksi yritykselle tehtiin tason 2 auditointi ja viimeisenä osallistujat täyttivät arvioinnin jälkeisen arviointi-lomakkeen. Yrityksen X palaute sisälsi seuraavaa: malli antaa hyvän suunnan ja ohjeistuksen Scrumin käyttöönottoon, ja malli tarjoaa päämäärät ja tavoitteet eri tasoille, joten organisaatio voi vähitellen parantaa kypsyystasoaan, vaikkapa yksi päämäärä kerrallaan. Yleisesti ottaen yritys piti mallia merkittävänä ja sen perusteella organisaatiossa tehtiin joitakin muutoksia nykyiseen prosessin.

Yrityksessä Y tehtiin kolme arviointia kolmelle projektille. Projektit olivat melko eritasoisia saavuttaen tasot 1, 1 ja 4. Kaikki projektipäälliköt pitivät mallia hyvänä työkaluna ja he uskoivat mallin mahdollisuuksiin nousta standardiksi. Yhden projektipäällikön mielestä mallin tuella on myös aika helppo nostaa organisaatio tasolle 3, jota hän piti hyväksyttävänä tasona. Yrityksessä Y nähtiin myös tärkeäksi, että on olemassa tällainen malli, jonka avulla Scrum voidaan oikeaoppisesti ottaa käyttöön.

Yrityksessä Z tehtiin kaksi arviointia kahdelle projektille. Projektipäälliköt pitivät mallia mielenkiintoisena ja hyvänä. Mallin etenemisvaiheita pidettiin hyvinä ja mallin uskottiin helpottavan Scrum-menetelmän käyttöönottoa. Haastatteluista ja arvioinneista saadun palautteen perusteella mallia on sanottu kehitettävän edelleen (Yin ym., 2011).

4.13 Yhteenveto

Tässä luvussa esiteltiin yksitoista ketterään ohjelmistokehitykseen tarkoitetuttua kypsyysmallia. Aluksi kerrottiin, miten malleja on etsitty ja valittu tähän esitykseen. Sen jälkeen kutakin mallia kuvattiin erikseen tekijöiden mukaisessa aakkosjärjestyksessä. Tavoitteena oli kuvata malleja siten, että niitä voidaan arvioida ja verrata kuvausten perusteella. Kustakin mallista esitettiin, mitä tarkoitusta varten malli on kehitetty, minkälaisista tasoista se koostuu sekä onko mallia käytetty ja/tai validoitu. Seuraavassa luvussa edellä esitettyjä ketterän kehittämisen kypsyysmalleja vertaillaan toisiinsa.

5 KYPSYYSMALLIEN VERTAILU

Tämän luvun tarkoituksena on vertailla edellisessä luvussa esiteltyjä ketterän ohjelmistokehityksen kypsyysmalleja. Aluksi kerrotaan aiemmista tutkimuksista, joissa on tarkasteltu kypsyysmalleja. Samassa yhteydessä määritetään kriteerit, joiden mukaisesti tässä tutkielmassa kypsyysmalleja vertaillaan. Sen jälkeen kerrotaan vertailun tuloksista määriteltyjen kriteerien mukaisessa järjestyksessä. Luvun lopussa esitetään yhteenveto ja johtopäätöksiä vertailusta.

5.1 Aiempia tutkimuksia ja vertailukriteerit

Ketterän ohjelmistokehityksen kypsyysmallien arviointia ja vertailua ei ole kovin paljon aiemmin tehty. Kirjallisuudesta löytyy neljä tutkimusta, jotka jollakin tavalla liittyvät tähän aihealueeseen: Ozcan-Top & Demirörs (2013), Leppänen (2013), Schweigert ym. (2013a) ja Schweigert ym. (2013b). Ozcan-Top ja Demirörs (2013) pyrkivät selvittämään, kuinka riittäviä (sufficient) kypsyysmallit ovat tarjoamaan näkemyksiä organisaation ketterästä kypsyydestä ja mikä ovat ketterien kypsyysmallien vahvuuksia ja heikkouksia. He tutkivat asiaa käsitteellisesti ja empiirisesti. Tutkimuksessa käytettiin seuraavia arviointikriteerejä: sopivuus tarkoitukseen, kattavuus (completeness), ketteryyden tasojen määrittely, objektiivisuus, oikeellisuus (correctness) ja johdonmukaisuus. Arvioinnin kohteeksi valittiin viisi kypsyysmallia: Ambler (2009), Benefield (2010), Patel & Ramachandran (2009), Sidky (2007) ja Yin ym. (2011).

Leppänen (2013) käyttää kuutta kriteeriä kahdeksan kypsyysmallin vertailuun. Kriteerit ovat: kohdealue (domain), tarkoitus, käsitteellinen ja teoreettinen perusta, lähestymistapa ja periaatteet, rakenne sekä käyttö ja validointi. Vertailtavina olivat seuraavat mallit: Ambler (2010), Lui & Chan (2005), Nawrocki ym. (2001), Packlick (2007), Patel & Ramachandran (2009), Qumer & Henderson-Sellers (2008) ja Sidky ym. (2007). Schweigert ym. (2013a) ja Schweigert ym. (2013b) vertailevat suurta joukkoa kypsyysmalleja tasojen vastaavuuksien ja nimien perusteella pitäen lähtökohtana CMMI:n (SEI, 2010) tasojäsennystä. Ta-

soihin perustumattomien mallien osalta he nostavat esille muiden muassa avainominaisuuksia, skaalautuvuustekijöitä, suosituksia ja johtamisperiaatteita. Esityksessä tutkitaan myös valittujen neljäntoista ketterän kypsyysmallin vertautuvuutta SPICE malliin (SPICE – Software Process Improvement and Capability dEtermination, (Paulk & Konrad, 1994)). Esitykset on luonteeltaan alustavia ja heikosti jäsenneltyjä.

Tässä tutkimuksessa vertaillaan ensin yleisesti kaikkia luvussa 4 esiteltyjä, ketterään kehittämiseen tarkoitettuja, kypsyysmalleja ja sen jälkeen tehdään menetelmäkohtaiset vertailut XP-menetelmään liittyvistä sekä Scrum-menetelmään liittyvistä kypsyysmalleista. Yleisvertailussa käytetään seuraavia vertailukriteereitä:

- *menetelmäsidonnaisuus:* minkä menetelmän (XP, Scrum, ketterät menetelmät yleensä) yhteydessä mallia on tarkoitus käyttää
- *kohdealue* (domain): onko malli tarkoitettu organisaation, projektin vai tiimin arviointiin
- *käyttötarkoitus*: mallin käyttötarkoitus (esim. arviointiin, kehittämiseen, suunnitteluun), (vrt. Leppänen, 2013)
- *rakenne*: mallien tasot, vaiheet tai muunlainen jäsennys
- *esitys*: mallin selkeys, esimerkiksi sisältääkö mallin kuvaus taulukoita tai kuvioita
- *käyttö*: miten paljon aikaa ja asiantuntemusta kypsyysmallin soveltamisen voidaan arvioida edellyttävän
- *testaus ja validointi*: onko mallia käytetty ja miten sen toimivuutta on tutkittu?

5.2 Yleisvertailu

Yleisvertailussa kaikkia valittuja ketterän kehittämisen kypsyysmalleja vertaillaan edellä esitettyjen kriteerien mukaisesti. Vertailu tapahtuu edellä esitetyssä järjestyksessä, alkaen menetelmäsidonnaisuudesta ja päättyen testauksen ja validoinnin arvioimiseen.

Menetelmäsidonnaisuus
Valitusta yhdestätoista ketterästä kypsyysmallista seitsemän mallia on menetelmäriippumattomia (Benefield, 2010; Ambler, 2010; Packlick, 2007; Patel & Ramachandran, 2009; Pettit, 2006; Qumer & Henderson-Sellers, 2008; Sidky ym., 2007) eli niitä voidaan käyttää kaikkien ketterien menetelmien yhteydessä. Kaksi mallia (Proulx, 2010; Yin ym., 2011) on tarkoitettu ainoastaan Scrum-menetelmän yhteyteen ja kaksi mallia (Lui & Chan, 2005; Nawrocki ym., 2001) on tarkoitettu ainoastaan XP-menetelmän yhteydessä käytettäväksi.

Kohdealue

Seuraavaksi mallien kohdealuetta analysoidaan kolmiportaisella luokittelulla, organisaatio, projekti, tiimi, perustuen siihen, minkä organisaatiotason arviointiin malli on tarkoitettu. Kaksi malleista (Ambler, 2010; Patel & Ramachandran, 2009) on suunnattu organisaatiotasoiseen arviointiin ja kolmea mallia (Pettit, 2006; Qumer & Henderson-Sellers, 2008; Sidky ym., 2007) voidaan organisaatiotason lisäksi käyttää myös projektitasolla. Neljää mallia (Benefield, 2010; Nawrocki ym., 2001; Proulx, 2010; Yin ym., 2011) voidaan käyttää niin organisaatio-, projekti- kuin tiimitasollakin. Kaksi malleista (Lui & Chan, 2005; Packlick, 2007) on tarkoitettu ainoastaan tiimien toiminnan ketteryyden arviointiin. Edellä oleva jaottelu perustuu siihen, onko mallissa on annettu ohjeita tai määritelty käytänteitä organisaatio-, projekti- tai tiimitasolle. Osa malleista (esim. Benefield, 2010) oli suunnattu organisaatiotasolle, mutta koska ne sisältävät ohjeistusta myös esimerkiksi tiimeille, voidaan malleja näin ollen käyttää myös tiimitasolla. Luin ja Chanin (2005) malli on tarkoitettu pienille, kehittymättömille tiimeille. Muissa malleissa ei ole esitetty rajoituksia liittyen kohderyhmän kokoon tai sen kehitysasteeseen. Mallien menetelmäsidonnaisuudet ja kohdealueet on koottu taulukkoon 5.

Käyttötarkoitus

Seuraavaksi tutkitaan, mihin tarkoitukseen kukin kypsyysmalli on kehitetty. Ambler (2010) määrittelee mallinsa käyttötarkoituksen seuraavasti: ”Malli tarjoaa ohjeistuksen ketterän ohjelmistokehityksen tehokkuuden parantamiseen”. Benefieldin (2010) mallin tavoitteena on nopeuttaa ketterän kehittämisen käyttöönottoa ja paljastaa sen riskialueita. Luin ja Chanin (2005) malli esittää vaiheistuksen XP-menetelmän käyttöönottoon. Nawrocki ym. (2001) esittävät yksinkertaisen kypsyysmallin XP-menetelmän käyttöönotolle. Packlick (2007) pitää malliaan ajattelutapana, joka korostaa tavoitteita ennemmin kuin käytäntöjä sekä ihmisiä ja vuorovaikutusta enemmän kuin prosessia. Mallin tarkoituksena on kehittää ketterien menetelmien käyttöä organisaatiossa (Packlick, 2007). Patel ja Ramachandran (2009, 6) määrittelevät mallinsa käyttötarkoituksen seuraavasti: ”Malli on suunniteltu parantamaan ja tehostamaan ketterän ohjelmistokehityksen menetelmiä sekä edistämään ketterän kehittämisen tavoitteiden saavuttamista, kuten halvempaa hintaa, parempaa asiakastyytyväisyyttä ja korkeampaa ohjelmistojen laatua”. Pettitin (2006) mallin avulla voidaan nopeasti arvioida nykyistä prosessia ja asettaa tavoitteita sen parantamiseksi. Mallin avulla organisaatio voi myös tunnistaa omia vahvuuksiaan ja heikkouksiaan ketterän kehittämisen alueella. Proulxin (2010) malli on tarkoitettu kypsyyden arviointiin Scrum-menetelmää käyttävässä organisaatiossa. Qumer ja Henderson-Sellers (2008, 1909) määrittelevät mallinsa käyttötarkoituksen seuraavasti: ”Malli ohjaa ohjelmistokehitys- organisaatiota omaksumaan ja parantamaan ketteriä käytäntöjä tietyssä tilanteessa tai projektissa”. Sidky ym. (2007, 215) määrittelevät mallinsa käyttötarkoitukseksi sen, että ”Malli tarjoaa jäsennellyn ja toistettavissa olevan lähestymistavan ohjaamaan ja auttamaan ketterien käytäntöjen omaksumista”. Yinin ym. (2011) mallin tarkoituksena on auttaa ja ohja-

ta ohjelmistokehitys-organisaatioita Scrum-menetelmän käytössä ja käyttöönotossa sekä rohkaista niitä kehittämään itse itseään. Yhteenveto mallien käyttötarkoituksista on esitetty taulukossa 5.

TAULUKKO 5 Kypsyysmallien menetelmäsidonnaisuudet, kohdealueet ja käyttötarkoitukset

Lähde	Menetelmä	Kohdealue	Kohdealueen lisärajoitus	Käyttötarkoitus
Ambler (2010)	Ketterät menetelmät	Organisaatio		Ketterän kehittämisen kypsyysmalli ohjelmistokehityksen tehokkuuden parantamiseen
Benefield (2010)	Ketterät menetelmät	Organisaatio, Projekti, Tiimi		Ketterän kehittämisen käyttöönoton nopeuttaminen ja riskien paljastaminen
Lui & Chan (2005)	XP	Tiimi	Pieni, kehittymätön tiimi	Vaiheistus XP-menetelmän käyttöönottoon
Nawrocki ym. (2001)	XP	Organisaatio, Projekti, Tiimi		Kypsyysmalli XP:n käyttöönotolle
Packlick (2007)	Ketterät menetelmät	Tiimi		Ketterien menetelmien käytön kehittäminen organisaatiossa
Patel & Ramachandran (2009)	Ketterät menetelmät	Organisaatio		Ketterän ohjelmistokehittämisen parantaminen ja tehostaminen
Pettit (2006)	Ketterät menetelmät	Organisaatio, Projekti		Ketterän kehittämisen arviointi ja kehittäminen organisaatiossa
Proulx (2010)	Scrum	Organisaatio, Projekti, Tiimi		Scrumia käyttävän organisaation kypsyyden arviointi
Qumer & Henderson-Sellers (2008)	Ketterät menetelmät	Organisaatio, Projekti		Ketterien käytäntöjen omaksuminen ja niiden parantaminen ohjelmistokehitysorganisaatiossa
Sidky ym. (2007)	Ketterät menetelmät	Organisaatio, Projekti		Ketterien käytäntöjen käyttöönoton systematisoiminen
Yin ym. (2011)	Scrum	Organisaatio, Projekti, Tiimi		Ohjelmistokehitysorganisaation auttaminen ja ohjaaminen Scrum-menetelmän käytössä

Beckerin ym. (2005) mukaan kypsyysmalleilla on kolme käyttötarkoitusta; kuvaileva, ohjaileva ja vertaileva. Kuvailevassa mallissa määritellään organisaation nykytila ennalta määritellyn kriteeristön pohjalta. Ohjailevassa tavassa mallia käytetään toiminnan kehittämiseen. Vertailevassa tavassa pääpaino on vertailussa. Malleista kolme (Ambler, 2010; Patel & Ramachandran, 2009; Pettit, 2006) on enimmäkseen kuvailevia. Viisi mallia (Benefield, 2010; Nawrocki ym.,

2001; Packlick, 2007; Proulx, 2010; Qumer & Henderson-Sellers, 2008) on kuvailevia, mutta myös jonkin verran ohjailevia. Kolme mallia (Lui & Chan, 2005; Sidky ym., 2007; Yin ym., 2011) on ohjailevia ja jonkin verran myös kuvailevia. Kaikkia malleja voidaan käyttää myös vertailussa, mutta se ei ole pääpaino missään mallissa.

Rakenne

Useimmissa esitellyistä malleista on määritelty joko etenemistasoja tai -vaiheita. Tällaisia tasoja tai vaiheita malleissa on neljästä kuuteen. Tasomäärittelyjen sisällöt ja vaatimukset olivat malleissa hyvin erilaisia. Esimerkiksi osassa malleista (Ambler, 2010; Benefield, 2010; Lui & Chan, 2005; Packlick, 2007; Proulx, 2010; Qumer & Henderson-Sellers, 2008; Sidky ym., 2007) ensimmäinen taso sisältää jo ketterän kehittämisen menetelmien käyttämistä. Osassa malleista (Nawrocki ym., 2001; Patel & Ramachandran, 2009) ensimmäinen taso tarkoittaa sitä, että ketteriä menetelmiä ei ole vielä ollenkaan käytössä. Esitellyistä yhdestätoista mallista viidessä (Ambler, 2010; Benefield, 2010; Patel & Ramachandran, 2009; Packlick, 2007; Yin ym., 2011) on määritelty viisi ketteryyden tasoa, yhdessä (Nawrocki ym., 2001) on neljä ketteryyden tasoa, Sidkyn ym. (2007) mallissa on neljä prosessin vaihetta ja mittausindeksi SAMI:ssa (Sidky ym., 2007, 204) 5 tasoa. Qumerin ja Henderson-Sellersin (2008) mallissa on kolme lohkoa, joissa on yhteensä kuusi ketterää tasoa. Proulxin (2010) mallissa puolestaan on kuusi tasoa, joista viisi ensimmäistä esitellään tarkemmin. Luin ja Chanin (2005) mallissa käyttöönotto toteutetaan neljässä vaiheessa. Pettit (2006) ei esitä selkeitä ketteryyden tasoja tai vaiheita, vaan hänen mallissaan valittujen toimintojen tilat arvioidaan 7-tasoisella asteikolla. Mallien tasot/vaiheet on koottu taulukkoon 6. Taulukosta puuttuvat Luin ja Chanin (2005) malli, sillä mallin vaiheita ei ole nimetty, sekä Pettitin (2006) malli, sillä mallissa ei ole tasoja eikä vaiheita.

Esitystapa

Seuraavaksi kerrotaan millä tavalla kypsyysmallit on esitetty. Amblerin (2010) mallista annetaan ainoastaan sanalliset selitykset mallin tasoille. Esityksen tueksi ei esitetä kuvia tai taulukoita. Sanalliset tasokuvaukset on esitetty aika yleisellä tasolla. Benefieldin (2010) mallista on selitetty sanallisesti mallin tasot ja ulottuvuudet. Sen sijaan, tasoja ja ulottuvuuksia ei ole jaettu tarkempiin osiin. Organisaation arviointia varten on käytettävissä taulukko, jonka riveinä ovat ulottuvuudet ja sarakkeina tasot. Näin ollen arviointitaulukko kuvaa myös mallin rakenteen. Taulukkoon voidaan merkitä myös tavoitetaso.

Benefield (2010) ehdottaa, että taulukon lisäksi, sen sisältö voidaan esittää myös ”seitti”-diagrammina. Luin ja Chanin (2005) malli on hyvin yksinkertainen, sillä se on tarkoitettu kokemattomille tiimeille. Mallissa on selkeät neljä tasoa, jolle XP-käytänteet on sijoitettu suoraan. Lui ja Chan (2005) kuvaavat mallin tasot myös taulukkona. Nawrockin ym. (2001) mallissa tasot on sanallisen selityksen lisäksi esitetty kuvana. Jokaiselle tasolle on tarkasti ja selkeästi määritelty avainprosessialueet ja niihin liittyvät XP-käytänteet. Mallin arviointitaulukon avulla voidaan nopeasti määrittää organisaation ketterän kehittämisen taso.

TAULUKKO 6 Ketterien kypsyysmallien tasot

	Ambler (2010)	Benefield (2010)	Nawrocki ym. (2001)	Packlick (2007)	Patel & Ramachandran (2009)	Proulx (2010)	Qumer & Henderson-Sellers (2008)	Sidky ym. (2007)		Yin ym. (2011)
	5 tasoa	5 tasoa	4 tasoa	5 tasoa	5 tasoa	6 tasoa	6 tasoa	5 tasoa	4 vaihetta	5 tasoa
			XP			Scrum		SAMI	Prosessi	Scrum
1	retorinen (rhetorical)	uudet parhaat käytännöt	ei ollenkaan yhteensopiva (not compliant at all)	tietoisuus (awareness)	alustava (initial)	tiimitaso	ketteryyden alku (agile infancy)	yhteistyö (collaborative)	esteiden tunnistaminen	alustava (initial)
2	sertifioitu (certified)	jatkuvat käytänteet komponenttitasolla	alustava (initial)	muutos (transformation)	tutkiva (explored)	osastotaso	alustava ketteryys (agile initial)	evolutionäärisyys (evolutionary)	projektitasoinen arviointi	hallittu (managed)
3	uskottava (plausible)	komponenttien välinen jatkuva integraatio	kehittynyt (advanced)	läpimurto (breakthrough)	määritelty (defined)	liiketoimintataso	ketteryyden toteuttaminen (agile realization)	tehokkus (effective)	organisaation valmiuden arviointi	määritelty (defined)
4	kunnioitettava (respectable)	projekttitasoinen jatkuva integraatio	kypsä (mature)	optimointi (optimizing)	parennettu (improved)	projektitaso	ketteryyden arvo (agile value)	mukautuvuus (adaptive)	sovittelu	mitattavasti hallittu (quantitatively managed)
5	mitattu (measured)	kysyntään juuri ajallaan vastaavat toimitukset		mentorointi (mentoring)	jatkuva (sustained)	johtotaso	ketterä äly (agile smart)	kattavuus (encompassing)		optimoiva (optimizing)
6						yritystaso	ketterä edistymien (agile progress)			

Packlickin (2007) mallin tasoista ja ketteryys-tavoitteista on sanalliset kuvaukset. Arviointia varten on käytettävissä taulukko, jonka riveinä ovat ketteryys-tavoitteet ja sarakkeina tasot. Näin ollen arviointitaulukko kuvaa myös mallin rakenteen. Arviointitaulukon avulla voidaan nopeasti määrittää organisaation ketterän kehittämisen taso. Patelin ja Ramachandranin (2009) mallin tasot on sanallisen selityksen lisäksi esitetty kuvana. Ylemmät tasot (tasot 2-5) on lisäksi esitetty taulukon-omaisin kuvin, joihin on merkitty myös avainprosessialueet ja niihin kuuluvat kysymykset. Avainprosessialueisiin kuuluvat kysymykset helpottavat ja selkeyttävät mallin käyttöä. Edistymisen seurantaan ei ole esitetty erillistä tapaa. Mahdollisesti avainprosessien kysymyksiä voitaisiin jossakin muodossa käyttää edistymisen seurantaan. Pettit (2006) määrittelee ainoastaan karkean kehyksen mallille, joten mallin tarkempi määrittely jää käyttäjän vastuulle. Edistymistä seurataan taulukon avulla. Valitut osa-alueet sijoitetaan kaavioon, jonka avulla voidaan arvioida osa-alueen alkutila, nykyinen tila sekä tavoitetila. Taulukko edistymiselle kuvaa myös mallin karkealla tasolla.

Proulxin (2010) malli sisältää sanalliset kuvaukset mallin tasoista, yleiskuvan mallista sekä tasokohtaiset kuvat. Tasokohtaiset kuvat ovat niin karkealla tasolla, että ne eivät helpota mallin käyttöä. Mallin tasojen kuvaukset on kirjoitettu melko yleisellä tasolla, kuten esimerkiksi ”Scrum-mestari rooli on käytössä”, joten käyttäjän vastuulle jää arvioida tarkemmin, mitä lause oikein tarkoittaa. Qumerin ja Henderson-Sellersin (2008) mallin sanallista kuvausta tuetaan kuvalla, johon mallista on kuvattu sen kolme lohkoa sekä niiden kuusi tasoa. Kutakin tasoa on lisäksi kuvattu muutamalla avainkäsitteellä. Kaiken kaikkiaan malli on aika monimutkainen, eikä tasoja ole selitetty kovin tarkasti. Mallin käyttäminen vaatii myös 4-DAT työkalun käyttöönoton (Qumer & Henderson-Sellers, 2006b). Edistymiselle ei mallissa ole esitetty kuvaustapaa.

Sidkyn ym. (2007) malli on monimutkainen, kahdesta eri osasta (SAMI ja prosessi), koostuva malli. Näitä osia tulee käyttää yhdessä. SAMI:sta on kuvattu sekä kuvina/taulukoina että sanallisessa muodossa tasot, ketterät periaatteet ja näiden yhdistelmille, indikaattorit. Indikaattorit selkeyttävät mallin käyttöä. Myös prosessista on sekä sanallinen että graafinen kuvaus. Sidky ym. (2007) esittelee myös esimerkinomaisesti osan taulukosta, johon arvioinnin tuloksia voidaan kirjata. Yinin ym. (2011) mallia kuvataan sanallisen selityksen lisäksi prosessikuvalla sekä tasokohtaisilla taulukoilla (tasot 2-5). Malli on selkeärakenteinen, mutta käytäntöjen ja metriikoiden määrittelyjen puuttuminen tuo haasteita mallin käyttämiseen. Tasokohtaisia taulukoita voidaan käyttää edistymisen seurantaan.

Taulukossa 7 on esitetty yhteenveto kypsyysmallien esitysmuodoista. Siitä käy kunkin mallin osalta ilmi, tarjoaako malli ylipäänsä mitään kuvaa, kaaviota tai taulukkoa, onko mallissa esitetty tarkempaa kuvaa, tai taulukkoa mallin osasta ja tarjoaako malli kuvaa tai taulukkoa edistymisen seurantaan.

TAULUKKO 7 Ketterien kypsyysmallien kuvat, aikatarve sekä testaus ja validointi

Lähde	Kuva/kaavio/ taulukko mallista	Tarkempi kuva/taulukko mallin osasta	Kuva/ taulukko edistymiselle	Tarvittava aika ja asiantuntemus	Testaus ja validointi
Ambler (2010)	ei	ei	ei	pieni	ei tietoja
Benefield (2010)	on (1)	ei	on	keski-määräinen	luotu tapaustutkimuksen yhteydessä
Lui & Chan (2005)	on	ei	ei	pieni	testattu pienemmässä mittakaavassa
Nawrocki ym. (2001)	on	ei	ei	pieni	testattu pienemmässä mittakaavassa
Packlick (2007)	on (1)	ei	on	keski-määräinen	luotu tapaustutkimuksen yhteydessä
Patel & Ramachandran (2009)	on	5 taulukkoa	ei	keski-määräinen	testattu pienemmässä mittakaavassa, validointitutkimusta tehty
Pettit (2006)	on (1)	ei	on	keski-määräinen	ei tietoja
Proulx (2010)	on	5 kuvaa	ei	pieni	ei tietoja
Qumer & Henderson-Sellers (2008)	on	ei	ei	suuri	testattu tapaustutkimuksella
Sidky ym. (2007)	on	1 taulukko	on	suuri	validoitu
Yin ym. (2011)	on	4 taulukkoa	on (2)	keski-määräinen	testattu tapaustutkimuksella ja validoitu

(1) taulukko edistymiselle kuvaa myös mallin
(2) vaihetaulukoita voidaan käyttää edistymisen seurantaan

Käyttö

Seuraavaksi malleja vertaillaan tarvittavan ajan ja tarvittavan asiantuntemuksen suhteen. Tarvittavalla ajalla tarkoitetaan tässä tapauksessa aikaa, joka kuluu menetelmän opettelemiseen ja käyttämiseen siten, että saavutetaan se, mitä mallin avulla halutaan saavuttaa. Tarvittavalla asiantuntemuksella tarkoitetaan puolestaan sitä tietotaidon määrää, joka tarvitaan menetelmän oppimiseen ja käyttämiseen. Tarkastelu perustuu tutkielman tekijän subjektiivisiin arvioihin.

Amblerin (2010) malli ei sisällä hankalia tai uusia termejä, joten ketterän kehittämisen normitiedot riittävät mallin käyttämiseen. Mallin avulla voidaan määritellä nopeasti organisaation ketterän kehittämisen taso, vaikkei mallin esityksessä olekaan kuvia tai kaavioita tukemassa mallin käyttöä. Benefieldin (2010) mallissa on jonkin verran omia termejä (esimerkiksi keskitetty toimitus,

interlock delivery), ja tasojen ja ulottuvuuksien ymmärtäminen ja sisäistäminen vie keskimääräisesti aikaa. Myös asiantuntemuksen tarve on keskimääräinen, sillä koska malli on luotu tapaustutkimuksen yhteydessä, se on luotu kyseisen yrityksen tarpeisiin ja mallin sopiminen suoraan muihin organisaatioihin ei ole varmaa. On mahdollista, että tarvitaan jonkin verran uusia määrittelyjä tasojen sisältöihin ja/tai ulottuvuuksiin. Kun mallin määrittelyt ovat kohdallaan, mallin arviointitaulukon avulla voidaan arvioida nopeasti organisaation ketterän kehittämisen taso. Luin ja Chanin (2005) malli on hyvin yksinkertainen, ja se on tarkoitettu kokemattomille tiimeille. Sen opettelu ja käyttöönottaminen on helppoa ja nopeaa, sillä se perustuu suoraan XP:n käytänteisiin. Erityistä asiantuntemusta ei tarvita, ja välttämätöntä ei ole edes tuntea XP-käytänteitä, sillä ne tulevat mallin myötä vähitellen tutuiksi. Nawrockin ym. (2001) mallissa XP:n käytänteitä on jaoteltu tarkemmin mallin alueisiin, jolloin yksi käytänne voi olla useammassa alueessa. Mallin käyttämiseen riittää normaalit tiedot XP-menetelmästä ja mallin käyttöönottoon tarvittava aika on lyhyt.

Packlickin (2007) mallissa on jonkin verran omia käsitteitä (esimerkiksi termi läpimurto, breakthrough), joiden ymmärtämiseen ja omaksumiseen tarvitaan aikaa ja asiantuntemusta. Mallin tasojen ja ketterien tavoitteiden ymmärtäminen ja sisäistäminen vie keskimääräisesti aikaa. Myös asiantuntemuksen tarve on keskimääräinen, sillä koska malli on luotu tapaustutkimuksen yhteydessä, se on luotu juuri kyseisen yrityksen tarpeisiin, ja mallin sopiminen suoraan muihin organisaatioihin ei ole itsestään selvää. On mahdollista, että tarvitaan jonkin verran uusia määrittelyjä tasojen sisältöihin ja/tai tavoitteisiin. Kun mallin määrittelyt ovat valmiit, mallin arviointitaulukon avulla voidaan nopeasti määrittää organisaation ketterän kehittämisen taso.

Patelin ja Ramachandranin (2009) mallin tavoitteiden ja avainprosessialueiden termien merkitys ja sisäistäminen vie jonkin verran aikaa ja siihen tarvitaan myös jonkin verran asiantuntemusta. Mallin käyttämistä helpottavat tasojen avainprosessialueisiin liittyvät kysymykset. Pettitin (2006) mallissa jokainen toiminto voidaan organisaatiokohtaisesti vaiheistaa vähiten ketterästä eniten ketterään. Tämä toimintojen määrittely ja vaiheistus vie keskimääräisesti aikaa ja siihen tarvitaan jonkin verran asiantuntemusta, sillä Pettit (2006) määrittelee toiminnot hyvin yleisellä tasolla. Malli sinänsä ei sisällä hankalia termejä tai välineitä. Sen jälkeen kuin määrittelyt on tehty, on organisaation uudelleenarviointi nopeampaa. Proulxin (2010) mallin käyttämiseen riittää normaalit tiedot Scrum-menetelmästä. Esimerkiksi organisaation Scrum-mestari (Scrum-tietämys) yhdessä projektipäällikön (tietämys projekteista ja organisaatiosta) kanssa, pystyy nopeasti arvioimaan mallin avulla esimerkiksi projektin ketterän kehittämisen tilan.

Qumerin ja Henderson-Sellersin (2008) mallin sisäistäminen ja käyttäminen vaatii runsaasti aikaa. Mallissa ei ole selitetty tarkkaan tasojen sisältöä, joten tarvitaan huomattavasti asiantuntemusta ymmärtämään, mitä termeillä ja käytänteillä tarkoitetaan. Esimerkki tällaisesta käytänteestä on mallin tasolta neljä, ”huomioi myös prosessit ja välineet”. Mallin käyttämiseen tarvitaan lisäksi myös 4-DAT työkalu (4-Dimensional Analytical Tool).

Sidkyn ym. (2007) malli on puolestaan hidas omaksua, ja sen käyttämiseen tarvitaan runsaasti asiantuntemusta. Malliin kuuluu kaksi eri osaa, joita tulee käyttää yhdessä. Jo mallin kuvauksessa todetaan, että mallin käyttäminen vaatii, että organisaatiossa on joku asiantuntija, joka tuntee ketterät menetelmät hyvin. Apuna voidaan käyttää myös ulkopuolista asiantuntijaa. Yinin ym. (2011) malli tarjoaa selkeät taulukot, joiden avulla ketterän kehittämisen tilaa voidaan arvioida. Taulukoiden käytänteiden ja mittareiden määrittelyyn kuluu keskimääräisesti aikaa ja siihen myös tarvitaan jonkin verran asiantuntemusta. Yksi mallia arvioinut asiantuntija oli sitä mieltä, että mallin tasot 4 ja 5 tarvitsisivat tarkempia yksityiskohtia ja käytäntöjä tasojen toteuttamiseen.

Arviot kypsyysmallien käytön tarvitsemista ajoista ja asiantuntemuksesta on esitetty taulukossa 7. Arvioita on esitetty kolmiportaisella asteikolla: pieni, keskimääräinen, suuri.

Testaus ja validointi

Lopuksi kerrotaan, missä määrin ja millä tavalla kypsyysmalleja on mahdollisesti testattu ja validoitu. Neljä esitellyistä malleista on, joko tehty tapaustutkimuksen yhteydessä (Benefield, 2010; Packlick 2007) tai niitä on testattu laajan tapaustutkimuksen avulla (Yin ym., 2011; Qumer & Henderson-Sellers, 2008). Kolmen mallin (Patel & Ramachandran, 2009; Lui & Chan, 2005; Nawrocki ym., 2001) käyttöönottoa on testattu pienemmässä mittakaavassa. Neljästä mallista (Sidky ym., 2007; Ambler, 2010; Proulx, 2010; Pettit, 2006) ei ole saatavilla tietoja siitä, että niitä olisi käytetty tai niiden käyttöönottoa testattu. Yinin ym. (2011) mallia on validoitu usealla tavalla. Ensin haastateltiin kahta Scrumin, ketteryyden ja CMMI:n asiantuntijaa ja sen jälkeen tehtiin kolmessa yrityksessä yhteensä kuusi arviointia. Patelin ja Ramachandranin (2009) mallin validointitutkimus oli meneillään vuonna 2009, mutta sen tuloksista ei ole julkaistua tietoa. Sidkyn ym. (2007) mallia on validoitu esittämällä se 28 ketterän yhteisön jäsenelle. SAMI ja prosessi arvioitiin erikseen. Muista malleista ei ole validointitietoja saatavilla. Mallien testaus ja validointitiedot on esitetty taulukossa 7.

5.3 XP-menetelmään liittyvät kypsyysmallit

Osa kypsyysmalleista on tarkoitettu käytettäväksi tietyn ketterän menetelmän yhteydessä. Tässä yhteydessä tarkastellaan malleja (Lui & Chan, 2005; Nawrocki ym., 2001), jotka on tarkoitettu XP-menetelmän yhteydessä käytettäväksi.

Luin ja Chanin (2005) ja Nawrockin ym. (2001) malleissa on neljä tasoa, joille XP-käytänteet on jaettu. Luin ja Chanin (2005) mallissa käytetään XP:n käytänteitä suoraan. Nawrockin ym. (2001) mallissa XP:n käytänteitä on jaoteltu tarkemmin mallin alueisiin, jolloin yksi käytänne voi olla useammassa alueessa ja näin ollen käytänteitä on mallissa enemmän. Esimerkiksi XP-käytänne pariohjelmointi liittyy Nawrockin ym. (2001) mallin käytänteisiin ”Parien/tehtävien kierrätys” (Yleissuunnittelu), ”Kaikki tuotantokoodi on pariohjelmoitua” (Koodaus) ja ”Vain yksi pari integroi kerrallaan” (Koodaus). Nämä

käytänteet liittyvät mallin tasolle 3. Nawrockin ym. (2001) käytänteet voivat liittyä myös useampiin XP-käytänteisiin, esimerkiksi ”Parien/tehtävien kierrätys”, liittyy pariohjelmoinnin lisäksi myös käytänteeseen yhteisomistajuus. Nawrockin ym. (2001) mallissa on lisäksi myös muutamia ketteriä käytänteitä, jotka eivät ole XP-käytänteitä, mutta jotka on mallissa ajateltu kuuluvan kuitenkin ketterään kehittämiseen. Esimerkkinä tällaisesta on käytänne ”Projekti on jaettu iteraatiohin”.

Vaikka molemmat mallit käyttävät XP-käytänteitä, käytänteiden käyttöönottovaiheistus on malleissa osin hyvin erilainen. Osa Nawrockin ym. (2001) mallin toisen tason käytänteistä on Luin ja Chanin (2005) mallissa vasta viimeisellä, eli neljännellä tasolla. Samoin myös yksi Luin ja Chanin (2005) mallin ensimmäisen tason käytänne on Nawrockin ym. (2001) mallissa vasta neljännellä tasolla. Yhtäläisyyksiäkin tosin löytyy, Lui ja Chan (2005) sijoittivat käytänteet ”pariohjelmointi” ja ”yhteisomistajuus” mallinsa tasolle 3, myös Nawrocki ym. (2001) sijoittivat mallinsa vastaavat käytänteet tasolle 3. Taulukossa 8 on vertailtu näiden kahden mallin XP-käytänteiden sijoittelua mallien tasoille. Vertailu on tehty Luin ja Chanin (2005) mallin käyttöönottovaiheisiin perustuen, koska Luin ja Chanin (2005) mallissa on vähemmän XP-käytänteitä. Vertailu on tehty käyttäen värikoodausta. Taulukoihin on esimerkiksi keltaisella merkitty Luin ja Chanin (2005) mallin ensimmäisen tason käytänteet ja Nawrockin ym. (2001) mallin käytänteet, jotka vastaavat Luin ja Chanin (2005) mallin ko. käytänteitä tai liittyvät läheisesti niihin. Osa Nawrockin ym. (2001) mallin käytänteistä ei sovi mihinkään Luin ja Chanin (2005) mallin käytänteeseen, joten nämä käytänteet on jätetty värikoodamatta.

Nawrockin ym. (2001) mallia voidaan käyttää niin organisaatio, projekti kuin tiimitasoisestikin. Luin ja Chanin (2005) malli puolestaan on tarkoitettu ainoastaan tiimitasolle käytettäväksi XP:n käyttöönoton suunnitteluun pienissä tiimeissä, jotka vasta aloittavat ketterää kehittämistä. Mallia toteuttavia ketterän kehittämisen alkuvaiheen tiimien tasoa voidaan arvioida mallin avulla, mutta sen jälkeen kun mallin kaikki käytänteet ovat tiimissä käytössä, malli ei tarjoa enää mittaria tai tasoja käytänteiden käyttämisen kypsyydelle. Myös Nawrockin ym. (2001) malli on kypsyysmalli XP:n käyttöönotolle, eli kun kaikki käytänteet ovat käytössä mallin korkeimmalla tasolla, malli ei tarjoa enää mittaria käytänteiden käytön kypsyydelle. Erona mallien käytänteissä on kuitenkin se, että Nawrockin ym. (2001) mallissa sama käytänne saattaa olla useammalla tasolla, esim. käytänne ”Integroi usein” on sekä tasolla 2 että 3. Tasolla 2 vaatimuksena on, että integrointi tapahtuu kerran viikossa, tasolla 3 puolestaan edellytetään, että integrointi tapahtuu vähintään kerran päivässä. Nawrockin ym. (2001) mallissa on muutenkin selitetty tarkemmin, mitä käytänteen noudattaminen ja saavuttaminen tarkoittaa.

TAULUKKO 8 Luin & Chanin (2005) malli vs. Nawrockin ym. (2001) malli

Lui & Chan (2005)

1	testaus, yksinkertainen suunnittelu, refaktorointi, koodaussäännöt
2	jatkuva integraatio
3	pariohjelmointi, yhteisomistajuus
4	metafora, 40 h viikko, pienet julkaisut, läsnäoleva asiakas, suunnittelupeli

Nawrocki ym. (2001)

Taso 2		**Taso 3**	**Taso 4**
P0. Suunnittelupelin käyttö projektien suunnitteluun	C2. Ensin koodataan yksikkötestit	P8. Parien/tehtävien kierrätys	C0. Asiakas kehitystiimin tiloissa
P2. Käyttäjätarinoiden kirjoittaminen	C5a. Integrointi usein	C0b. Asiakas käy säännöllisesti kehitystiimin luona	C8. Ei ylitöitä
P3. Suunnitelma luo aikataulun	C7. Optimoinnin jättäminen viimeiseksi	C1. Koodi kirjoitetaan sovittujen sääntöjen mukaan	T1. Kaiken koodin täytyy läpäistä yksikkötestit
P4. Säännölliset, pienet julkaisut	T0. Kaikella koodilla on yksikkötestit	C3. Kaikki tuotantokoodi on pariohjelmoitua	
P5. Projektin käytettyä aikaa (velocity) mitataan	T1a. Kaiken koodin täytyy läpäistä yksikkötestit	C4. Vain yksi pari integroi kerrallaan	
P6. Projekti on jaettu iteraatiohin	T2. Löytyneestä viasta täytyy luoda testitapaus	C5b. Integrointi usein	
P7. Jokainen iteraatio alkaa suunnittelulla	T3. Hyväksyntätestejä ajetaan usein ja niiden tulokset julkaistaan	C6. Koodin yhteisomistajuus	
D1. Järjestelmän metaforan valinta		C8b. Ylityötiedot kerätään ja julkaistaan	
D3. Julkaisu aikaisin riskien minimoimiseksi		C9. Versionhallinta on käytössä	
D4. Toimintoja ei saa lisätä ilman vaatimuksia		T4. Automatisoituja testejä käytetään tukemaan säännöllisiä integraatiotestejä	
C0a. Tehokas yhteistyö asiakkaan kanssa		F0. Tiimillä on avoin työtila	

5.4 Scrum-menetelmään liittyvät kypsyysmallit

Tässä yhteydessä pyritään vertaaman malleja (Proulx, 2010; Yin ym., 2011), jotka on tarkoitettu Scrum-menetelmän käytön yhteyteen.

Yin ym. (2011) ja Proulxin (2010) mallit ovat hyvin erilaisia, ja niitä on hankala verrata keskenään. Yinin ym. (2011) mallin tasojen ja niihin liittyvien lomakkeiden avulla organisaatio voi määritellä tarkasti selkeän etenemistavoitteen ja/tai arvion Scrumin käytön tilasta. Mallissa käytetään Scrumin perustermejä kuten esimerkiksi tuoteomistaja ja sprintin arviointipalaveri. Proulxin (2010) malli puolestaan antaa kuvan Scrum-menetelmän käyttöönoton laajenemisesta organisaatiossa. Scrumin periaatteiden ja käytäntöjen osalta malli pysyy hyvin yleisellä tasolla, käyttäen esimerkiksi termejä Scrum-käytännöt, Scrum-roolitus ja dokumentointitaso. Molempia malleja voidaan käyttää ketterän kehittämisen tason arvioinnissa organisaatiossa monella tasolla, kuten esimerkiksi tiimikohtaisesti, projektikohtaisesti ja organisaatiokohtaisesti. Yinin ym. (2011) mallin käyttämiseen tarvitaan enemmän aikaa ja asiantuntemusta kuin Proulxin (2010) mallin käyttämiseen. Yinin ym. (2011) malli tarjoaa selkeämmät tasot ja tarkemmat tasokuvaukset kuin Proulxin (2010) malli.

5.5 Yhteenveto vertailusta ja johtopäätöksiä

Tässä luvussa raportoitiin ketterän ohjelmistokehityksen kypsyysmallien vertailusta. Aluksi kerrottiin aiemmista tutkimuksista, joissa on tarkasteltu kypsyysmalleja. Sen jälkeen määriteltiin kriteerit, joiden mukaisesti tämän tutkimuksen vertailu suoritettiin. Ketterään kehittämiseen esitettyjä kypsyysmalleja vertailtiin kahdella tavalla, ensin yleisesti kaikkia malleja yhdessä ja sen jälkeen tehtiin menetelmäkohtaiset vertailut, ensin XP-menetelmään liittyville malleille ja sitten Scrum-menetelmään liittyville malleille. Yleisvertailussa malleja vertailtiin seitsemän vertailukriteerin mukaisesti. Vertailukriteereinä käytettiin menetelmäsidonnaisuutta, kohdealuetta, käyttötarkoitusta, rakennetta, esitystapaa, käyttöä sekä testausta ja validointia.

Mallien menetelmäsidonnaisuudesta voidaan todeta, että malleista suurin osa (7/11), oli menetelmäriippumattomia ja niitä voidaan käyttää kaikkien ketterien menetelmien yhteydessä. XP-menetelmän yhteydessä käytettäviä malleja oli kaksi (Lui & Chan, 2005; Nawrocki ym., 2001) ja Scrum-menetelmän yhteydessä käytettäviä malleja myös kaksi (Yin ym., 2011; Proulx, 2010). Kohdealuekohtaisessa vertailussa tutkittiin, onko malli tarkoitettu käytettäväksi tiimi-, projekti- vai organisaatiotasolla. Analyysissa todettiin, että suurinta osaa malleista voidaan käyttää organisaatio- ja projektitasoisen kypsyyden tarkasteluun. Pari mallia (Lui & Chan, 2005; Packlick, 2007) on kehitetty käytettäväksi vain tiimitasolla.

Seuraavaksi tutkittiin mallien käyttötarkoitusta sen mukaisesti kuin mallin tekijät ovat sitä kuvanneet. Suurin osa malleista oli tarkoitettu ketterien mene-

telmien käytön kehittämiseen organisaatiossa. Käyttötarkoitusta verrattiin myös Beckerin ym. (2005) jaottelutavan mukaisesti, ryhmitellen mallit joko kuvailevan, ohjailevan tai vertailevan käyttötarkoituksen mukaisesti. Suurin osa malleista on kuvailevia malleja.

Mallien rakennetta vertailtiin tutkimalla niiden tasoja, vaiheita tai muunlaista jäsennystä. Tasoja tai vaiheita malleissa oli pääosin neljästä kuuteen. Tasomäärittelyjen sisällöt ja vaatimukset olivat malleissa hyvin erilaisia. Mallien esitystapaa puolestaan vertailtiin tutkimalla niiden sisältämiä kuvia ja taulukoita, sekä arvioimalla mallin kuvauksen selkeyttä. Mallit (esim. Yin ym., 2011), joissa kattavia sanallisia kuvauksia tuettiin kuvin ja taulukoin, olivat helpommin ymmärrettävissä. Kaikissa malleissa kuvat eivät kuitenkaan tuoneet lisäarvoa (vrt. Proulx, (2010)). Yleensä ottaen selkeä ja vaiheittainen kuvaus helpottaa ja nopeuttaa mallin käyttöönottoa. Seuraavaksi selviteltiin mallien käyttöönoton vaatimaa aikaa ja asiantuntemusta. Tätä arvioitiin subjektiivisesti kolmiportaisella asteikolla: pieni, keskimääräinen, suuri. Ajan ja asiantuntemuksen tarpeeseen vaikuttavat eniten mallin monimutkaisuus, laajuus, selkeys sekä käytetty termistö. Viimeisenä vertailtiin mallien testaus- ja validointitapoja. Muutamaa malleista (Benefield, 2010; Packlick, 2007; Qumer & Henderson-Sellers, 2008; Yin ym., 2011) oli testattu ja validoitu hyvinkin laajasti, kolmea mallia (Ambler, 2010; Pettit, 2006; Proulx, 2010) puolestaan ei oltu testattu tai validoitu lainkaan. Taulukoissa 5-7 esitettiin yhteenvetoja vertailuista, ja niistä saakin nopean yleiskuvan mallien yhtäläisistä ja eroavista piirteistä vertailukriteerien mukaisesti.

Menetelmäkohtaisessa vertailussa vertailtiin ensin kahta XP-menetelmään liittyvää mallia (Lui & Chan, 2005; Nawrocki ym., 2001). Vaikka molemmat mallit perustuvat XP käytänteisiin, oli niiden käyttöönottovaiheistus melko erilainen. Scrum-menetelmään liittyvät kaksi mallia kaksi (Yin ym., 2011; Proulx, 2010) puolestaan ovat jo lähestymistavaltaan hyvin erilaisia. Toinen malleista kuvasi enemmänkin ketterän kehittämisen menetelmien käytön laajenemista organisaatiossa (Proulx, 2010) ja toinen (Yin ym., 2011) taas keskittyy organisaation ketterän kehittämisen tason arviointiin organisaatiossa.

Kuten edellisestä voi päätellä, vertaillut mallit ovat monella tapaa erilaisia. Malleja ei voi asettaa tärkeys tai paremmuusjärjestykseen, vaan mallin sopivuus riippuu tilanteesta, organisaatiosta sekä siitä, mitä mallin avulla halutaan saavuttaa. Esimerkiksi tilanteessa, jossa *organisaatio haluaa selvittää ohjelmointitiimien ketteryyden tason*, voidaan arviointiin käyttää malleja, jotka on tarkoitettu tiimien ketteryyden arviointiin (esim. Lui & Chan, 2005; Nawrocki ym., 2001; Packlick, 2007; Proulx, 2010; Yin ym., 2011). Tiimin käyttämän ketterän menetelmän mukaan voidaan sopivien mallien määrää rajata tarvittaessa lisää. Vaikka tiimi käyttäisi vain yhtä ketterää menetelmää, esim. Scrumia, voidaan arviointiin käyttää menetelmää joka sopii kaikille ketterille menetelmille. Jos tiimit ovat vasta ketterän kehittämisen alkutaipaleella, sopii niille erilainen malli (esim. Lui & Chan, 2005) kuin tiimille, joka on jo kauan käyttäneet ketteriä menetelmiä ja haluavat lähinnä kehittää ketterien menetelmien käyttöä (esim. Packlick, 2007).

Jos taas halutaan *selvittää organisaation ketteryyden taso*, tulisi valita malli, joka soveltuu organisaatiotasoisen ketteryyden arviointiin (esim. Ambler, 2010; Benefield, 2010; Nawrocki ym., 2001; Patel & Ramachandran, 2009; Pettit, 2006; Proulx, 2010; Qumer & Henderson-Sellers, 2008; Sidky ym., 2007). Organisaatiossa käytetty ketterä menetelmä vaikuttaa myös mallin valintaan, jos organisaatiossa käytetään useita ketteriä menetelmiä, ei esim. XP-menetelmän arviointiin kehitetyt mallit (esim. Nawrocki ym., 2001) sovi, vaan olisi parempi valita malli, joka soveltuu ketterille menetelmille yleensä (esim. Ambler, 2010; Benefield, 2010; Patel & Ramachandran, 2009; Pettit, 2006; Qumer & Henderson-Sellers, 2008; Sidky ym., 2007). Valintaan voi vaikuttaa myös organisaation koko ja käytettävissä olevat resurssit. Pienen organisaation tuskin kannattaa valita kovin laajaa ja monimutkaista mallia (esim. Qumer & Henderson-Sellers, 2008; Sidky ym., 2007). Suuri organisaatio, jolla on runsaasti resursseja käytössään ja jolle on tärkeää, että mallia on käytetty ja testattu aikaisemmin (esim. Qumer & Henderson-Sellers, 2008; Sidky ym., 2007) tekee valintansa sen mukaisesti.

Organisaation, joka haluaa *suunnitella ketterien käytäntöjen käyttöönottoa organisaatiossa*, kannattaa valita malli, joka on kehitetty tähän tarkoitukseen (esim. Benefield, 2010; Nawrocki, 2001; Qumer & Henderson-Sellers, 2008; Yin ym., 2011). Kun organisaatio vasta suunnittelee ketterien käytäntöjen käyttöönottoa, sille voi olla tärkeää valita malli, josta on jo olemassa käyttökokemuksia tai testaustietoja (esim. Benefield, 2010; Qumer & Henderson-Sellers, 2008). Myös valittu ketterä menetelmä voi vaikuttaa mallin valintaan, esim. jos organisaatio suunnittelee Scrum-menetelmän käyttöönottoa, mallin tulisi tukea Scrum menetelmän käyttöönottoa (esim. Benefield, 2010; Qumer & Henderson-Sellers, 2008; Yin ym., 2011).

6 YHTEENVETO

Tämän tutkielman tavoitteena oli kuvata kirjallisuudessa esitettyjä ketterän ohjelmistokehityksen kypsyysmalleja ja verrata niitä toisiinsa. Työn tutkimusongelma muotoiltiin seuraavasti: *Millaisia yhtäläisiä ja erilaisia piirteitä ketterään ohjelmistokehitykseen tarkoitetuilla kypsyysmalleilla on*? Tutkimusongelmaa tarkennettiin kolmella tutkimuskysymyksellä, jotka olivat:

- Mitä tarkoitetaan ketterällä ohjelmistokehityksellä ja millaisia ketteriä menetelmiä on olemassa?
- Mitä tarkoitetaan kypsyysmallilla ja millaisia kypsyysmalleja on olemassa?
- Millaisia kypsyysmalleja ketterään ohjelmistokehitykseen on esitetty ja miten ne vertautuvat toisiinsa?

Ensimmäisellä tutkimuskysymyksellä tähdättiin yleiskuvan muodostamiseen ketterästä ohjelmistokehityksestä ja sen menetelmistä siten, että kypsyysmallien tarkasteluun saataisiin hyvä käsitteellinen perusta. Tutkielmassa kerrottiin ensin yleisesti ketterästä lähestymistavasta, sen arvoista ja periaatteista. Ketterän ohjelmistokehityksen yhteiset perusarvot ja periaatteet on määritelty Agile-manifestissa (Beck ym., 2001). Ketterillä menetelmillä tarkoitetaan keveitä, joustavia ja nopeasti muutoksiin reagoivia ohjelmistokehitysmenetelmiä. Niissä kehitysprosessi toteutetaan lyhyinä iteratiivisina ja inkrementaalisina sykleinä, jolla ohjelmistoa kasvatetaan vähitellen (Boehm, 2002). Samalla esiteltiin myös ketteryyttä koskevia erilaisia käsityksiä sekä ketterästä ohjelmistokehityksestä koettuja hyötyjä ja ongelmia. Ketterät menetelmät on todettu helposti omaksuttaviksi ja toimiviksi (Bustard ym., 2013; Rodriguez ym., 2012). Tutkimuksissa on todettu, että ketteriä menetelmiä käyttämällä asiakastyytyväisyys on lisääntynyt, prosessi on kehittynyt sekä lopputuotteen laatu on parantunut (Boehm & Turner, 2003; Highsmith, 2004; Anderson, 2005). Ketterien menetelmien käyttöönotto edellyttää usein suuria muutoksia. Käyttöönotto voikin olla työlästä, aikaa vievää ja haastavaa, mikä puolestaan voi aiheuttaa muutosvastarintaa kehitystiimin keskuudessa. (Schatz & Abdelshafi, 2005). Ketterän lähestymista-

van jälkeen kuvattiin kahta yleisimmin käytössä olevaa ketterää menetelmää, XP:tä (Beck, 1999) ja Scrumia (Schwaber, 2004). Näitä menetelmiä yhdistää näkemys inkrementaalisesta ja iteratiivisesta prosessista, jolle on ominaista kiinteästi yhdessä työskentelevät tiimit ja asiakkaan edustajat. XP on painottunut kuvaamaan ohjelmistokehityksen käytänteitä, kun taas Scrumin painopiste on työnhallinnassa tiimi- ja projektitasolla.

Toisen tutkimuskysymyksen tarkoituksena oli jatkaa tutkielman käsitteellisen perustan luomista kypsyysmallien osalta. Tutkimuskysymykseen vastaamiseksi kerrottiin aluksi perinteisistä kypsyysmalleista, niiden taustasta, rakenteesta ja periaatteista. Kypsyysmalleilla pyritään arvioimaan jonkin kohteen kypsyyttä (maturity) tai/ja kyvykkyyttä (capability) tietyn kriteeristön pohjalta (Jokela ym., 2006). Tyypillisesti mallit koostuvat kypsyystasoista ja näkökulmista, joiden kautta kohdetta arvioidaan (Jokela ym., 2006). Kahta kypsyysmallia, CMM (Paulk, 2001) ja CMMI (SEI, 2010), kuvattiin hieman tarkemmin. Software Engineering Instituten (SEI) julkaisema CMM (Capability Maturity Model) (Paulk, 2001) on yksi tunnetuimmista kypsyysmalleista. CMM mallin korvasi vuonna 2000 julkaistu CMMI (Capability Maturity Model Integration) (SEI, 2006). CMMI malli koostuu viidestä kypsyystasosta ja 22 prosessialueesta (SEI, 2010). Sen jälkeen kerrottiin, miten perinteiset kypsyysmallit nähdään ketterän kehittämisen näkökulmasta. Perinteisesti CMMI ja ketterät menetelmät on nähty lähes vastakkaisina lähestymistapoina, mutta monet tutkijat ovat sitä mieltä, että niitä voisi käyttää yhtä aikaa (Boehm & Turner 2003; Paulk, 2001; Kähkönen & Abrahamsson, 2004). Ongelmana on kuitenkin se, että perinteisten kypsyysmallien käyttö on turhan raskasta ketterän ohjelmistokehityksen yhteydessä.

Kolmatta tutkimuskysymystä lähestyttiin esittämällä ensin, miten malleja on etsitty ja valittu tähän tutkimukseen. Ketterän ohjelmistokehityksen kypsyysmalleja koskevia tutkimuksia etsittiin tutkimuskannoista (IEEExplore ja ACM Digital Library) sekä käyttämällä Google Scholaria. Näin löydettiin neljätoista mallia. Alustavassa tarkastelussa mallijoukko rajattiin yhdeksitoista. Rajauksen perusteina käytettiin sitä, että mallit kuvaavat tarpeeksi laajasti ja kattavasti ketterää ohjelmistokehitystä, kun ajatellaan koko organisaatiota ja sen toimintoja. Rajaus oli tehtävä myös sen takia, että neljäntoista mallin vertaileminen tässä tutkimuksessa olisi ollut liian työlästä. Lisäksi vertailuun haluttiin mukaan myös muutamia malleja, jotka olivat tarkoitettu käytettäväksi ainoastaan tietyn menetelmän yhteydessä. Tämän jälkeen kutakin valittua mallia kuvattiin erikseen siten, että sitä voitiin arvioida ja verrata kuvausten perusteella. Kustakin mallista kerrottiin, mitä tarkoitusta varten malli on kehitetty, minkälaisista tasoista, vaiheista tai ulottuvuuksista se koostuu sekä onko mallia käytetty, testattu tai validoitu.

Valittuja, ketterän ohjelmistokehityksen kypsyysmalleja, vertailtiin kahdella tavalla, ensin yleisesti kaikkia malleja yhdessä ja sen jälkeen menetelmäkohtaisesti, XP-menetelmään liittyviä malleja ja Scrum-menetelmään liittyviä malleja. Yleisvertailussa malleja vertailtiin seitsemän vertailukriteerin mukaisesti. Vertailukriteereinä käytettiin menetelmäsidonnaisuutta, kohdealuetta,

käyttötarkoitusta, rakennetta, esityksen selkeyttä, käyttöä sekä testausta ja validointia.

Neljä malleista on tarkoitettu käytettäväksi vain tietyn menetelmän yhteydessä, kaksi XP:n ja kaksi Scrumin. Muita malleja voidaan käyttää kaikkien ketterien menetelmien yhteydessä. Suurin osa malleista on tarkoitettu organisaatiotasoiseen tarkasteluun kuitenkin niin, että niitä voidaan käyttää myös projekti ja tiimitasolla. Mallien käyttötarkoitusta tutkittiin kahdesta näkökulmasta, ensin sen mukaisesti, kuin mallin tekijät ovat sitä itse kuvanneet. Seuraavaksi malleja arvioitiin sen mukaisesti, olivatko ne enemmän kuvailevia, ohjailevia vai vertailevia. Suurin osa malleista on kuvailevia malleja. Mallien rakennetta vertailtiin tutkimalla niiden tasoja, vaiheita tai ulottuvuuksia. Yleisin tasojen/vaiheiden määrä on viisi. Taso- ja vaihemäärittelyjen sisällöt ja vaatimukset ovat malleissa hyvin erilaiset. Mallien esitystapaa vertailtiin niiden sisältämien kuvien ja taulukoiden pohjalta, sekä arvioimalla mallin kuvaustavan selkeyttä ja termistöä. Mallit, joissa kattavia sanallisia kuvauksia tuettiin kuvin ja taulukoin, olivat helpommin ymmärrettävissä. Seuraavaksi arvioitiin mallien käyttöönoton vaatimaa aikaa ja asiantuntemusta kolmiportaisella asteikolla. Ajan ja asiantuntemuksen tarpeeseen vaikuttavat eniten mallin monimutkaisuus, laajuus, selkeys sekä käytetty termistö. Viimeisenä vertailtiin mallien testaus- ja validointitapoja. Muutamaa malleista oli testattu ja validoitu hyvinkin laajasti, kolmea mallia puolestaan ei oltu testattu tai validoitu ollenkaan.

Menetelmäkohtaisessa vertailussa vertailtiin ensin kahta XP-menetelmään liittyvää mallia. Vaikka nämä molemmat mallit perustuvat XP-käytänteisiin, oli käytänteiden käyttöönottovaiheistus hyvin erilainen. Scrum-menetelmään liittyvät kaksi mallia puolestaan olivat jo lähestymistavaltaan hyvin erilaisia. Toinen keskittyi organisaation ketterän kehittämisen tason arviointiin organisaatiossa ja toinen malleista taas kuvasi enemmänkin ketterän kehittämisen menetelmien käytön laajenemista organisaatiossa.

Johtopäätöksenä kypsyysmallien vertailusta voidaan todeta, että osa malleista on vasta lähinnä ideatasoisia, ja niiden saattaminen tasolle, joka takaisi niiden käytännön hyödyllisyyden, edellyttäisi niiden huomattavaa tarkentamista ja lisää empiiristä tutkimusta. Tämä näkyi selkeimmin mallien tasojen yleisluonteisena kuvauksena sekä validointitietojen ja testaustulosten puuttumisena.

Verrattuna perinteisiä kypsyysmalleja koskevaan tutkimukseen, ketterän ohjelmistokehityksen kypsyysmallien arvioinnista tai vertailusta ei ole tehty vielä kovin montaa tutkimusta. Kirjallisuudesta löytyy neljä tutkimusta, jotka jollakin tavalla liittyvät tähän aihealueeseen: Ozcan-Top & Demirörs (2013), Leppänen (2013), Schweigert ym. (2013a) ja Schweigert ym. (2013b). Ozcan-Top ja Demirörs (2013) pyrkivät selvittämään, kuinka riittäviä kypsyysmallit ovat tarjoamaan näkemyksiä organisaation ketterästä kypsyydestä ja mikä ovat ketterien kypsyysmallien vahvuuksia ja heikkouksia. Viiden kypsyysmallin tutkimuksessa käytettiin arviointikriteereinä sopivuutta tarkoitukseen, kattavuutta, tasojen määrittelyä, objektiivisuutta, oikeellisuutta ja johdonmukaisuutta. Leppänen (2013) käytti kuutta kriteeriä kahdeksan kypsyysmallin vertailuun. Kriteerit ovat: kohdealue, tarkoitus, käsitteellinen ja teoreettinen perusta, lähesty-

mistapa ja periaatteet, rakenne sekä käyttö ja validointi. Schweigert ym. (2013a) ja Schweigert ym. (2013b) vertailevat suurta joukkoa kypsyysmalleja tasojen vastaavuuksien ja nimien perusteella pitäen lähtökohtana CMMI:n (SEI, 2010) tasojäsennystä. Tasoihin perustumattomien mallien osalta he käyttivät kriteereinä muun muassa avainominaisuuksia, skaalautuvuustekijöitä, suosituksia ja johtamisperiaatteita.

Tässä tutkimuksessa käytettiin seitsemää kriteeriä yhdentoista kypsyysmallin vertailuun. Tutkimus tavallaan jatkaa ja laajentaa Leppäsen (2013) sekä Ozcan-Topin ja Demirörsin (2013) tutkimuksia. Leppäsen (2013) ja Ozcan-Topin ja Demirörsin (2013) tutkimuksissa oli kaksi yhteistä mallia, jota molemmat vertailivat. Tässä tutkimuksessa on kaikki Leppäsen (2013) tarkastelemat kypsyysmallit ja neljä Ozcan-Topin ja Demirörsin (2013) tutkimuksen viidestä mallista. Näiden lisäksi tähän tutkimukseen on sisällytetty kaksi muuta mallia, jotka eivät ole mukana Leppäsen (2013) eikä Ozcan-Topin ja Demirörsin (2013) tutkimuksissa. Niitä on käsitelty kyllä Schweigertin ym. (2013a) ja Schweigertin ym. (2013b) tutkimuksissa. Schweigertin ym. (2013a) ja Schweigertin ym. (2013b) tutkimukset ovat kuitenkin hyvin karkealla tasolla. Vertailukriteerit tässä tutkimuksessa ovat osittain samoja kuin Leppäsen (2013) sekä Ozcan-Topin ja Demirörsin (2013) tutkimuksissa. Uusina vertailukriteereinä olivat menetelmäsidonnaisuus, kohdealue, esitys ja käyttö. Uutta tässä tutkimuksessa on myös menetelmäkohtaiset vertailut.

Tämä tutkimus on tuottanut tiiviit kuvaukset laajasta joukosta ketterän ohjelmistokehityksen kypsyysmalleista ja niitä koskevia vertailutietoja. Tutkimuksen tuloksia voidaan hyödyntää monella tavalla käytännön työssä. Tutkimuksen perusteella saa hyvän yleiskuvan siitä, mitä kypsyysmalleilla ketterän ohjelmistokehityksen yhteydessä tarkoitetaan. Se antaa myös konkreettisen kuvan olemassa olevasta mallitarjonnasta, jonka pohjalta kukin organisaatio, projekti ja tiimi voi omalta kohdaltaan miettiä, mikä voisi olla sovelias sen käyttötarkoituksiin. Koska mikään malleista ei ole ylivertainen, on kuvausten ja vertailutietojen perusteella mahdollista räätälöidä käyttökohteeseen paremmin sovelias malli.

Kuten Schweigertin ym. (2013a) ja Schweigertin ym. (2013b) tutkimuksista käy ilmi, ketterän ohjelmistokehityksen kypsyysmalleja on runsaasti enemmän, kuin mitä tässä tutkimuksessa pystyttiin esittelemään ja vertailemaan. Tosin joitakin pois jätettyjä malleja ei ole tuotettu akateemisen tutkimuksen tuloksina. Malleja olisi voitu tutkia myös muiden kriteerien suhteen kuin mitä tässä työssä on tehty. Esimerkkinä tällaisesta on se, millä perusteella kypsyysmallin rakenne on muodostettu. Malleja tutkittiin ainoastaan kirjallisen materiaalin perusteella. Jotta kypsyysmallien sopivuudesta ja toimivuudesta saataisiin syvällinen ja luotettava käsitys, se edellyttäisi niiden käyttämistä kontrolloidusti käytännön tilanteessa, samaan tapaan kuin mitä perinteisten kypsyysmallien tutkimuksessa on tehty.

Ketterien menetelmien käyttö ovat nykyään yleistä, ja niiden käyttäminen organisaatioissa lisääntyy. Näin ollen myös menetelmien toimivuuden ja hyödyllisyyden arviointitarve lisääntyy. Koska vielä ei ole saatu aikaan yhteisesti

hyväksyttyä kypsyysmallia, uusia ketterän kehittämisen kypsyysmalleja julkaistaan ja olemassa olevia kehitetään. Jatkossa tutkimusta voisi laajentaa näihin uusiin malleihin. Kiinnostavaa olisi myös yrittää selvittää, millä edellytyksillä jostakin kypsyysmallista voisi tulla yleisesti hyväksytty ja minkälaisia vaatimuksia ja oletuksia ketterällä yhteisöllä olisi tällaista mallia kohtaan.

LÄHTEET

Abbas N., Gravell A. & Wills G. (2008). Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From? Teoksessa *Proceedings of the 9th International Conference, XP 2008.* Limerick, Ireland, June 10-14, 2008. Springer Berlin, Heidelberg. Haettu 15.5.2015 osoitteesta http://eprints.soton.ac.uk/266606/1/xp2008camera_ready.pdf

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta J. (2002). *Agile Software Development Methods: Review and Analysis*. VTT Publications 478. Haettu 1.6.2012 osoitteesta http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

Ambler S. (2009). The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments. Haettu 5.6.2015 osoitteesta http://www.webfinancialsolutions.com/wp-content/uploads/2011/10/Adapting-Agile-Methods-for-Complex-Environments.pdf

Ambler S. (2010). *The Agile maturity model (AMM)*, Dr Dobb's. Haettu 31.1.2011 osoitteesta http://www.drdobbs.com/architecture-and-design/224201005;jsessionid=P1KHI0JRB4C5ZQE1GHPCKH4ATMY32JVN

Anderson D. (2005). *Agile Management for Software Engineering, Applying the Theory of Constraints for Business Results*, Prentice Hall.

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

Baker, S.W. (2005). Formalizing agility: an agile organization's journey toward CMMI accreditation. Teoksessa *Proceedings of the Agile Development Conference*, (s. 185 – 192). IEEE Computer Society, Denver, CO, USA.

Basili V. (1992) *Software modeling and measurement: the Goal/Question/ Metric paradigm*. University of Maryland, College Park. Haettu 15.5.2015 osoitteesta http://drum.lib.umd.edu/bitstream/1903/7538/1/Goal_Question_Metric.pdf

Beck, K. (1999a), Embracing change with extreme programming. *Computer,* 32(10), 70-77.

Beck, K. (1999b). *eXtreme Programming Explained: Embrace Change*, Addison-Wesley, Kanada.

Beck K. ym. (2001). *Manifesto for Agile Software Development*. Haettu 2.5.2015 osoitteesta http://agilemanifesto.org/

Beck K. & Anders C. (2004). *Extreme Programming Explained: Embrace Change*, 2nd Edition, Addison-Wesley, Boston, United States.

Beck K. & Boehm B. (2003). Agility through discipline: A debate. *Computer,* 36(6), 44-46.

Becker, J., Knackstedt R. & Pöppelbuß, J. (2009). Developing maturity models for IT management – a procedure model and its application. *Business and Information Systems Engineering*, 1(3), 213-222.

Benefield R. (2010). Seven dimensions of agile maturity in the global enterprise: a case study. Teoksessa *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, Honolulu, Hawaji, January 5-8, 2010.

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer*, 35(1), 64-69.

Boehm B. & Turner R. (2003). *Balancing Agility and Discipline -A Guide for the Perplexed,* Addison-Wesley, 2003.

Bos E. & Vriens C. (2004). An Agile CMM. Teoksessa *Extreme Programming and Agile Methods - XP/Agile Universe 2004, Proceedings of the 4th Conference on Extreme Programming and Agile Methods*, (s. 129-138). Calgary, Canada Berlin: Springer-Verlag.

Bustard C., Wilkie G. & Greer D. (2013). The Maturation of Agile Software Development Principles and Practice: Observations on Successive Industrial Studies in 2010 and 2012. Teoksessa *Proceedings of the 20th IEEE International Conference and Workshops on the Engineering of Computer Based Systems (ECBS)* 2013, (s. 139 – 146). IEEE.

Cao L. & Ramesh B. (2008). Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*. 25(1), 60-67.

Cohen D., Lindvall M. & Costa P. (2004). An Introduction to Agile Methods, *Advances in Computers*, 62.

Conboy K. (2009). Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research,* 20(3), 329–354.

Conboy K., Coyle S., Wang X. & Pikkarainen M. (2011). People Over Process: Key People Challenges in Agile Development. *IEEE Software.* 28(4), 48-57.

Conboy, K. & Fitzgerald, B. (2004). Toward a Conceptual Framework for Agile Methods: a Study of Agility in Different Disciplines. Teoksessa Proceedings of the ACM Workshop on Interdisciplinary Software Engineering Research (WISER), 37–44.

Coram M. & Bohner S. (2005). The impact of agile methods on software project management. Teoksessa *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS '05,* (s. 363 – 370), 4-7 April, IEEE.

Crosby P. (1979) *Quality is Free*. New York, McGraw-Hill

Crosby P. (1986) *Running things*. New York, McGraw-Hill

De Bruin T., Freeze R., Kulkarni U. & Rosemann M. (2005). Understanding the Main Phases of Developing a Maturity Assessment Model. Teoksessa *Proceedings of ACIS 200516th Australasian Conference on Information Systems*, Sydney.

DeMarco T. & Boehm B. (2002). The agile methods fray. *Computer*, 35(6), 90-92.

Dybå, T. & Dingsøyr, T. (2008). Empirical studies of agile software development: A Systematic review. *Information and Software Technology*. 50(9-10), 833-859.

Figueiredo A. (2009). An Executive Scrum Team. Teoksessa *Agile Conference, 2009. AGILE '09.* (s. 147–150). Chicago, IL. IEEE.

Forrester Research Inc. (2010). *Forrester/Dr. Dobb's Global Developer Technographics® Survey*, Q3 2010.

Fowler, M. & Highsmith, J. (2001), *The Agile Manifesto*. Haettu 1.6.2012 osoitteesta http://www.pmp-projects.org/Agile-Manifesto.pdf

Fraser, P., Moultrie, J. & Gregory, M. (2002). The use of maturity models/grids as a tool in assessing product development capability. Teoksessa *Proceedings of the IEMC 02 2nd Engineering Management Conference*, Minneapolis, 244-249.

Glass R. (2001). Agile versus traditional: Make love, not war! *Cutter IT Journal*, 14(12), 12-18.

Glazer H. (2001). Dispelling the Process Myth: Having a Process Does Not Mean Sacrificing Agility or Creativity. CrossTalk. *The Journal of Defense Software Engineering,* 27-30. Haettu 15.5.2015 osoitteesta http://static1.1.sqspcdn.com/static/f/702523/9457149/1290003936870/200111-Glazer.pdf?token=pYatSBrcXE7IKtPKqrCOOnOlDO4%3D

Glazer H. (2010). Love and Marriage: CMMI and Agile Need Each Other. CrossTalk. *The Journal of Defense Software Engineering*, 23(1), 29-34.

Gujral R. & Jayaraj S. (2008, 2. elokuuta). *The Agile Maturity Model, The Agile Philly.* Haettu 10.5.2015 osoitteesta http://www.shaunjayaraj.com/2008/08/agile-maturity-model.html

Heeager L. (2013). The Agile and the Disciplined Software Approaches: Combinable or Just Compatible? Teoksessa Pooley R. ym. (toim.) *Information Systems Development: Reflections, Challenges and New Directions,* (s. 35-50). New York: Springer Science+Business Media.

Highsmith J. (2000). Retiring Lifecycle Dinosaurs, *Software Testing & Quality Engineering*, July/August, 2000, 22-28.

Highsmith J. (2000). *Adaptive Software Development*. Addison-Wesley

Highsmith J. (2004). *Agile Project Management, Creating innovative products.* Addison-Wesley.

Highsmith J. (2006) *Agile: from Rogue Team to Enterprise Acceptance*, Cutter-consortium: business technology trend and impacts.

Highsmith & Cockburn (2001). *Agile software development: the business of innovation.* 34(9). IEEE.

Humble J. & Russell R. (2009). *The Agile Maturity Model – Applied to building and releasing software.* Haettu 15.5.2015 osoitteesta http://tg-tatiana-oquendo.googlecode.com/svn/trunk/Agile%20Maturity%20Model%20Applied%20to%20Building%20and%20Releasing%20Software.pdf

Humphrey W. (1989). *Managing the Software Process*. SEI series in software engineering. Reading, Mass: Addison-Wesley.

Humphrey W. (1995). *Discipline for Software Engineering*. Addison-Wesley, Reading.

Huo, M., Verner, J., Zhu, L. & Babar, M. (2004). Software quality and agile methods. Teoksessa *Proceedings of the 28th Annual International Computer*

Software and Applications Conference (COMPSAC'04), (s. 520-525). IEEE Computer Society, Hong Kong.

Hämäläinen P. (2007). Kypsyyden monta astetta, *Tietokone*, 12, 55-56.

Jokela, T., Siponen, M., Hirasawa, N. & Earthy, J. (2006). A survey of usability capability maturity models: Implications for practice and research, *Behaviour and Information Technology*, 25(3), 263–282

King J. & Kraemer K. (1984). Evolution and organizational information systems: An assessment of Nolan's Stage model. *Communications of the ACM,* 27(5), 466-475.

Kollanus S. (2003) *Laatujärjestelmät*. Haettu 20.4.2012 osoitteesta http://users.jyu.fi/~kolli/ohjelmistotekniikka/files/laatujarjestelmat.pdf

Kollanus S. (2007) *CMMI*. Haettu 20.4.2012 osoitteesta http://users.jyu.fi/~kolli/OHTU2007/materiaali/CMMI.pdf

Komi-Sirviö S. (2004). *Development and evaluation of software process improvement methods*. Oulun yliopisto. VTT Publications 535, Espoo.

Kähkönen, T., & Abrahamsson, P. (2004). Achieving CMMI level 2 with enhanced extreme programming approach. Product Focused Software Process Improvement, 2004. Teoksessa *Proceedings of the 5th International Conference, PROFES 2004*, (s. 378-392). Kansai Science City, Japan, April 5-8, 2004.

Laanti M. (2012), *Agile Method in Large-scale Software Development Organizations, Applicability and Model for Adoption*. Väitöskirja, Oulun Yliopisto. Juvenes Print Tampere. Haettu 22.5.2015 osoitteesta http://herkules.oulu.fi/isbn9789526200347/isbn9789526200347.pdf

Laanti M., Similä J. & Abrahamsson P. (2013). Definitions of Agile Software Development and Agility. Teoksessa *Proceedings of the 20th European Conference, EuroSPI 2013* (s.247-258). Berlin: Springer –Verlag.

Larman C. & Basili V. (2003). Iterative and Incremental Development: A Brief History, *Computer* 36(6), 47-56.

Leppänen M. (2013). A Comparative Analysis of Agile Maturity Models. Teoksessa Pooley R. ym. (toim.) *Information Systems Development: Reflections, Challenges and New Directions*, (s. 329-343). New York: Springer Science+Business Media.

Lui K. & Chan K. (2005). A road map or implementing extreme programming. Internal Software Process Workshop Teoksessa M. Li, B. Boehm & L.J. Osterweil (toim.): *Proceedings of the SPW 2005, LNCS 3840*, (s. 474 – 481). Berlin: Springer-Verlag.

Mahnic V. & Zabkar N. (2007). Introducing CMMI Measurement and Analysis Practices into Scrumbased Software Development Process. *International Journal of Mathematics and Computers in Simulation*, 65-72.

Mahnic V. & Zabkar N. (2008). Measurement repository for Scrum-based software development process. Teoksessa *Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications, World Scientific and Engineering Academy and Society (WSEAS)*, (s. 23-28). Stevens Point, Wisconsin, USA. Haettu 15.5.2015 osoitteesta

http://www.wseas.us/e-library/conferences/2008/mexico/cea/1-%20CEA.pdf

Malik N. (2007). *Simple lifecycle agility maturity model*. Haettu 10.5.2015 osoitteesta http://blogs.msdn.com/b/nickmalik/archive/2007/06/23/simple-lifecycle-agility-maturity-model.aspx

Mann C. & Maurer F. (2005). *A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction*. Haettu 15.5.2015 osoitteesta http://ase.cpsc.ucalgary.ca/uploads/Publications/MannMaurerAU2005.pdf

Marcal A., de Freitas B., Furtado Soares F., Furtado M., Maciel T. & Belchior A. (2008). Blending Scrum practices and CMMI project management process areas. *Innovations of System Software Engineering* 2008(4), 17–29.

McMahon P. (2005). Extending Agile Methods: A Distributed Project and Organizational Improvement Perspective. Teoksessa *Proceedings of Systems and Software Technology Conference,* April 2005.

Mettler, T. (2010). Thinking in terms of design decisions when developing maturity models. *International Journal of Strategic Decision Sciences*, 1(4), 76-87.

Mettler T. & Rohner P. (2009) Situational maturity models as instrumental artifacts for organizational design. Teoksessa *DESRIST'09 Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology.* New York, USA. Haettu 20.5.2015 osoitteesta http://desrist2009.ist.psu.edu/Papers/desrist2009_submission_5.pdf

Moe N., Dingsøyr T. & Dybå T. (2010). A teamwork model for understanding an agile team: A case study of a Serum project. *Information and Software Technology,* 52(5), 480-491.

Nawrocki J., Walter B. & Wojciechowski A. (2001). Towards the maturity model for extreme programming. Teoksesssa *Proceedings of the 27th EUROMICRO Conference 2001: A Net Odyssey*, Warsaw, Poland, September 4-6, (s. 233-239). IEEE Computer Society.

Ngwenyama O. & Nielsen P. (2003) Competing values in software process improvement: an assumption analysis of CMM from an organizational culture perspective, *IEEE Transactions on Engineering Management*, 50(1), 100-112.

Nolan R. (1973). Managing the computer resource: a stage hypothesis *Communications of the ACM*, 16(7), 399-405.

Ozcan-Top O., Demirörs O. 2013. Assessment of agile maturity models: a multiple case study. Teoksessa *Proceedings of the EuroSpi 2013 Conference.*

Packlick J. (2007). The agile maturity map: a goal oriented approach to agile improvement. Teoksessa J. Eckstein, F. Maurer, R. Davies, G. Melnik & G. Pollice (toim.) *Proceedings of the Agile 2007*, Washington D.C., August 13-17, 2007, (s. 266-271). Los Alamitos: IEEE Computer Society.

Palmer S. & Felsing M. (2002). *A Practical Guide to Feature Driven Development*. Pearson Education.

Patel C. & Ramachandran M. (2009). Agile maturity model (AMM): a software process improvement framework of agile software development practices. *International Journal of Software Engineering* 2(1), 3-28.

Paulk M. (2001). Extreme programming from a CMM perspective. *IEEE Software*, 18(6), 19-26.

Paulk M. (2002). *Agile Methodologies and Process Discipline*. Carnagie Mellon University, Institute for Software Research. Haettu 15.5.2015 osoitteesta http://repository.cmu.edu/cgi/viewcontent.cgi?article=1012&context=isr

Paulk M. (2009). A History of the Capability Maturity Model for Software. *The Software Quality Profile*, 1(1), 5-19.

Paulk M., Curtis B., Chrissis M. & Weber C. (1993). *Capability maturity model for software, Version 1.1,* CMU/SEI-93-TR-24, Software Engineering Institute. Pittsburg, PA: Carnegie Mellon University.

Paulk M. & Konrad D. (1994). An Overview of ISO's SPICE Project. *American Programmer*, 7(2), 16-20.

Pettit, R. (2006). An ''agile maturity model?". *Agile Journal 2006*. Haettu 2.2.2011 osoitteesta http://www.agilejournal.com/articles/columns/the-agile-manager/52-an-qagile-maturity-modelq.

Phillips M. (2003). *CMMI V1.1 Tutorial*. Haettu 1.6.2012 osoitteesta http://www.sei.cmu.edu/

Pikkarainen M. (2008). *Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices*. Väitöskirja, VTT Publications 695, Helsinki.

Pikkarainen M., Haikara J., Salo O., Abrahamsson P. & Still J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering* 13(3). 303–337.

Pikkarainen M. & Mäntyniemi A. (2006). An Approach for Using CMMI in Agile Software Development Assessments: Experiences from Three Case Studies. Teoksessa *Proceedings of SPICE 2006 Conference*, Haettu 15.5.2015 osoitteesta http://ulir.ul.ie/bitstream/handle/10344/2257/2006_Pikkarainen.pdf?sequence=2

Pikkarainen M. & Wang X. (2011). *An Investigation of Agility Issues in Scrum Teams Using Agility Indicators.* Teoksessa W.W. Song ym. (toim.) *Information Systems Development* (s. 449-459), Springer Science + Business Media

Proulx M. (2010, 12. elokuuta*). Yet another agile maturity model (AMM) – The 5 levels of Maturity.* Haettu 20.4.2011 osoitteesta http://analytical-mind.com/2010/07/12/yet-another-agile-maturity-model-the-5-levels-of-maturity/

Pöppelbuß J., Niehaves B., Simons A. & Becker J. (2011). Maturity models in information systems research: literature search and analysis. *Communications of the Association for Information Systems*, 29(27), 504-533.

Qumer, A. & Henderson–Sellers, B. (2006a). Comparative evaluation of XP and Scrum using the 4D Analytical Tool (4-DAT). Teoksessa Z. Irani, O. D.

Sarikas, J. Llopis, R. Gonzalez & J. Gasco (toim.), *Proceedings of the European and Mediterranean Conference on Information Systems (EMCIS)*, (s. 1-8). London: Brunel University.

Qumer, A. & Henderson–Sellers, B. (2006b). Measuring agility and adoptability of agile methods: a 4–dimensional analytical tool. Teoksessa N. Guimares, P.Isaias A. Goikoetxea (toim.), *Proceedings of the IADIS International Conference on Applied Computing,* (s. 503–507). San Sebastian, Spain. IADIS.

Qumer A. & Henderson-Sellers B. (2008). A Framework to support the evaluation, adoption and improvement of agile methods in practice. *The Journal of Systems and Software,* 81, 1899-1919.

Renken J. (2004). Developing an IS/IT management capability maturity framework. Teoksessa *Proceedings of SAICSIT, the Annual Conference of the South African Institute of computer scientists and information technologists*, 53-62.

Rodriguez P., Markkula J., Oivo M. & Turula K. (2012). Survey on agile and lean usage in Finnish software industry. Teoksessa *ESEM'12 Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement* (s. 139-148). New York, USA, ACM New York.

Salo O. & Abrahamsson P. (2008). Agile methods in European embedded software development organizations: a survey on the actual use and usefulness of Extreme Programming and Scrum. *IET Software*, 2(1), 58-64.

Schatz B. & Abdelshafi I. (2005). Primavera Gets Agile: A Successful Transition to Agile Development. *IEEE Software*, 36-42.

Schwaber K. (1995). SCRUM Development Process. Teoksessa Patel D., Casanave C., Hollowell G., Miller J. & Sutherland J. (toim.*) Proceedings of the Business Object Design and Implementation Workshop*, (s. 1-23). Austin, Texas, USA, October 15-19. Springer.

Schwaber K. (2000). Against a Sea of Troubles: Scrum Software Development, *Cutter IT Journal*, 13(11), 34-39.

Schwaber K. (2004). *Agile Project Management with Scrum*. Microsoft Press.

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

Schwaber K. & Sutherland J. (2013). *The Scrum Guide*. Haettu 15.5.2015 osoitteesta http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

Schweigert T., Nevalainen R., Vohwinkel D., Korsaa M. & Biro M. (2012). Agile Maturity Model: Oxymoron or the Next Level of Understanding. Teoksessa *Proceedings of the 12th International Conference, SPICE 2012*, (s. 289-294). Palma, Spain, Berlin: Springer-Verlag.

Schweigert T., Vohwinkel D., Korsaa M., Nevalainen R. & Biro M. (2013a). Agile maturity model: a Synopsis as a First Step to Synthesis. Teoksessa *Proceedings of the 20th European Conference, EuroSPI-2013*. Dundalk, Ireland. Berlin: Springer-Verlag.

Schweigert T., Vohwinkel D., Korsaa M., Nevalainen R. & Biro M. (2013b). Agile maturity model: analysing agile maturity characteristics from the SPICE perspective. *Journal of Software: Evolution and Process*, 26(5), 513-520.

SEI 2006. *CMMI for development, version 1.2* - Improving processes for better products. Carnagie Mellon. Software Engineering Institute. Haettu 6.6.2012 osoitteesta http://www.sei.cmu.edu/reports/06tr008.pdf

SEI 2010 *Capability Maturity Model Integration CMMI, version 1.3.* CMMI product Team 2010. Haettu 6.6.2012 osoitteesta http://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15287.pdf

Sidky A., Arthur J. & Bohner S. (2007). A disciplined approach to adopting agile practice: the agile adoption framework. *Innovations in Systems and* Software Engineering, 3, 203-216.

Stapleton J. (1997). *Dynamic systems development method - The method in practice.* Addison Wesley.

Turner R. & Jain A. (2002). Agile Meets CMMI: Culture Clash or Common Cause? Teoksessa *Extreme Programming and Agile Methods – XP/Agile Universe Conference, Proceedings of the Second XP Universe and First Agile Universe,* (s. 153-165). Chicago, IL, USA, August 4–7. Berlin: Springer-Verlag.

Vanderburg G. (2005). A Simple Model of Agile Software Processes or Extreme Programming Annealed. Teoksessa *Proceedings of the Annual ACM SIGPLAN Conference of Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA*), 539-545. ACM San Diego.

VersionOne 2010. *5th Annual Survey* "*The State of Agile Survey*". Haettu 1.10.2011 osoitteesta http://www.versionone.com/state_of_agile_development_survey/10/

VersionOne 2014. *9th Annual Survey* "*The State of Agile Survey*". Haettu 10.5.2015 osoitteesta http://info.versionone.com/state-of-agile-development-survey-ninth.html

Vriens C. (2003). Certifying for CMM Level 2 and ISO9001 with XP@Scrum. Teoksessa Proceedings of the Agile Development Conference, (s. 120-124).

Yin A., Figueiredo S. & da Silva M. (2011). Scrum maturity model. Teoksessa *ICSEA2011: The Sixth International Conference on Software Engineering Advances,* IARIA 2011. Haettu 1.6.2012 osoitteesta http://www.thinkmind.org/download.phparticleid=icsea_2011_1_40_10440